Tivoli software

IBM

Integrating IBM Tivoli Workload Scheduler with Tivoli Products

- Integrate Tivoli Workload Scheduler with other IBM products
- Experiment with real-life scenarios
- Learn the benefits of integration

Vasfi Gucer
Michael Ferguson
Warren Gill
Martin R Grange
Dean Harrison
Tina Lamacchia
Sriraman Padmanabhan
Steve Sando
Sharon Wheeler
Anthony Yen

Redbooks

ibm.com/redbooks

International Technical Support Organization

Integrating IBM Tivoli Workload Scheduler with Tivoli Products

May 2005

SG24-6648-00

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

First Edition (May 2005)

This edition applies to IBM Tivoli Workload Scheduler and IBM Tivoli Workload Scheduler for z/OS Version 8.2.

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 illustrates 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. 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 IBM's application programming interfaces.

Trademarks

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

@server®	GDDM®	Parallel Sysplex®
@server®	Informix®	Passport Advantage®
Redbooks (logo) ™	IBM®	Planet Tivoli®
ibm.com®	IMS™	QMF™
z/OS®	Lotus Notes®	Redbooks™
zSeries®	Lotus®	RACF®
AIX®	Maestro™	Tivoli Enterprise™
CICS®	MVS™	Tivoli Enterprise Console®
Domino®	NetView®	Tivoli®
DB2®	Notes®	TME 10™
DFS™	OMEGAMON®	TME®
DFSMSrmm™	OS/390®	WebSphere®

The following terms are trademarks of other companies:

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

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

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

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

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

Preface

This IBM® Redbook explains the benefits and technical merits of integrating IBM Tivoli® Workload Scheduler Distributed and IBM Tivoli Workload Scheduler for z/OS® with other IBM products. Scheduling is a mission critical process for any company. However, when you talk about scheduling, you are really talking about an ecosystem. In this ecosystem, each solution is a building block that adds value to the overall solution. With IBM Tivoli Workload Scheduler, you can collect and add data to and from each component. In addition, expanding the scheduling ecosystem to include monitoring, management, help desk, storage, and business systems management provides greater value.

This book discusses all these integration points and provides detailed scenarios on how to integrate IBM Tivoli Workload Scheduler with these types of applications.

Because workload management is widely considered the nucleus of the data center, there are numerous opportunities for you to integrate IBM Tivoli Workload Scheduler with other products. This book addresses just some of these many opportunities. In terms of integration with IBM Tivoli Workload Scheduler, do not limit yourself to the products that this book discusses. Integration points discussed in this book should give you an idea of the potential value that IBM Tivoli Workload Scheduler integration can provide for your company.

The team that wrote this redbook

This IBM Redbook was produced by a team of specialists from around the world working at the International Technical Support Organization (ITSO), Austin Center.

Vasfi Gucer is an IBM Certified Consultant IT Specialist working at the ITSO, Austin Center. He has worked with IBM Turkey for 10 years and has been with the ITSO since January 1999. He has more than 10 years experience in systems management, networking hardware, and distributed platform software. He has worked on various Tivoli customer projects as a Systems Architect in Turkey and in the U.S. Vasfi is also a Certified Tivoli Consultant.

Michael Ferguson is a Senior IT specialist in the IBM Support center in Australia. He has 19 years of experience in the z/OS software field, including Tivoli Workload Scheduler for z/OS. His areas of expertise include Tivoli Workload Scheduler for z/OS, Parallel Sysplex®, and z/OS. He teaches and

performs services in both Tivoli Workload Scheduler for z/OS and Parallel Sysplex in the Asia Pacific region.

Warren Gill is working in the SWAT team in Austin to assist with sales of various deployments in Tivoli Workload Scheduler and other Tivoli products. He has been a courseware designer and trainer for Tivoli Workload Scheduler and its predecessor Maestro™, as well as a customer support engineer. Warren has been with Tivoli Systems since December 1997, when Tivoli acquired Unison Software. He has been using Tivoli Workload Scheduler since 1990.

Martin R Grange is an Advisory IT Specialist working in Server Systems Operations (SSO) in IBM Global Services in Portsmouth, UK. He has a Computer Science degree from Portsmouth Polytechnic and has worked for IBM since 1983. He has 18 years of experience working with Tivoli Workload Scheduler for z/OS (formerly OPC) and supporting reporting databases using Tivoli Performance Reporter for OS/390® and TDS for OS/390 (formerly EPDM, PR/MVS™). He designed and supports the UK standard SMF process and currently supports several multi-TDS for OS/390 database environments for both the UK internal account and many commercial accounts.

Dean Harrison has had 20 years experience in scheduling and automation, ranging from doing the day-to-day scheduling and JCL creation, to configuration and customization of scheduling and automation products. Dean is a regular contributor to the Tivoli Workload Scheduler community, having presented at Tivoli Workload Scheduler and Tivoli events in the UK and across Europe. Now working for Diversified Software Systems, Inc., Dean is using his Tivoli Workload Scheduler experience to assist the Research and Development team to further integrate their products with Tivoli Workload Scheduler, delivering even more value and productivity in the field of JCL management.

Tina Lamacchia has been in the IT realm for 18 years. Her career started with programming, and then administrating hundreds of UNIX® systems. She has been with Tivoli Systems for six years. Tina has worked with Tivoli Workload Scheduler for nine years as a customer and as an Tivoli Workload Scheduler advocate. Currently, she is working in the SWAT team in Austin to assist with sales of various deployments in Tivoli Workload Scheduler and all PACO products.

Sriraman Padmanabhan is currently a SWAT Customer Solutions Engineer for Tivoli Software, IBM. He is engaged in client-based consulting for Tivoli solutions. Sriraman has been a software engineer in product development for more than five years.

Steve Sando has been developing mainframe oriented packaged software products for 25 years. His job experience includes designing and implementing products for Valor Software, ADS, and Compuware. He is currently chief architect for all C++ based products at Diversified Software in Morgan Hill. He has worked directly on product design with a number large scale data centers , such as Merrill Lynch, Citigroup, and Exxon.

Sharon Wheeler is an Tivoli Customer Support Engineer, based in Research Triangle Park, North Carolina. She is currently a member of the Americas Tivoli Workload Scheduler L2 Support Team. She began working for IBM as a member of the Tivoli services team in 1997, joined the Tivoli Customer Support organization in 1999, and has supported a number of products, most recently TBSM. In 2004, she began working on the Tivoli Workload Scheduler for z/OS L2 Support team.

Anthony Yen is a Senior IT Consultant with IBM Business Partner Automatic IT Corporation, http://www.AutomaticIT.com, in Austin, Texas, United States. He has delivered 19 projects involving 11 different Tivoli products over the past six years. His areas of expertise include Enterprise Console, Monitoring, Workload Scheduler, Configuration Manager, Remote Control, and NetView®. He has given talks at Planet Tivoli® and Automated Systems and Planning OPC and Tivoli Workload Scheduler Users Conference (ASAP). Anthony has taught courses on Tivoli Workload Scheduler. He has been an IBM Certified Tivoli Consultant since 1998.

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

Budi Darmawan, Mike Foster
ITSO, Austin Center

Robert Haimowitz, Debbie Willmschen
ITSO, Raleigh Center

Terri Buchanan, JC Clement, Art Eisenhour, Jaynna Sims, Doug Specht, Kim Nguyen-Theunissen
IBM USA

Luke Marshall
IBM UK

Finn Bastrup Knudsen
IBM Denmark

Domenico Chillemi
IBM Italy

Jean Gabriel Weyer
IBM France

Tom Clark, Sal Deconte, Robert.Sackett, Ron Sheets
Diversified Software

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners, or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll 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 Redbooks™ to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

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

  ibm.com/redbooks

- Send your comments in an e-mail to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. JN9B Building 905
  11501 Burnet Road
  Austin, Texas 78758-3493

Part 1

Introduction

This part describes the benefits and architecture of integrating IBM Tivoli Workload Scheduler (Distributed and z/OS) with various IBM and Tivoli products.

Because workload management is widely considered the nucleus of the data center, there are numerous opportunities for you to integrate Tivoli Workload Scheduler (TWS) with other products. This book addresses just some of these many opportunities. In terms of integration with TWS, do not limit yourself to the products that this book discusses. Integration points discussed in this book should give you an idea of the potential value that TWS integration can provide for your company.

1

Architecture of integration

This chapter introduces various IBM Tivoli products that can integrate with Tivoli Workload Scheduler (z/OS and Distributed). It briefly identifies the function of each product, its benefits, and how it integrates with Tivoli Workload Scheduler (TWS). This chapter also lists potential synergistic integrations with other IBM Tivoli products.

While this chapter focuses on individual product integration and the relationship of those individual products to TWS, other products can interact with each other as well. There are many potential combinations of these products that can be integrated to deliver solutions to many batch scheduling problems. Each section that covers a product includes a matrix that illustrates other IBM Tivoli products that can be integrated against the product to help you identify possible combinations, which is discussion in detail in Appendix A, “Using the right tool for the job” on page 655.

We installed TWS and other Tivoli products on the test systems in our lab (or, in some cases, the products were already installed for us). Our scenarios, then, start from the point of adding any functions, configurations, and so forth as needed to integrate the products in such a way as to create a *best practices* environment. Because each product integration scenario expects the user to have installed the Tivoli products in the system environment as a baseline, this book does not duplicate the documentation of that effort.

1.1 IBM TWS for z/OS solution scenario

To integrate the mainframe-based products, we built a small TWS for z/OS environment consisting of a single mainframe, one Windows® machine, and one AIX® system.

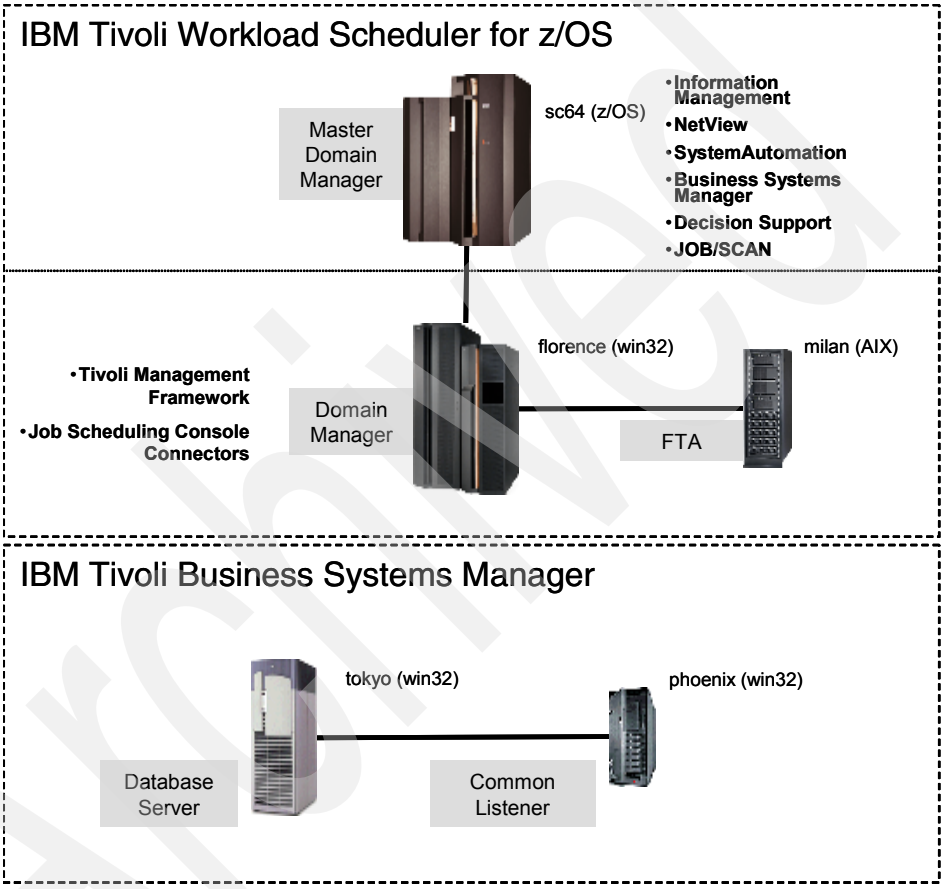

Figure 1-1 The TWS for z/OS network

All of the Tivoli products for zSeries® were installed on this single z/OS image, and the configurations were performed there. An existing Tivoli Business Systems Manager installation was running on two Windows machines in our lab, so we have added our integrations to that existing environment rather than creating a new one.

This section discusses the following end-to-end products:

- Integrating with Tivoli Information Management for z/OS
- Integrating with Tivoli NetView for z/OS
- Integrating with Workload Manager for z/OS
- Integrating with Tivoli Decision Support for OS/390
- Integrating with Tivoli Business Systems Manager (for z/OS)
- Integrating with Tivoli System Automation for z/OS
- Integrating with DFSMSrmm™
- Integrating with JOB/SCAN

You can find information about distributed products in 1.2, “IBM TWS solution scenario” on page 25.

1.1.1 Integrating with Tivoli Information Management for z/OS

Tivoli Information Management for z/OS (Infoman) is a systems management product which provides an integrated set of tools and services for customizing and automating your:

- Help desk or call center
- Problem management
- Change management
- Configuration management

This section looks at the integration of TWS for z/OS with the problem management application of Infoman.

Integration benefits

By integrating TWS for z/OS with Infoman problem management, problem records are created for all batch job failures in Infoman at the time of job failure. Without this integration, the batch operator creates a problem record for each job failure manually. During busy periods, it can be hours before the operator has a chance to create the required problem reports. In some cases, problem records might not be raised at all due to an oversight.

Another important benefit is that the initial data that is entered into the problem record is consistent because it is extracted from TWS for z/OS at the time of failure. For example, the problem description can contain certain keywords in specific locations in the description about the job that failed. Thus, reports have a consistent look for the problem description.

For more information and real-life scenarios that use this integration, refer to Chapter 2, “Integrating Tivoli Infoman” on page 41.

Integration matrix

Potential integration points between Tivoli Information Management and other Tivoli products for end-to-end TWS environments include:

- Tivoli NetView for z/OS
- Workload Manager for z/OS
- Tivoli Decision Support for z/OS
- Tivoli OMEGAMON®
- Tivoli System Automation for z/OS

1.1.2 Integrating with Tivoli NetView for z/OS

With Tivoli NetView for z/OS, you can manage complex systems and networks from a single point. When you customize Tivoli NetView for z/OS, you can use it to filter data and messages and to automate responses to events in the environment. Through the use of Tivoli NetView for z/OS, problems in TWS can be used to trigger responses, including automated notification of personnel when there is a problem.

Integration benefits

Integration of TWS for z/OS with Tivoli NetView for z/OS allows support center personnel to focus on the results of batch applications, instead of focusing upon the batch applications themselves. Tivoli NetView for z/OS has the capacity for filtering messages from TWS for z/OS and responding to the messages. You can integrate Tivoli NetView for z/OS with other applications, such as Tivoli System Automation and SMTP, to respond to messages that are generated by TWS for z/OS. This integration can save both resources and response time.

For more information and real-life scenarios that use this integration, refer to Chapter 3, “Integrating Tivoli NetView for z/OS” on page 79.

Integration matrix

Potential integration points between Tivoli NetView for z/OS and other Tivoli products for end-to-end TWS environments include:

- Tivoli Business Systems Manager
- Tivoli System Automation for z/OS
- Tivoli Information Management
- Tivoli Decision Support/390

1.1.3 Integrating with Workload Manager for z/OS

With Workload Manager for z/OS (WLM), you can define performance goals and business importance for your workload. Workload Manager for z/OS then assigns systems resources, such as processor, storage, and I/O-based resources, on these goals. When Workload is not meeting its goals, Workload Manager for z/OS uses the business importance to decide which workload receives resources. This decision might involve taking resources away from a workload which has a lower business importance than the workload which requires the extra resources to meet its goal.

Workload is broken into groups which have similar performance goals. These groups are known as *service classes*. Workload Manager for z/OS assigns resources to each service class.

Integration benefits

Workload Manager for z/OS bases its decisions on real-time performance. When a workload starts running in the system, Workload Manager for z/OS assigns resources with regard to response time or the amount of time that the job must wait for system resources. These decisions are made dynamically through the entire time that the job is running. TWS for z/OS also has goals. The TWS for z/OS goals differ from the Workload Manager for z/OS goals in that TWS for z/OS has a specific time by which a job must complete. Integration of TWS for z/OS and Workload Manager for z/OS allows TWS for z/OS to request Workload Manager for z/OS to place a batch job into a service class with higher performance goals. This placement allows a batch job to receive more resources than it would in the service class to which it is normally assigned.

The benefit is that if TWS for z/OS decides that a job stream will not meet its deadline or is running too long, TWS for z/OS can request Workload Manager for z/OS to give the batch jobs in this stream more resources until they are back on schedule.

Integration with TWS for z/OS

TWS for z/OS integrates with Workload Manager for z/OS (WLM) using Workload Manager for z/OS Services as shown in Figure 1-2.

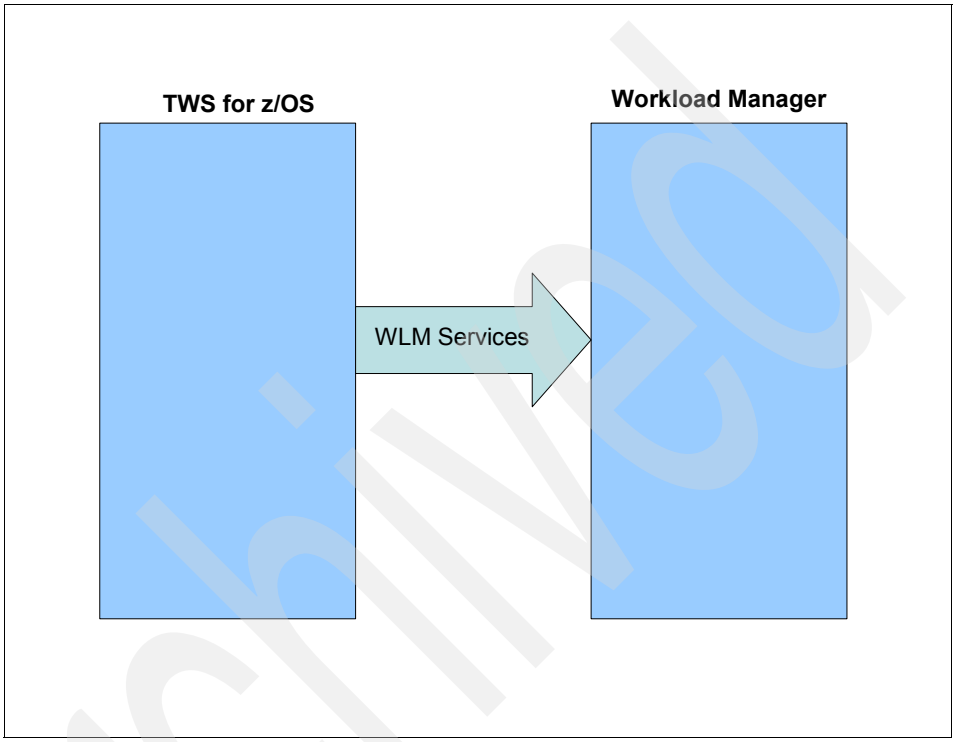

Figure 1-2 TWS for z/OS interface to Workload Manager

TWS for z/OS uses these services to request Workload Manager for z/OS to run each batch job that is not meeting its deadline in a predefined high performance service class. The batch job should run quicker than normal in this service class. TWS for z/OS continues to run each batch job in a stream in this higher performance service class until the stream is back on schedule.

The interface is controlled by the Workload Manager for z/OS parameter on the OPCOPTS initialization statement that is used by the TWS for z/OS controller.

This parameter has two fields:

- The name of the high performance service class.
- The default criteria that is used by TWS for z/OS to determine which jobs to promote to the high performance service class. This criteria can be overridden at the individual job level.

Integration matrix

Potential integration points between Workload Manager for z/OS and other Tivoli products for end-to-end TWS environments include Tivoli System Automation for z/OS.

1.1.4 Integrating with Tivoli Decision Support for OS/390

Tivoli Decision Support for OS/390 is a DB2®-based relational database used for the collection of systems management data and the presentation of it in both graphical and tabular formats.

Integration benefits

By integrating TWS for z/OS with Tivoli Decision Support for OS/390 all data collection and report production can be automated. Without this integration the Tivoli Decision Support for OS/390 administrator would have to collect the TWS for z/OS data and run many reports manually.

Another benefit is that the data collection and report production can be scheduled to run overnight, during a predefined batch window for example, to ensure that the database is available during business hours for the users and reports are available for meetings, and so forth.

For more information and real-life scenarios that use this integration, refer to Chapter 6, “Integrating Tivoli Decision Support for OS/390” on page 209.

Integration matrix

Potential integration points between Tivoli Decision Support for OS/390 and other Tivoli products for end-to-end TWS environments include:

- Tivoli NetView for z/OS
- Tivoli Information Management
- Tivoli Data Warehouse
- Tivoli Service Level Advisor

1.1.5 Integrating with Tivoli Business Systems Manager (for z/OS)

Tivoli Business Systems Manager monitors critical business systems, allowing support organizations, application owners, and management to view system components according to organizational priority or as they relate to overall business needs. When resources are defined to Tivoli Business Systems Manager, they are available from the console and the executive dashboard, which allows quick visualization of the health of any critical business system. Business systems can include resources ranging from z/OS to distributed system resources.

Integration benefits

Tivoli Business Systems Manager integrates with TWS for z/OS to load the daily plan, applications and jobs build business systems into the Tivoli Business Systems Manager database.

Status from the batch applications can be used to create lines of business that reflect the current state of critical jobs and applications. This allows organizations to monitor the applications in meaningful lines of business. By monitoring the impact of relevant jobs and applications from the daily plan, organizations can focus resources more responsively upon the outcome of the jobs and applications.

For more information and real-life scenarios that use this integration, refer to Chapter 5, “Integrating Tivoli Business Systems Manager” on page 183.

Integration matrix

Potential integration points between Tivoli Business Systems Manager (TBSM) and other Tivoli products for TWS for z/OS environments include:

- Tivoli Service Level Advisor
- Tivoli NetView for z/OS
- Tivoli Data Warehouse
- Tivoli System Automation for z/OS
- Tivoli Enterprise™ Console
- Tivoli Monitoring
- Tivoli OMEGAMON

1.1.6 Integrating with Tivoli System Automation for z/OS

System Automation for z/OS automates, monitors, and manages resources in the enterprise, both software and hardware. System Automation for z/OS has powerful features for starting, stopping, and monitoring resources in the system. It also has timers that it can use. However, it does not have the scheduling or restart and recovery functions that TWS for z/OS has.

Integration benefits

Integration of System Automation for z/OS and TWS for z/OS allows the advanced scheduling functions of TWS for z/OS to schedule resources which System Automation for z/OS manages. It also allows TWS for z/OS to perform restart and recovery actions on failed subsystems. Communication between System Automation for z/OS and TWS for z/OS is 2-way. This means System Automation for z/OS can notify TWS for z/OS when resources are not available and update the status of corresponding special resources defined in TWS for z/OS.

A good example of this integration is using TWS for z/OS to manage DB2. Many batch jobs require that DB2 be available for them to execute. TWS for z/OS can be used to request that DB2 be started at a specific time of day. System Automation for z/OS starts DB2 then responds to TWS for z/OS when DB2 has successfully initialized. At the same time, System Automation for z/OS changes the status of the DB2 special resource in TWS for z/OS to available. Batch jobs dependant on DB2 can now run.

Should DB2 terminate, System Automation for z/OS automatically changes the status of the DB2 special resource in TWS for z/OS to unavailable preventing any DB2 batch jobs from running as these would also terminate. When the problem with DB2 is fixed, it is restarted and System Automation for z/OS changes the status of the DB2 special resource to available, allowing the DB2 batch jobs to run.

TWS for z/OS also performs dependency checking. It can schedule a subsystem to start at a particular time but if its predecessor is not complete, the subsystem will not start. This works well with CICS®. CICS may need to be down until overnight batch jobs complete. If batch is running late, CICS remains down until it is complete.

This interface can also be used by TWS for z/OS to enter z/OS commands. System Automation for z/OS and TWS for z/OS complement each other and provide a truly powerful automation platform for your systems.

Integration with TWS for z/OS

System Automation for z/OS has an OPC automation extension. This extension is still called OPC but works exactly the same with TWS for z/OS. This extension provides additional functions used specifically to interface with TWS for z/OS.

TWS for z/OS uses its Operation-Status-Change exit 7 to send requests to System Automation for z/OS. This exit is more commonly referred to as its module name EQQUX007. Each time an operation changes status in TWS for z/OS, this exit is called. System Automation for z/OS provides its own copy of this exit. When the TWS for z/OS workstation data this exit receives has a specific format it passes the request to the NetView Program to Program Interface (PPI). The NetView PPI has specific TWS for z/OS related automation operators which perform the TWS for z/OS request. When the request has been processed, System Automation for z/OS uses the TWS for z/OS supplied program routine EQQUSINT to update TWS for z/OS as shown in Figure 1-3 on page 12.

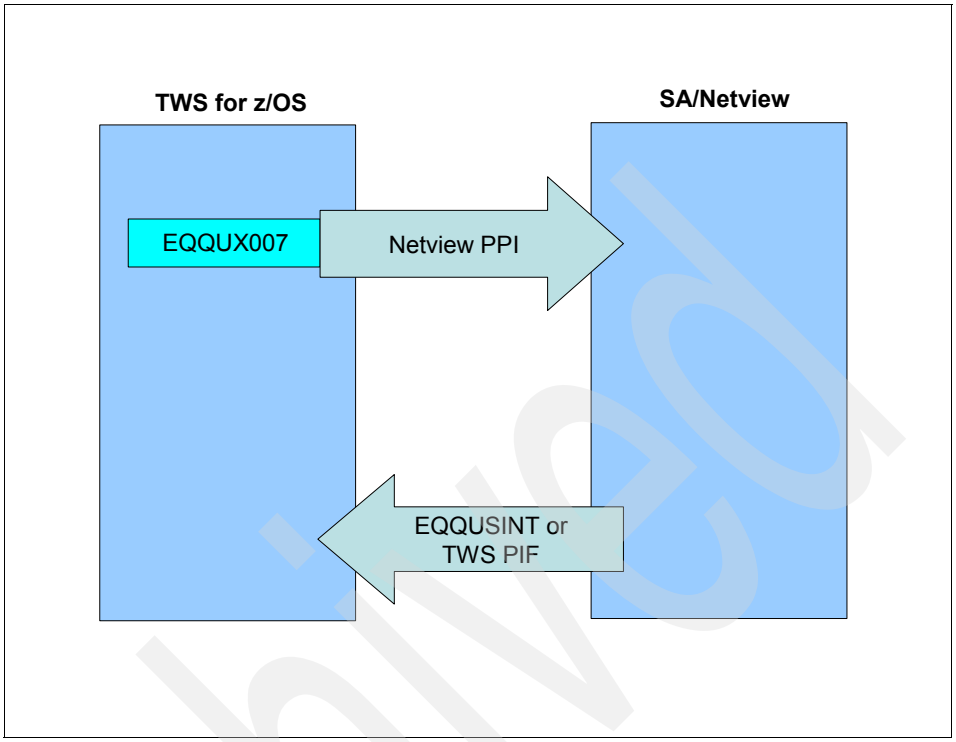

Figure 1-3 TWS for z/OS interface to System Automation for z/OS

Often, it is System Automation for z/OS which triggers the interface. This is the case where a TWS for z/OS special resource represents a system resource. In this case, when the resource that System Automation for z/OS is monitoring has a problem and is no longer available, System Automation for z/OS uses a TWS for z/OS supplied program routine EQQUSINT to change the status of the corresponding special resources in TWS for z/OS to unavailable.

The operator can also access TWS for z/OS data from within System Automation for z/OS. System Automation for z/OS uses the TWS Program Interface (PIF) to read the TWS data and present the data to the System Automation for z/OS user.

Integration matrix

Potential integration points between System Automation for z/OS and other Tivoli products for TWS for z/OS environments include:

- Tivoli NetView for z/OS
- Workload Manager for z/OS

Note: Apart from these products, System Automation for z/OS also integrates with subsystems such as CICS and IMS™.

1.1.7 Integrating with DFSMSrmm™

DFSMSrmm (Data Facility System Managed Storage removable media manager) manages your removable media such as tape volumes and the data sets contained on the volumes. It also manages the location of tape volumes. For example, volumes could be:

- On a shelf
- On a shelf in another location
- In a vital records store
- In an automated, virtual, or manual tape library

Integration benefits

TWS for z/OS has a function called Restart and Cleanup. This function is used to delete data set to enable a failed job to be restarted or rerun.

The benefit of integrating TWS for z/OS with DFSMSrmm is that TWS for z/OS can perform cleanup actions on tape volumes using the DFSMSrmm Application Programming Interface (API).

Integration with TWS for z/OS

The integration of TWS for z/OS with DFSMSrmm is seamless. The interface is activated using the DSTRMM(YES) parameter of the RCLOPTS initialization statement. TWS for z/OS then uses the DFSMSrmm API (as shown in Figure 1-4 on page 14) for tape volumes defined to DFSMSrmm.

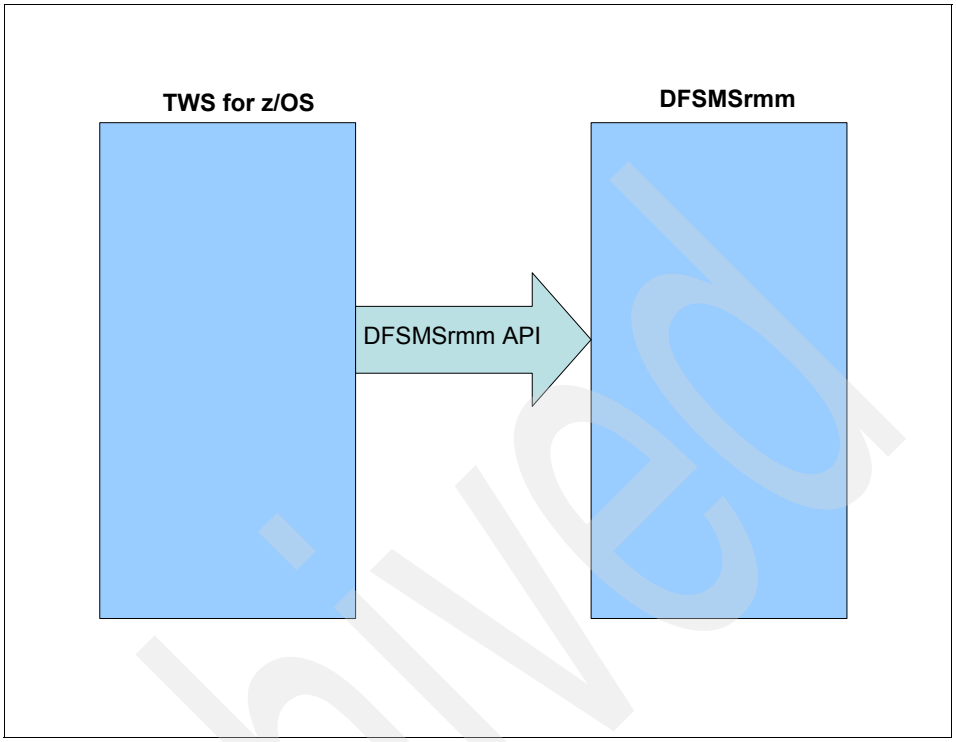

Figure 1-4 TWS for z/OS interface to DFSMSrmm

TWS for z/OS then uses this API as required during restart and cleanup.

1.1.8 Integrating with JOB/SCAN

JOB/SCAN is a product from IBM Business Partner, Diversified Software Systems Incorporated. You can order JOB/SCAN through IBM using the IBM product number 5620-FIB. (IBM is a reseller of JOB/SCAN.)

JOB/SCAN provides a variety of facilities to ensure that production z/OS batch JCL is as error free as possible, the system is ready to execute batch JCL, and they are consistent with your company's standards. JOB/SCAN can start from a single job, a TWS for z/OS application, or the current and long term plans. From there JOB/SCAN sequences the jobs according to TWS for z/OS dependencies, resolves variables, expands PROCs, controls cards for key utilities, and simulates the execution of the JCL in order to form a complete picture of a collection of jobs.

The z/OS JCL parameter, **`TYPRUN=SCAN`**, is of limited value. This facility only checks the JCL for syntax errors, such as missing commas. More important than

just validating the JCL for syntax is the evaluation of the job in context of the current environment and the resolution of scheduler variables.

In addition to TWS for z/OS, JOB/SCAN includes the following interfaces:

- z/OS System Catalogs
- Volume Table of Contents (VTOCs)
- Removable Media Manager (RMM)
- Hierarchical Storage Manager (HSM)
- Storage Management Subsystem (SMS)
- Resource Access Control Facility (RACF®)
- Access Method Services (IDCAMS)
- Information Management System (IMS)
- DB2

JOB/SCAN is used while developing JCL, TWS for z/OS applications, and certain day-to-day TWS for z/OS processes. For example, it is considered a best practice to validate any major scheduling changes before they become part of the current plan.

Integration benefits

JOB/SCAN, on its own, has numerous features that benefit any z/OS data center. However, data centers using TWS are faced with unique situations that make JOB/SCAN more valuable and integrating TWS for z/OS and JOB/SCAN essential.

Structured JCL listing

After integration, you can create Structured JCL Listing for JOB/SCAN to see:

- Resolved TWS for z/OS variables and directives
- Overrides for DDs and where these overrides are applied
- Both PROCs and INCLUDE members
- Interleaving of JCL and potential errors
- Numerous other features to assist in reading JCL

JOB/SCAN requests the resolved JCL from TWS for z/OS and merges it with any referenced cataloged procedures. It shows the before and after effects of any JCL symbolic parameters.

Finally, any messages that are produced by the JOB/SCAN validation process are included both in context with the statement to which it applies and in a summary, together with all the messages produced (see “JOB/SCAN sample output” on page 683 for an example).

When JOB/SCAN is used to validate multiple jobs together, a single Structured JCL Listing contains output for every job that is scanned (see Example 1-1).

Thus, you could scan the JCL for an entire application and save the output as part of your quality control process.

Example 1-1 Sample JOB/SCAN output

```
*** ERROR SUMMARY AND COUNTS ***   JEM  6.2.4A    02/18/2005  10:27:06
    1  ADVISORY LEVEL
    0  WARNING LEVEL
    2  ERROR LEVEL
    3  TOTAL ISSUED        0 SUPPRESSED.
     THE HIGHEST SEVERITY CODE ENCOUNTERED WAS    08.
     THE  STDS  PGM  CALLED FOR THIS RUN WAS "TWSSTD1".
 LABEL   SV MSG.NO.    ERROR MESSAGE
-------- -- --------   ------------------------------------------------
.JAAA     8 DSS4420E - GENERATION DATA GROUP DOES NOT EXIST
.JAAA     0 DSS8900A - DSN = "FC.GLPROD.RMT2IN(+1) "
.JAAB     8 DSS8081E - MISSING  SYSPRINT  DD STATEMENT
//TWSRES6 JOB  (1),NOTIFY=&SYSUID
--TWSRES6 JOB  (1),NOTIFY=TWSRES5
//   EXEC PGM=IEBGENER
//SYSUT1   DD *
//SYSUT2   DD DSN=FC.GLPROD.RMT2IN(+1),UNIT=SYSDA,SPACE=(TRK,1),
//      DCB=FC.GLPROD.MODEL,DISP=(NEW,CATLG)
***ERROR  - DSS4420E - GENERATION DATA GROUP DOES NOT EXIST
*ADVISORY - DSS8900A - DSN = "FC.GLPROD.RMT2IN(+1) "
//SYSIN    DD DUMMY
***ERROR  - DSS8081E - MISSING  SYSPRINT  DD STATEMENT
```

See “JOB/SCAN sample output” on page 683 for an example report.

Dealing with complex TWS for z/OS JCL

JCL operations often contain TWS for z/OS information that is unknown until execution time. Occurrence variables, Conditional Variables, Variable Tables, Calendars, Input Arrival Dates, and so on, all affect the final, submitted JCL. This complexity makes it difficult to predict exactly what your JCL will look like and what it will do. Thus, it is critical that you employ every method possible to ensure a high level of success with your batch schedule. This drives the need to fully integrate TWS for z/OS and JOB/SCAN.

Some of the challenges of not having an integrated JCL validation solution include:

- Occurrence variables

  TWS for z/OS provides a great deal of information in the form of occurrence variables. These variables cover such things as Application Name, Input Arrival, Operation Number. For jobs that run in multiple applications, or several times within the same application, this information is often crucial to

make the job run differently within its TWS for z/OS context. This can affect data set names, member names, PARM statements, or even in JCL tailoring directives.

- JCL variable tables

  These variable tables contain the values for user defined TWS for z/OS variables that might be included in your JCL. The variable tables in use for a job can be influenced by the run cycle or period that is used to schedule the application, modifications to the plans or even directives in the JCL, which themselves can contain variables in their definition.

  It is not uncommon to see the following:

  ```
  //*%OPC TABLE NAME=&OADID
  ```

  This line means that the job will use a variable table of the same name as the application, making it impossible to predict the resolved JCL without knowing what application is scheduling the job.

- Conditional variables

  There is a special form of user variables — conditional variable. TWS for z/OS determines the values for these based on the value of other variables in the job. Most common jobs use conditional variables with occurrence variables to translate TWS for z/OS information into a more useful context.

  For example, you might have a job scheduled several times in the same application. It can be useful to know the first run of the occurrence and the last. You could have a variable dependent on `OOPNO` that is set to `FIRST` for operation `5`, `LAST` for operation `250`, and `OTHER` for operations `10` to `245`.

- Directives

  TWS for z/OS has many JCL tailoring directives which can alter dramatically the way it presents JCL to the internal reader. They can include or exclude lines of JCL. They might even fetch JCL from other sources. These directives appear to the internal reader as normal JCL comments. So, a JCL validation tool without integration would be unaware of their impact on the JCL.

- Calendars

  TWS for z/OS determines some variables by what calendar is in use when the job is scheduled. These could be calendar related variables, for example **`&OFREEDAY`**, or effect calculations that are based on Work or Free days using the **`SETVAR`** directive.

  Determining what calendar is in effect is not possible from the JCL alone. You must know the context of the job within TWS for z/OS to be able to determine the calendar in use, and its impact on the JCL.

- Dependencies

  Validating what the submitted JCL will look like is the first benefit of TWS for z/OS and JOB/SCAN integration. The second benefit is validating a JOB in its proper context. No job is an island. Each scheduled job relies on a variety of resources, such as data sets, programs, SMS rules, security rules, tape management systems, and so forth. Also, each job, in some way, relies on and influences other jobs. JOB/SCAN evaluates JCL and compares it to the current environment and the JOBs that are scheduled around it.

  To that end, JOB/SCAN processes multiple jobs as a unit to ensure the data sets are created before referenced or deleted before re-created. JOB/SCAN does this by internally tracking every change a job makes to catalogs, VTOCs, and tape volumes and compares that to the requirements of subsequent jobs. For the best possible results, JOB/SCAN must evaluate each job in dependency sequence.

  Fortunately, after integration, TWS for z/OS informs JOB/SCAN of the jobs that are scheduled and the dependencies for those jobs. From there, JOB/SCAN can sequence the jobs appropriately, with no further interaction.

When to use JOB/SCAN and TWS for z/OS integration

There are a number of points in your JCL life cycle where the integration of TWS for z/OS and JOB/SCAN are beneficial. The JCL life cycle covers all the phases JCL goes through on its path to a fully scheduled production job. By integrating JOB/SCAN and TWS for z/OS, you have a JCL development environment that covers everything from conception to production. For the purposes of this book, the integration is focused on getting test JCL into production status.

Production JCL normally begins by being written and then copied from somewhere else (such as test JCL or another job) and can include nonstandard methods or styles. This type of JCL normally does not take into account scheduling and the operational environment. The JCL can also have the wrong variables or none at all. In this situation, JOB/SCAN shows how TWS for z/OS resolves variables and notifies you of potential JCL errors directly from ISPF EDIT.

Note: Your company's JCL standards are also enforceable at this point in the JCL life cycle. Refer to *JOB/SCAN Standards User Guide* (which ships with JOB/SCAN) for more information.

JOB/SCAN shows how TWS for z/OS resolves variables and notifies you of potential JCL errors directly from ISPF EDIT. This function is achieved using the JOB/SCAN edit macro: **JTWS**. You can invoke JTWS from ISPF EDIT or directly from TWS for z/OS using the JOB/SCAN for TWS JCL edit panel. For information about how to use the JOB/SCAN edit macro with TWS for z/OS to help bring primitive JCL to full operational maturity, see 7.2.1, “Preparing and validating JCL within the ISPF editor” on page 293.

Another benefit is the creation or modification of an application. Here, you invoke JOB/SCAN from TWS for z/OS while editing the application database. In this case, JOB/SCAN scans the application's operations in dependency sequence. The output is a report that shows all the JCL (including PROCs and key PARMLIB members) for the complete application, along with any potential error conditions.

This integration can also provide validation for complete or partial production schedules. It is not uncommon for JOB/SCAN users to scan 10 000 or more jobs as a single unit to identify potential dependency problems. Finding even one production problem before it happens can save untold hours of frustration.

Finally, there are times in any TWS for z/OS data center when special circumstances arise and you must change and re-submit JCL. If this type of instance occurs, timing is critical, and hand editing of production JCL is required. At this point, it is essential to re-validate the JCL before it is submitted.

Integration with TWS for z/OS

Figure 1-5 shows the architecture of integration between TWS for z/OS and JOB/SCAN.

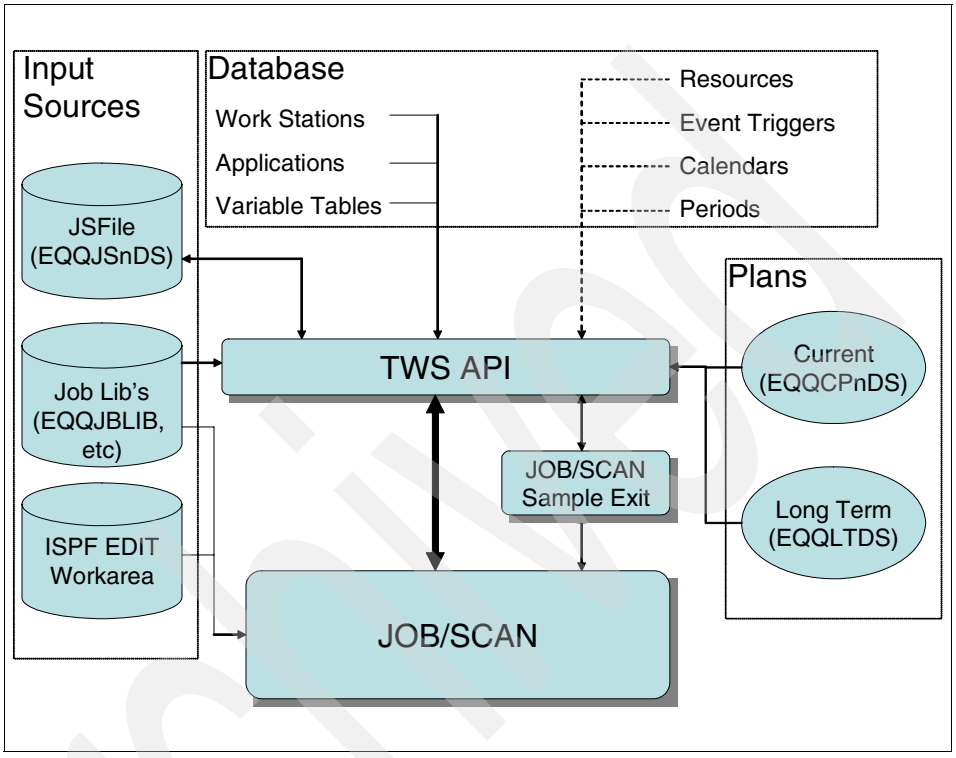

Figure 1-5 Architectural overview of JOB/SCAN integration

There are a number of ways that JOB/SCAN interacts with TWS for z/OS. In all cases, JOB/SCAN uses the standard TWS for z/OS API to ensure that the resolved JCL is the same as the JCL when TWS for z/OS eventually submits the job.

For example, when a user is editing JCL, JOB/SCAN uses the API to resolve TWS for z/OS variables that are embedded in the JCL. When validating one or more applications, JOB/SCAN uses the API to obtain a list of jobs and their dependencies.

From the TWS for z/OS database, JOB/SCAN makes use of workstation, application, and variable table definitions. For your company, there might be occasions where it is useful to further integrate the two products. (For an example of this type of integration, see 7.6, “Tuning JOB/SCAN for special situations” on page 319.) For input, JOB/SCAN reads JCL from the JS File or any

of the JOB Libraries using the TWS for z/OS API. At times, it uses the ISPF EDIT WORKAREA as a source for JCL.

JOB/SCAN does not directly access the current and long term plan data sets. These data sets are accessed by the TWS for z/OS API when servicing requests from JOB/SCAN.

How TWS for z/OS and JOB/SCAN cooperate to read JCL

JOB/SCAN relies on TWS for z/OS to retrieve the JCL because TWS for z/OS uses a variety of sources (EQQJSnDS, EQQJBLIB, or the Job-Library-Read Exit). When TWS for z/OS attempts to retrieve JCL for submission, it first looks in the EQQJSnDS data set, often referred to as the JS file. If there is no JCL in the JS file, by default, TWS for z/OS looks in the EQQJBLIB data set, which is either a single library or a concatenation of many libraries.

In some instances, you might use the Job-Library-Read Exit (EQQUX002) to allow TWS for z/OS to use alternate JCL sources to the EQQJBLIB, which could be alternate libraries or even VSAM files. Typically, this exit is used either for performance or security reasons and keeps JCL for different services in separate concatenations, which can be selected conditionally by the exit. This case provides secure separation of services and reduces seek times for large concatenations.

Figure 1-6 illustrates the two paths that are used for JCL retrieval.

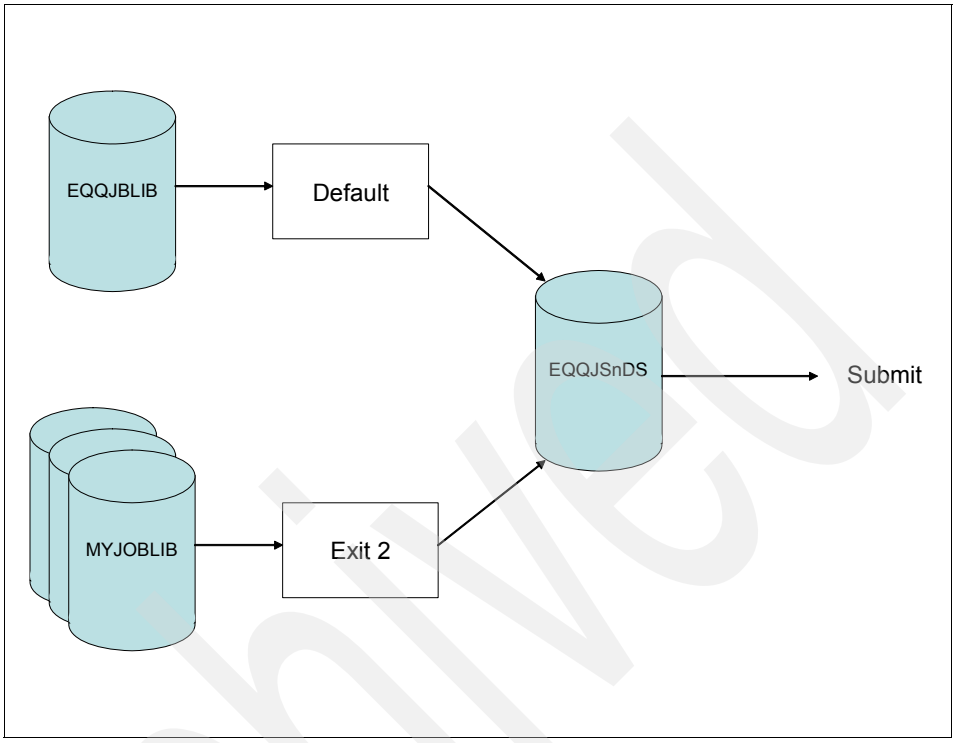

Figure 1-6 JCL source for TWS for z/OS

To choose the correct location for your JCL and to correctly integrate with JOB/SCAN, you must understand how your job retrieval process is configured. For more information about this type of exit, see *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning*, SC32-1285.

JOB/SCAN automatically retrieves JCL for validation from TWS for z/OS. So, regardless of the use of exits, the JCL that is scanned is the same as the JCL that is submitted. Use of the Job-Library-Read exit only impacts the use of the JCL Edit Tool, because the TWS for z/OS API does not expose the ability to find the JCL directly at its source. If you use this Job-Library-Read exit, then ensure that you have taken this into account with the customization of **`JOOCTWS`** (see 7.1.5, “Configuring JOB/SCAN to match your environment” on page 290). See also *IBM Tivoli Workload Scheduler for z/OS - Customization and Tuning*, SC32-1285, for more information about the exit.

JCL exists in one of two forms:

- Unresolved JCL

  In EQQJBLIB, or wherever your source JCL is kept, the JCL is in a completely unresolved format. Thus, TWS for z/OS JCL variables are not resolved, and any JCL Tailoring Directives have not been processed.

  Validating unresolved JCL is likely to show errors that are not representative of the actual operation. However, JCL is written and maintained in unresolved form.

- Resolved JCL

  In normal use, TWS for z/OS retrieves the JCL only at submission time (Figure 1-6 on page 22). It resolves the variables and directives and stores the results in the JS file before submitting the job, which is usually too late to avoid problems. JOB/SCAN follows this same process to retrieve JCL for accurate, preemptive validation.

  Whether TWS for z/OS resolves JCL depends on the setting of the **OPCOPTS VARSUB** statement in the following TWS for z/OS parameters:

  YES Always resolve the JCL

  NO Never resolve the JCL

  SCAN Only resolve JCL when an **//*%OPC SCAN** card has been detected in the JCL (default)

  Note: TWS for z/OS supplies some variables, which contain current date and time, that you can use in your JCL (for example, CDDMMYY). Though these variables would ordinarily show the current date and time, this could look strange when validating JCL for some time in the future. For simplicity, JOB/SCAN sets these variables to use the Input Arrival as their source.

  In some cases, you can precede a submission operation with a setup operation. Certain variables and directives can be identified to be resolved at the SETUP stage. In the case of certain variables, the operator is prompted to provide values after setup variables and directives have been resolved. Typically, variables for which you can be prompted include variables such as tape volume serial numbers or processing parameters for ad hoc or recovery applications. The partially resolved JCL is stored in the JS file at the end of the setup operation and then retrieved from there to be resolved fully at submission.

  The resolved JCL has all the known TWS for z/OS variables substituted into the JCL and all JCL tailoring directives processed. The processes JCL directives are marked as processes by changing the **//*%OPC** prefix to

//*>OPC, preventing TWS for z/OS from reprocessing them on re-submission of the job.

If a variable is not found anywhere within TWS for z/OS variable tables or the supplied list of TWS for z/OS system variables, then one of two things happens, depending of the setting of the **OPCOPTS VARFAIL** statement in your TWS for z/OS parameters. If you have included the prefix (&, % or ?) for the "missing" variable in the **VARFAIL** statement, then the variable is left as is, unresolved in the output JCL. If the prefix was not included, then submission fails with an error code of **OJCV**.

Tips: If you include in-stream procedures in your executing JCL, then by default, TWS for z/OS does not resolve JCL between the PROC and PEND statement to prevent conflicts between TWS for z/OS variables and JCL symbolic parameters. You can enable TWS for z/OS to resolve JCL between the PROC and PEND statements by specifying **OPCOPTS VARPROC(YES)** in your TWS for z/OS parameters.

To ensure that you understand how variables are being resolved by TWS for z/OS, examine your settings on the **OPCOPTS** statement in your TWS for z/OS configuration. You can find details about the settings that influence variables in *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning*, SC32-1265.

For more information and real-life scenarios that use this integration, refer to Chapter 7, “Integrating JOB/SCAN” on page 283.

Integration matrix

A potential integration point between JOB/SCAN and other Tivoli products for end-to-end TWS environments include Tivoli NetView for z/OS.

1.2 IBM TWS solution scenario

In our test lab, we created a small network of systems running UNIX, Linux®, and Windows where various Tivoli products were installed. Figure 1-7 shows which products were installed on each system.

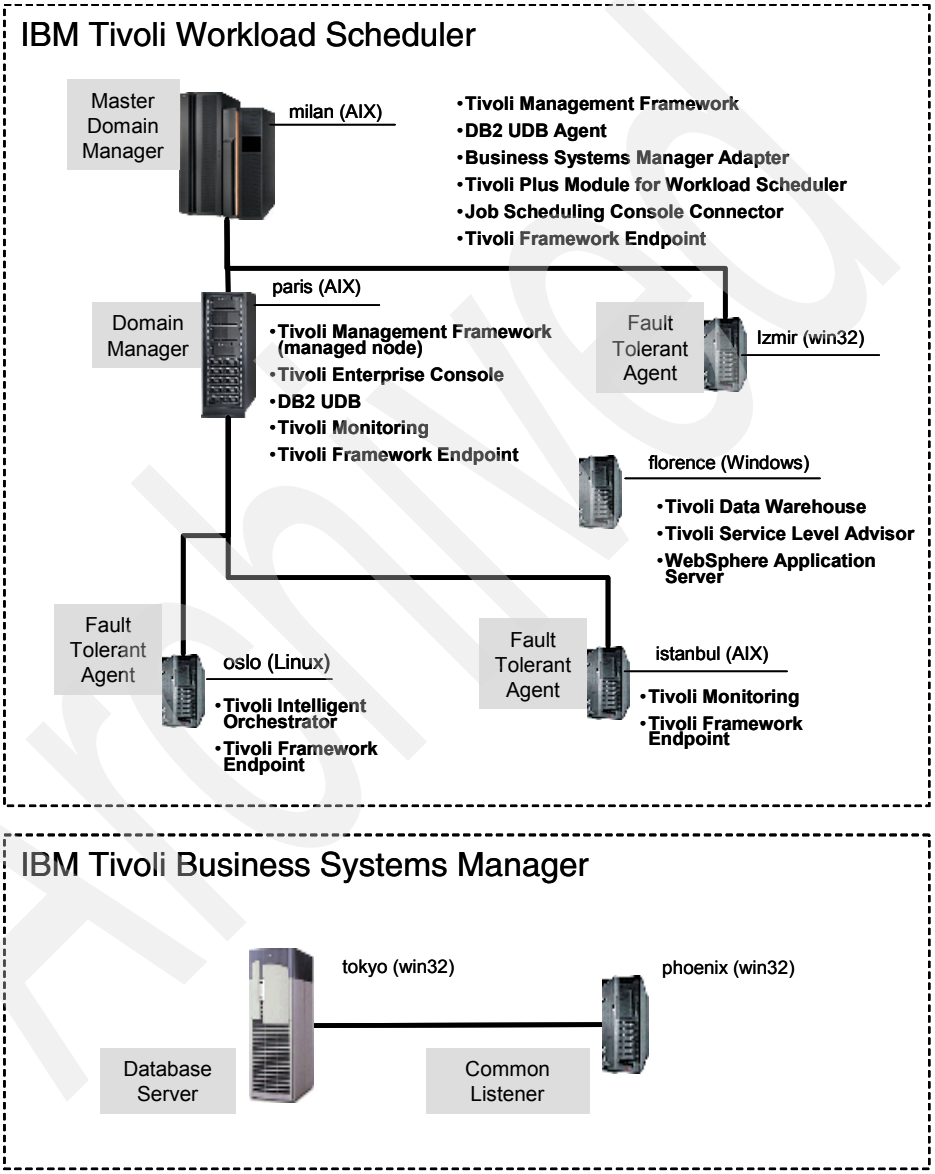

Figure 1-7 The TWS network

Note: The Windows system that housed the Tivoli Data Warehouse and Tivoli Service Level Advisor was not actually connected to the TWS network.

This section discusses the following distributed products:

- Integrating with Tivoli Service Level Advisor
- Integrating with Tivoli Intelligent Orchestrator
- Integrating with Tivoli Data Warehouse
- Integrating with Tivoli Business Systems Manager
- Integrating with Tivoli Storage Manager
- Integrating with Tivoli NetView
- Integrating with Tivoli Enterprise Console
- Integrating with Tivoli Monitoring
- Integrating with Tivoli Configuration Manager
- Integrating with IBM DB2 Content Manager OnDemand

1.2.1 Integrating with Tivoli Service Level Advisor

Tivoli Service Level Advisor is designed to provide predictive service level management capabilities by enabling you to proactively predict when SLA violations are likely to occur and then take corrective actions to avoid an SLA violation. The integration of these capabilities with the real-time availability management process delivered in Tivoli Business Systems Manager produces a service delivery solution that allows executives to manage IT based on business priorities.

Integration benefits

Using Tivoli Service Level Advisor with TWS identifies batch scheduling service delivery problems before they occur, allowing you to take action to maintain service levels rather than simply report them. This maintains batch customer productivity and satisfaction with the batch services they depend on to meet business objectives. The integration identifies problem areas with the batch schedule and operations, presenting executive summaries and detailed operations status of service level agreements. Using these summaries, executives can see the impact of service levels to the business and preemptively notify batch customers.

Integration with TWS

Tivoli Service Level Advisor uses job status data sent to Tivoli Data Warehouse by TWS to determine service levels and report on them. Figure 1-8 shows the relationship between these architectural elements.

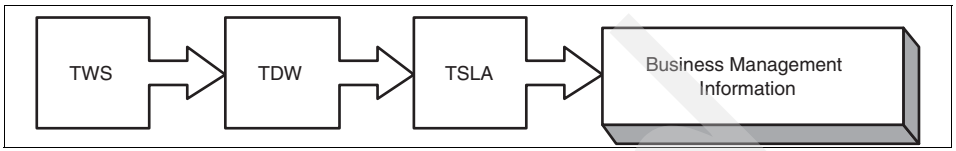

Figure 1-8 Architectural overview of Tivoli Service Level Advisor integration

For more information and real-life scenarios that use this integration, refer to Chapter 12, “Integrating Tivoli Data Warehouse and Tivoli Service Level Advisor” on page 585.

Integration matrix

Potential integration points between Tivoli Service Level Advisor and other Tivoli products for distributed TWS environments include:

- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Business Systems Manager
- Tivoli Storage Manager
- Tivoli NetView
- Tivoli Enterprise Console®
- Tivoli Monitoring
- Tivoli Configuration Manager

1.2.2 Integrating with Tivoli Intelligent Orchestrator

Tivoli Intelligent Orchestrator improves the use of existing hardware, software, storage and network devices without rewiring or changing the network architecture. It delivers this improvement by automating repetitive, manual tasks performed by system, network, and storage administrators, thereby saving time and money. It boosts administrator productivity by automatically triggering the execution of the steps necessary to provision, configure, and deploy a complete solution into productive use.

Orchestration triggers automatically the execution of the steps that are necessary to provision, configure, and deploy a complete solution into productive use. It senses the increase or decrease in IT resource demand and takes action to re-allocate resources accordingly throughout the entire system. It also allows multiple applications to be run efficiently according to business priorities on a common, dynamic, intelligently managed IT infrastructure. This automatic

reallocation of resources enables it to anticipate, plan and dynamically provide server capacity to meet peak business needs on demand.

Integration benefits

TWS can give Tivoli Intelligent Orchestrator advance reservation notice for resources needed for efficient processing of batch workloads. Tivoli Intelligent Orchestrator can make appropriate decisions to make available appropriate resources to give continuous service to scheduled batch workloads.

Integration with TWS

For more information and real-life scenarios that use this integration, refer to Chapter 11, “Integrating Tivoli Intelligent Orchestrator” on page 579. Full integration with Tivoli Intelligent Orchestrator falls outside the scope of this book.

Integration matrix

Potential integration points between Tivoli Intelligent Orchestrator and other Tivoli products for distributed TWS environments include:

- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Business Systems Manager
- Tivoli Storage Manager
- Tivoli NetView
- Tivoli Enterprise Console
- Tivoli Monitoring
- Tivoli Configuration Manager

1.2.3 Integrating with Tivoli Data Warehouse

Tivoli Data Warehouse is an embedded technology that provides the backbone repository for all historical systems management data and the basis for all Tivoli reporting solutions.

Integration benefits

Using Tivoli Data Warehouse with TWS aggregates and correlates historical job status. This helps identify batch service trends, predict new business demands for batch services and support line of business planning. The reports provide the batch scheduling team an efficient vehicle to communicate their value-add to their internal customers.

Integration with TWS

Job history in archived production plans from TWS are extracted into flat files and uploaded into the Tivoli Data Warehouse server’s DB2 Data Warehouse

from the TWS Master Domain Manager. An IBM DB2 client is used for communicating with the IBM DB2 Data Warehouse application server.

Figure 1-8 on page 27 shows the high level architectural relationship between these products.

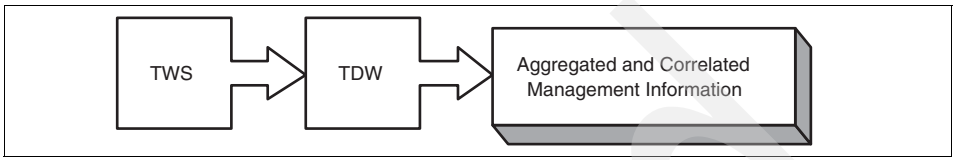

Figure 1-9 Architectural overview of Tivoli Data Warehouse integration

For more information and real-life scenarios that use this integration, refer to Chapter 12, “Integrating Tivoli Data Warehouse and Tivoli Service Level Advisor” on page 585.

Integration matrix

Potential integration points between Tivoli Data Warehouse and other Tivoli products for distributed TWS environments include:

- Tivoli Service Level Advisor
- Tivoli Intelligent Orchestrator
- Tivoli Business Systems Manager
- Tivoli Storage Manager
- Tivoli NetView
- Tivoli Enterprise Console
- Tivoli Monitoring
- Tivoli Configuration Manager

1.2.4 Integrating with Tivoli Business Systems Manager

Tivoli Business Systems Manager provides intelligent management software to help businesses increase operational agility by aligning IT operations to business priorities. The executive dashboard visualizes the health of the most critical business services and any associated service level agreements.

Integration benefits

Using Tivoli Business Systems Manager with TWS helps optimize batch services with business goals to the organization, rather than focusing on the batch scheduling technology itself. These goals might be related to streamlining business processes and optimizing resources to help manage costs, increase efficiency to manage productivity and increase revenue, and assure service availability to enhance customer satisfaction.

Tivoli Business Systems Manager also enables the batch scheduling team to become responsive to their customers rather than technology. It delivers real-time knowledge so that scheduling resource decisions can be made to reduce the high cost of application downtime and ultimately drive the greatest impact to the business.

Integration with TWS

TWS provides direct, out-of-the-box integration support for Tivoli Business Systems Manager. Figure 1-10 shows the high-level architectural relationship between these products.

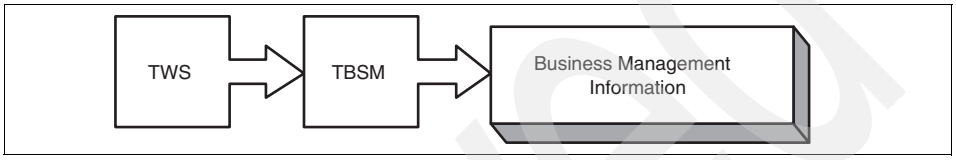

Figure 1-10 Architectural overview of TBSM integration

For more information and real-life scenarios that use this integration, refer to Chapter 10, “Integrating Tivoli Business Systems Manager” on page 553.

Integration matrix

Potential integration points between Tivoli Business Systems Manager (TBSM) and other Tivoli products for distributed TWS environments include:

- Tivoli Service Level Advisor
- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Storage Manager
- Tivoli NetView
- Tivoli Enterprise Console
- Tivoli Monitoring
- Tivoli Configuration Manager

1.2.5 Integrating with Tivoli Storage Manager

Tivoli Storage Manager provides centralized, automated data protection that helps reduce the risks that are associated with data loss while also helping to reduce complexity, manage costs and address compliance with regulatory data retention requirements. It stores backup, archive, space management, and bare-metal restore data, as well as compliance and disaster-recovery data in a hierarchy of offline storage.

Important: The IBM Redbook *Implementing TWS Extended agent for Tivoli Storage Manager*, SG24-6030, discusses Tivoli Storage Manager and TWS integration. This book was written for TWS Version 7.0 and Tivoli Storage Manager Version 4.1. A new book (*Implementing TSM Extended Agent for Tivoli Workload Scheduler 8.2*, SG24-6696) that discusses this integration with the latest versions of both products is currently under development.

Integration benefits

Using Tivoli Storage Manager with TWS lets backups work around applications that use batch scheduling. For example, backups and restores can be scheduled around when a business process is finished with a database. Backup windows are then determined by the business instead of the technical priorities.

Integration with TWS

TWS invokes Tivoli Storage Manager commands through the TWS Extended Agent interface. A custom Extended Agent method is used to interact with Tivoli Storage Manager.

Figure 1-11 shows the high-level architectural relationship between these products.

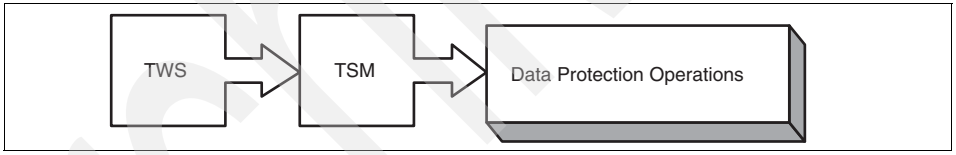

Figure 1-11 Architectural overview of Tivoli Storage Manager integration

Integration matrix

Potential integration points between Tivoli Storage Manager and other Tivoli products for distributed TWS environments include:

- Tivoli Service Level Advisor
- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Business Systems Manager
- Tivoli NetView
- Tivoli Enterprise Console
- Tivoli Monitoring
- Tivoli Configuration Manager

1.2.6 Integrating with Tivoli NetView

Tivoli NetView discovers TCP/IP networks, displays network topologies, correlates and manages events and SNMP traps, monitors network health, and gathers performance data. It measures availability and provides fault isolation for problem control and management, maintains device inventory for asset management, and reports on network trends for further analysis.

Integration benefits

Using Tivoli NetView with TWS lets network managers monitor and diagnose TWS networks. The scheduler network can be viewed topographically, the status of job scheduling activity can be determined, and critical TWS processes on each workstation are monitored. Menu actions are provided to start and stop scheduler processing, and to run **conman** on any workstation in the network.

Integration with TWS

TWS provides direct, out of the box support for integration with Tivoli NetView. SNMP traps are sent from TWS to Tivoli NetView for asynchronous job scheduling events such as terminated jobs and stuck job streams.

Manager and agent software for Tivoli NetView poll for scheduler processing status and send notification via SNMP traps when necessary. A process called **muser** runs TWS commands issued by NetView users. Figure 1-12 shows the high-level architectural relationship between these products.

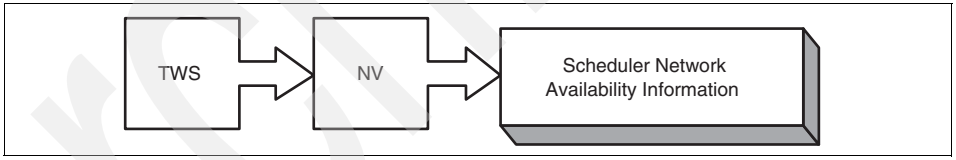

Figure 1-12 Architectural overview of Tivoli NetView integration with TWS

For more information and real-life scenarios that use this integration, refer to Chapter 13, “Integrating Tivoli NetView Distributed” on page 615.

Integration matrix

Potential integration points between Tivoli NetView and other Tivoli products for distributed TWS environments include:

- Tivoli Service Level Advisor
- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Business Systems Manager
- Tivoli Storage Manager
- Tivoli Enterprise Console
- Tivoli Monitoring
- Tivoli Configuration Manager

1.2.7 Integrating with Tivoli Enterprise Console

Tivoli Enterprise Console provides sophisticated, automated problem diagnosis and resolution to improve system performance and reduce support costs. It uses pre-configured rules providing best-practices event management out-of-the-box.

Integration benefits

Using Tivoli Enterprise Console with TWS enables root cause analysis on scheduling events. For example, if a job is still running after an estimated duration, an event is displayed by Tivoli Enterprise Console.

Integration with TWS

TWS is integrated with Tivoli Enterprise Console through the TWS Plus Module. This product extends the software on both TWS and Tivoli Enterprise Console. Scheduling events on TWS are forwarded via an Tivoli Enterprise Console logfile adapter to Tivoli Enterprise Console.

Figure 1-13 shows the high-level architectural relationship between these products.

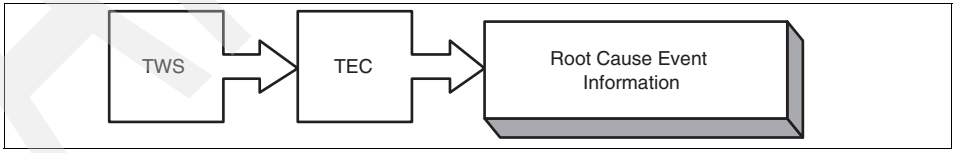

Figure 1-13 Architectural overview of Tivoli Enterprise Console integration with TWS

For more information and real-life scenarios that use this integration, refer to Chapter 8, “Integrating Tivoli Enterprise Console” on page 325.

Integration matrix

Potential integration points between Tivoli Enterprise Console and other Tivoli products for distributed TWS environments include:

- Tivoli Service Level Advisor
- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Business Systems Manager
- Tivoli Storage Manager
- Tivoli NetView
- Tivoli Monitoring
- Tivoli Configuration Manager

1.2.8 Integrating with Tivoli Monitoring

Tivoli Monitoring provides monitoring for essential hardware and software system resources, to detect bottlenecks and potential problems, and to recover from critical situations automatically.

Integration benefits

Using Tivoli Monitoring with TWS adds monitoring of critical infrastructure hardware and software that jobs as well as TWS itself depend upon. Job execution can be put off if Tivoli Monitoring detects that infrastructure resources are unavailable, thus avoiding job termination that can require costly manual intervention. For example, Tivoli Monitoring can detect if a database is almost full or too busy to accept batch transactions, then run a command so that jobs that use the database have their start times pushed back by 10 minutes.

Integration with TWS

Tivoli Monitoring uses software called resource models to monitor critical hardware and software. These can be configured to monitor such resources as disk, processor, database performance, and so forth. Typically Tivoli Monitoring watches over TWS, but does not directly receive data from TWS. In many configurations Tivoli Monitoring sends event information to Tivoli Enterprise Console.

When a problem is detected, Tivoli Monitoring can either interact directly with TWS or through Tivoli Enterprise Console. Figure 1-14 on page 35 shows the high-level architectural relationship between these products.

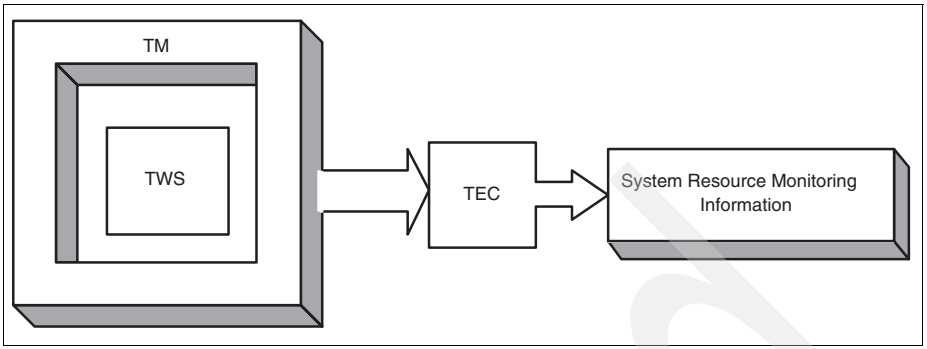

Figure 1-14 Architectural overview of Tivoli Monitoring integration with TWS

For more information and real-life scenarios that use this integration, refer to Chapter 9, “Integrating Tivoli Monitoring” on page 395.

Integration matrix

Potential integration points between Tivoli Monitoring and other Tivoli products for distributed TWS environments include:

- Tivoli Service Level Advisor
- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Business Systems Manager
- Tivoli Storage Manager
- Tivoli NetView
- Tivoli Enterprise Console
- Tivoli Configuration Manager

1.2.9 Integrating with Tivoli Configuration Manager

Tivoli Configuration Manager provides the ability to capture your best practices for software distribution, automate those best practices and enforce corporate standards. The software distribution module enables you to rapidly and efficiently deploy complex mission-critical applications to multiple locations from a central point. The inventory module lets you automatically scan for and collect hardware and software configuration information from computer systems across your enterprise. This inventory configuration data can be used in the reference models to automatically remediate systems that are not compliant.

Integration benefits

Using Tivoli Configuration Manager with TWS, administrators can distribute, inventory and keep up-to-date TWS Fault Tolerant Agent (FTA) and Extended

Agent (XA) software on all TWS workstations in a scheduler network. The programs, scripts, and data files that are used by TWS jobs can also be distributed, inventoried, and updated.

Integration with TWS

TWS FTA and XA software is packaged in a format that is directly compatible with Tivoli Configuration Manager. So, maintaining this software follows usual Tivoli Configuration Manager processes. Maintaining the programs, scripts, and data files that are used by jobs using Tivoli Configuration Manager is performed by adding these files to Tivoli Configuration Manager software packages, inventory profiles, and reference models. Figure 1-15 shows the high-level architectural relationship between these products.

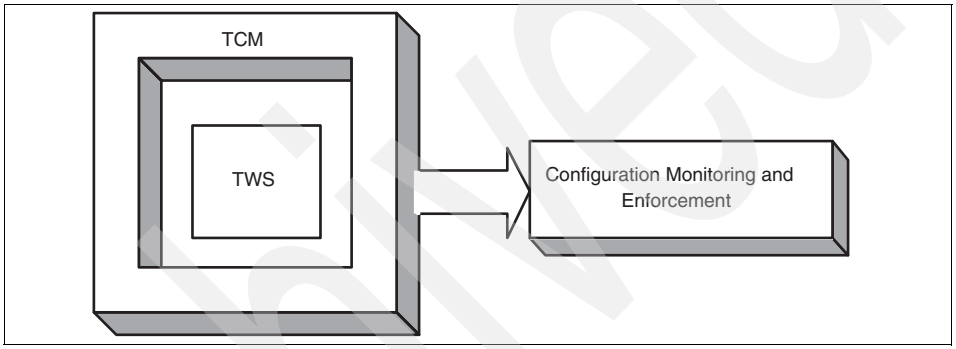

Figure 1-15 Architectural overview of Tivoli Configuration Manager integration with TWS

Integration matrix

Potential integration points between Tivoli Configuration Manager and other Tivoli products for distributed TWS environments include:

- Tivoli Service Level Advisor
- Tivoli Intelligent Orchestrator
- Tivoli Data Warehouse
- Tivoli Business Systems Manager
- Tivoli Storage Manager
- Tivoli NetView
- Tivoli Enterprise Console
- Tivoli Monitoring

1.2.10 Integrating with IBM DB2 Content Manager OnDemand

IBM DB2 Content Manager OnDemand is an archival and retrieval system for documents, log files, and reports. It provides online access to hard copy and is an excellent tool for microfiche replacement, providing instant access to information.

Although IBM DB2 Content Manager OnDemand is not a Tivoli branded product, this section briefly discusses the integration of DB2 Content Manager OnDemand and TWS. This integration has the promise of creating great synergy for your environment. Specifically, this solution can provide immediate benefit by integrating the job logs into an online, electronic information archive and retrieval system, which you can use for quick search and problem resolution purposes.

This integration solution (for both TWS Distributed and z/OS) is covered in detail in the IBM Redbook *IBM Tivoli Workload Scheduler and Content Manager OnDemand to Provide Centralized Job Log Processing*, SG24-6629. The book also includes scripts that make up this integration.

Part 2

TWS for z/OS integrations

This part of the book discusses the product integrations with Tivoli Workload Scheduler for z/OS (TWS for z/OS) and includes the following chapters:

- Chapter 2, “Integrating Tivoli Infoman” on page 41
- Chapter 3, “Integrating Tivoli NetView for z/OS” on page 79
- Chapter 4, “Integrating with OPC Automation extension of System Automation” on page 97
- Chapter 5, “Integrating Tivoli Business Systems Manager” on page 183
- Chapter 6, “Integrating Tivoli Decision Support for OS/390” on page 209
- Chapter 7, “Integrating JOB/SCAN” on page 283

2

Integrating Tivoli Infoman

This chapter describes the scenario that we used to demonstrate the integration of Tivoli Workload Scheduler for z/OS (TWS for z/OS) with Tivoli Information/Management (Infoman). This integration allows TWS for z/OS to raise problem records in Infoman when a batch job under TWS for z/OS control fails. We set up this scenario on a single z/OS system. This system ran both the TWS for z/OS controller and Infoman and was part of a Parallel Sysplex. We used the samples that are provided in the TWS for z/OS SEQQSAMP library to activate the interface.

This chapter includes detailed information about the configuration of the text environment, how TWS for z/OS created problem records in Infoman, and how we implemented the interface.

The scenario presented in this chapter is not the only way to create problem records in Infoman automatically. There are other scenarios which create problem records in Infoman. However, these scenarios require other products.

2.1 The TWS for z/OS and Infoman configuration

The TWS for z/OS and Infoman configuration that we used in our scenario runs on a single z/OS system called SC64. SC64 was part of a 4-way Parallel Sysplex. The TWS for z/OS controller and all trackers were connected using XCF. We did not exploit the Parallel Sysplex to share the Infoman databases. In our configuration, it was possible to run Infoman on all systems in the Parallel Sysplex and to share the Infoman database using VSAM Record Level Sharing (RLS). We used a batch job to create problem records in Infoman.

The advantage of running an instance of Infoman on every system in the sysplex is that this batch job could run on any system in the Parallel Sysplex and create problem records. Because we were running Infoman only on one system in our Parallel Sysplex, we had to run the batch job on this same system.

The diagram in Figure 2-1 illustrates this configuration.

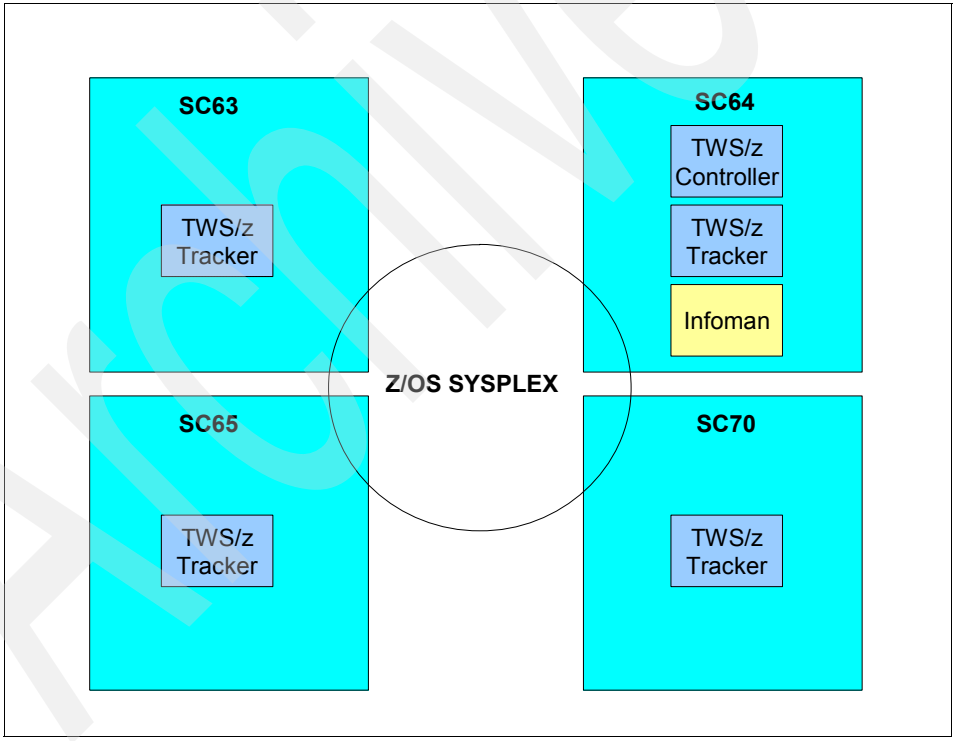

Figure 2-1 TWS for z/OS and Infoman configuration in the z/OS Parallel Sysplex

2.2 Creating problem records in Infoman

Each time a job fails, TWS for z/OS changes the status of the job to E (for error). This status change is the trigger for TWS for z/OS to create a problem record in Infoman. There are three different types of processing that occur to create the Infoman problem record, in the following order:

1. The Operation-Status-Change TWS for z/OS exit (EQQUX007)
2. Batch jobs which execute a CLIST and Infoman in batch
3. Infoman Stored Response Chain (SRC)

Some processing occurs within TWS for z/OS, some in batch, and then the final piece in Infoman. TWS for z/OS provides samples for everything except the Infoman SRC.

Figure 2-2 shows how the problem records are created in this scenario. This scenario uses the samples that are provided by TWS for z/OS.

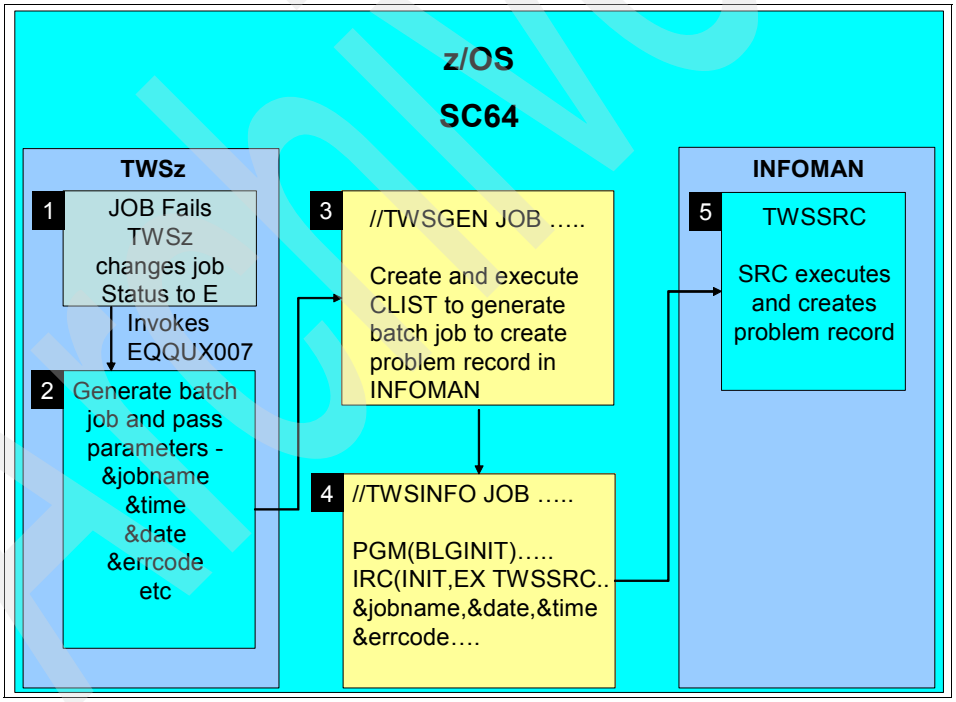

Figure 2-2 Step-by-step creation of problem record in Infoman

The figure shows the TWS for z/OS and Infoman address spaces in the z/OS system called SC64. Between them are the two batch jobs which execute as part of the problem record creation process.

The steps are as follows:

1. A TWS for z/OS controlled job fails and has its status changed to E. This status change triggers TWS for z/OS to call the Operation-Status-Change exit 7 (EQQUX007).
2. TWS for z/OS passes the following data to EQQUX007:

   ```
   New status
   Old status
   Operation number
   Caller task name
   Error code
   Workstation name
   Application name
   Application owner name
   Resource group name
   Extended status
   ```

   TWS for z/OS also passes data areas to EQQUX007:

   – Job-related data area, which contains the following:

   ```
   Job name
   Job number
   Current date
   Job start time
   Job end time
   Step name
   System abend code
   User abend code
   Origin Network Job Entry (NJE) node
   Proc step name
   ```

   – Operation-related data area, which contains the following:

   ```
   Operation description
   Application input arrival
   Operation input arrival
   Planned start time
   Planned end time
   Operation deadline
   User data field
   Restart information data area
   Extended job name
   ```

Note: These parameters and the use of EQQUX007 are documented in the *IBM Tivoli Workload Scheduler for z/OS V 8.2 Customization & Tuning*, SC32-1265.

Although the exit has access to all this data, the sample EQQUX007 does not use them all. To use more than the sample, you need to change the assembler source.

Example 2-1 shows the section of the assembler source where the variables are set and then inserted into the CLIST.

Example 2-1 Extract of EQQX7ASM source code where variables are created

```
*--------------------------------------------------------------------*
* INSERT ADDITIONAL RECORDS IN OUTPUT FILE                           *
*--------------------------------------------------------------------*
INSERT   DS    0H
         LA    R2,FIRSTREC         * POINT TO FIRST INSERT RECORD
NEXT     DS    0H
         C     R2,=A(LASTREC)      * CHECK IF FINISHED
         BC    GE,AGAIN            * BR IF NO MORE INSERT RECORDS
         LA    R1,NEXT1            * GET BRANCH ADDR
         AL    R1,=X'80000000'     * REQUEST 31 BIT MODE
         BSM   0,R1                * SWITCH TO AMODE 31
NEXT1    DS    0H
         MVC   RECORD,0(R2)        * COPY CURRENT RECORD
         CLC   =CL8'ERRCODE',RECORD+5 * IS IT ERROR CODE ?
         BC    NE,NEXT010          * BR IF NOT
         L     R1,PERRCODE         * GET ERROR CODE ADDRESS
         MVC   RECORD+21(4),0(R1)  * COPY ERROR CODE TO RECORD
         B     PUTIT               * WRITE THIS RECORD
```

The sample EQQUX007 reads skeleton JCL from the UX07IN DD statement. This DD statement needs to be placed in the TWS for z/OS controller procedure, as shown in Example 2-2 on page 46.

Example 2-2 TWS for z/OS controller procedure

```
//TWSC       EXEC PGM=EQQMAJOR,REGION=0M,PARM='TWSC',TIME=1440
//*********************************************************************
//* THIS IS A SAMPLE STARTED TASK PROCEDURE FOR AN OPC CONTROLLER ONLY
//* IT SHOULD BE REVIEWED AND MODIFIED AS REQUIRED
//* TO SUIT THE NEEDS OF THE INSTALLATION.
//*********************************************************************
//STEPLIB   DD  DISP=SHR,DSN=TWS.INST.LOADLIB
//EQQMLIB   DD  DISP=SHR,DSN=EQQ.SEQQMSG0
//EQQMLOG   DD  SYSOUT=*
//EQQPARM   DD  DISP=SHR,DSN=TWS.INST.PARM
//SYSMDUMP  DD  SYSOUT=*
//EQQDUMP   DD  SYSOUT=*
//EQQBRDS   DD  SYSOUT=(A,INTRDR)
//EQQEVDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.EV
//EQQCKPT   DD  DISP=SHR,DSN=TWS.INST.TWSC.CKPT
//EQQWSDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.WS
//EQQADDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.AD
//EQQRDDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.RD
//EQQSIDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.SI
//EQQLTDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.LT
//EQQJS1DS  DD  DISP=SHR,DSN=TWS.INST.TWSC.JS1
//EQQJS2DS  DD  DISP=SHR,DSN=TWS.INST.TWSC.JS2
//EQQOIDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.OI
//EQQCP1DS  DD  DISP=SHR,DSN=TWS.INST.TWSC.CP1
//EQQCP2DS  DD  DISP=SHR,DSN=TWS.INST.TWSC.CP2
//EQQNCPDS  DD  DISP=SHR,DSN=TWS.INST.TWSC.NCP
//EQQCXDS   DD  DISP=SHR,DSN=TWS.INST.TWSC.CX
//EQQJTARC  DD  DISP=SHR,DSN=TWS.INST.TWSC.JTARC
//EQQJTO1   DD  DISP=SHR,DSN=TWS.INST.TWSC.JT1
//EQQJTO2   DD  DISP=SHR,DSN=TWS.INST.TWSC.JT2
//EQQJTO3   DD  DISP=SHR,DSN=TWS.INST.TWSC.JT3
//EQQJTO4   DD  DISP=SHR,DSN=TWS.INST.TWSC.JT4
//EQQJTO5   DD  DISP=SHR,DSN=TWS.INST.TWSC.JT5
//EQQJCLIB  DD  DISP=SHR,DSN=TWS.INST.JCLIB
//EQQINCWK  DD  DISP=SHR,DSN=TWS.INST.INCWORK
//EQQSTC    DD  DISP=SHR,DSN=TWS.INST.STC
//EQQJBLIB  DD  DISP=SHR,DSN=TWS.INST.JOBLIB
//EQQPRLIB  DD  DISP=SHR,DSN=TWS.INST.JOBLIB
//*
//*   DATASETS FOR EQQUX007
//UX07IN    DD  DISP=SHR,DSN=TWS.INST.EQQUX007.INPUT.JCL
//UX07OUT   DD  SYSOUT=(A,INTRDR)
```

The skeleton JCL is also provided by the product in the EQQSAMP member EQQX7JOB. EQQUX007 searches the skeleton JCL for the string AAAA. This string is in the CLIST section of the skeleton JCL. At this point, EQQUX007 inserts CLIST variable definitions for the some of the parameters that it has been passed. Example 2-3 shows the statements inserted in the CLIST.

Example 2-3 CLIST variables placed in generated batch job by EQQUX007

```
SET ERRCODE  = &STR(S806)
SET APPLNAME = &STR(U9APPL1         )
SET NEWSTAT  = &STR(E)
SET OLDSTAT  = &STR(S)
SET CALLTASK = &STR(EM  )
SET OPERNUM  = &STR(  24)
SET WSNAME   = &STR(CPU1)
SET OWNER    = &STR(OPC COURSES     )
SET GROUP    = &STR(        )
SET JOBNAME  = &STR(SM409J5 )
SET JOBNUM   = &STR(JOB21455)
SET DATE     = &STR(03/02/2005)
SET JOBSTART = &STR(14.03.44)
SET JOBEND   = &STR(14:03.44)
SET STEPNAME = &STR(BR14#1  )
SET PSTEPNAM = &STR(        )
SET ABCODE   = &STR(S806)
SET USERCODE = &STR(    )
SET OPERTEXT = &STR(                    )
SET APPLIA   = &STR(0503021000)
SET OPERIA   = &STR(          )
SET PSTART   = &STR(0503021000)
SET PLNEND   = &STR(0503021000)
SET DEADLINE = &STR(0503031300)
SET EXSTAT   = &STR(B)
```

EQQUX007 submits the skeleton JCL by writing to the UX07OUT DD statement. This DD statement also needs to be added to the TWS for z/OS controller procedure, as shown in Example 2-10, “Skeleton for TWSINFO batch job” on page 52. The name of this job in our scenario is TWSGENIB.

3. The batch job TWSGENIB that is generated by EQQUX007 creates and then executes a CLIST. This CLIST contains the defined variables listed in Example 2-3 that were created by EQQUX007. The CLIST issues a WTO about the job failure and then uses ISPF file tailoring to generate another batch job. This batch job is called TWSINFO in our scenario. As it creates TWSINFO batch job, it resolves the variables in the INFOMAN request with the defined variable values.

Example 2-4 shows the SYSTSIN DD statement which contains the INFOMAN request before the following variables have been resolved:

- &DATE
- &TIME
- &JOBNAME
- &ERRCODE

Example 2-4 INFOMAN request in batch job before resolution by the CLIST

```
/SYSTSIN  DD *
  PROFILE   PREFIX(TWSRES1)
  ISPSTART  PGM(BLGINIT) PARM(SESS(00) CLASS(TWSBATCH) +
            IRC(INIT,EX TWSERROR,+
            &DATE,+
            &TIME,+
            TWS JOB &JOBNAME FAILED CC=&ERRCODE,END,END,Q)
```

Example 2-5 shows the SYSTSIN DD statement that contains the INFOMAN request after the variables have been resolved by execution of the CLIST.

Example 2-5 INFOMAN request in batch job after submission by the CLIST

```
//SYSTSIN  DD *
  PROFILE   PREFIX(TWSRES1)
  ISPSTART  PGM(BLGINIT) PARM(SESS(00) CLASS(TWSBATCH) +
            IRC(INIT,EX TWSERROR,+
            03/02/2005,+
            14:03,+
            TWS JOB SM409J5  FAILED CC=S806,END,Q)
```

4. The TWSINFO batch job invokes INFOMAN by executing the BLGINIT program. This program invokes the INFOMAN Stored Response Chain (SRC) and passes it the data that is shown in Example 2-3 on page 47. TWSINFO must run on the same system as Infoman. To ensure the batch job runs on the same system as Infoman, use one of two methods:
   a. Use a system affinity JCL card in the batch job as we have done, as shown in Example 2-6.

Example 2-6 JCL card with system affinity specified

```
//TWSINFO  JOB 0,CLASS=A,MSGCLASS=X,NOTIFY=TWSRES4
/*JOBPARM SYSAFF=SC64
```

b. Use a Workload Manager (WLM) scheduling environment. A WLM scheduling environment is a list of resource names and their required states, ON or OFF. The resources are defined in the WLM service definition as is the name of the scheduling environment. The batch job has the required scheduling environment specified on the job card as shown in Example 2-7.

Example 2-7 Scheduling environment specified on job card

```
//TWSINFO  JOB 0,CLASS=A,MSGCLASS=X,NOTIFY=TWSRES4,SCHENV=INFOYES
```

5. The TWSINFO batch job creates the problem record in INFOMAN by using a Stored Response Chain (SRC). An SRC simulates a user entering data into Infoman. It does this by following the same prompting sequence that a user does when entering records through the ISPF Infoman dialog. The SRC receives input from the batch job in the form of parameters separated by commas. The SRC must be created by the user in Infoman. The name of the SRC that we used is TWSERROR, as shown in Example 2-3 on page 47. The data after TWSERROR is the data that is entered into the problem record. Each piece of data is separated by a comma.

   At the end of this process, a new problem record is created in Infoman. The operator generally uses the Infoman ISPF interface and a different SRC to list all records that are created by TWS for z/OS. All our Infoman records were created with the reporter field set to TWSINFO. We created another SRC that generates a search with the reporter field equal to TWSINFO. This provides us with a list of all problem records that are created by the TWSINFO jobs, as shown in Example 2-8.

Example 2-8 List of problem records created by TWSINFO

```
Environment   Dialog   Record   Search   Scroll   Window   Options   Help
------------------------------------------------------------------ MORE: +-
===>
BLG1TSRL                 SEARCH RESULTS LIST                     LINE 1 OF 10
DATABASE: 5
RECORD ID     DESCRIPTION ABSTRACT
   1. 00000004  TWS JOB EA732J5  FAILED CC=0012
   2. 00000042  TWS JOB PJGL0030 FAILED CC=0012
   3. 00000043  TWS JOB PJGL0030 FAILED CC=JCL
   4. 00000050  TWS JOB JOB1     FAILED CC=JCL
   5. 00000075  TWS JOB SM404J5  FAILED CC=JCLI
   6. 00000077  TWS JOB EA736VR1 FAILED CC=CAN
   7. 00000087  TWS JOB JOB1     FAILED CC=JCL
   8. 00000088  TWS JOB T01T01D9 FAILED CC=0012
   9. 00000188  TWS JOB T01T01MA FAILED CC=0008
  10. 00000189  TWS JOB SM409J5  FAILED CC=S806
*** BOTTOM OF DATA ***
Line Cmds:  C=Copy  D=Delete  P=Print  S=Select  U=Update
 Type DOWN or UP to scroll the panel, or type END to exit the panel.
```

These records are then updated by the user to:

- Add more detailed information about the failed job.
- Document any recovery actions taken.
- Add any other detail which was not placed in the problem record by TWS for z/OS.

2.3 TWS for z/OS set up

To set up the TWS for z/OS part of this configuration, you need to:

1. Create EQQUX007.
2. Set up the skeleton JCL.
3. Activate EQQUX007.
4. Add the required DD statements to the TWS for z/OS controller.

2.3.1 Creating TWS for z/OS exit 7

TWS for z/OS provides a sample of EQQUX007 in the EQQX7ASM member of the SEQQSAMP library. The sample consists of a batch job to assemble and link-edit the exit into module EQQUX007. It contains the assembler program source for EQQUX007 in the SYSIN for the first step of the job. The second step of EQQS7ASM is the link-edit step. The exit must be link-edited into an authorized library, placed in a data set in the LNKLST concatenation, or placed on the STEPLIB DD statement of the TWS for z/OS controller. You must tailor this job to your installation before you run it.

Restriction: The interface from TWS for z/OS to the OPC Automation extension of System Automation for z/OS uses EQQUX007. OPC Automation provides its own copy EQQUX007. Therefore, you cannot use EQQUX007 as the module name for the TWS for the z/OS to Infoman interface. To continue to use EQQUX007 for other purposes, the OPC Automation copy of EQQUX007 calls modules UX007005 to UX007009 after it has execute. In our scenario, we implemented OPC Automation after the TWS for z/OS to Infoman interface. We renamed our EQQUX007 to UX007005.

We created a new authorized library and placed it on the STEPLIB DD statement of the TWS for z/OS controller. Our library is called TWS.INST.LOADLIB. Example 2-9 on page 51 shows the link-edit step of EQQX7ASM with our library specified on the SYSLMOD DD statement.

Example 2-9 Link-edit step from the EQQX7ASM sample

```
//LKED     EXEC PGM=IEWL,PARM='RENT,MAP,LIST,CALL',
//          COND=(4,LT,ASMH),REGION=512K
//SYSPRINT DD  SYSOUT=*
//SYSUT1   DD  UNIT=SYSDA,SPACE=(CYL,(3,3))
//SYSLMOD  DD  DISP=SHR,DSN=TWS.INST.LOADLIB
//SEQOBJ   DD  DISP=(OLD,DELETE),DSN=&&OBJ
//SYSLIN   DD  *
 INCLUDE SEQOBJ
 ENTRY   EQQUX007
 NAME    EQQUX007(R)
/*
```

The EQQX7ASM should complete with a condition code of zero.

2.3.2 Setting up the skeleton JCL

The skeleton JCL is provided in the EQQX7JOB member of the TWS for z/OS SEQQSAMP library. You need to tailor this JCL to your installation standards.To set up the skeleton JCL, you run the following commands in this order:

1. **CLIST**, which create creates a CLIST. This CLIST is tailored by EQQUX007.
2. **SKEL**, which copies the skeleton TWSINFO job into a temporary data set called &&SKEL.
3. **ISPF**, which executes the CLIST that was created. The CLIST uses file tailoring against the TWSINFO job and then submits TWSINFO.

The CLIST contained in the skeleton JCL does not produce problem records for any jobs that fail with a condition code of:

- JCL
- JCLI
- any numeric return code

These codes cause a WTO to be issued. Abends are the only codes which cause a problem record to be created. We removed this test in our scenario so that jobs failing with these errors would also create problem records.

The **SKEL** command generates the TWSINFO batch job. TWSINFO executes a procedure called TSOBATCH. You need to change this procedure to match your procedure which executes TSO in batch. In our test environment, we did not have a TSO batch procedure. So, we added the required program and DD statements to the skeleton as shown in Example 2-10, “Skeleton for TWSINFO batch job” on page 52.

Example 2-10 Skeleton for TWSINFO batch job

```
//TWSINFO  JOB 0,CLASS=A,MSGCLASS=X,NOTIFY=TWSRES4
/*JOBPARM SYSAFF=SC64
//* THIS JOB REPORTS AN ERROR TO THE INFO/MAN PROBLEM MANAGEMENT
//* SYSTEM.
//* THIS STEP EXECUTES ISPF IN BATCH
//ISPF    EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=2048K
//SYSPROC  DD DISP=SHR,DSN=CPAC.CMDPROC
//         DD DISP=SHR,DSN=ISP.SISPCLIB
//ISPSLIB  DD DISP=SHR,DSN=ISP.SISPSLIB
//ISPPLIB  DD DISP=SHR,DSN=ISP.SISPPENU
//         DD DISP=SHR,DSN=BLM.SBLMPNLS
//         DD DISP=SHR,DSN=BLM.SBLMSAMP
//ISPMLIB  DD DISP=SHR,DSN=ISP.SISPMENU
//* THE ISPLLIB DD-CARD MAY BE UNNECESSARY IN YOUR INSTALLATION.
//* IF IT IS, YOU SHOULD REMOVE THE NEXT TWO DD-CARDS.
//ISPLLIB  DD DISP=SHR,DSN=ISP.SISPLOAD
//         DD DISP=SHR,DSN=INFOUSER.INST.LOADLIB
//         DD DISP=SHR,DSN=BLM.SBLMMOD1
//ISPTLIB  DD DISP=SHR,DSN=ISP.SISPTENU
//ISPPROF  DD UNIT=SYSDA,SPACE=(CYL,(1,1,5)),
//         DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
//ISPLOG   DD UNIT=SYSDA,SPACE=(CYL,(1,1)),
//         DCB=(RECFM=VBA,LRECL=125,BLKSIZE=3120)
//SYSHELP  DD DISP=SHR,DSN=SYS1.HELP
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
  PROFILE   PREFIX(TWSRES1)
  ISPSTART  PGM(BLGINIT) PARM(SESS(00) CLASS(TWSBATCH) +
            IRC(INIT,EX TWSERROR,+
            &DATE,+
            &TIME,+
            TWS JOB &JOBNAME FAILED CC=&ERRCODE,END,Q)
/*
```

The last part of tailoring the skeleton is the data on the SYSTSIN DD statement. This part invokes Infoman in batch and executes the SRC. The parameters that you need are:

- PREFIX(), which is the TSO user ID that is used when the batch job accesses Infoman.
- CLASS(), which is an Infoman privilege class. A privilege class allows you to control which users have authority to perform which specific tasks and facilities.
- EX, which tells Infoman which SRC to execute. It is followed by the SRC name, which in our scenario is TWSERROR.
- Variables prefixed with an ampersand (&). These are the values passed to the SRC. In the skeleton JCL they are variables. These are replaced by values by the CLIST before the job runs.
- Description, which is data that is placed in the description field of the problem record by the SRC.

Example 2-11 shows the Infoman data in TWS for z/OS supplied sample. This sample:

- Does not include a CLASS() parameter.
- Has SRCNAME as the name of the SRC to execute.
- Contains many variables.

Example 2-11 EQQX7JOB skeleton job Infoman data default

```
//SYSTSIN  DD *
  PROFILE   PREFIX(TSOUSER)
  ISPSTART  PGM(BLGINIT) PARM(SESS(00) +
            IRC(INIT,EX SRCNAME,+
            &DATE,+
            &TIME,+
            &JOBNAME,+
            &JOBNUM,+
            &STEPNAME,+
            &PSTEPNAM,+
            &ERRCODE,+
            &EXSTAT,+
              OPC   TRACKING REPORTED &ERRCODE AT &TIME,Q,END)
/*
```

In our scenario, we decided:

- Not to use all the variables.
- To call our SRC TWSERROR.
- To add the CLASS(TWSBATCH) parameter to the Infoman request.
- To change the short description text.

Example 2-12 shows a copy of the INFOMAN invocation after the changes for our scenario.

Example 2-12 Infoman request used in our scenario

```
PROFILE   PREFIX(TWSRES1)
  ISPSTART  PGM(BLGINIT) PARM(SESS(00) CLASS(TWSBATCH) +
            IRC(INIT,EX TWSERROR,+
            &DATE,+
            &TIME,+
            TWS JOB &JOBNAME FAILED CC=&ERRCODE,END,Q)
```

The description field ends with CC=&ERRCODE. There are two additional commands to Infoman which follow the description field:

- **END**, which is the SRC that creates the problem record completes on the PROBLEM SUMMARY panel. At this point the problem record has not been saved. This end command saves the record.
- **Q**, which quits Infoman.

We created a sequential data set called TWS.INST.EQQUX007.INPUT.JCL in which we placed the tailored skeleton JCL. This data set is placed on the UX07IN DD statement in the TWS for z/OS controller procedure.

2.3.3 Activating EQQUX007

EQQUX007 is used by the TWS for z/OS controller. The exit is activated by adding the EXITS initialization statement to the TWS for z/OS controller parameter member as shown in Example 2-13 on page 55.

Note: If the EXITS initialization statement is not specified in the TWS for z/OS controller initialization statements, the TWS for z/OS controller will automatically try and load a number of the TWS for z/OS exits including EQQUX007. If the exit is not found, it issues a message in the MLOG and continues processing. The automatic load of exits can be prevented by coding EXITS CALLxx(NO) where xx is the exit number. WE recommend that you do this for all exits you do not intend to use.

Example 2-13 Exits initialization statements

```
EXITS    CALL00(NO)
         CALL01(NO)
         CALL02(NO)
         CALL03(NO)
         CALL07(YES)
         CALL09(NO)
         CALL11(NO)
```

When you are ready to activate the EQQUX007, you simply shutdown and restart the TWS for z/OS controller.

2.3.4 Adding the required DD statements

EQQUX007 identifies the DD statements that it will use when reading the skeleton JCL and then submitting the JCL. The sample uses two DD statements:

- UX07IN, which identifies the data set where the skeleton JCL is found.
- UX07OUT, which is the internal reader.

Example 2-14 shows the DD statements that we inserted in the TWS for z/OS controller procedure.

Example 2-14 DD statements inserted for z/OS Controller procedure

```
//*   DATASETS FOR EQQUX007
//UX07IN    DD  DISP=SHR,DSN=TWS.INST.EQQUX007.INPUT.JCL
//UX07OUT   DD  SYSOUT=(A,INTRDR)
```

Once these statements have been added and the EQQUX007 activated, the TWS for z/OS controller is shut down and restarted.

This step completes the setup of the TWS for z/OS side of the interface.

2.4 Infoman set up

In our scenario, we used the standard problem management application which comes as part of Infoman. We did not tailor any of the Infoman ISPF panels. To set up the Infoman:

1. Create a privilege class.

   Problem records created by TWS for z/OS are in this class.

2. Create a Stored Response Chain (SRC).

   The TWS for z/OS generated batch job TWSINFO uses this SRC to create problem records in Infoman.

2.4.1 Creating a privilege class

You might not need to create a privilege class at your site. We created this privilege class to show the SRC running under a specific privilege class in the batch job. The name of the privilege class that we created is `TWSBATCH`.

You might need to be defined in a specific privilege class to be able to define another privilege class. Your Infoman administrators should know which class you need to use. It is most likely that they will perform these steps for you. However, we have included the steps here for completeness.

To define a privilege class you must use the SYSTEM application. The top right hand corner of the Infoman PRIMARY OPTION MENU shows which application you are currently using. Figure 2-3 on page 57 shows that the MANAGEMENT application is currently in use. This application needs to be changed to SYSTEM before we proceed.

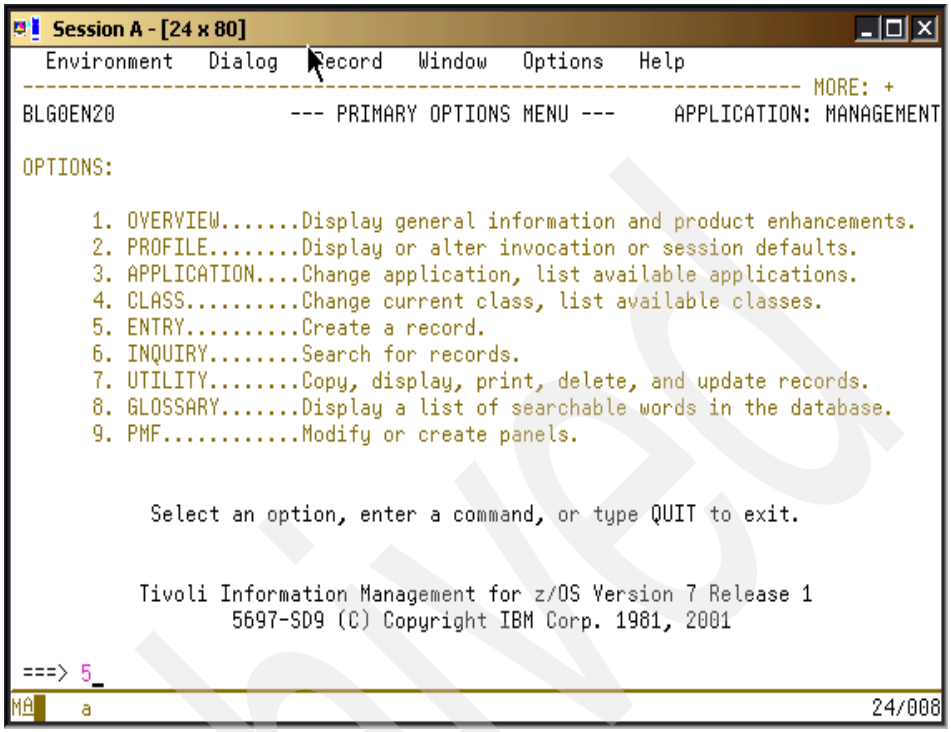

Figure 2-3 Infoman primary options menu showing application: management

Changing the Infoman application currently in use

To change the application which is currently in use:

1. Select option **3 - APPLICATION**. The APPLICATION SELECTION panel shown in Figure 2-4 displays.

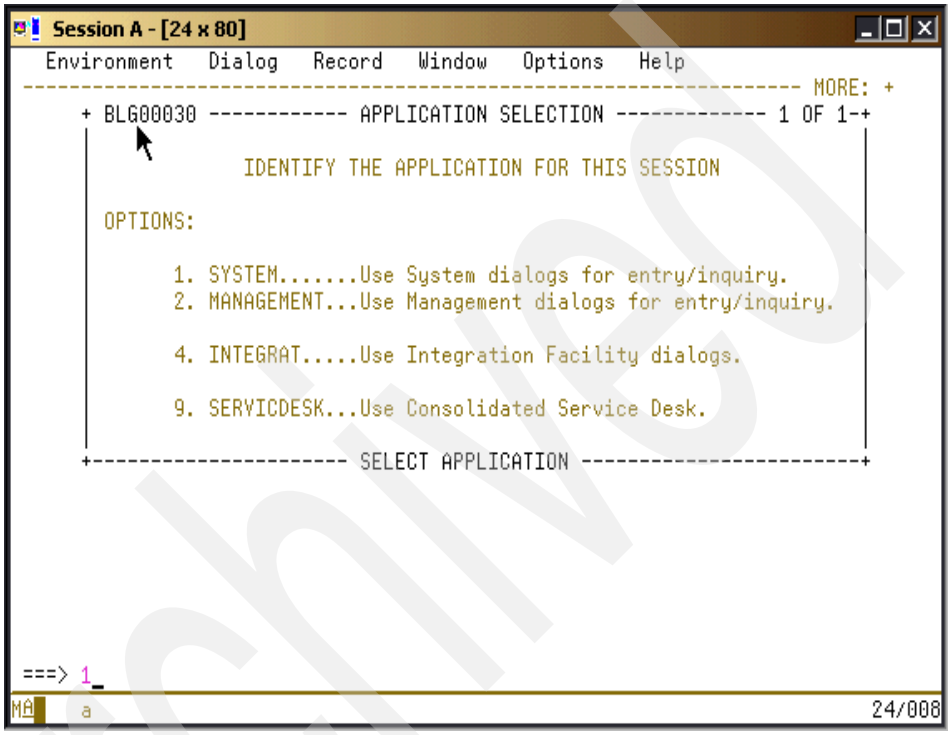

Figure 2-4 Application selection menu

2. Choose option **1 - SYSTEM** and press Enter. This sets your application to SYSTEM and returns you to the Infoman PRIMARY OPTIONS MENU.

   The top right-hand corner now shows the SYSTEM application is in use as shown in Figure 2-5 on page 59.

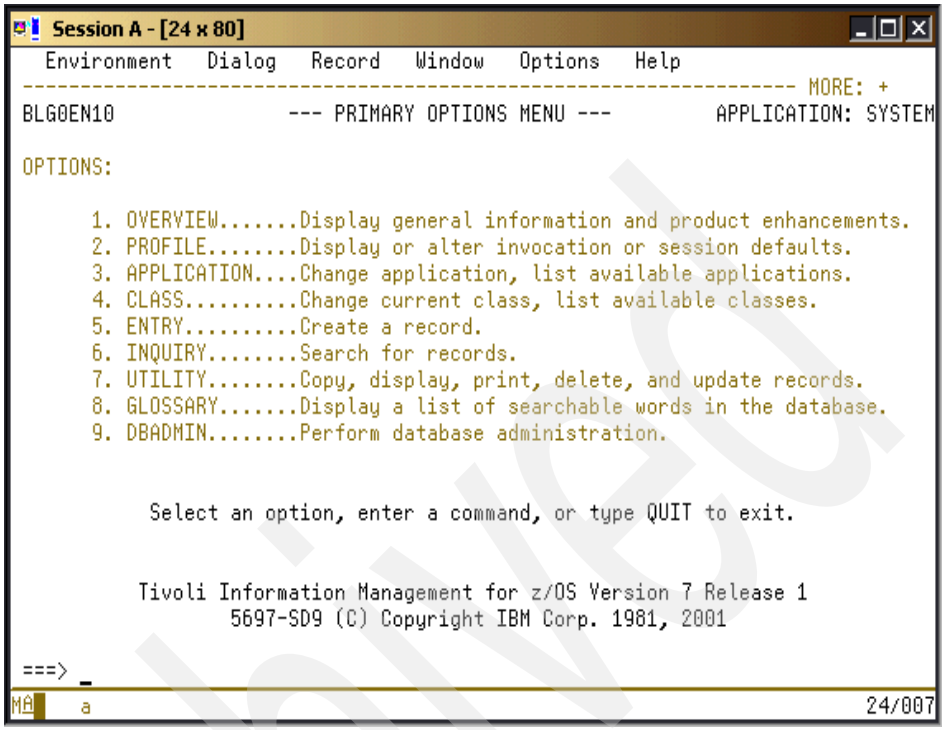

Figure 2-5 Infoman primary options menu showing application: system

When the correct application is in use, you can create a privilege class.

Creating a privilege class

To create a privilege class:

1. Select option **5 - ENTRY** on the Infoman PRIMARY OPTIONS MENU. The SYSTEM RECORD ENTRY panel shown in Figure 2-6 displays.

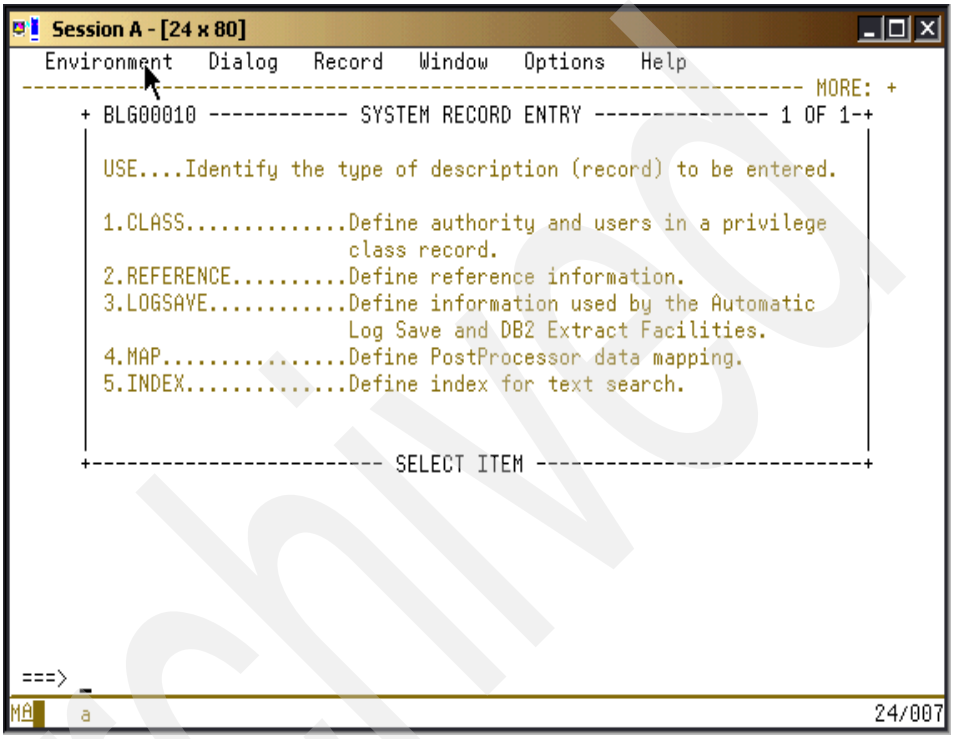

Figure 2-6 System record entry panel

2. From the SYSTEM RECORD ENTRY panel, select option **1 - CLASS**. The CLASS DESCRIPTION ENTRY panel as shown in Figure 2-7 on page 61 displays.

.

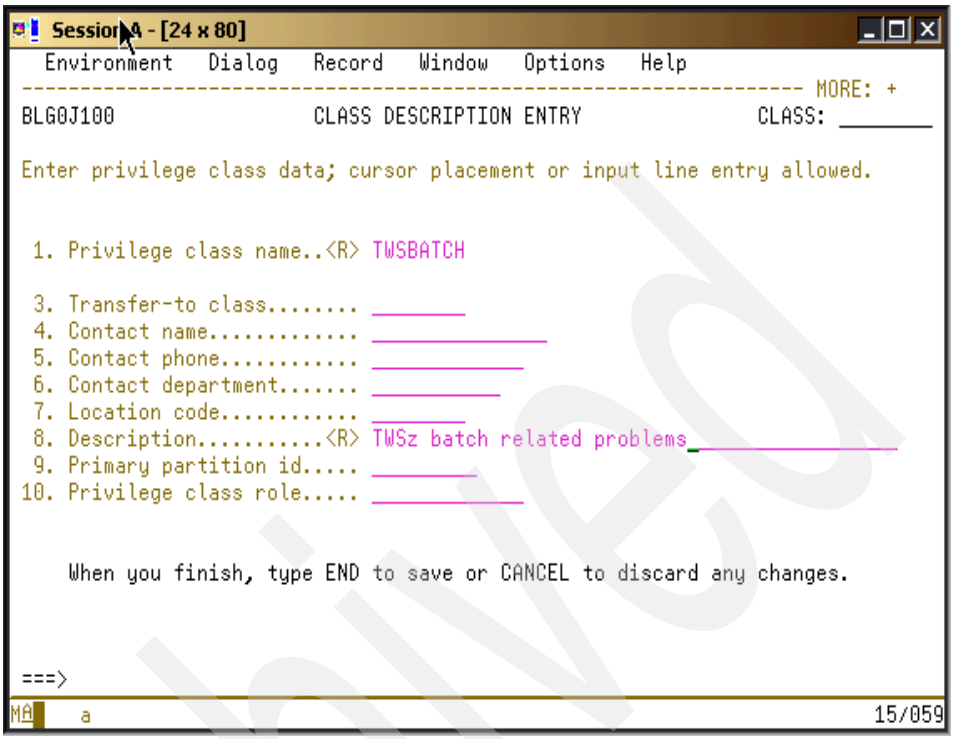

Figure 2-7 Class description entry panel

There are two required entry fields, which are highlighted by the <R>:

- Privilege class name
- Description

3. Complete these fields, press **PF3**, or type **end** and press Enter. This saves the data and takes you to the CLASS SUMMARY panel shown in Figure 2-8 on page 62.

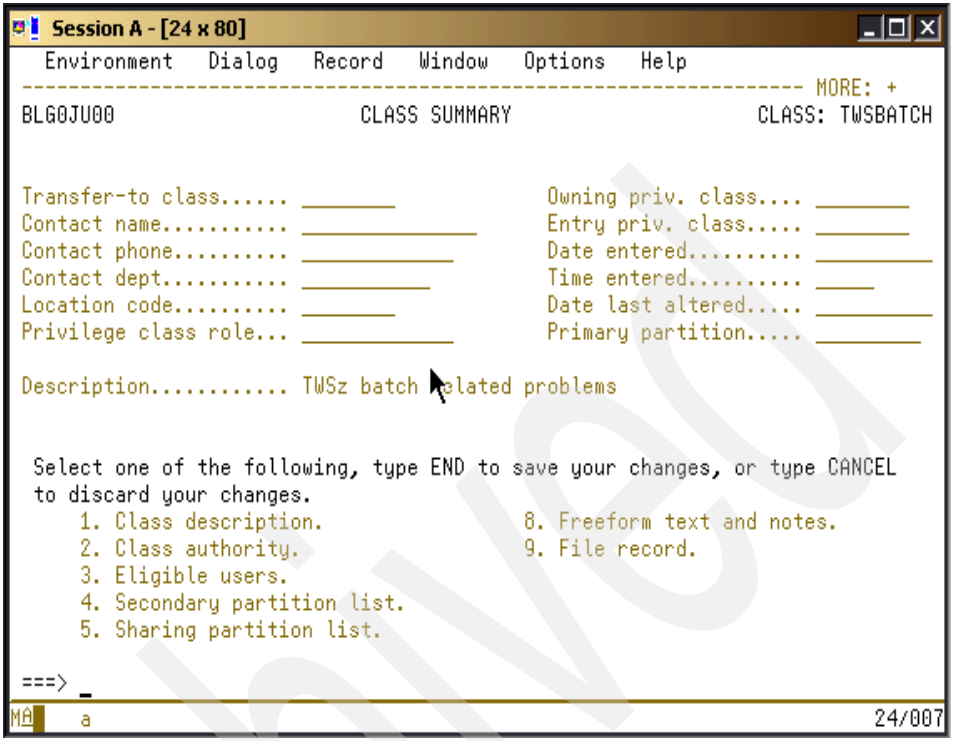

Figure 2-8 Class summary panel

4. From the CLASS SUMMARY panel, you can enter additional data by selecting the other options. There are two additional pieces of data that we need to enter:
   - Class authority, which describes what access a privilege class has to records and data in the Infoman database.
   - Eligible users, which is the list of user IDs that are eligible to use this privilege class.
5. Define the authorities for the TWSBATCH privilege class. Select option **2 - Class authority** and press Enter. The AUTHORITY ENTRY panel shown in Figure 2-9 on page 63 displays.

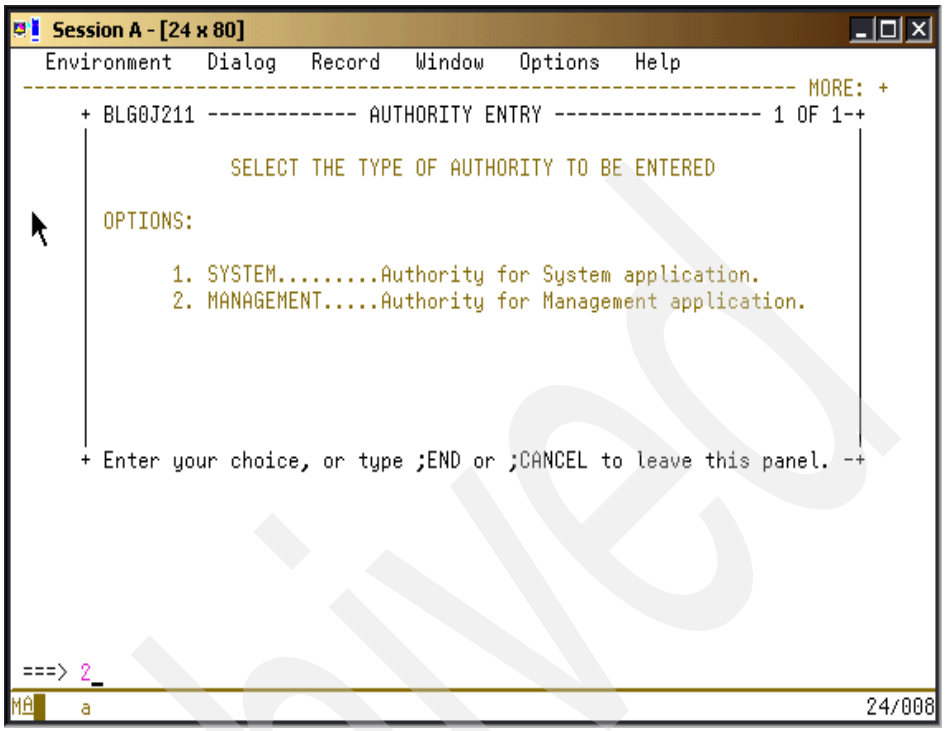

Figure 2-9 Authority Entry panel

We are using the SYSTEM application to create this privilege class record. The TWSBATCH privilege class is used to create problem records only. The problem record application is part of the MANAGEMENT application. Therefor, e the TWSBATCH privilege class does not have any authority in the System application.

6. Select option **2 - MANAGEMENT**. The AUTHORITY ENTRY panel shown in Figure 2-10 on page 64 displays.

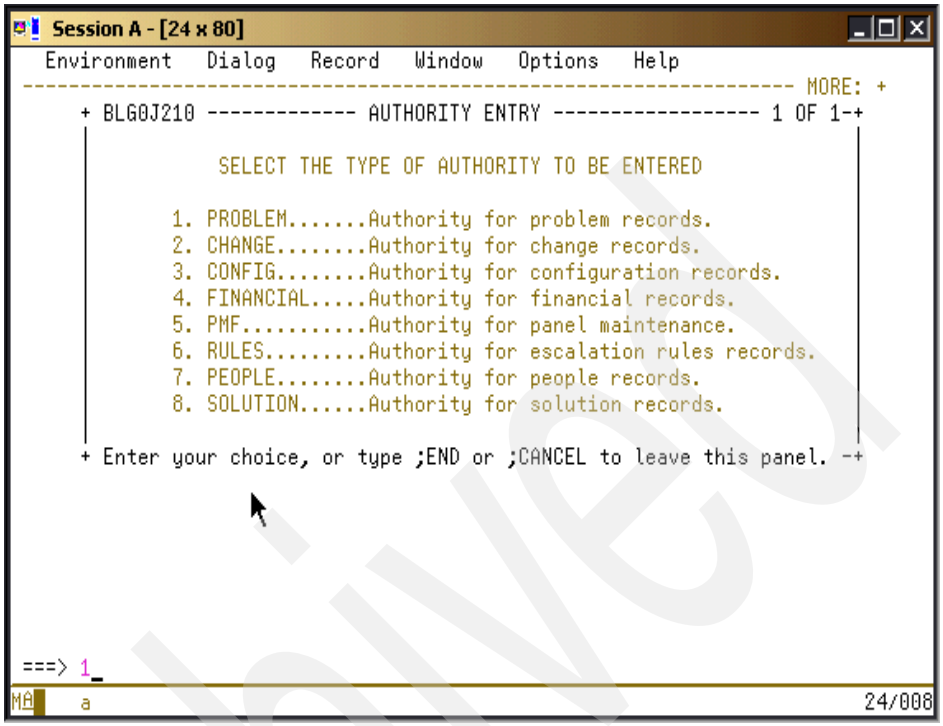

Figure 2-10 Authority Entry panel with different types of management applications

There are number of applications in the MANAGEMENT class. Because TWSBATCH is only used for managing problems, it only has authority in the problem application.

7. Select option **1 - PROBLEM** and press Enter. The PROBLEM AUTHORITY panel shown in Figure 2-11, “Problem Authority panel” on page 65 displays.

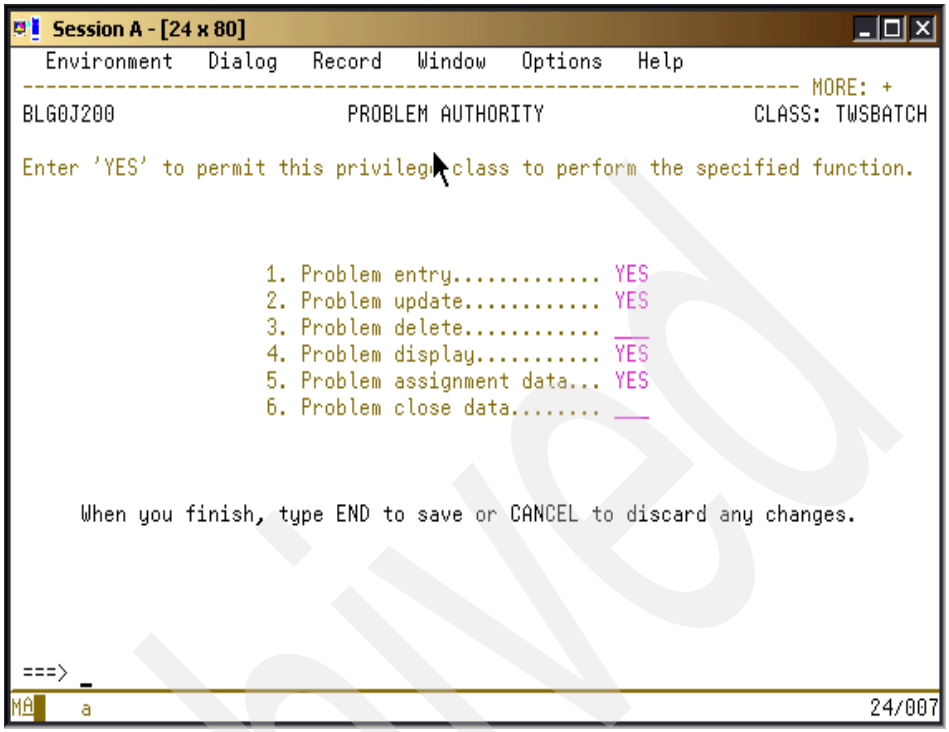

Figure 2-11 Problem Authority panel

8. On the PROBLEM AUTHORITY panel, enter **YES** so that the TWSBATCH privilege class can perform the specific function on problems records. For this example, we selected:
   - Problem entry
   - Problem update
   - Problem display
   - Problem assignment data

   We do not want TWSBATCH to be able to delete or close problem records. The batch operators also use this class to update the problem records after they are created by the TWSINFO batch job, because the operators are responsible for providing additional information to the problem record. Therefore, they must be able to update it. They are not to make decisions as to whether a record was validly created. Therefore, they do not have delete authority. Close data is the responsibility of the group which owns the batch job which has failed. Because the operators in our scenario do not own any batch jobs, they do not need close authority.

9. After this data has been entered, press **PF3**, or type `end`, and press Enter. Continue to press **PF3**, or type **end** and press Enter until you return to the CLASS SUMMARY panel.

10. Define which users have access to this privilege class. Select option **3 - Eligible users** and press Enter. The GENERAL USER AUTHORIZATION TABLE shown in Figure 2-12 displays.

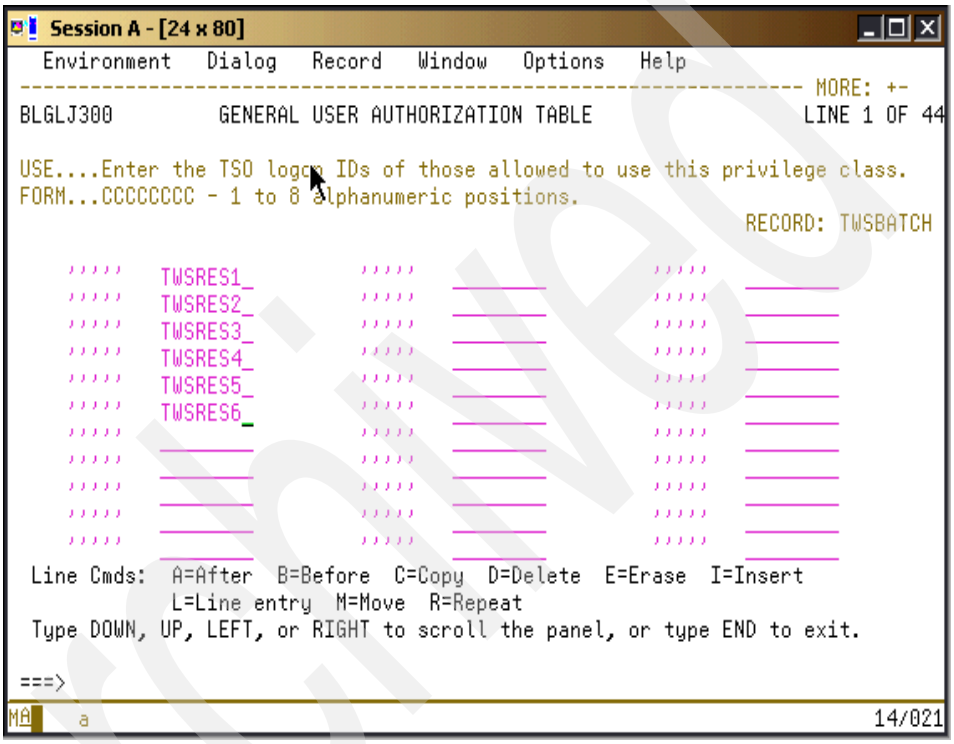

Figure 2-12 General user authorization table panel

11. On the GENERAL USER AUTHORIZATION TABLE panel, enter the TSO logon IDs that are permitted to use this privilege class. The profile specified on the TWSINFO batch job must be one of these TSO logon IDs.

12. After you have entered the TSO logon IDs, press **PF3**, or type **end** and press Enter. This returns you to the CLASS SUMMARY panel. Press **PF3**, enter **end** and press Enter, or select option **9 - File record** which files the privilege class record and returns you to the Infoman PRIMARY OPTIONS MENU.

The TWSBATCH privilege class has now been created and is available for use. This is confirmed by the message as shown in Figure 2-13 on page 67.

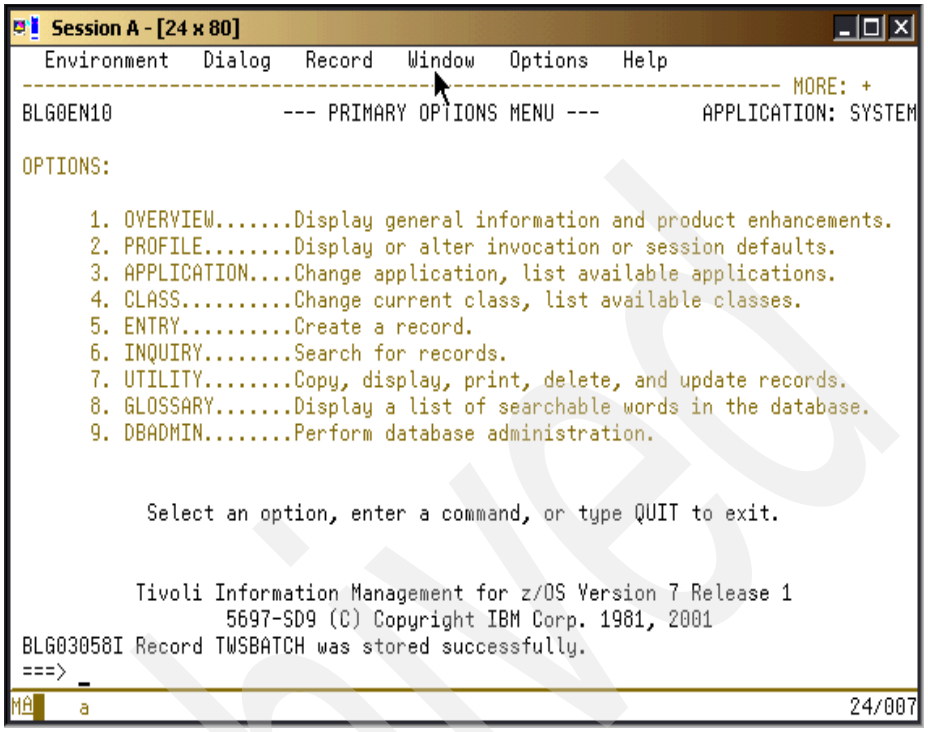

Figure 2-13 Primary options menu after TWSBATCH record created

2.4.2 Creating a stored response chain

The Stored Response Chain (SRC) is used by the TWSINFO batch job to create the problem records in Infoman. The name of the SRC that we created is TWSERROR. We used the sample Infoman Problem application that is included with the product. Your site might want to tailor the panels to make them more relevant to a batch job and add your own fields to these panels before creating the SRC. In this case, the SRC you create would differ.

The SRC we show here is very simple and is used for creating very basic problem records. Creating an SRC is fully documented in *Information Management for z/OS User's Guide*, SC31-8756.

To create a privilege class you must use the MANAGEMENT application. The top right-hand corner of the Infoman PRIMARY OPTION MENU shows which application you are currently using. Figure 2-3 on page 57 shows that the MANAGEMENT application is currently in use. If this is not the case, then follow the steps starting at "Changing the Infoman application currently in use" on page 58. Where this section selects SYSTEM as the application, you need to

select MANAGEMENT instead. When your application is set as MANAGEMENT, you continue to create the SRC.

An SRC simulates a user entering data into Infoman. The process of creating an SRC involves using the ISPF Infoman dialog and recording what options are selected and any data entered. When the SRC expects to receive input from the TWSINFO batch job, a question mark is entered in the SRC.

To create the SRC from the Infoman PRIMARY OPTIONS MENU.

1. On the command line, type **generate** and press Enter to start the creation of the SRC as shown in Figure 2-14.

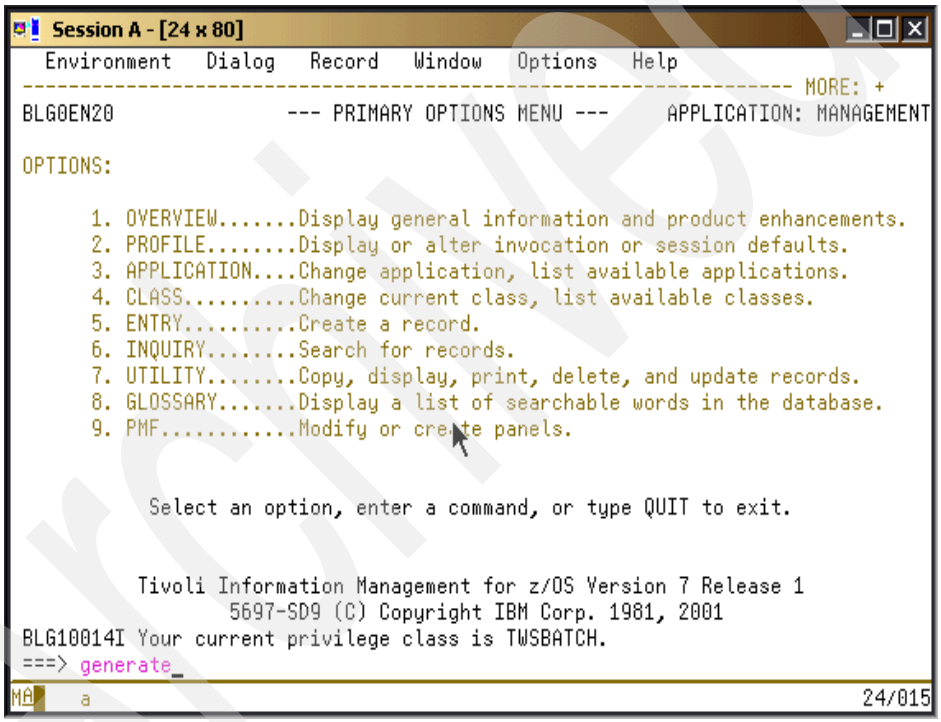

Figure 2-14 Start to generate the SRC

2. The SRC DESCRIPTION ENTRY panel as shown in Figure 2-15 displays.

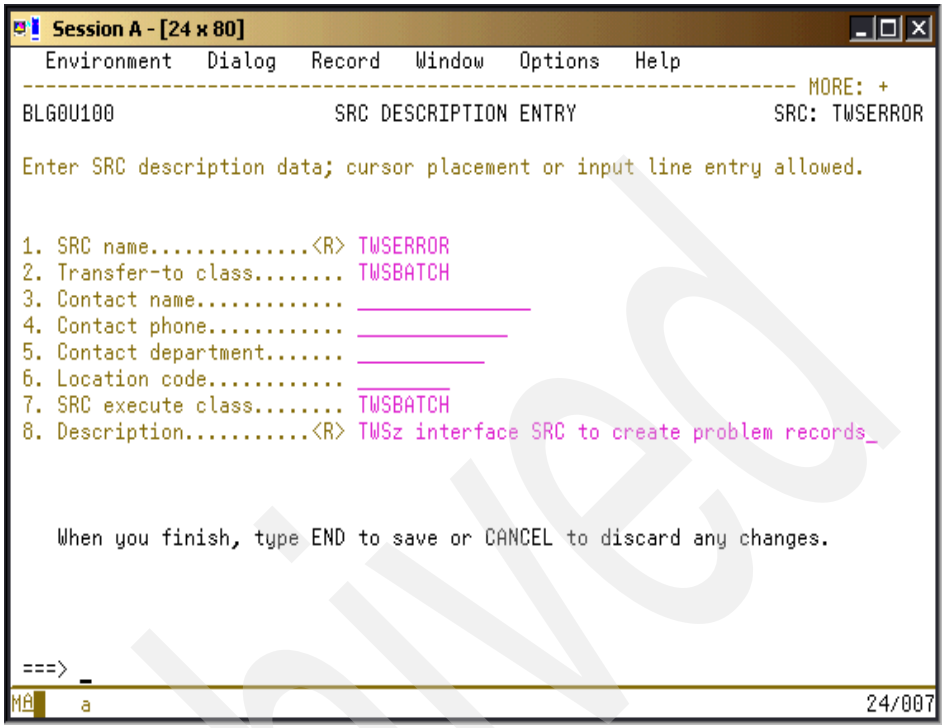

Figure 2-15 SRC description entry panel

3. On the SRC description entry panel, enter information about the SRC. The <R> in the field description signifies a mandatory field. Enter data in the following fields:
   - SRC name, which is the name that is executed by the TWSINFO batch job. Our SRC name is TWSERROR.
   - Transfer-to class, which is the privilege class to which the record is initially assigned. We assign this the TWSBATCH class.
   - SRC execute class, which is the privilege class permitted to execute this SRC. This is TWSBATCH. If this SRC is to be executed by more than one class, you provide this data later in the SRC creation process.
   - Description, which is a description of the SRC.

   For our test scenario, we did not enter any data into the other fields. After you have finished entering the data, press **PF3,** or type **end** and press Enter. This causes the SRC SUMMARY panel as shown in Figure 2-16 on page 70 to be displayed. At this point, you have not recorded any keystrokes or user simulation.

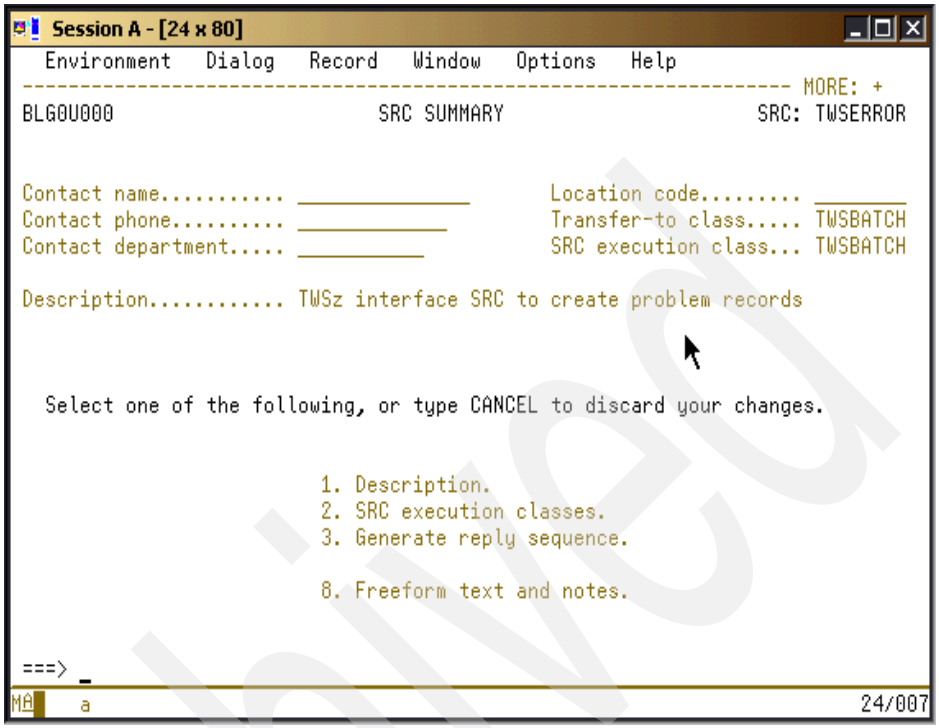

Figure 2-16 SRC Summary panel

4. To define additional privilege classes which might execute this SRC, use option **2 - SRC execution classes**. Because this SRC is executed only by the TWS for z/OS TWSINFO batch job, there is no need to use this option.
5. To start recording the user simulation, select option **3 - Generate reply sequence**. This action takes you back to the Infoman PRIMARY OPTION MENU because this is where the **`generate`** command was first entered. From here, type in the options and press Enter as though you are creating a problem record. Each response that you enter is recorded in the SRC.

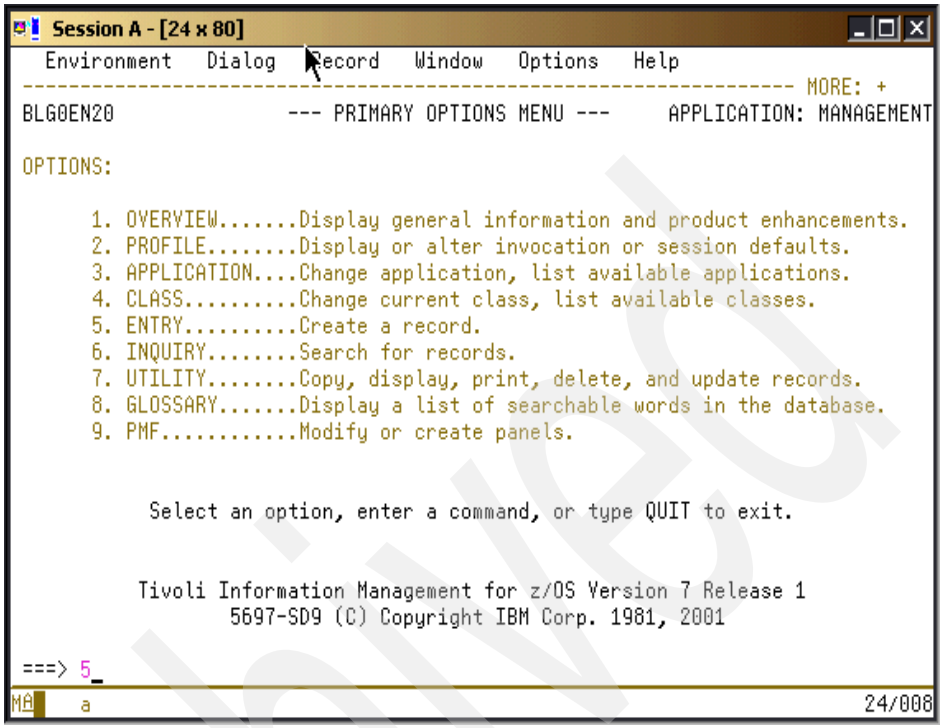

Figure 2-17 Starting to generate SRC

6. Enter the options and data as though you are creating a problem record manually. Select option **5 - ENTRY** and press Enter. The ENTRY panel shown in Figure 2-18 on page 72 displays.

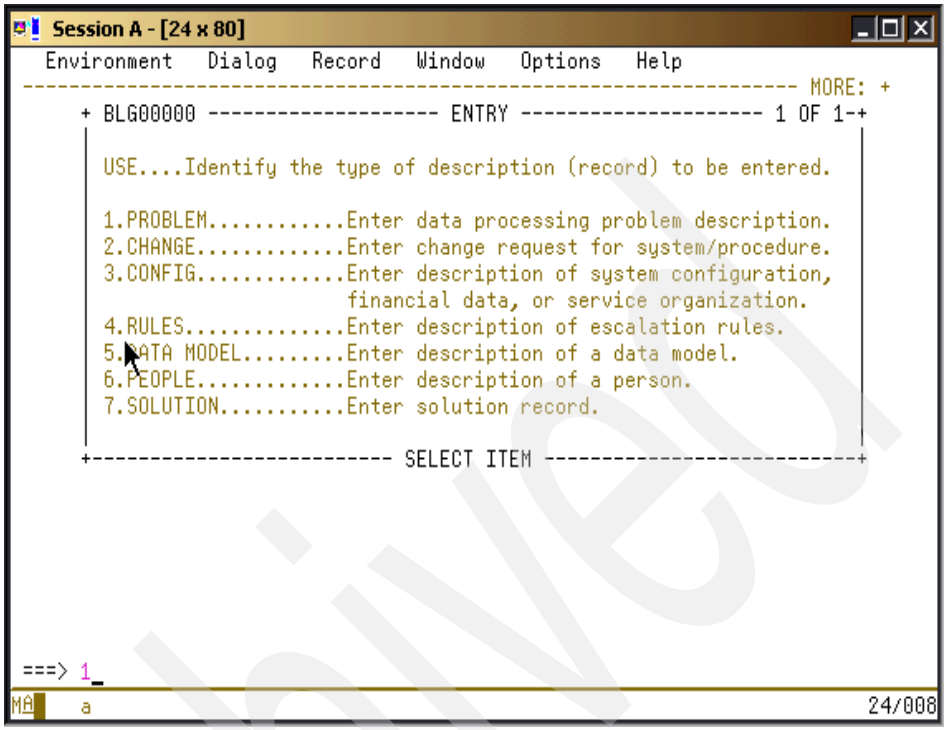

Figure 2-18 Problem entry panel

7. Select option **1- PROBLEM** and press Enter. The PROBLEM REPORTER panel displays. We enter data into this panel by typing the field number followed by data on the command line. The format is the field number followed by a comma and then the data. When you enter data into multiple fields, just follow the data of the previous field with another comma. So, the format is:

   ```
   field name 1,data 1,field name 2,data 2,field name 3, data 3
   ```

   An example is shown in Figure 2-19 on page 73.

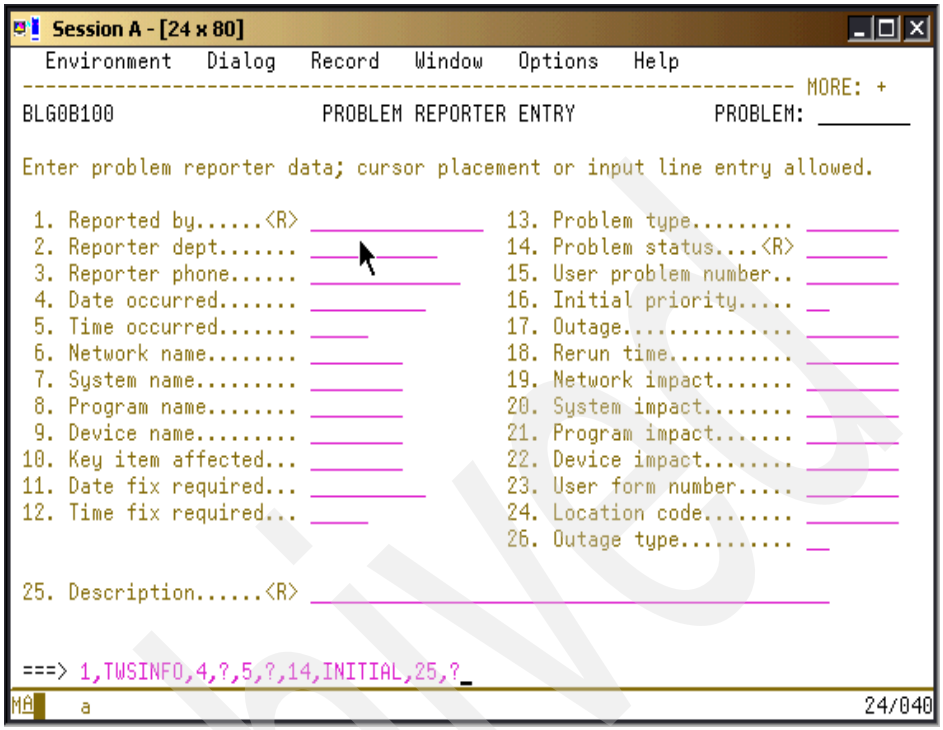

Figure 2-19 Problem Reporter Entry panel during SRC generation

The question mark suspends processing of the SRC so Infoman can collect data from the TWSINFO batch job. After it has collected this data, the SRC continues to process. Our data entry for the SRC is:

```
1,TWSINFO,4,?,5,?,14,INITIAL,25,?
```

Important: Data entered is case sensitive. If you need data to be in upper case, enter here in upper case. We want TWSINFO to appear in uppercase, so it must be entered in upper case. The Problem Status field is not case sensitive. It will always appear in upper case.

In our scenario, collecting data from the TWSINFO batch job works as follows:

- The Reported by field is filled in by the SRC with TWSINFO as this data exists in the SRC.
- The Date occurred field looks for input from the TWSINFO batch job because the field number is followed by a question mark. It takes the first

piece of data which follows the SRC name in the TWSINFO batch job. It finds the data and returns to the SRC.

- The Time occurred field looks for input from the TWSINFO batch job because it is also followed by a question mark. It looks for the next piece of data in the batch job. It finds the data and returns to the SRC.
- The Problem status field is filled in by the SRC with INITIAL.
- The Description field looks for input from the TWSINFO batch job because it is followed by a question mark. It looks for the next piece of data in the batch job. It finds the data and returns to the SRC.

Now that you have entered the sequence of data that will be collected from the TWSINFO batch job, you can continue with the SRC generation.

8. Press Enter on the panel which fills in the fields on the panel with the data that you have just placed on the command line. The PROBLEM SUMMARY panel as shown in Figure 2-20 displays.

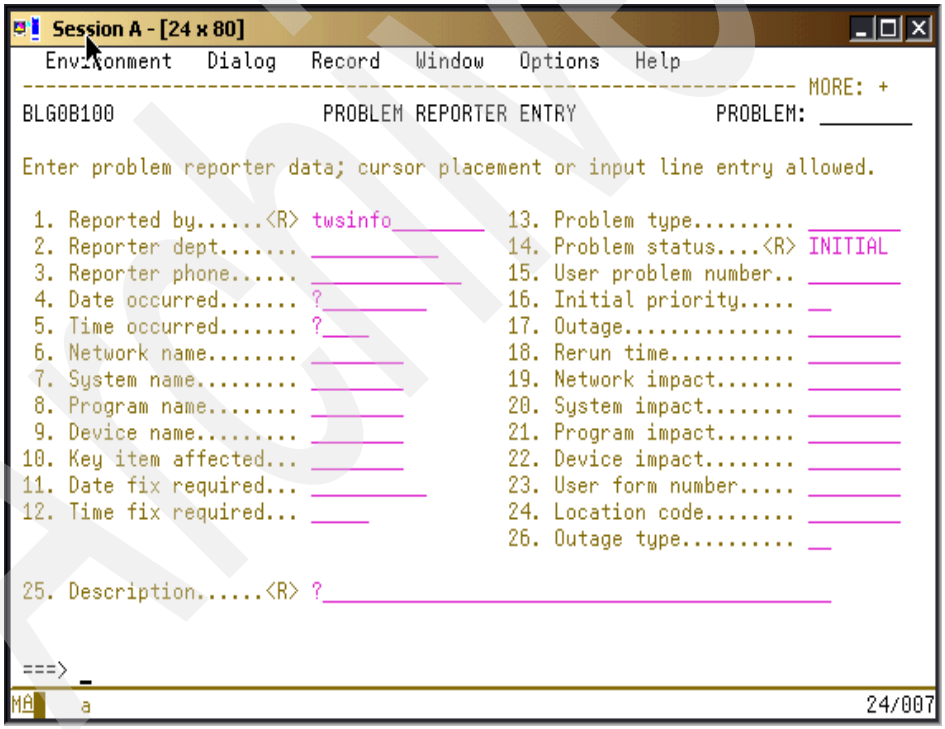

Figure 2-20 Problem reporter entry panel showing fields with SRC prompts

9. Press **PF3**, or type **end** and press Enter to return to the PROBLEM SUMMARY panel. You have now finished entering the data that is used by the SRC. To stop the user simulation at this point, type **end generate** and press Enter as shown in shown in Figure 2-21.

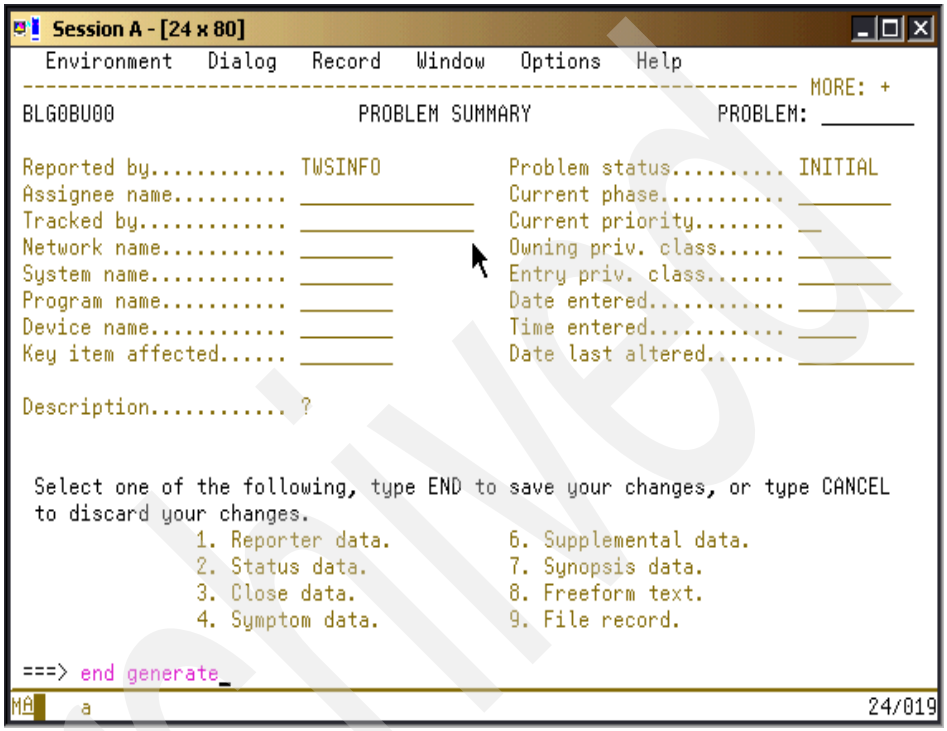

Figure 2-21 Problem summary panel at SRC generation end

After you enter the **end generate** command, you are returned to the SRC SUMMARY panel that is shown in Figure 2-22 on page 76.

Important: The SRC completes on the PROBLEM SUMMARY panel. At this point, the problem record and the data that was received from the TWSINFO batch job has not been saved. To save the record, enter the **end** command as shown here:

```
PROFILE   PREFIX(TWSRES1)
  ISPSTART  PGM(BLGINIT) PARM(SESS(00) CLASS(TWSBATCH) +
            IRC(INIT,EX TWSERROR,+
            &DATE,+
            &TIME,+
            TWS JOB &JOBNAME FAILED CC=&ERRCODE,END,Q)
```

This command saves the record. The **Q** command then quits Infoman.

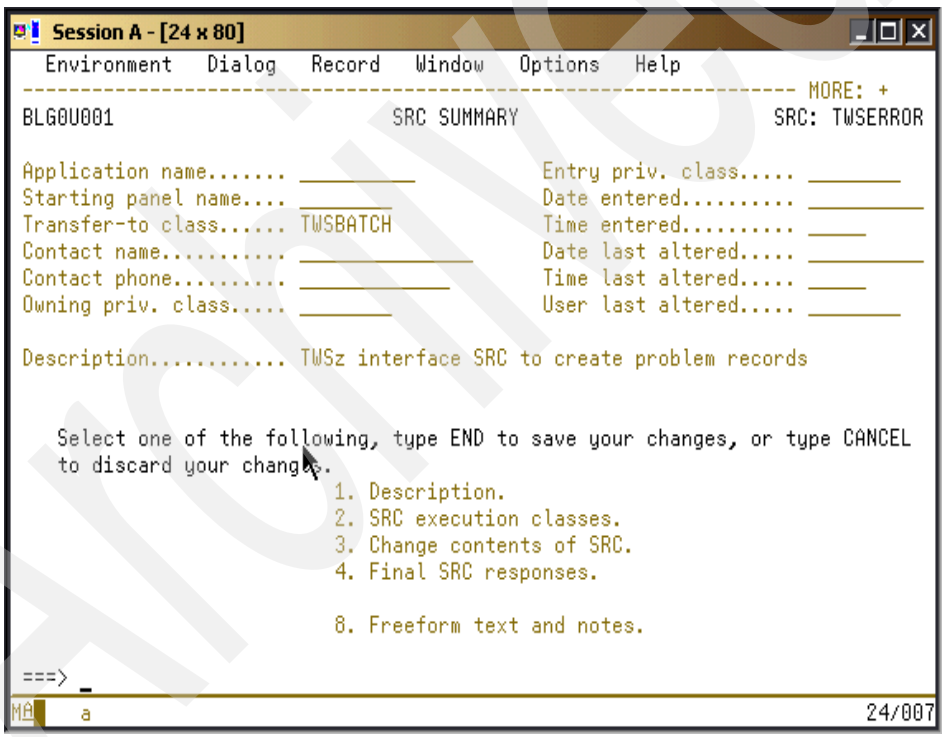

Figure 2-22 SRC summary at end of SRC generation

On the SRC SUMMARY panel, you have the option to change any of the data that you entered into the SRC before filing it.

10. You have now finished creating the SRC. You next need to file the SRC. Press **PF3**, or type **end** and press Enter to save the SRC. This action returns

you to the Infoman PRIMARY OPTIONS MENU and displays the following message:

```
BLG03840I SRC TWSERROR was filed successfully.
```

This message is shown in Figure 2-23.

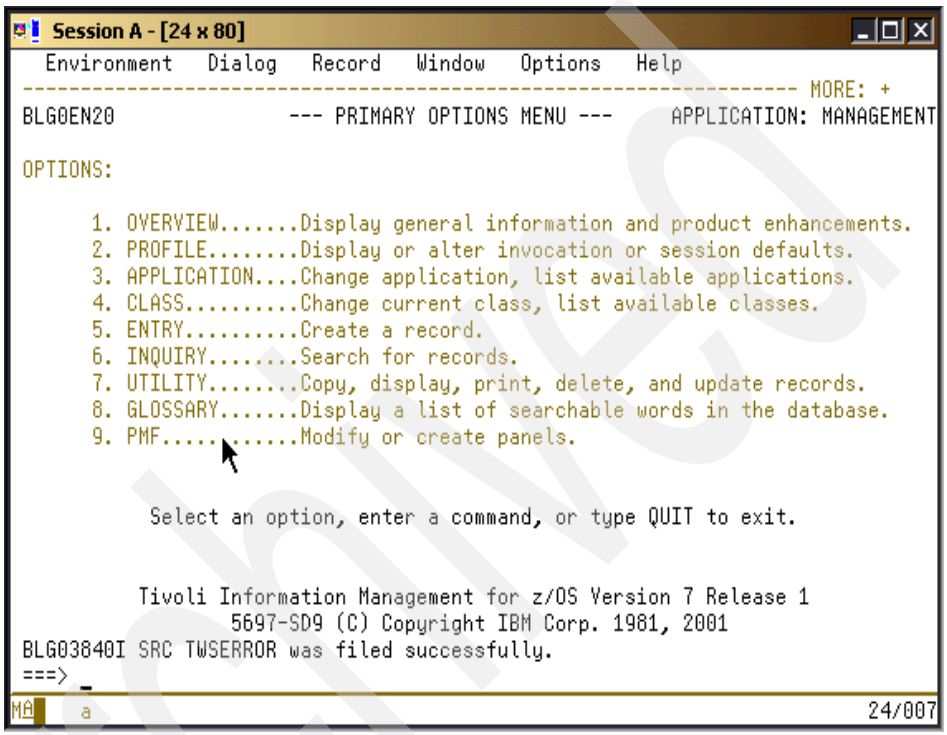

Figure 2-23 SRC file successful message

The SRC has now been created and is ready for use by the TWSINFO batch job.

2.5 Solving issues with the interface

Each batch job should complete with a zero return code. We suggest that you use the TWS for z/OS Event Triggered Tracking function to track both the TWSGENIB and TWSINFO jobs. This ensures that if either batch job has an issue, the operators are notified.

During our scenario we only had an issue with the TWSINFO batch job and Infoman.

We had two different issues:

1. There were insufficient variables to match the prompts in the SRC.

   Our TWSBATCH SRC expects to receive three pieces of data from the batch job. This is because there are three question marks in the SRC. Our initial batch jobs missed a comma and, therefore, provided only two pieces of data. This issue also caused the length one piece of data to be too long.

2. CC=16 with the following message in the batch job output

   ```
   BLG05009W THIS SRC CANNOT BE RUN BY PRIVILEGE CLASS.
   ```

   This issue was confusing because no privilege class is in the message. The problem was caused because the TSO logon ID that was used by the TWSINFO batch job had access to more than one privilege class. This issue was rectified by adding the CLASS(TWSBATCH) to the Infoman request.

When issues occur within Infoman, messages and panels are printed in the batch output that is referenced by the SYSPRINT DD statement.

We did not have any issues with EQQUX007. However, should there be a issue with the exit code, TWS might stop using the module. To determine if the exit has a problem, check the TWS for z/OS controller Message Log (MLOG). The MLOG is defined on the EQQMLOG DD statement of the TWS for z/OS controller.

3

Integrating Tivoli NetView for z/OS

This chapter discusses the use of Tivoli Workload Scheduler for z/OS (TWS for z/OS) with Tivoli NetView for z/OS. TWS for z/OS is a job scheduling application that allows users to schedule jobs or groups of jobs to be processed at a given time or under given circumstances. NetView for z/OS enables users to manage complex systems and networks from a single point. When NetView for z/OS is customized, it can be used to filter data and messages and automate responses to events in the environment. Through the use of Tivoli NetView for z/OS, issues in TWS can be used to trigger responses, including alerting personnel.[1]

[1] The material in this chapter is based on a whitepaper that was written by Art Eisenhour from IBM Americas Advanced Technical Support Center.

3.1 Description of the environment and scenarios

This solution was implemented using Tivoli Workload Scheduler for z/OS 8.2 and Tivoli NetView for z/OS 5.1. It requires an SMTP server running on the same z/OS image or in the same JES Multi Access Spool (MAS) system as Tivoli NetView for z/OS and that TWS messages are available to Tivoli NetView for z/OS for automation.

3.1.1 Scenarios shown in this solution

This chapter uses the following scenarios:

1. If certain TWS for z/OS scheduled jobs fail, a notification is sent.
2. If certain jobs or jobs above a certain priority are late, a notification is sent.
3. If any workstation goes into a status of *Offline*, *Unlinked*, or *Failed* and does not return within a specified number of minutes (*nn*), the workstation is considered unavailable, and a notification is sent. When a workstation becomes available that was unavailable for more than *nn* minutes, a notification is sent. This policy is designed to avoid false alarms if a workstation is interrupted temporarily (such as when the daily plan is updated). Finally, when a failed or new workstation comes online or is linked, a notification is sent.

3.2 Exposing TWS for z/OS messages to Tivoli NetView for z/OS automation

This section outlines how to expose messages by:

- Specifying the appropriate TWS ALERT WTO options in the controller and engine initialization statements.
- Editing the TWS SEQQMSG0 message library to specify `WTO=YES` for messages that are not provided by the ALERT options.

3.2.1 Reporting job and TWS for z/OS failures

You need to specify the appropriate TWS for z/OS controller and engine ALERT WTO initialization options. Specify `ERROROPER` to expose job ended-in-error message EQQE036I and `LATEOPER` for job late message EQQE037I. The following is a brief description of the ALERT WTO options. You can find more

details in *Tivoli Workload Scheduler for z/OS 8.2 Customization and Tuning*, SC32-1265.

- **DURATION**: When an operation is active for an unexpectedly long time, message

  EQQE028I is issued for an operation at a general workstation
  EQQE038I for an operation at a computer or printer workstation
  EQQE039I for computer operations that have been submitted but not started

- **ERROROPER:** When an operation is set to ended-in-error status, message

  EQQE026I is issued for an operation at a general workstation
  EQQE036I for an operation at a computer or printer workstation

- **LATEOPER:** When an operation in the current plan becomes late, message

  EQQE027I is issued for an operation at a general workstation
  EQQE037I for an operation at a computer or printer workstation.

- **OPCERROR:** Message EQQZ045W is issued when a subtask or subsystem ends unexpectedly.

  EQQN019E is issued if unable to build a VSAM LSR buffer pool.

- **QLIMEXCEED**: Message EQQZ106W is issued when a subtask queue exceeds a threshold value.

- **RESCONT:** Message EQQQ515W is issued when an operation exceeds the resource queue time specified on the CONTENTIONTIME keyword of RESOPTS.

For example, in the controller initialization options, specify:

```
ALERTS   WTO   (DURATION
ERROROPER
LATEOPER
…….)
```

Example 3-1 contains a sample of these TWS for z/OS messages as they appear in the MVS SYSLOG:

Example 3-1 Sample job failure messages in the MVS SYSLOG

```
EQQE036I JOB TWSEXTCP(JOB04259), OPERATION(0020), ENDED IN ERROR 0012
PRTY=9, APPL = DAILYPLAN , WORK STATION = CPU1, IA = 0501180700
EQQE037I JOB OVERTIME(        ), OPERATION(0090), IN APPLICATION
PAYROLL         , IS LATE, WORK STATION = CPU1, IA = 0501211030
EQQE038I LONG DURATION FOR JOB TWSEXTCP(JOB04303), OPERATION(0020),
IN APPLICATION DAILYPLAN , WORK STATION = CPU1, IA = 0501190700
```

3.2.2 Reporting workstation failures

For this step, edit the TWS for z/OS message library SEQQMSG0 member EQQWL1, and add WTO=YES to message EQQWL10 (Example 3-2). It is recommended that you implement message library changes with SMPE as a usermod to SEQQMSG0.

Alternatively, you could edit the TWS for z/OS supplied message member into a private message library concatenated before SEQQMSG0 in EQQMLIB DD in the TWS for z/OS controller/engine STC JCL for testing purposes.

Example 3-2 Sample change to EQQWL10 in SEQQMSG0(EQQWL1) member

```
EQQWL10 ' ' WTO=YES                <== add WTO=YES to expose the message
'W WORK STATION &WSID, HAS BEEN SET TO &STAT STATUS'
EQQWL11 ' '
'I ACTIONS ARE: &FAILACT, &REACT, &STARTACT'
EQQWL12 ' '
'I WORK STATION &WSID IS MANUALLY VARIED TO &STATUS STATUS'
EQQWL13 ' '
'I REROUTING IN EFFECT FOR WORK STATION &WSID'
EQQWL15 ' '
'I REROUTING IS WITHDRAWN'
```

Example 3-3 shows a sample of the TWS for z/OS messages as they appear in the MVS SYSLOG:

Example 3-3 Sample workstation failure messages in SYSLOG

```
EQQWL10W WORK STATION LNO2, HAS BEEN SET TO OFFLINE  STATUS
EQQWL10W WORK STATION LNO2, HAS BEEN SET TO UNLINKED STATUS
EQQWL10W WORK STATION LNO2, HAS BEEN SET TO ACTIVE   STATUS
EQQWL10W WORK STATION LNO2, HAS BEEN SET TO LINKED STATUS
```

This scenario sends a notification only if a workstation is offline or unlinked for more than 12 minutes. This outage toleration is defined in the wait4cmd variable in the TWSWSDN REXX routine, which is described in the next section.

3.3 Customizing NetView to enable SMTP notification

For customizing NetView to enable SMTP notification, you can use the *Tivoli NetView for z/OS 5.1 Automated Operations Network Customization Guide*, SC31-8871 and the *Tivoli NetView for z/OS 5.1 Automation Guide*, SC31-8853 for reference.

Important: SMTP must be running on the same MAS where NetView is running. The name of the SMTP started task defaults to SMTP but can be specified in CNMSTYLE in variable COMMON.EZLsmtpNAME.

These scenarios use the following NetView supplied REXX routines, which are located in CNMCLST:

- **INFORM:** NetView command that generates an immediate inform action based on the INFORM policy member, EZLINSMP, in DSIPARM. An operator can enter the name of the individual or group policy to contact, and optionally specify message text. INFORM is a synonym for EZLECALL. Our scenario REXX routines call INFORM.
- **INFORMTB:** REXX routine that reads and processes the inform policy member, EZLINSMP, and initializes common global variables used by INFORM. It is normally invoked during NetView initialization by auxInitCmd.INFORM in CNMSTYLE. It can be executed from a command line to update the inform policy in memory after startup. INFORMTB is a synonym for EZLEITBL.
- **EZLESMTP:** REXX routine that sends a 1-line e-mail. It takes input parameters, builds SMTP e-mail commands, puts them in a safe, and calls EZLESSMT to send the e-mail. Our scenario policy specifies EZLESMTP as the SMTP Interface routine.
- **EZLESSMT:** REXX routine that sends SMTP commands to SYSOUT class B with a destination defined by global variable EZLsmtpNAME - the default is SMTP. Called by EZLESMTP.

Also worth knowing about and used to verify the NetView to SMTP function are:

- **EZLEMAIL:** REXX window routine that is used by an operator to send mail. It takes operator input, builds SMTP e-mail commands, puts them in a safe, and calls EZLESSMT to send the e-mail.
- **EZLKMAIL:** The view panel that EZLEMAIL uses.

To customize Tivoli NetView for z/OS:

1. Check that NetView is customized to be able to submit jobs to JES.

   Verify that the DSIRQJOB is active by listing the DSIRQJOB task status:

   ```
   LIST DSIRQJOB
   ```

   NetView should respond:

   ```
   TYPE: OPT TASKID: DSIRQJOB TASKNAME: DSIRQJOB STATUS: ACTIVE
   LOADMOD: DSIRQJOB
   ```

   If DSIRQJOB is not active, then the following customization steps should bring up DSIRQJOB the next time NetView starts.

   a. Customize CNMSTYLE so that the following statement is commented out as shown here:

      ```
      *TASK.DSIRQJOB.MOD=*NONE*
      ```

   b. Customize CNMSTASK so that the following statements are present and not commented:

      ```
      TASK.DSIRQJOB.MOD=DSIRQJOB          // JES job ID requests
      TASK.DSIRQJOB.PRI=8
      ```

   You can start the DSIRQJOB task without recycling NetView by executing the command: shown in Example 3-4.

Example 3-4 Starting the DSIRQJOB task

```
start task=DSIRQJOB,MOD=DSIRQJOB,PRI=8
NetView should respond:
DSI166I DSIRQJOB IS ACTIVATED BY your userid
CNM493I CNMSTDAT : (NO SEQ) : CNMETDAS
DSI530I 'DSIRQJOB' : 'OPT' IS READY AND WAITING FOR WORK
DWO154I NETVIEW HAS RECEIVED JES JOBID STC.....
```

2. Check the SMTP address space name. If it is not SMTP, then customize the NetView CNMSTYLE variable COMMON.EZLsmtpNAME to specify the name of your SMTP address space:

   ```
   COMMON.EZLsmtpNAME = yourSMTPname
   ```

   Set the current value without recycling NetView with the SETCGLOB command:

   ```
   SETCGLOB EZLsmtpNAME TO yourSMTPname
   ```

3. Customize the NetView SMTP INFORM policy; DSIPARM member EZLINSMP. Example 3-5 on page 85 shows our scenario example. Specify a meaningful name for who is to be informed and specify the ROUTE for that person's e-mail or phone_service address.

Example 3-5 DSIPARM member EZLINSMP- our scenario example

```
SETUP LOG=YES,MEMBER=INFLOG,LOGCALLS=YES;
GROUP SAMBROWN,
LIST=SAMBROWNVIAMAIL,SAMBROWNVIAPHONE;
*
INFORM SAMBROWNVIAMAIL;                          Who to Inform – 'whoinform'
CONTACT ONCALLDAY=*,
SP=DUMMY,
CONNECTION=EMAIL,
ROUTE=SAMBROWN@US.IBM.COM,
INTERFACE=EZLESMTP,
NAME=Sam Brown;
*
INFORM SAMBROWNVIAPHONE;
CONTACT ONCALLDAY=*,
SP=DUMMY,
CONNECTION=EMAIL,
ROUTE=3015551212@MOBILE.ATT.NET,
INTERFACE=EZLESMTP,
NAME=Sam Brown;
```

4. Add the following auxInitCmd statement to CNMSTYLE to initialize the INFORM policy when NetView starts:

   ```
   auxInitCmd.INFORM = INFORMTB EZLINSMP
   ```

   Execute the INFORMTB command to initialize and to re-initialize the INFORM policy without recycling NetView.

   ```
   INFORMTB EZLINSMP
   ```

   NetView should respond as follows:

   ```
   EZL452I DSIPARM MEMBER EZLINSMP IS BEING USED FOR INFORM POLICY
   ```

   Check that your SMTP INFORM policy variables have been initialized:

   ```
   QRYGLOBL COMMON VARS=EZL*
   ```

   NetView should respond with the information: shown in Example 3-6 on page 86.

Example 3-6 Output from NetView

```
…… . . .
BNH036I GLOBAL VARIABLE NAME:       GLOBAL VARIABLE VALUE:
……..… .
EZLINFORMMEM                         EZLINSMP Á INFORM Policy name
EZLINFORMOP                          EZLAUTO1 Á Who initialized policy
EZLINFORMDATE                        02/10/05 Á When initialized
EZLINFORMTIME                        16:17:49
EZLINFORM.LOG.ENABLED                YES
EZLINFORM.LOG.MEMNAME                INFLOG
EZLTRACED                            NONE
EZLINFORM.LOGCALLS.ENABLED           YES
EZLSMTPNAME                          SMTP     Á Name of SMTP STC
……..… .
EZLINF_POLICY.2                      SAMBROWNVIAPHONE …..
EZLINF_POLICY.1                      SAMBROWNVIAMAIL  …..
EZLINF_POLICY.0                      …
EZL_INFORM_GROUP.1                   SAMBROWN
EZL_INFORM_GROUP.0                   …
…… . . .
```

5. Add a NetView autotask operator to run the SMTP automation, or choose an available existing autotask. Here is our autotask definition in DSIOPFU:

   ```
   EZLAUTO1   OPERATOR    PASSWORD=EZLAUTO1  ← SMTP autotask name
                PROFILEN      DSIPROFC
   ```

   If you add an operator to DSIOPFU, you can enable the operator to start without recycling NetView by executing the following NetView NCCF refresh **opers** command:

   ```
   nccf refresh opers
   ```

6. Add the following AUTOTASK statement to CNMSTYLE to start the SMTP automation autotask when NetView starts. Here is our CNMSTYLE definition:

   ```
   AUTOTASK.EZLAUTO1.Console = *NONE*
   ```

   You can start an autotask operator without recycling NetView with the following command:

   ```
   autotask opid=EZLAUTO1
   ```

   You can verify that the autotask operator is defined and started with the following command:

   ```
   qos op=EZLAUTO1
   ```

   This should result in the following response from NetView:

   ```
   DWO837I EZLAUTO1 IS DEFINED AND LOGGED ON TO NETVIEW
   ```

7. Customize your NetView Automation Table to trap the TWS for z/OS messages.

   Example 3-7 shows the section of our automation table that supports this scenario.

Example 3-7 Section of our automation table

```
*****************************************************************
* AUTOMATION TABLE STATEMENTS FOR TWS INFORMS                   *
*****************************************************************
* INFORM WORKSTATION STATUS CHANGES                             *
*   EQQWL10W WORK STATION LNO2, HAS BEEN SET TO OFFLINE STATUS  *
*   EQQWL10W WORK STATION LNO2, HAS BEEN SET TO UNLINKED STATUS *
*****************************************************************
*
IF MSGID = 'EQQWL1' . & TEXT = MESSAGE THEN
BEGIN;
ALWAYS
CONTINUE(Y);
IF ( MSGID = 'EQQWL10' . & ( TOKEN(9) = 'ACTIVE' . |
TOKEN(9) = 'LINKED' . ) ) THEN
EXEC(CMD('TWSWSUP ' MESSAGE)
ROUTE(ONE EZLAUTO1));   /* Specify appropriate autotask-oper */
IF ( MSGID = 'EQQWL10' . & ( TOKEN(9) = 'FAILED' . |
TOKEN(9) = 'OFFLINE' . | TOKEN(9) = 'UNLINKED' . ) ) THEN
EXEC(CMD('TWSWSDN ' MESSAGE)
ROUTE(ONE EZLAUTO1));
ALWAYS;
END;
*************************************************************************
* INFORM JOB FAILURES                                                   *
* EQQE036I JOB TWSEXTCP(JOB04259), OPERATION(0020), ENDED IN ERROR 0012*
*  PRTY=1, APPL = DAILYPLAN , WORK STATION = CPU1, IA = 0501180700     *
* EQQE037I JOB JOBX    (       ), OPERATION(0020), IN APPLICATION     *
*  PAYROLL         , IS LATE, WORK STATION = CPU1, IA = 0501180600     *
* EQQE038I LONG DURATION FOR JOB TWSEXTCP(JOB04303), OPERATION(0020),  *
*  IN APPLICATION DAILYPLAN , WORK STATION = CPU1, IA = 0501190700     *
*************************************************************************
IF MSGID = 'EQQE03' . & TEXT = MESSAGE THEN
BEGIN;
ALWAYS
CONTINUE(Y);
IF MSGID = 'EQQE036' . THEN
EXEC(CMD('TWSJBERR ' MESSAGE)
ROUTE(ONE EZLAUTO1));
IF MSGID = 'EQQE037' . THEN
EXEC(CMD('TWSJBLAT ' MESSAGE)
ROUTE(ONE EZLAUTO1));
ALWAYS;
END;
*****************************************************************
```

3.4 Adding REXX routines to DSICLD user library

This section contains sample REXX routines that you need to add to the DSICLD user library.

3.4.1 TWSWSDN

Example 3-8 shows the TWSWSDN REXX routine.

Example 3-8 TWSWSDN REXX routine

```
/*********************************************************************/
copyright='TWSWSDN (c) Copyright IBM Corp. 2005. All rights reserved.'
/* TWSWSDN IS TRIGGERED IN THE AUTOMATION TABLE BY MESSAGE EQQWL10W  */
/* when a WorkStation goes offline, failed, or unlinked. e.g.        */
/*   EQQWL10W WORK STATION LNO2, HAS BEEN SET TO OFFLINE  STATUS     */
/*   EQQWL10W WORK STATION LNO2, HAS BEEN SET TO UNLINKED STATUS     */
/*********************************************************************/
informwho = 'SAMBROWNVIAMAIL'
/*ChronTime hh.mm.ss */
wait4cmd = '00.12.00'        /* Outage toleration time – 12 minutes */
'GETMLINE mline1' 1
PARSE VAR mline1 . 'WORK STATION' WSID ',' . 'SET TO' STATE 'STATUS' .
WSID  = STRIP(WSID)
STATE = STRIP(STATE)
IF STATE = 'FAILED' THEN
STATE = 'OFFLINE'
notifycmd  = ''''INFORM' informwho mline1 ' at' MSGGTIME() ''''
'PIPE NETV CHRON AFTER='||wait4cmd||',COMMAND='||notifycmd ,
'| CORRWAIT 5 | VAR RDSI201I'
PARSE VAR RDSI201I . 'ID=' waitidin '''' .
TWSWSSTATE = WSID||STATE
INTERPRET TWSWSSTATE '= waitidin'
'GLOBALV PUTT' TWSWSSTATE
Return
*********************************************************************
```

3.4.2 TWSWSUP

Example 3-9 shows the TWSWSUP REXX routine.

Example 3-9 TWSWSUP REXX routine

```
/*********************************************************************/
copyright='TWSWSUP (c) Copyright IBM Corp. 2005. All rights reserved.'
/* TWSWSUP IS TRIGGERED IN THE AUTOMATION TABLE BY MESSAGE EQQWL10W  */
/* when a WorkStation goes active or linked. e.g.          .        */
/*   EQQWL10W WORK STATION LN02, HAS BEEN SET TO ACTIVE   STATUS     */
/*   EQQWL10W WORK STATION LN02, HAS BEEN SET TO LINKED STATUS       */
/*********************************************************************/
informwho = 'SAMBROWNVIAMAIL'
'GETMLINE mline1' 1
PARSE VAR mline1 . 'WORK STATION' wsid ',' . 'SET TO' state 'STATUS' .
wsid  = STRIP(wsid)
state = STRIP(state)
stated = ''
IF state = 'ACTIVE' THEN
stated = 'OFFLINE'
ELSE IF state = 'LINKED' THEN
stated = 'UNLINKED'
TWSWSSTATE = wsid||stated
'GLOBALV GETT' TWSWSSTATE
INTERPRET 'statev =' TWSWSSTATE
IF statev <> TWSWSSTATE | statev <> '' THEN DO
'PIPE NETV PURGE TIMER='||statev' | CORRWAIT 5 | VAR prgout'
PARSE VAR prgout 'DSI205I' tc 'TIMER' .
tc = STRIP(tc)
IF tc = 0 THEN DO
SAY 'INFORM' informwho mline1 'at' MSGGTIME()
'INFORM' informwho mline1 'at' MSGGTIME()
END
INTERPRET TWSWSSTATE '= '''''
'GLOBALV PUTT' TWSWSSTATE
END
Return
*********************************************************************
```

3.4.3 TWSJBERR

Example 3-10 shows the TWSJBERR REXX routine.

Example 3-10 TWSJBERR REXX routine

```
*********************************************************************/
copyright='TWSJBERR (c) Copyright IBM Corp. 2005. All rights reserved.'
/* TWSJBERR IS TRIGGERED IN THE AUTOMATION TABLE BY MESSAGE EQQE036I */
/* WHICH INDICATES THAT A JOB SCHEDULED BY TWS HAS FAILED.           */
/*********************************************************************/
informwho = 'SAMBROWNVIAMAIL'
notepri   = 7
'GETMLINE mline1' 1
PARSE VAR mline1 . 'JOB' jobname '(' jobnum '),' . '),' text '.' .
PARSE VAR text . 'ERROR' desc
'GETMLINE mline2' 2
PARSE VAR mline2 . 'PRTY=' prtyc 'APPL =' appl . 'STATION =' wsid ',' .
PARSE VAR prtyc  prty ',' .
jobname     = STRIP(jobname)
jobnum      = STRIP(jobnum)
appl        = STRIP(appl)
prty        = STRIP(prty)
desc        = STRIP(desc)
wsid        = STRIP(wsid)
sysname     = CURSYS()
date_occurd = DATE('s')
time_occurd = SUBSTR(msggtime(),1,8)
parmvar = 'JOB' jobname,
'JNUM:'||jobnum,
'HAD ERROR:'||desc,
'APPL:'||appl,
'PRTY:'||prty,
'WSID:'||wsid,
'SYSN:'||sysname,
' at' date_occurd time_occurd,
'MSGID:'||MSGID()
IF prty >= notepri THEN DO
SAY 'INFORM' informwho parmvar
'TWSECALL' informwho parmvar
END
Return(0)
/* Obtain Date and Time Occurred from the message */
Msgtime:
JULIANDATE = MSGGDATE()
CALL REXXDATE 'SGRJUL' JULIANDATE
GREGDATE   = result
DATE_OCCURRED=SUBSTR(GREGDATE,3,2)||'/'SUBSTR(GREGDATE,5,2)||'/',
||SUBSTR(GREGDATE,1,2)
TIME_OCCURRED=SUBSTR(MSGGTIME(),1,2)||':'SUBSTR(MSGGTIME(),4,2)
Return
/*********************************************************************/
```

3.4.4 TWSJBLAT

Example 3-11 shows the TWSJBLAT REXX routine.

Example 3-11 TWSJBLAT REXX routine

```
/*********************************************************************/
copyright='TWSJBLAT (c) Copyright IBM Corp. 2005. All rights reserved.'
/* TWSJBLAT IS TRIGGERED IN THE AUTOMATION TABLE BY MESSAGE EQQE037I */
/* WHICH INDICATES THAT A JOB SCHEDULED BY OPC/ESA HAS FAILED.       */
/*********************************************************************/
informwho = 'SAMBROWNVIAMAIL'
notepri   = 7
'GETMLINE mline1' 1
PARSE VAR mline1 . 'JOB' jobname '(' jobnum '),' .
'GETMLINE mline2' 2
PARSE VAR mline2 . appl ',' desc ',' . 'STATION =' wsid ',' .
jobn        = STRIP(jobname)
jobnum      = STRIP(jobnum)
appl        = STRIP(appl)
desc        = STRIP(desc)
wsid        = STRIP(wsid)
sysname     = CURSYS()
date_occurd = DATE('s')
time_occurd = SUBSTR(msggtime(),1,8)
PARMVAR = 'JOB' jobn,
'JNUM:'||jobnum,
'APPL:'||appl,
'ERR:'||desc,
'WSID:'||wsid,
'SYSN:'||sysname,
'at' date_occurd time_occurd,
'MSGID:'||MSGID()
IF SUBSTR(appl,1,5) = 'DAILY',
| SUBSTR(appl,1,3) = 'TWS',
| SUBSTR(appl,1,3) = 'ART',
| SUBSTR(appl,1,3) = 'PAY',
| SUBSTR(jobn,1,3) = 'TWS',
THEN DO
SAY 'INFORM' informwho parmvar
'TWSECALL' informwho parmvar
END
Return(0)
/*********************************************************************/
```

Note: We encountered an issue with our MVS system ID, which contained the dollar sign special character ($). This special character caused the error messages from SMTP that are shown in Example 3-12.

Example 3-12 Error messages from SMTP

```
mail from:<EZLAUTO1@HCB$>
501 Syntax Error. Unexpected Token '$'
rcpt to:<SAMBROWN@US.IBM.COM>
503 Sender must be specified before recipients data
503 No Sender specified
```

You can circumvent this issue by modifying NetView REXX EXEC EZLESMTP to specify the NetView domain instead of the system ID. Modify the following line in EZLESMTP:

```
sep' LIT /mail from:<'OPID()'@'CURSYS()'>/' ,
```

Change `CURSYS()` to something else, such as the NetView domain name, `DOMAIN()`:

```
sep' LIT /mail from:<'OPID()'@'DOMAIN()'>/' ,
```

If this change is necessary for EZLESMTP, then do the same for EZLEMAIL.

3.4.5 Verifying that NetView can send SMTP mail

To verify that NetView can send SMTP mail, execute the **`EZLEMAIL`** command.

This command involves no automation and does not use the INFORM Policy. Issuing this command establishes that NetView can talk to SMTP. You should issue the command as a NetView operator.

If **`EZLEMAIL`** does not send e-mail, then verify that SMTP is running within the same JES MAS and that it is working properly. Verify that the name of the SMTP address space matches the value in common global variable, EZLsmtpNAME, which if not specified defaults to SMTP. Check that the DSIRQJOB task is active. Check for related messages in the NetView NETLOG and the MVS SYSLOG because these logs usually pinpoint the issue.

3.5 Testing your NetView notification automation

After you have completed the configuration, you can test the Tivoli NetView for z/OS notification of TWS for z/OS events.

3.5.1 Trying the job failure scenario

To test the job failure scenario:

1. Set up an application with an operation which will fail, for example EXEC PGM=IEFBR13, and add the application to the current plan.
2. When the job fails, the automation table entry for message EQQE036I invokes the TWSJBERR REXX routine.
3. TWSJBERR calls the INFORM routine and passes it the informwho variable to specify which the INFORM Policy entry to be used.
4. The INFORM routine sends e-mail to the address that is specified in your INFORM Policy ROUTE. The e-mail looks similar to this:

```
JOB ATSTBR13 JNUM:JOB06653 HAD ERROR:S806 APPL:ADHOC PRTY:8 WSID:CPU1
SYSN:HCB$ AT 20050210 16.19.06 MSGID:EQQE036I
```

Troubleshooting tips

If the e-mail does not arrive, then check the following:

- MVS SYSLOG should contain a TWS EQQE036I message for the failed job:

```
16:19:06 EQQE036I JOB ATSTBR13(JOB06653), OPERATION(0005), ENDED IN ERR
S806 PRTY=8, APPL = ADHOC          , WORK STATION = CPU1, IA = 0502101110
```

- If the EQQE036I message is not in SYSLOG, then check the Controller options and insure that option ERROROPER is included in the ALERTS WTO options. If the Controller option is specified correctly, the message will appear.

  NetView NETLOGA should contain the following messages:

```
16:19:06 CNM493I INFORMSM : (NO SEQ) : TWSJBERR EQQE036I JOB
ATSTBR13(JOB06653), OPERATION(0005), ENDED IN ERROR S806
EZL460I EMAIL ACTION WAS SUCCESSFULLY ISSUED FOR POLICY SAMBROWNVIAMAIL BY
```

- If the CNM493I message is not in NETLOG for this event, then the automation table is not trapping the EQQE036I message. Check that the z/OS MPFLST allows automation for EQQ* messages.

  Review your automation table. Refer to the *Tivoli NetView for z/OS 5.1 Automation Guide*, GC31-8851 for guidance.

- If the EZL460I message is missing, then look in NETLOG for messages that describe what failed.

- If the EZL460I message is in NETLOG and the mail still has not arrived, then review the INFORM Policy definitions to verify that the ROUTE is specified correctly. The best way to do this is to execute the NetView command:

  ```
  QRYGLOBL COMMON VARS=EZL*
  ```

  NetView will respond with the current INFORM Policy values as shown in Example 3-13.

Example 3-13 NetView output

```
EZLINFORMMEM                EZLINSMP Á INFORM Policy name
EZLINFORMOP                 EZLAUTO1 Á Who initialized policy
EZLINFORMDATE               02/10/05 Á When initialized
EZLINFORMTIME                       16:17:49
EZLSMTPNAME                         SMTP     Á Name of SMTP STC
……….. .
EZLINF_POLICY.2                     SAMBROWNVIAPHONE …..
EZLINF_POLICY.1                     SAMBROWNVIAMAIL  …..
EZLINF_POLICY.0                     …
```

Browse NETLOGA, and scroll right to about column 123 on the line with the EZLINF_POLICY.n entry that you are trying to notify. You will see the ROUTE that INFORM is using, for example:

```
EZLINF_POLICY.1  SAMBROWNVIAMAIL  ….. EMAIL  SAMBROWN@US.IBM.COM
```

Correct any errors, and execute the INFORMTB command to re-initialize the INFORM policy without recycling NetView:

```
INFORMTB EZLINSMP
```

3.5.2 Trying the workstation scenario

To run the workstation scenario:

1. Determine a workstation where you can change to the offline state for some number of minutes. Our scenario uses 12 minutes.
2. Invoke TWS for z/OS workstation management to set the workstation to offline status:
   - On a 3270 screen:
     - Select TWS option **5.5** (from ISPF).
     - Select the workstation.
     - Set it offline.
   - On a Job Scheduling Console:
     - Select **Default Plan Lists**.
     - Select **Status of All Workstations**.
     - Right click and select the workstation.
     - Set it Offline.

3. When the workstation changes state to offline, the automation table entry for message EQQWL10W and OFFLINE status will invoke the TWSWSDN REXX routine.
4. TWSWSDN calls the CHRON timer command to invoke INFORM after the interval specified in the TWSWSDN wait4cmd variable.

   Note: You need to keep the workstation offline for at least as long as that interval.

5. INFORM is passed the TWSWSDN informwho variable as part of the CHRON command,
6. When the time interval elapses, the INFORM routine sends e-mail to the address that is specified in your INFORM Policy ROUTE.

   The e-mail looks similar to this:

   ```
   EQQWL10W WORK STATION LN02, HAS BEEN SET TO OFFLINE  STATUS  AT 16.40.27.79
   ```

   Attention: If the workstation comes online before the TWSWSDN wait4cmd time elapses, a message EQQWL10W for the ONLINE status occurs and that triggers the TWSWSUP routine, which cancels the CHRON command and e-mail is not sent.

7. Now, set the status of the workstation online, and e-mail should be received from NetView that looks similar to this:

   ```
   EQQWL10W WORK STATION LN02, HAS BEEN SET TO ACTIVE   STATUS AT 16.55.20.13
   ```

Troubleshooting tips

If the e-mail does not arrive, then check the following.

- MVS SYSLOG should contain a TWS for z/OS EQQWL10W message for the failed workstation:

  ```
  16.40.27 EQQWL10W WORK STATION LN02, HAS BEEN SET TO OFFLINE  STATUS
  ```

- If the EQQWL10W message is not in SYSLOG, then edit the TWS for z/OS message library, SEQQMSG0 member EQQWL1, and check that `WTO=YES` is set for message EQQWL10.

  ```
  EQQWL10 ' ' WTO=YES                <== add WTO=YES to expose the message
  W WORK STATION &WSID, HAS BEEN SET TO &STAT STATUS'
  ```

- NetView NETLOGA should contain the messages: shown in Example 3-14.

Example 3-14 NetView NETLOGA messages

```
16:40:27 CNM493I INFORMSM : (NO SEQ) : TWSWSDN EQQWL10W WORK STATION LN02, HAS
BEEN SET TO OFFLINE  STATUS
EZL460I EMAIL ACTION WAS SUCCESSFULLY ISSUED FOR POLICY SAMBROWNVIAMAIL BY
```

- If the CNM493I message is not in NETLOG for this event, then the automation table is not trapping the EQQE036I message. Check that the z/OS MPFLST allows automation for EQQ* messages.

  Review your automation table. Refer to the *Tivoli NetView for z/OS 5.1 Automation Guide*, GC31-8851 for guidance.

- If the EZL460I message is missing, then look in NETLOG for messages that describe what failed.
- If the EZL460I message is in NETLOG and the mail still has not arrived, then review the INFORM Policy definitions to insure that the ROUTE is specified correctly for job failure.

4

Integrating with OPC Automation extension of System Automation

This chapter describes the scenario that we used to demonstrate the integration of Tivoli Workload Scheduler for z/OS (TWS for z/OS) with Tivoli System Automation for z/OS. TWS for z/OS uses the OPC Automation extension to integrate with System Automation for z/OS. The scenario used in this chapter is set up on a single z/OS system. This system runs both the TWS for z/OS controller and System Automation with the OPC Automation extension. This system is part of a Parallel Sysplex although none of the Parallel Sysplex functions are used.

This chapter includes detailed information about the configuration that we used, a step-by-step description of how TWS for z/OS and System Automation for z/OS communicate with each other through the OPC Automation extension, and a step-by-step discussion of the implementation of the interface.

Note: This chapter does not cover the use of the System Automation z/OS OPC dialog.

4.1 The TWS for z/OS and System Automation for z/OS configuration

The TWS for z/OS and System Automation for z/OS (referred to as System Automation in this book) configuration that we used in our scenario ran on a single z/OS system called SC64. SC64 is part of a 4-way Parallel Sysplex. The TWS for z/OS controller and all trackers were connected using XCF. It is possible to run System Automation on all systems in the Parallel Sysplex. However, for our scenario, we used only one system. TWS for z/OS used one of its own exits and the NetView program-to-program interface to communicate with System Automation.

The diagram in Figure 4-1 shows the configuration for our scenario.

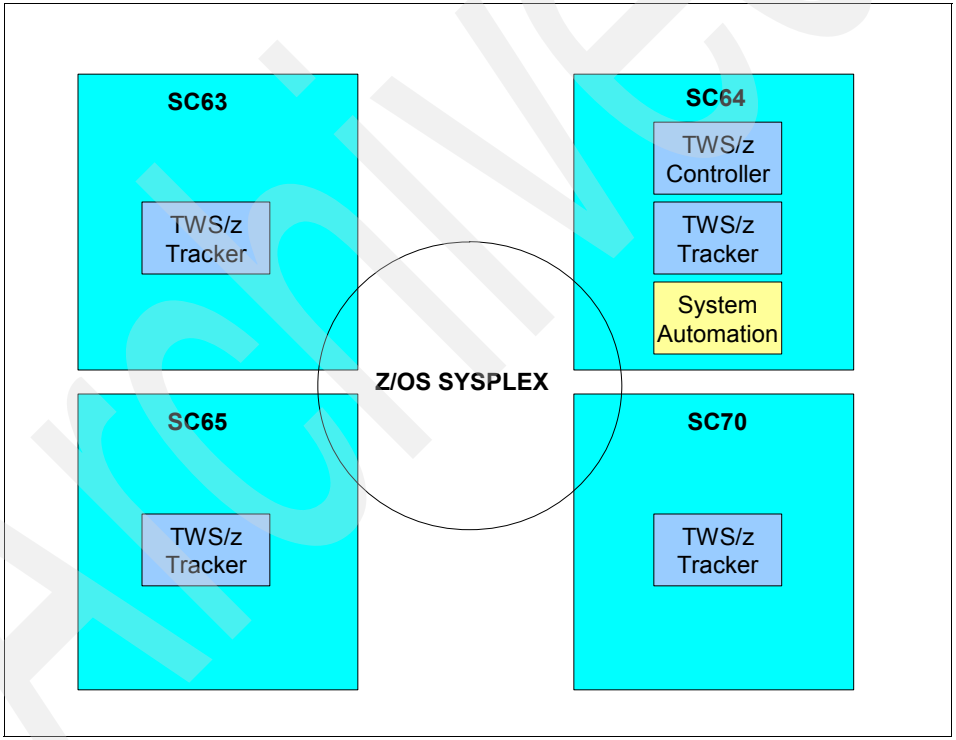

Figure 4-1 TWS for z/OS and System Automation configuration on SC64

4.1.1 How TWS for z/OS communicates with System Automation

Figure 4-2 illustrates how requests flow from TWS for z/OS to System Automation and how responses return to TWS for z/OS. The diagram shows TWS for z/OS initiating the start of a CICS subsystem.

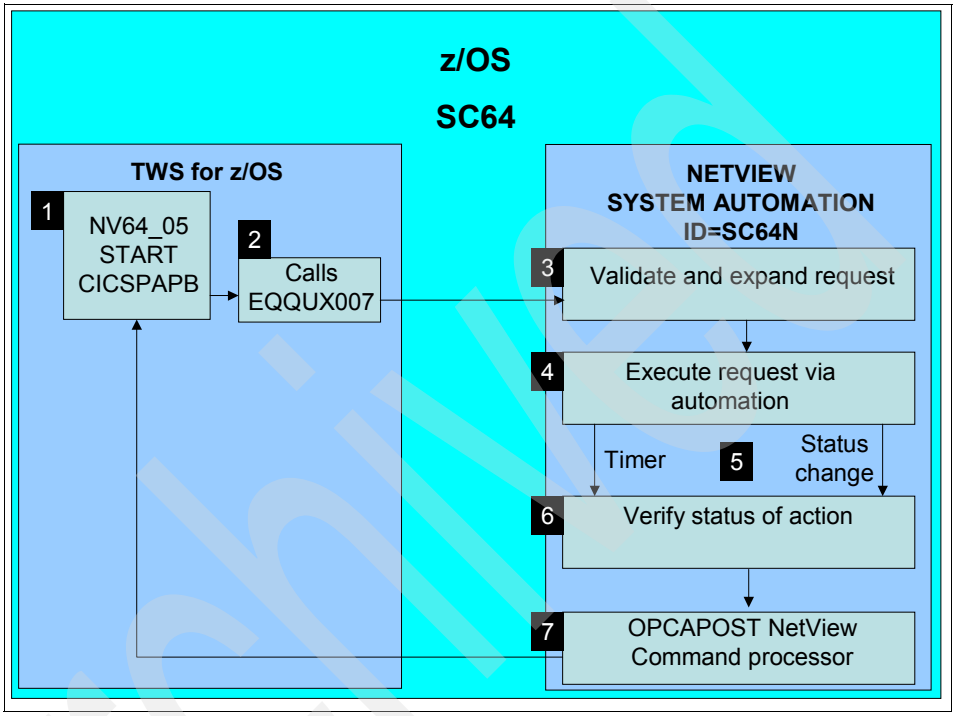

Figure 4-2 Step-by-step process request flow

The steps in this process request flow are as follows:

1. The CICSPAPB operation at the NetView workstation NV64 is submitted by TWS for z/OS. The text field for the operation has START in it.
2. The submission of the operation causes the System Automation copy of EQQUX007 to be called. EQQUX007 identifies that this is a request for System Automation based on the first two characters, NV, of the workstation. It uses the NetView PPI to pass the request to System Automation.
3. The request is received, verified, and expanded by NetView. The workstation name is mapped to a NetView domain. The START is expanded via the CICSPAPB STARTUP policy item.
4. The request is then passed to System Automation to execute the request. In this case it is an z/OS start command.

5. System Automation tracks the request by either a timer or status change.
6. When the timer has expired or the status changes, the success is verified.
7. The result is passed back to the TWS for z/OS operation using the TWS for z/OS EQQUSINT program routine. The status of the operation is changed to `complete` or `error` based on the result.

Our scenario also describes how System Automation can change the status of a Tivoli Workload Scheduler special resource that is based on the status of an System Automation resource. This change is driven by System Automation rather than the request originating in Tivoli Workload Scheduler. Figure 4-3 shows how a status change in an System Automation resource flows through to a change in the corresponding Tivoli Workload Scheduler special resources.

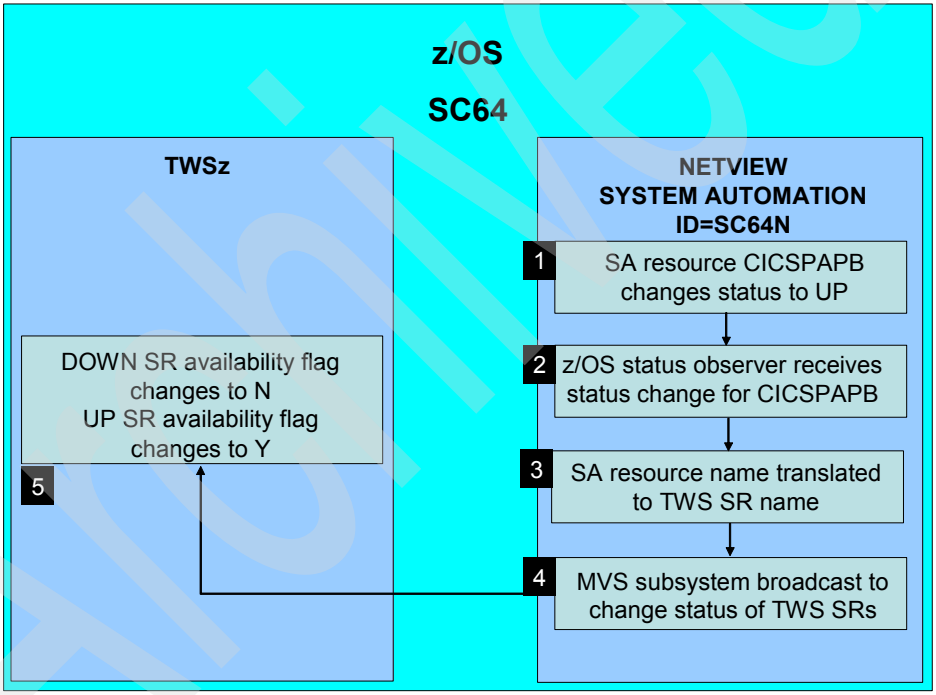

Figure 4-3 Step-by-step process for special resource status change

The steps in this special resource status change are as follows:

1. The System Automation resource CICSPAPB has a status change to `UP`.
2. The z/OS status observer non-MVS subsystem is notified of the status change as it is registered with the automation manager.

3. The name of the System Automation resource is translated into two Tivoli Workload Scheduler special resources:
   - One with an UP suffix
   - One with a DOWN suffix

   For example, if the name of the System Automation resource is CICSPAPB/APL/SC64, the corresponding TWS special resources are:
   - ING.SC64.APL.CICSPAPB.DOWN
   - ING.SC64.APL.CICSPAPB.UP
4. The z/OS status observer subsystem performs a subsystem broadcast which contains the special resource status change requests to all TWS for z/OS subsystems.
5. TWS for z/OS receives the status change request for the two corresponding special resources and processes them.

   Because CICSPAPB has a status of UP, the TWS for z/OS UP special resource availability flag is changed to `Y` and the DOWN special resource availability flag is changed to `N`.

Other interfaces not covered in this chapter

System Automation provides an interface through a NetView session which can access TWS for z/OS data. The user logs on to NetView and uses the System Automation z/OS dialogs. This interface uses various TWS for z/OS program routines to read and modify some TWS data. It does not have access to all TWS for z/OS data.

This chapter does not discuss this interface because it is activated automatically when the OPC Automation extension has been activated. You can find information about how to activate this automation extension in this chapter.

4.1.2 Configuring TWS for z/OS

To configure TWS for z/OS, you need to:

1. Provide access to System Automation and NetView libraries.
2. Activate EQQUX007.
3. Set up the TWS for z/OS workstation.

Providing access to System Automation and NetView libraries

TWS for z/OS must have access to modules in two libraries:

- The System Automation for z/OS SINGMOD1 target library
- The NetView CNMNETV target library.

One of the more important modules in this library is EQQUX007. This module uses TWS for z/OS Exit 7.

Both of these libraries are either placed in the LINKLST concatenation or placed on the STEPLIB DD statement of the TWS for z/OS controller.

Restriction: The interface from TWS for z/OS to the OPC Automation extension of System Automation for z/OS uses EQQUX007. OPC Automation provides its own copy EQQUX007. Therefore, you cannot use EQQUX007 as the module name for other uses of the exit.

To allow you to continue to use EQQUX007 for other purposes, the OPC Automation copy of EQQUX007 calls modules UX007005 to UX007009 after it has finished executing. If you are using EQQUX007, then rename your existing EQQUX007 to any name in the range UX007005 to UX007009.

We used EQQUX007 for the TWS for z/OS to INFOMAN interface. We renamed our existing EQQUX007 to UX007005. Our exit continued to operate normally.

Activating EQQUX007

EQQUX007 is used by the TWS for z/OS controller. Activating the exit is simple. It is activated by adding the EXITS initialization statement to the TWS for z/OS controller parameter member.

Note: If the EXITS initialization statement is not specified in the TWS for z/OS controller initialization statements, the TWS for z/OS controller tries to load a number of the TWS for z/OS exits automatically, including EQQUX007. If the exit is not found, it issues a message in the MLOG and continues processing. The automatic load of exits can be prevented by coding `EXITS CALL`*`xx`*`(NO)` where *xx* is the exit number. We recommend that you do this for all exits that you do not intend to use.

Example 4-1 shows our EXITS initialization statement.

Example 4-1 Exits initialization statements

```
EXITS    CALL00(NO)
         CALL01(NO)
         CALL02(NO)
         CALL03(NO)
         CALL07(YES)
         CALL09(NO)
         CALL11(NO)
```

When you are ready to activate the EQQUX007, shutdown and restart the TWS for z/OS controller.

Setting up the TWS for z/OS workstation

TWS for z/OS uses a workstation to communicate with System Automation. The format of the workstation name must be NV*xx*, where the first two characters are always NV and *xx* are characters that you choose.

The workstation we uses was called NV64. We used 64 because these two characters are the last two characters of the system name where the System Automation with which we communicated was running.

Important: Only use this format for workstations which communicate with System Automation. Using this format for other workstations can cause unpredictable results.

To create the workstation:

1. Use option 1 - Database of the Operations Planing and Control ISPF primary options menu as shown in Figure 4-4 on page 104.

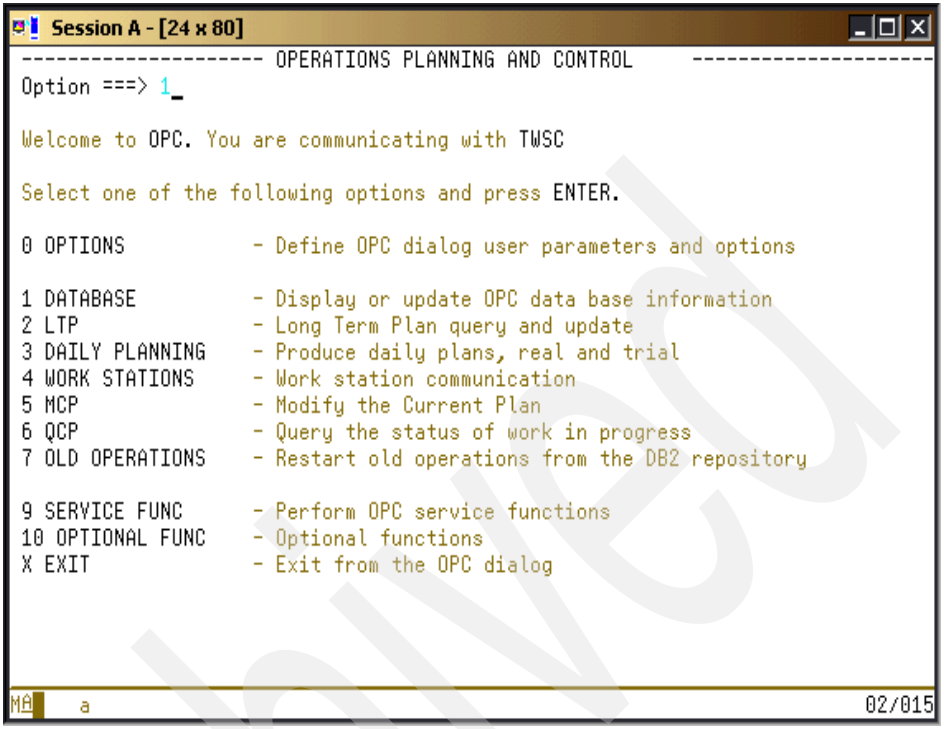

Figure 4-4 TWS for z/OS ISPF primary options menu

The MAINTAINING OPC DATA BASES panel displays as shown in Figure 4-5 on page 105.

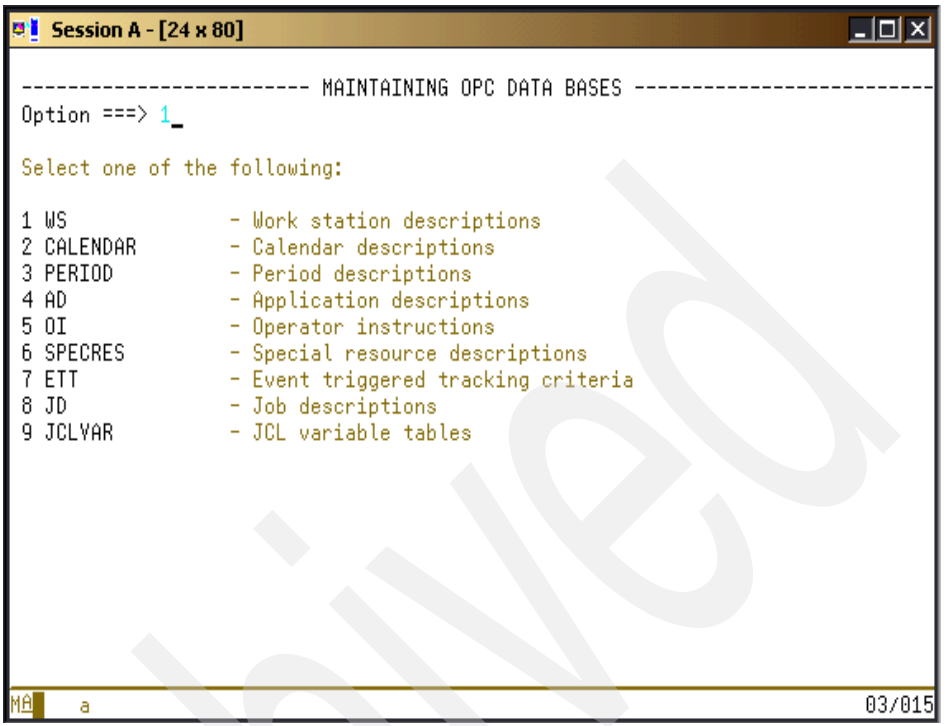

Figure 4-5 MAINTAINING OPC DATA BASES panel

2. Choose option **1 - WS**. The MAINTAINING WORK STATION DESCRIPTIONS panel that is shown in Figure 4-6 on page 106 displays.

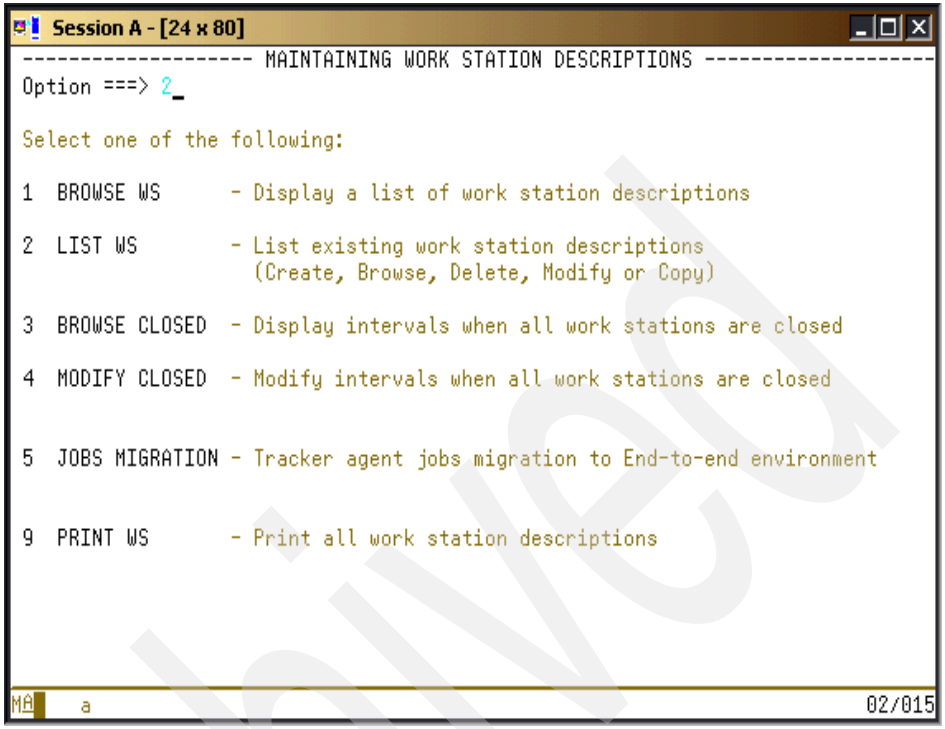

Figure 4-6 MAINTAINING WORK STATION DESCRIPTIONS panel

3. Choose **Option 2 - LIST WS**. The SPECIFYING WORK STATION LIST CRITERIA panel that is shown in Figure 4-7 on page 107 displays.

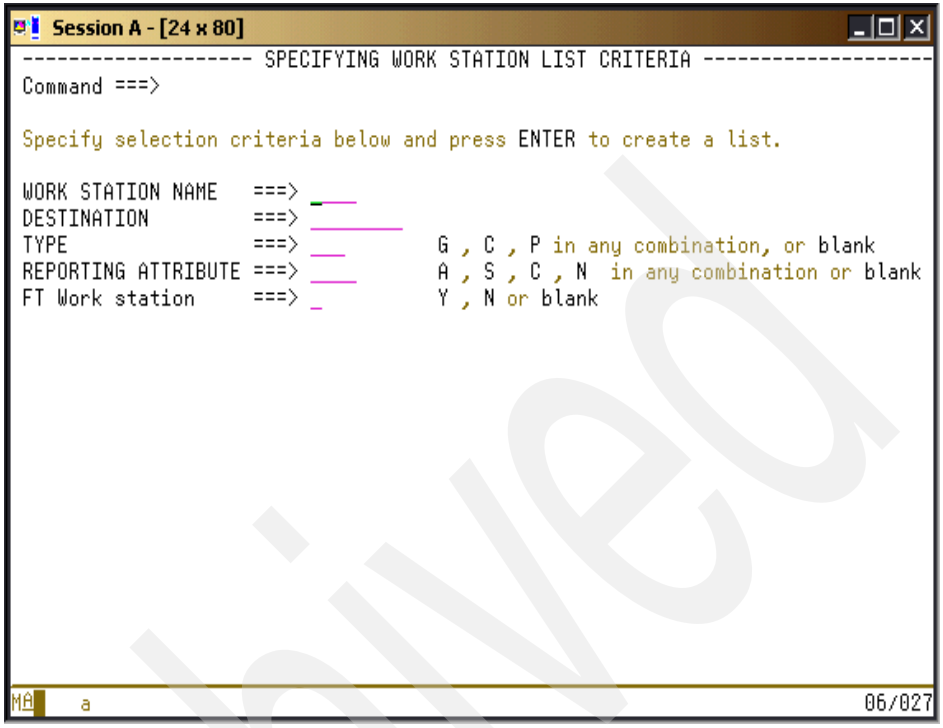

Figure 4-7 SPECIFYING WORK STATION LIST CRITERIA

4. All fields should be blank. Press Enter and the LIST OF WORK STATION DESCRIPTIONS panel that is shown in Figure 4-8 on page 108 displays.

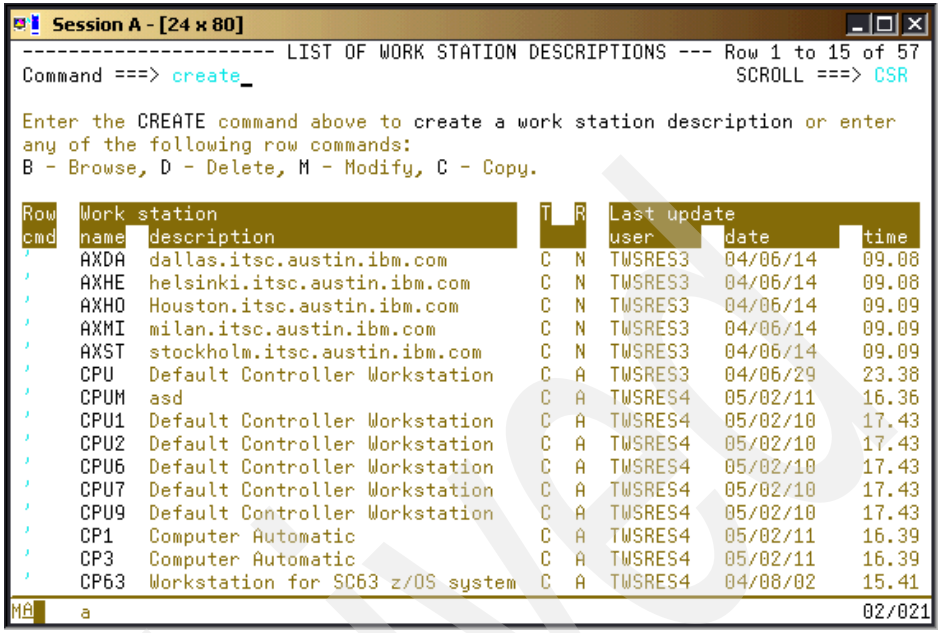

Figure 4-8 LIST OF WORK STATION DESCRIPTIONS panel

5. This panel lists all defined workstations to TWS for z/OS which RACF permits you to see. Type **create** on the command line and press Enter. The GENERAL CREATING INFORMATION ABOUT A WORK STATION panel that is shown in Figure 4-9 on page 109 displays.

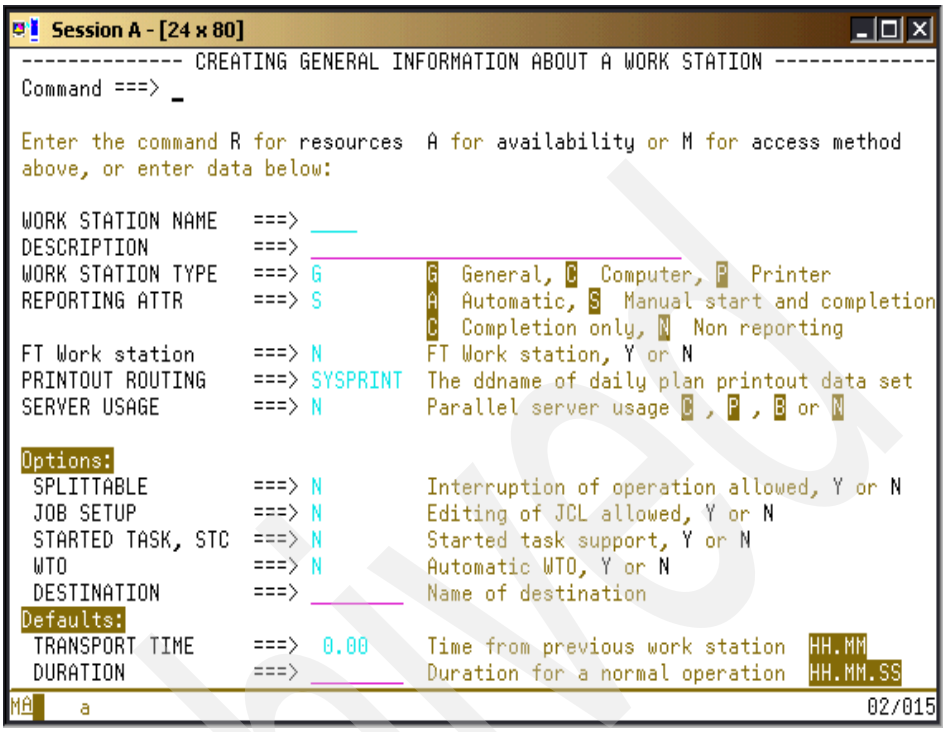

Figure 4-9 CREATING GENERAL INFORMATION ABOUT A WORK STATION panel

6. The CREATING GENERAL INFORMATION ABOUT A WORK STATION has the default values. You need to change some of these values to correctly define the work station to communicate with System Automation. Table 4-1 shows the data that you need to enter.

Table 4-1 Data required for work station to communicate with System Automation

Field Name	Description	Value
WORK STATION NAME	4 character Work Station name - Must start with NV	NVxx
DESCRIPTION	Free form description	
WORK STATION TYPE	Must be General	G
REPORTING ATTR	Must be Automatic	A
FT Work station	Must be N	N
PRINTOUT ROUTING	Leave as default	SYSPRINT

Field Name	Description	Value
SERVER USAGE	Site dependant - check with TWS for z/OS administrators	N
SPLITTABLE	Must be no	N
JOB SETUP	Must be no	N
STARTED TASK, STC	Must be no	N
WTO	Must be no	N
DESTINATION	Must be blank	
TRANSPORT TIME	Use default of zero	00.00
DURATION	Leave blank	

The name of each workstation is mapped to a NetView domain using the Workstations Domain Policy object entry (the ODM entry). The ODM entry is found in the System Automation ISPF dialog from the Entry Type Selection menu option 20 - PRD Product Automation and then option 33 - ODM Workstation domainID. The creation of the ODM entry and how to get there is explained fully in the 4.1.3, “Configuring System Automation” on page 112.

After you have entered the data for the work station, it should look similar to Figure 4-10. The last two characters of your workstation name might differ.

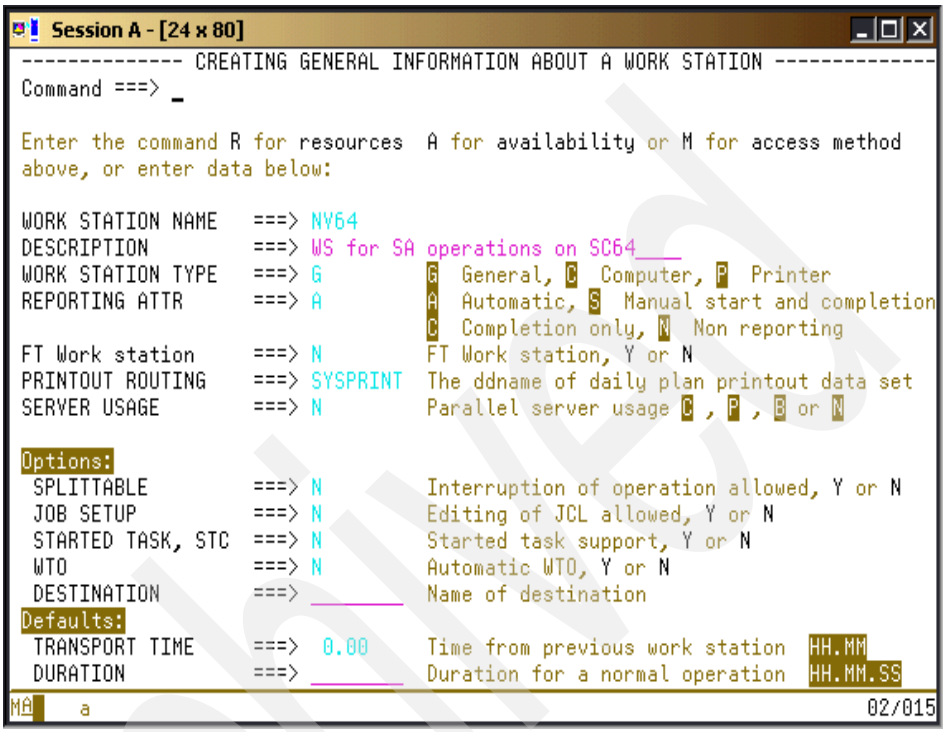

Figure 4-10 Data for creating work station NV64

7. Press **PF3** to complete the creation process and to return to the LIST OF WORK STATION DESCRIPTIONS panel. The message `WS CREATED` should appear in the top right-hand corner of the panel, as shown in Figure 4-11 on page 112.

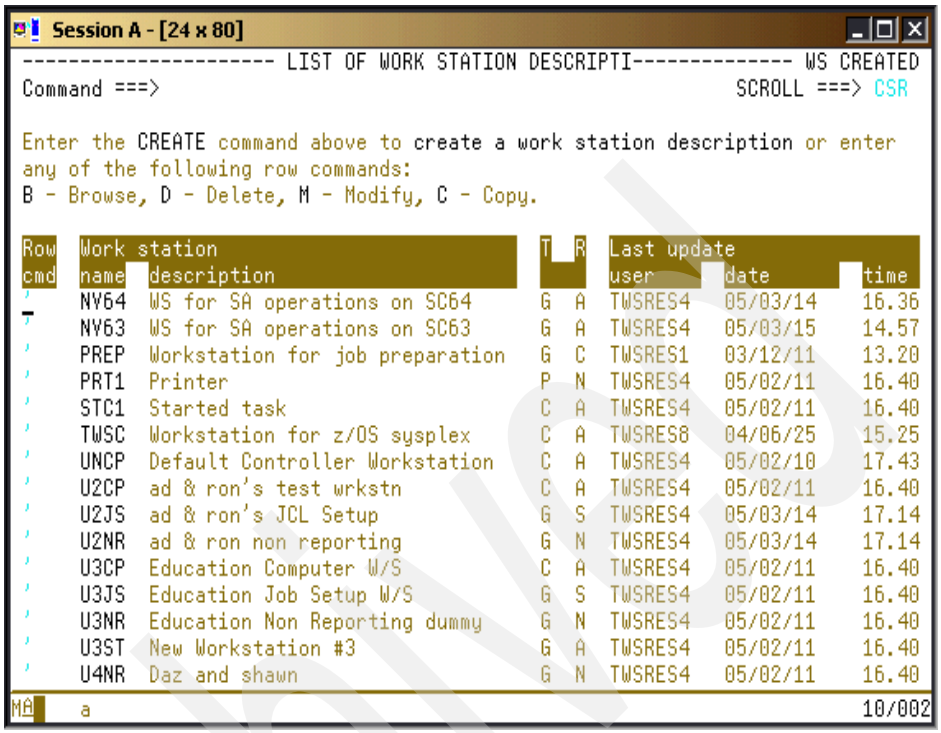

Figure 4-11 Work station NV64 WS CREATED message

8. The workstation has now been created. To interface with System automation running on other systems, define more workstations.

4.1.3 Configuring System Automation

The System Automation part of the configuration is performed in the System Automation ISPF dialog. This dialog is used to create the automation policy that System Automation uses to determine how to automate, monitor, and control resources in the enterprise. The dialog is also used to build the automation policy into a data set. The automation policy is then activated.

The sample SYSPLEX Policy Database (PDB) includes some of the definitions that are required for OPCAO. You can modify these definitions, or you can use them as a guide to create your own.

There are three different types of definitions to set up in the automation policy:

1. Define Automation Operators that are required by TWS for z/OS.

   These are tasks that perform work that operators normally perform (such as entering z/OS commands and monitoring write to operator's).

2. Define the Application entries.

   There are four mandatory application entries and one optional:

   a. Define the TWS for z/OS controller.

   b. Define the TWS for z/OS trackers.

   c. Define the TWS PPI request server.

   d. Define the new TWS for z/OS batch command server. (Optional)

      This subsystem allows NetView PPI requests from batch jobs to be processed by System Automation.

      If you do not use the batch command interface, then there is no need to define this application

   e. Define the TWS for z/OS observer subsystem to update the status of special resources in TWS for z/OS.

      This subsystem takes the status of a subsystem in System Automation and requests a change in status of a corresponding special resource in TWS for z/OS.

3. Define the Product Automation entries for OPCAO.

   There are four OPC components to be defined:

   a. OPC (TWS) System details

   b. Controller details

   c. Special Resources

   d. Workstation domainID

      This is how System Automation knows which NetView domain (which System Automation on which system) processes the request from TWS for z/OS.

These steps are documented in the *OPC Automation Programmer's Reference and Operator's Guide*, SC33-7046. Additional information is available in *System Automation for z/OS Defining Automation Policy*, SC33-7039. The remainder of this section consolidates information from these two sources.

We defined all definitions in our scenario. We used the samples provided as a guide. These samples all use the name *OPC*. We created our own using the name *TWS*.

To perform our definitions, we start the System Automation ISPF dialog. Each definition begins at the Entry Type Selection Menu. To reach this menu, start the System Automation ISPF dialog. The System Automation z/OS 2.3 Customization Dialog Primary Menu that is shown in Figure 4-12 displays.

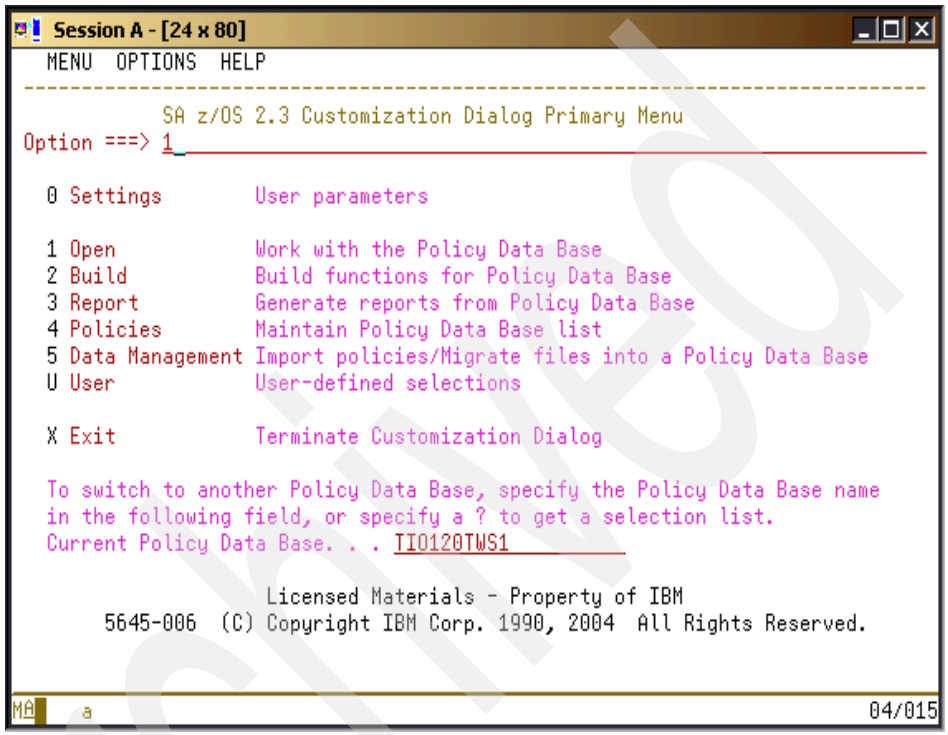

Figure 4-12 System Automation z/OS 2.3 Customization Dialog Primary Menu

On this menu, specify the Current Policy Data Base. Our database is called TIO120TWS1.

Select option **1 - Open**. The Entry Type Selection menu that is shown in Figure 4-13 on page 115 displays.

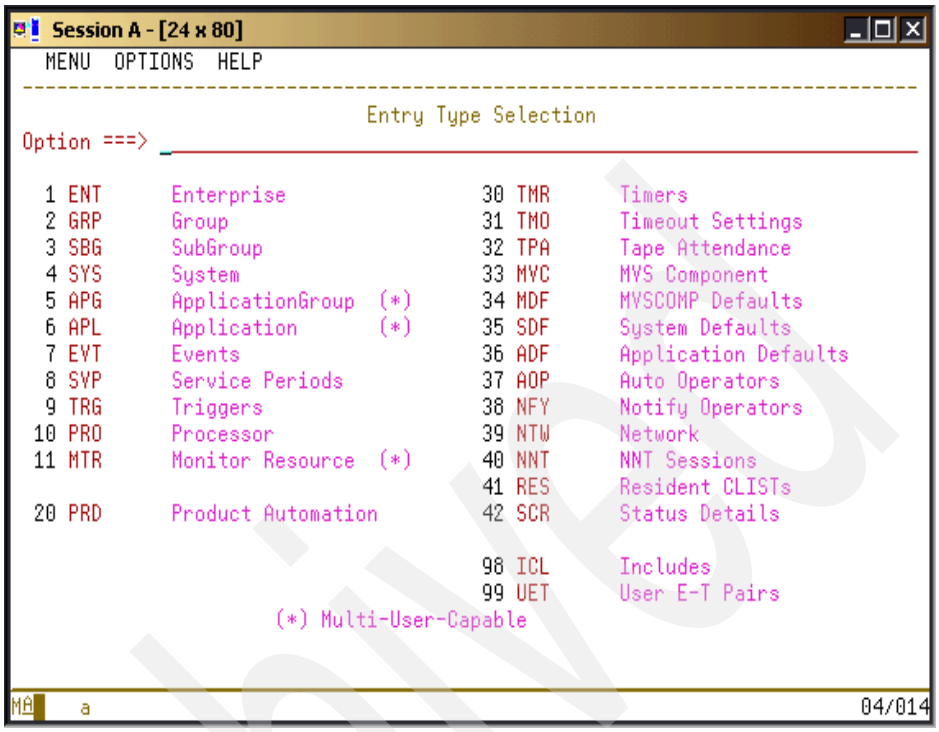

Figure 4-13 Entry Type Selection panel

The other definitions performed in this chapter begin from this panel.

Defining Automation Operators

TWS for z/OS requires three automation operators:

- **OPCAMSTR**, which is the main automation operator that is required to enable OPCAO. This operator processes EVJ* messages.
- **OPCAOPR2**, which is the TWS for z/OS request execution operator.
- **OPCAMDR**, which is the batch command execution operator.

The sample SYSPLEX PDB includes these definitions.

To create the automation operators:

1. Select option **37 AOP Auto Operators** on the Entry Type Selection panel (Figure 4-13 on page 115). The Entry Name Selection panel that is shown in Figure 4-14 displays.

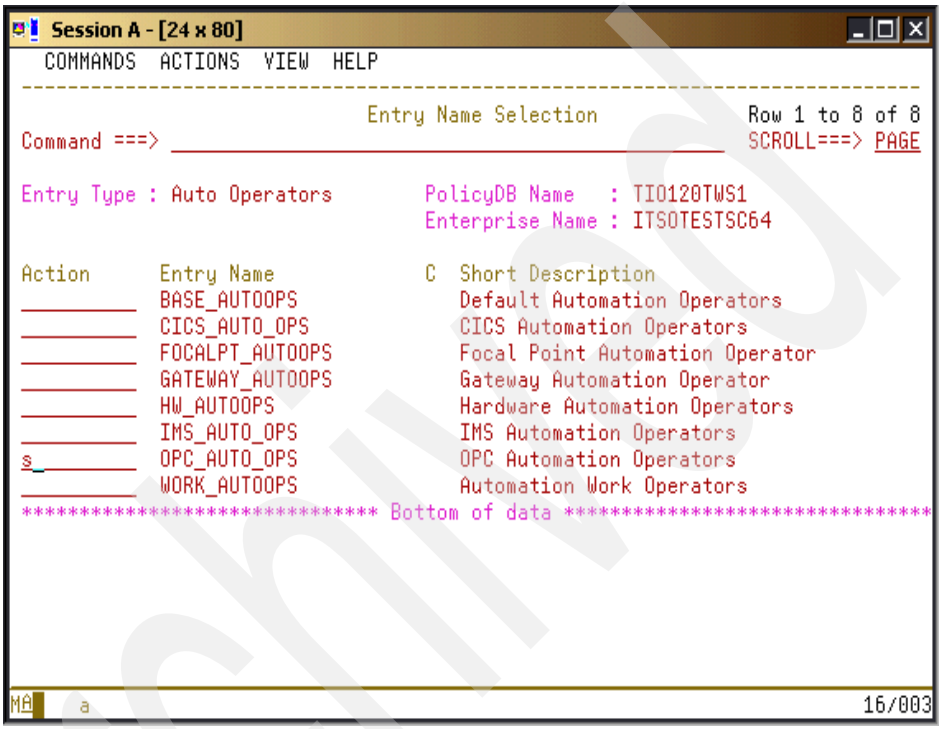

Figure 4-14 Auto Operators Entry Name Selection panel

2. Place an s in the Action column next to OPC_AUTO_OPS and press Enter. The Policy selection panel that is shown in Figure 4-15 on page 117 displays.

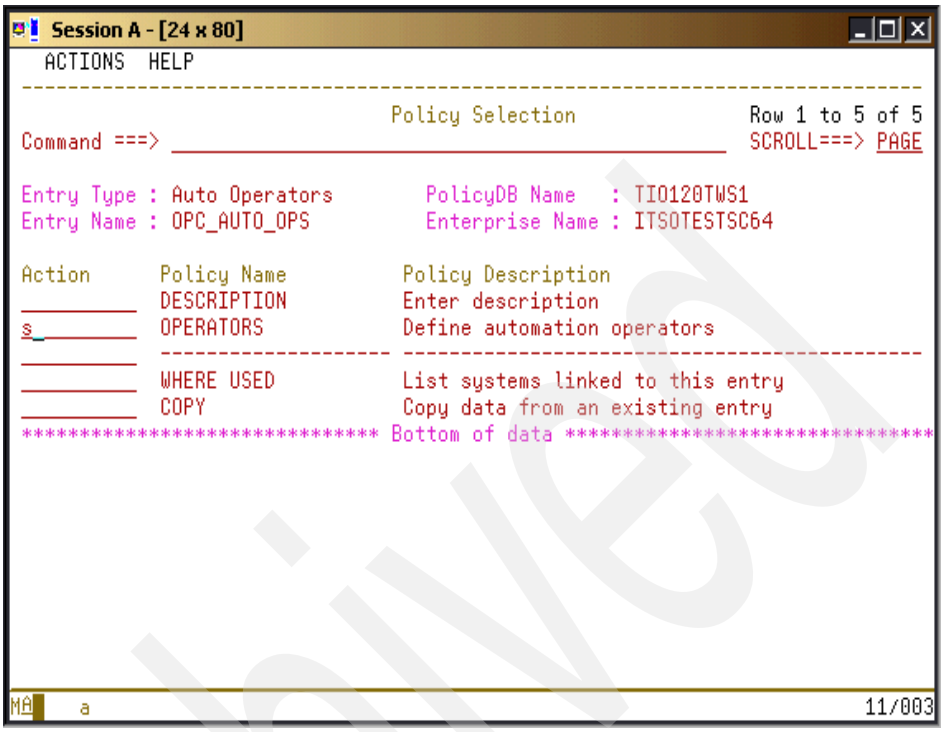

Figure 4-15 OPC_AUTO_OPS Policy Selection panel

3. Place an `s` in the action column next to `OPERATORS` and press Enter. The Automation Operator Definitions panel that is shown in Figure 4-16 on page 118 displays.

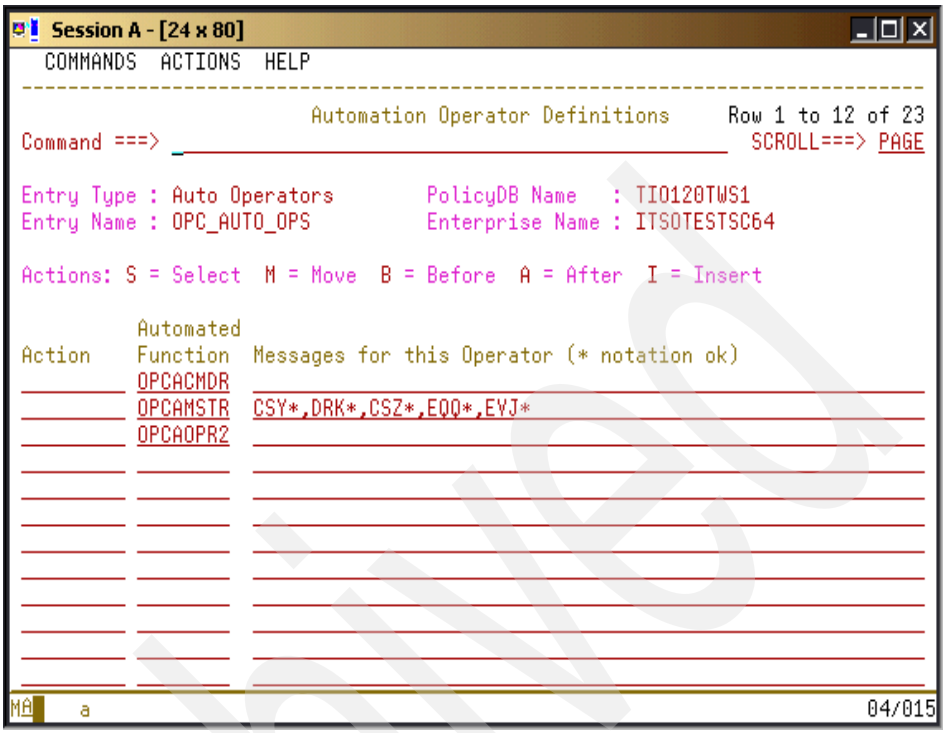

Figure 4-16 OPC_AUTO_OPS Automation Operator Definitions

4. Press **PF3** to save and to return to the Policy Selection panel. Place an s next to WHERE USED and press Enter. The Where Used panel that is shown in Figure 4-17 on page 119 displays.

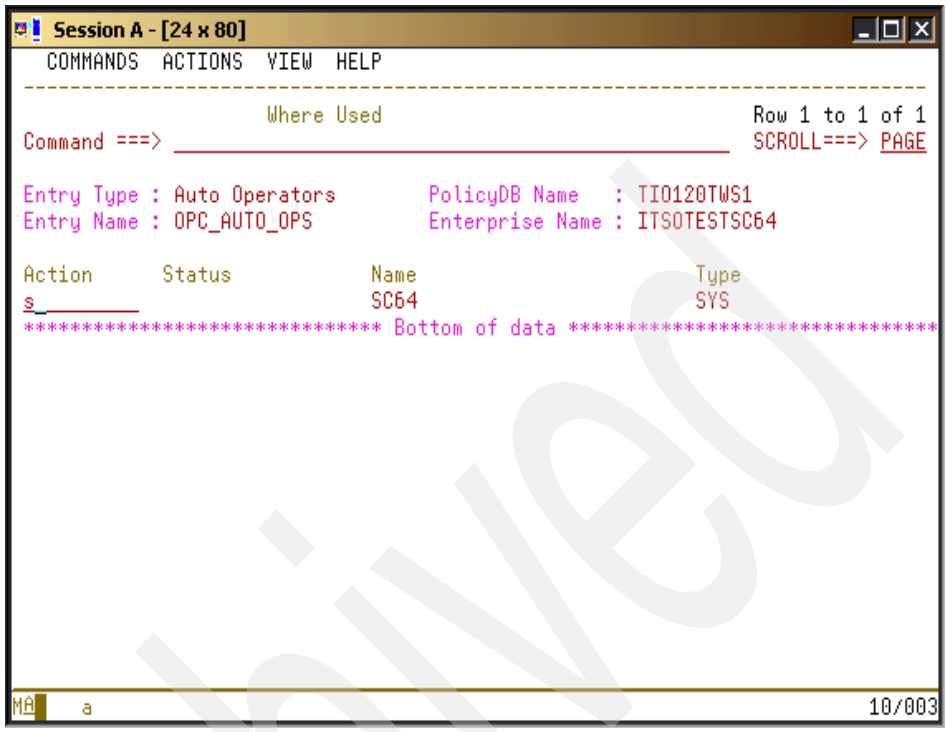

Figure 4-17 OPC_AUTO_OPS Where Used panel

5. The Where Used panel shows the name of systems that are defined in your automation policy. Because we have defined only one system, we have one choice. You can have more than one choice. Place an `s` in the action column of each system where these Auto Operators are to be used and press Enter. The Status field changes to `SELECTED` as shown in Figure 4-18 on page 120.

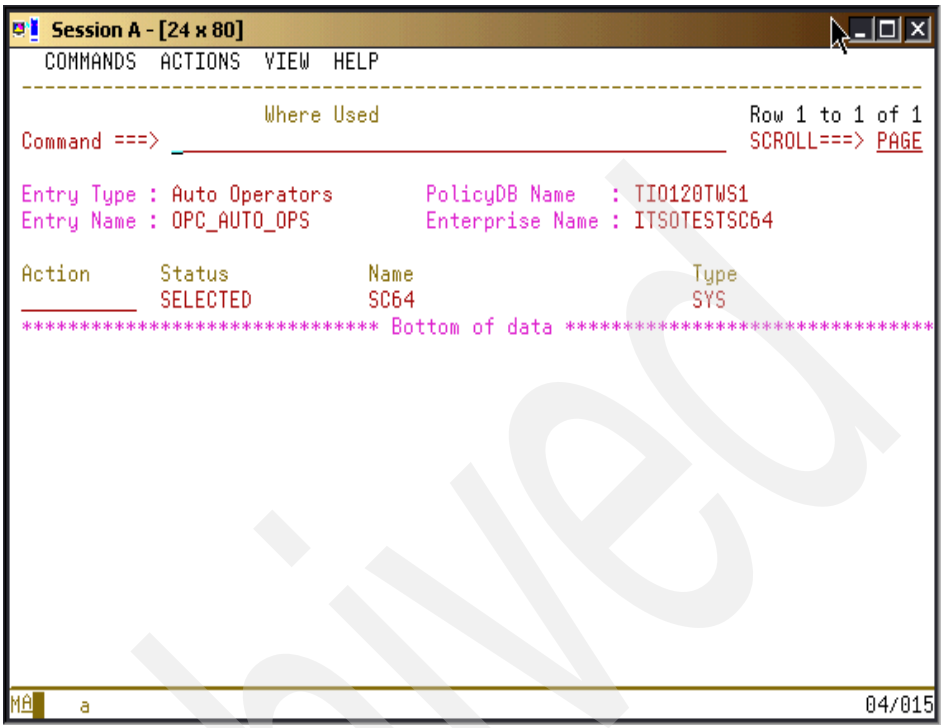

Figure 4-18 OPC_AUTO_OPS panel with system SC64 SELECTED

6. Press **PF3** to save the definitions. Continue to press **PF3** until you return to the Entry Type Selection menu. This completes creation of the automation operators.

   The sample SYSPLEX PDB includes these definitions.

Defining the Application entries

Subsystems are defined as Applications to System Automation. Each application must be linked to an application group. Many sites define one application group per system and then link all the applications that are defined for that system to the one application group. A group called TWS_SC64_APPL_GROUP was defined for our scenario. This section discusses:

- Defining the TWS for z/OS controller
- Defining the TWS for z/OS tracker
- Defining the TWS for z/OS PPI request server
- Defining the new Tivoli Workload Scheduler for z/OS batch command server (optional)
- Defining the z/OS status observer subsystem

To define each application, enter option **6 APL** - Application on the Entry Type Selection panel as shown in Figure 4-13 on page 115. The Entry Name Selection panel that is shown in Figure 4-19 displays.

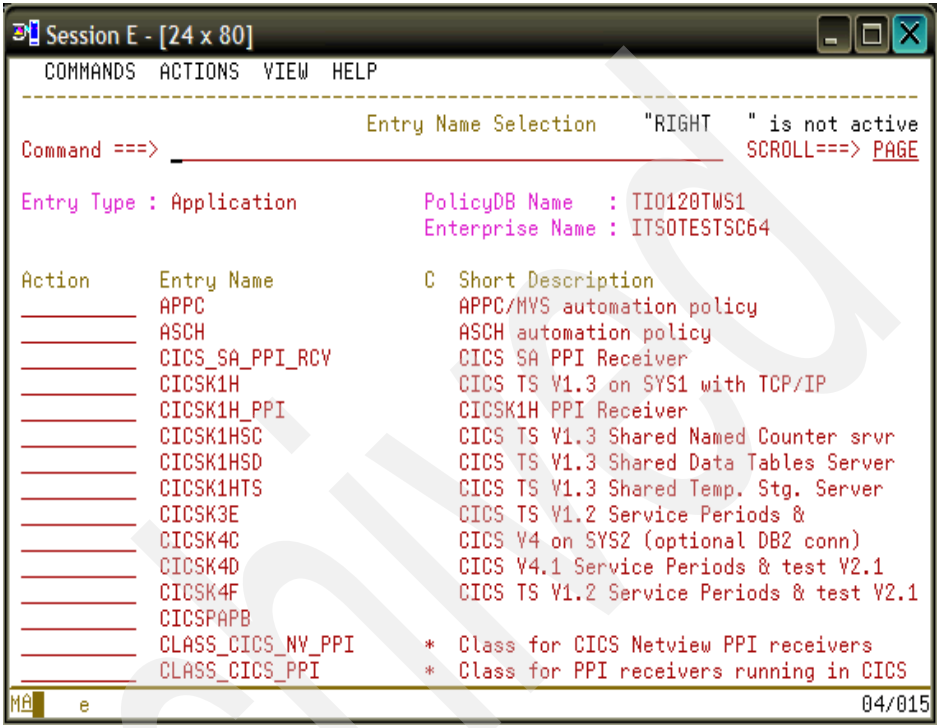

Figure 4-19 Application Entry Name Selection panel

Defining the TWS for z/OS controller

To define the TWS controller:

1. Type **new** on the command line of the Entry Name Selection panel that is shown in Figure 4-19 on page 121 and press Enter. The Define New Entry panel displays. On this panel, enter data into the fields as shown in Table 4-2.

Table 4-2 Define New Entry panel data

In this field	Enter this data
Name	Our controller is called TWSC, so use this as the name.
Subsystem Name	We use the same as the Name field for convenience. In our scenario, this is TWSC.
Object Type	INSTANCE
Application Type	OPC
Subtype	CONTROLLER
Job Type	MVS
Job Name	The same as the Subsystem Name, which in our scenario is TWSC.
Short description	(Optional) Enter a short description.

The completed panel is shown in Figure 4-20 on page 123.

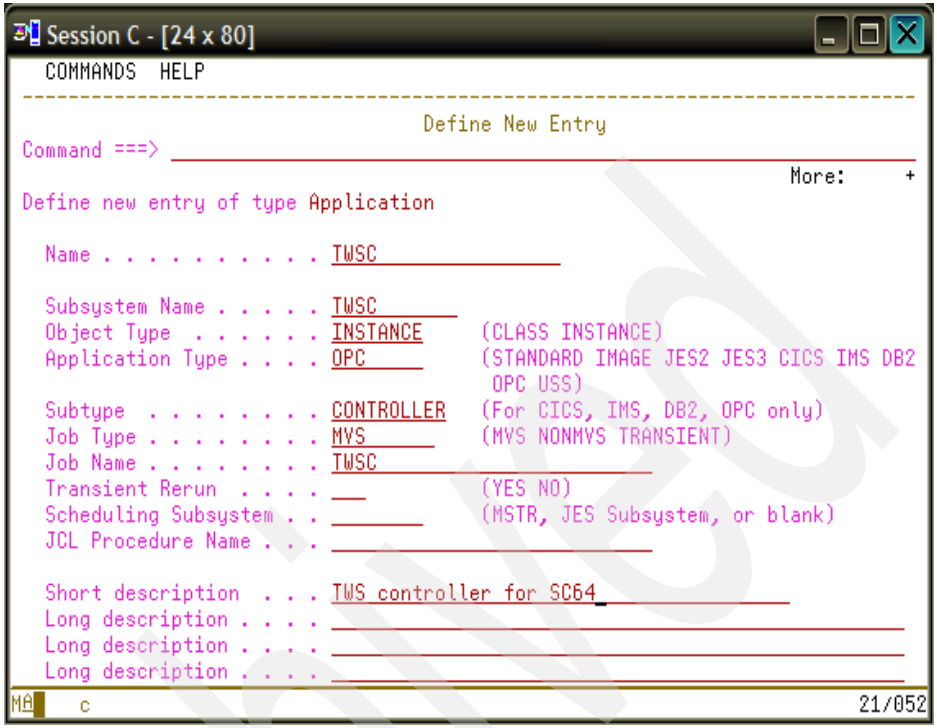

Figure 4-20 Define New Entry panel for TWS for z/OS controller

2. After you enter the data, press **PF3**. The top part of the Policy Selection panel that is shown in Figure 4-21 on page 124 displays.

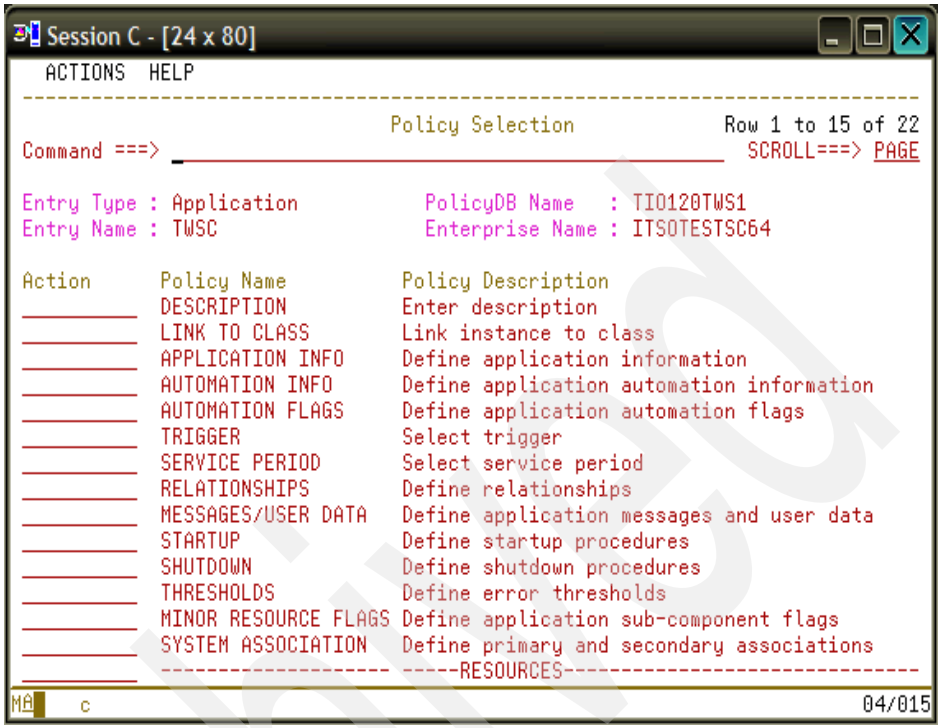

Figure 4-21 Top section of Policy Selection panel for TWS for z/OS controller

The Policy selection panel has two parts. Use the scroll key to see the bottom part of the panel. The bottom part of the panel is shown in Figure 4-22 on page 125.

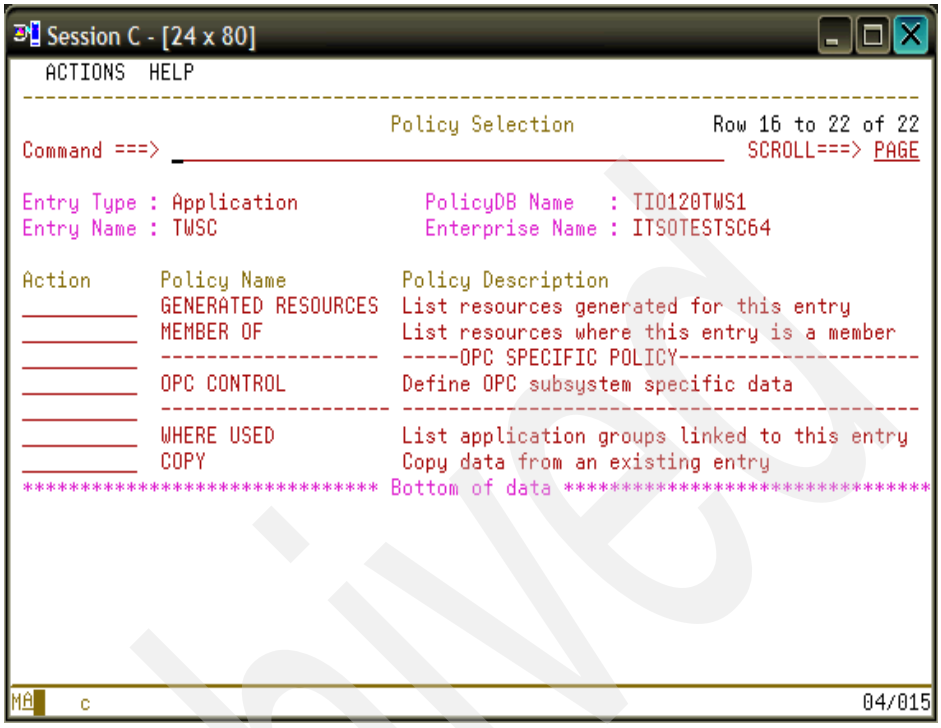

Figure 4-22 Bottom part of Policy Selection panel for TWS for z/OS controller

There are a number of parts to the TWSC policy. The Policy Selection panel has a number of different Policy Names on it. Some of these need to be updated for TWS for z/OS controller.

3. To select a policy, place an s in the Action column and press Enter. The Policy Names that need to be updated are:
   - OPC CONTROL
   - WHERE USED
4. Select the **OPC CONTROL** policy. The OPC Control Specifications panel that is shown in Figure 4-23 on page 126 displays.

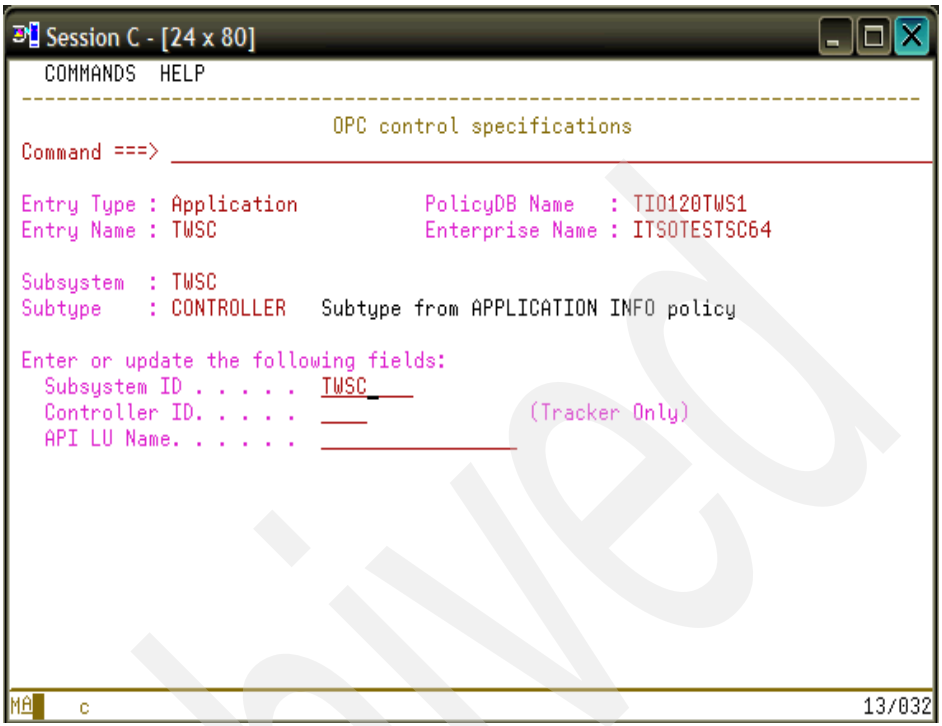

Figure 4-23 OPC Control Specifications panel for TWS for z/OS controller

5. Enter the subsystem name of your TWS for z/OS controller in the Subsystem ID field. In our scenario, this name is TWSC. Press **PF3** to return to the Policy Selection panel.
6. Select the **WHERE USED** policy. The Where Used panel that is shown in Figure 4-24 on page 127 displays. You can scroll this panel to see all the names that are displayed.

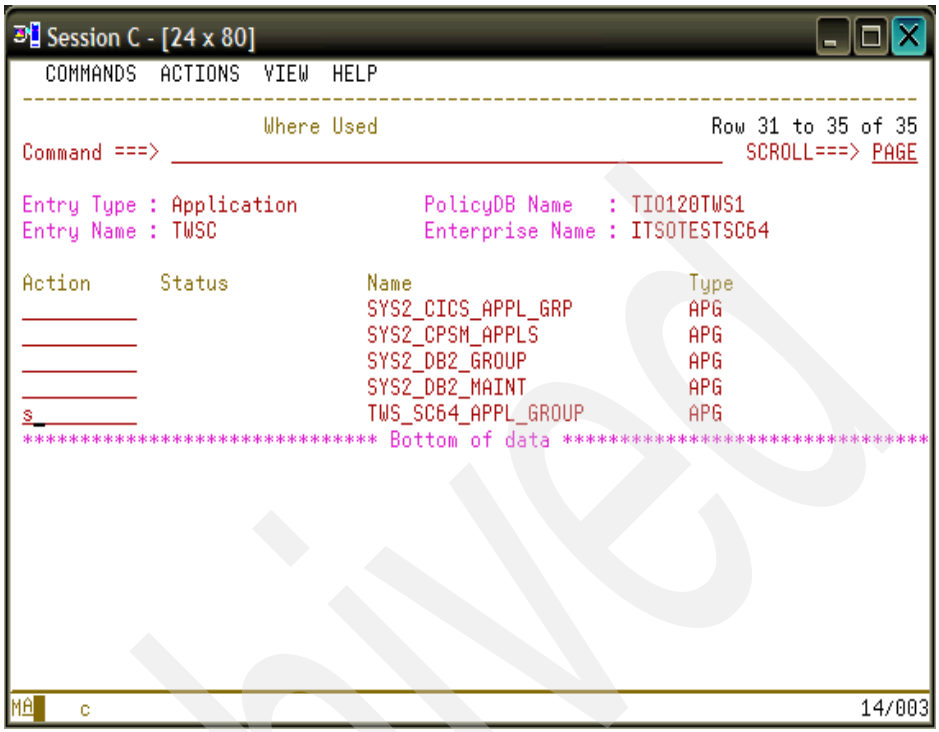

Figure 4-24 Where Used panel for TWS for z/OS controller

7. We have 35 possible names to select in our scenario. So, scroll down until TWS_SC64_APPL_GROUP appears. Place an `s` in the Action column to select the name of the application group (APG) to use and press Enter. The Status column for the name that you choose changes to `SELECTED` as shown in Figure 4-25 on page 128.

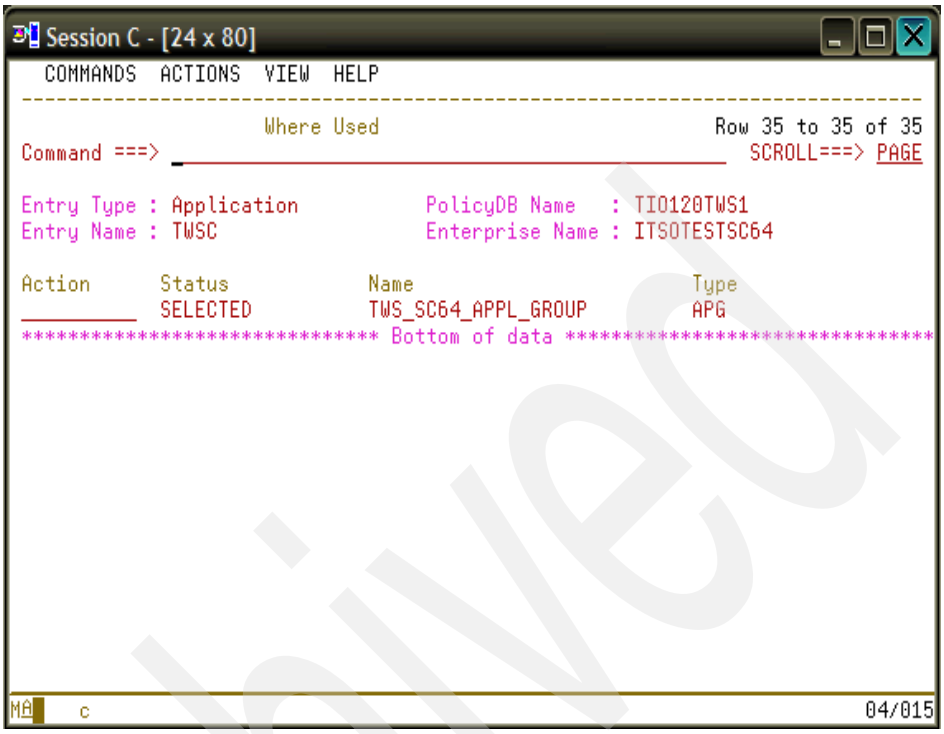

Figure 4-25 Where Used panel for TWS for z/OS controller with APG selected

8. This completes the definition for the TWS for z/OS controller. Press **PF3** until you return to the Entry Name Selection panel.

Defining the TWS for z/OS tracker

The process for defining the TWS for z/OS tracker is almost identical to defining the TWS for z/OS controller. The name of our tracker is TWST. There are changes on two panels: the Define New Entry panel and the OPC Control Specification panel.

1. Table 4-3 shows the changes for the Define New Entry panel.

Table 4-3 Changes for the Define New Entry panel

In this field	Enter this data
Name	Our tracker is called TWST, so use this as the name.
Subsystem Name	Use the same as the Name field for convenience. In our scenario, this is TWST.
Object Type	INSTANCE
Application Type	OPC
Subtype	TRACKER
Job Type	MVS
Job Name	The same as the Subsystem Name, which in our scenario is TWST.
Short description	(Optional) Enter a short description.

The Define New Entry panel for the TWS for z/OS tracker is shown in Figure 4-26 on page 130.

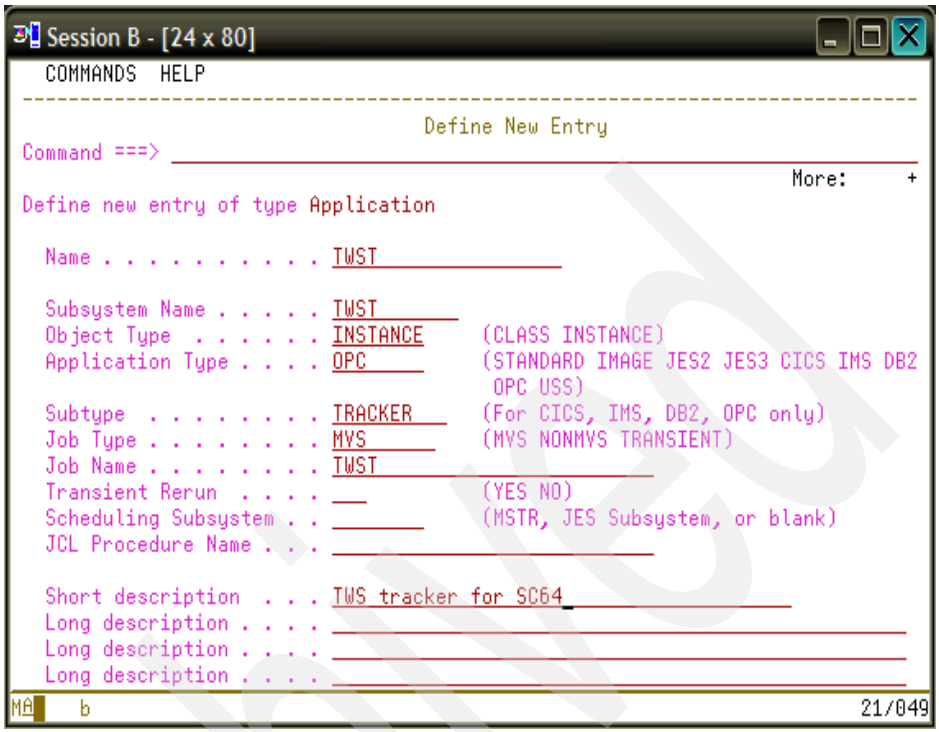

Figure 4-26 Define New Entry panel for TWS for z/OS tracker

2. Table 4-4 shows the changes for the OPC Control Specification panel. Because this is a tracker subsystem, it requires that we complete the Controller ID field.

Table 4-4 Changes for the OPC Control Specification panel

In this field	Enter this data
Subsystem ID	TWST
Controller ID	TWSC

The OPC Control Specification panel with the Subsystem ID and Controller ID fields completed for the TWS for z/OS tracker definition is shown in Figure 4-27 on page 131.

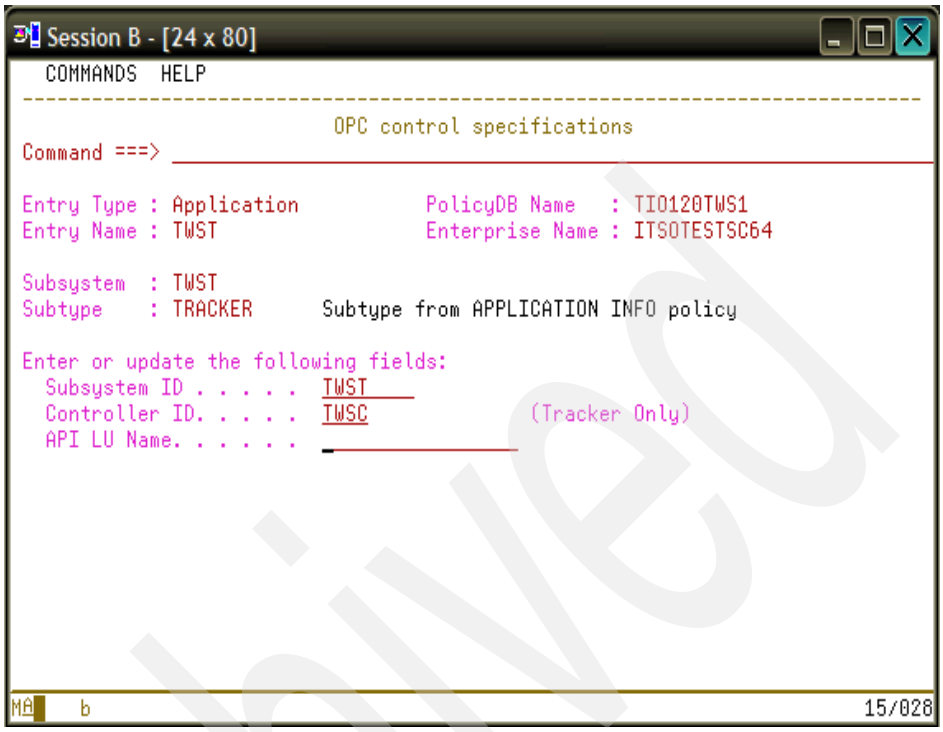

Figure 4-27 OPC control specification panel for TWS for z/OS tracker

3. Press **PF3** to save the OPC control specification definitions. Select the **Where Used** policy from the Policy Selection panel for the TWS for z/OS tracker. Complete the Where Used panel in the same manner as you did for the TWS for z/OS controller.
4. This completes the definition for the TWS for z/OS tracker. Press **PF3** until you return to the Entry Name Selection panel.

Defining the TWS for z/OS PPI request server

The TWS for z/OS request server is a non-MVS subsystem that handles requests from TWS for z/OS to System Automation using the NetView PPI. (This might still be referred to as the OPC request server in some documentation.)

The recommendation for the TWS for z/OS request server is to link the application to a SYSTEM type group which is linked to each system that runs System Automation. The simplest type of definition is to have an application group created for each z/OS. Then all application that runs on a particular system is linked to the one group using the Where Used field of the application entry.

In our scenario, we created an application group called TWS_SC64_APPL_GROUP. All of the applications that we defined to System Automation were linked to this group.

System Automation for z/OS V2.2 includes a sample SYSPLEX PDB that includes a definition for the TWS for z/OS PPI request server called TWS_REQUEST_SERVER. There is no need to create this definition. You just need to link it to your application group, which in our scenario is TWS_SC64_APPL_GROUP. The link is done using the Where Used policy of the provided TWS_REQUEST_SERVER application policy.

The following figures show what is defined in the supplied SYSPLEX PDB.

Figure 4-28 shows the Application Information panel. The data on this panel is the same as what is entered on the Define New Entry panel when creating an application entry from scratch.

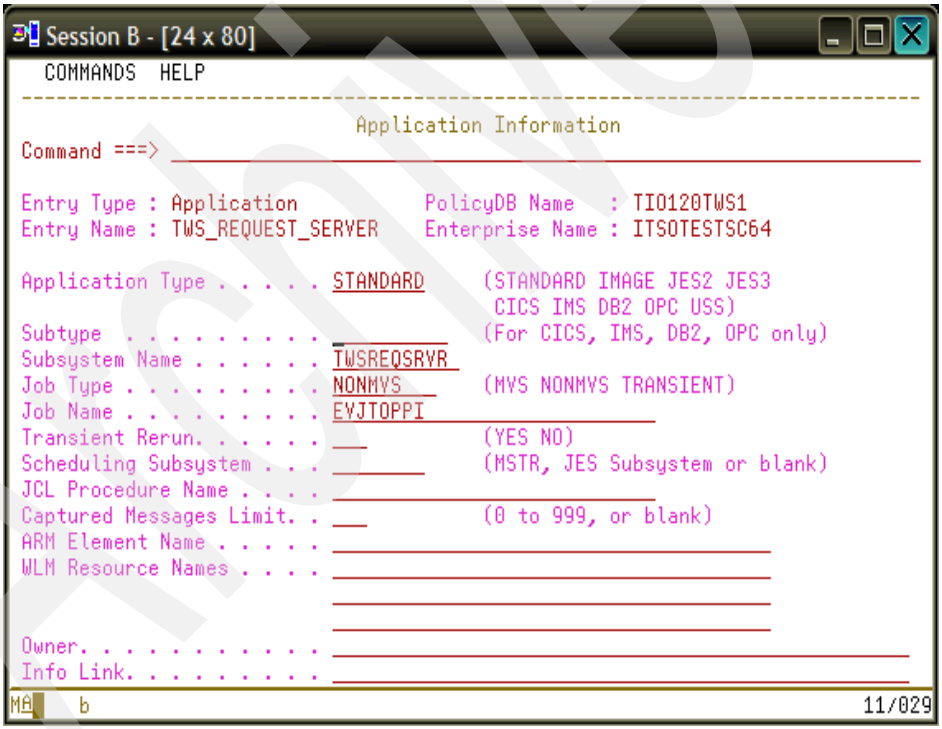

Figure 4-28 Application Information panel

You can see on the Application Information panel that the Job Type is NONMVS, where as both the TWS for z/OS controller and tracker are MVS. The Job Name also differs from the Subsystem Name. Press **PF3** which displays the Policy Selection panel.

There are three policy items on the Policy Selection panel that are used by the TWS for z/OS request server:

- STARTUP
- SHUTDOWN
- WHERE USED

To define the new TWS for z/OS PPI request server:

1. Place an s in the Action column next to STARTUP on the Policy Selection panel. The Subsystem Startup Processing panel that is shown in Figure 4-29 displays.

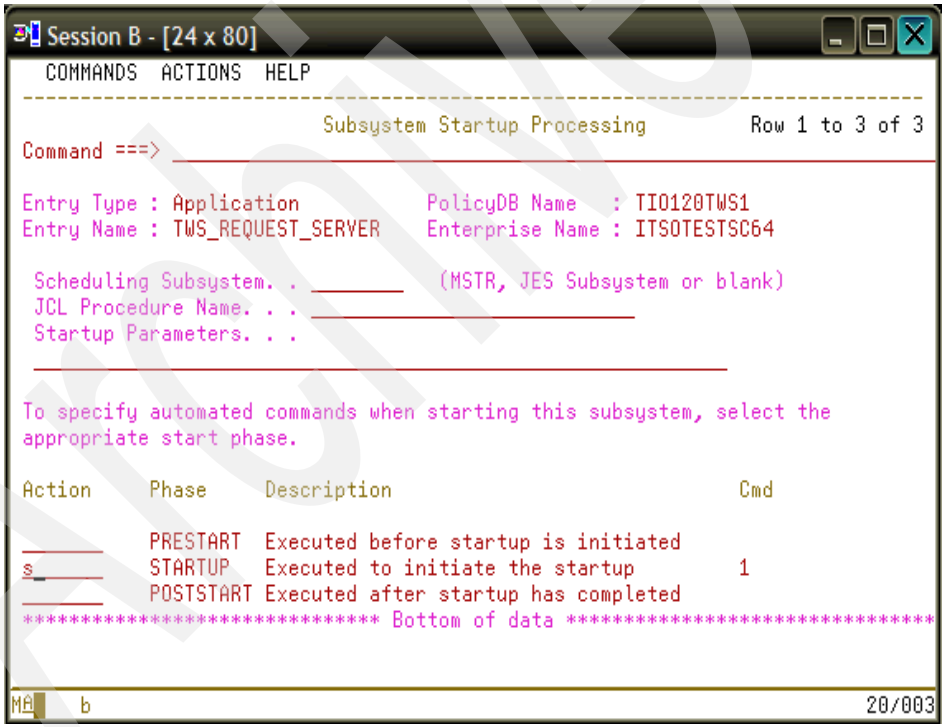

Figure 4-29 Subsystem Startup Processing panel

2. There is no PRESTART or POSTSTART Phase for the TWS for z/OS request server. Place an s in the Action column next to STARTUP. The Startup Command Processing panel that is shown in Figure 4-30 displays.

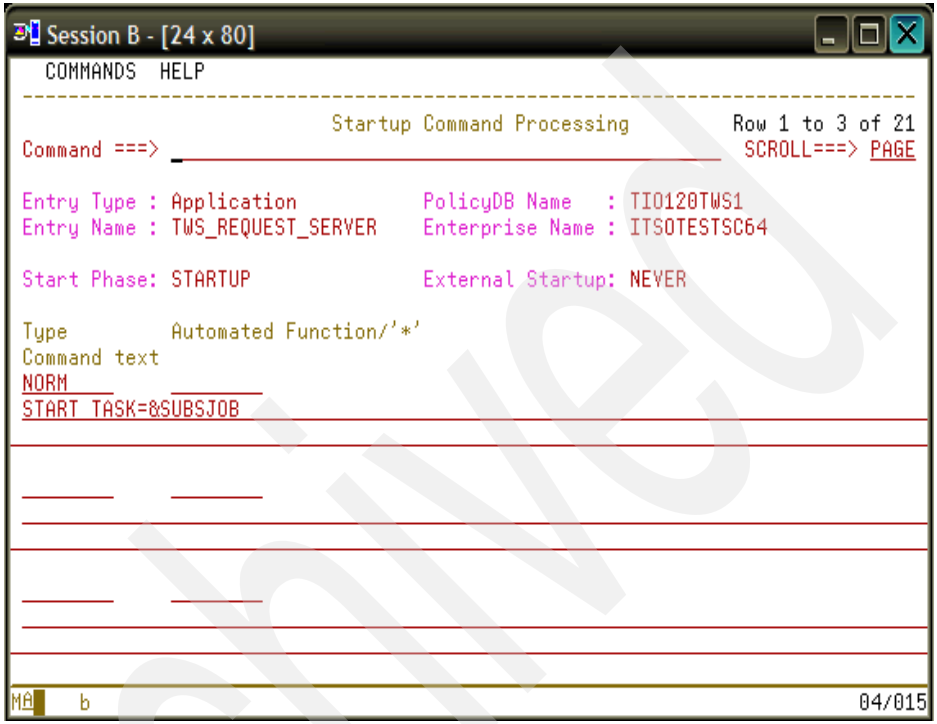

Figure 4-30 Startup Command Processing panel

3. This panel shows the startup command that is used by System Automation to start the TWS for z/OS request server. Press **PF3** twice to return to the Policy Selection panel.
4. Place an s in the Action column next to SHUTDOWN on the Policy Selection panel. The Subsystem Shutdown Processing panel that is shown in Figure 4-31 on page 135 displays.

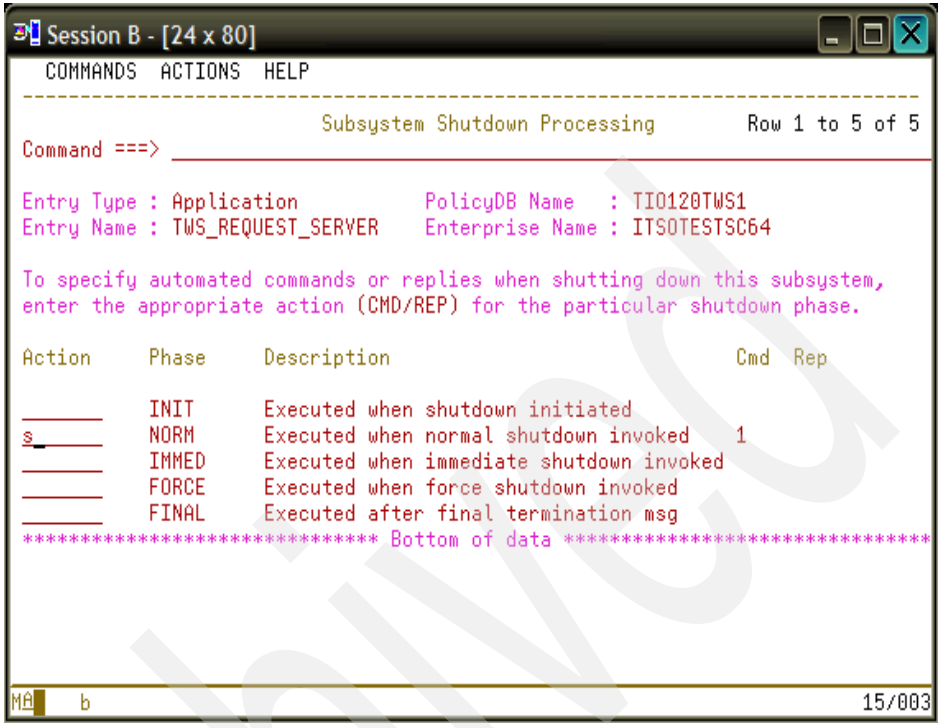

Figure 4-31 Subsystem Shutdown Processing panel

5. There is no INIT, IMMED, FORCE, or FINAL Phase for the TWS for z/OS request server. Place an s in the Action column next to NORM. The Shutdown Command Processing panel that is shown in Figure 4-32 on page 136 displays.

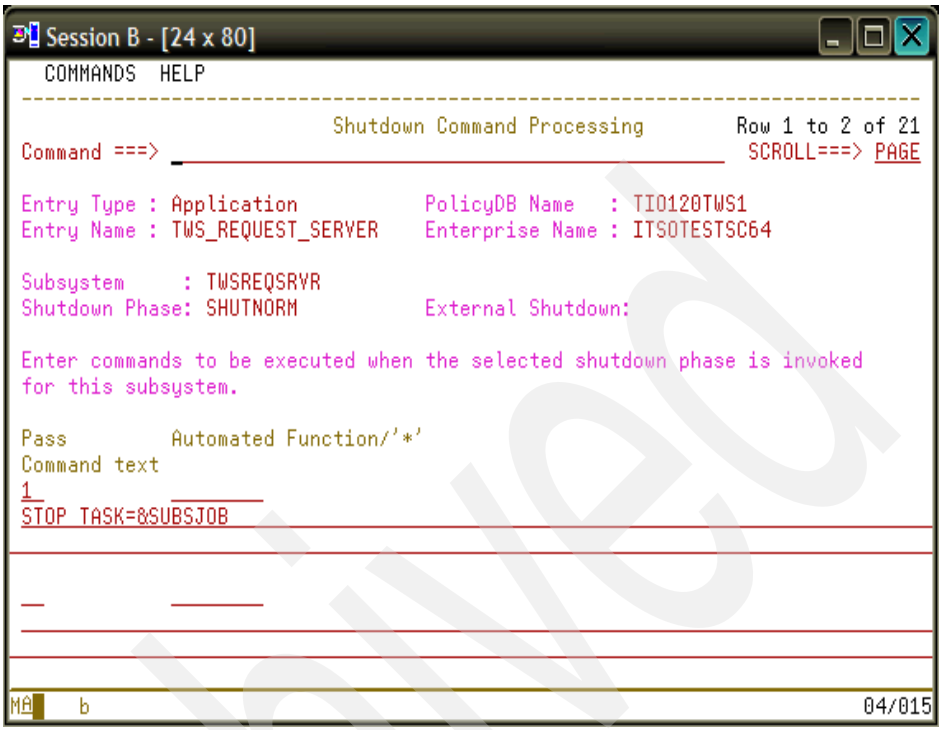

Figure 4-32 Shutdown Command Processing

6. This panel shows the shutdown command that is used by System Automation to stop the TWS for z/OS request server. Press **PF3** twice to return to the Policy Selection panel.
7. Place an s in the Action column next to WHERE USED on the Policy Selection panel. The Where Used panel displays. Complete the Where Used panel in the same manner that you did for the TWS for z/OS controller.
8. This completes the definition for the TWS for z/OS request server. Press **PF3** until you return to the Entry Name Selection panel.

Defining the new Tivoli Workload Scheduler for z/OS batch command server (optional)

The TWS for z/OS request server is a non-MVS subsystem that handles requests from batch jobs using the new Batch Command Interface to System Automation. If you do not use the batch command interface, then there is no need to define this application entry, and you can skip this section.

The recommendation for the TWS for z/OS batch command server is to link the application to a SYSTEM type group which is linked to each system which runs System Automation. The simplest type of definition is to have an application group created for each z/OS. Then all applications that run on a particular system are linked to the one group using the Where Used field of the application entry.

In our scenario, we created an application group called TWS_SC64_APPL_GROUP. All of the applications that we defined to System Automation were linked to this group.

System Automation for z/OS V2.2 includes a sample SYSPLEX PDB which includes a definition for the TWS for z/OS batch command server called TWS_COMMAND_SERVER. You do not need to create this definition. It just needs to be linked to your application group which in our scenario is TWS_SC64_APPL_GROUP. The link is done using the Where Used policy of the provided TWS_COMMAND_SERVER application policy.

The following figures show what is defined in the supplied SYSPLEX PDB.

Figure 4-33 on page 138 shows the Application Information panel. The data on this panel is the same as the data that is entered on the Define New Entry panel when creating an application entry from scratch.

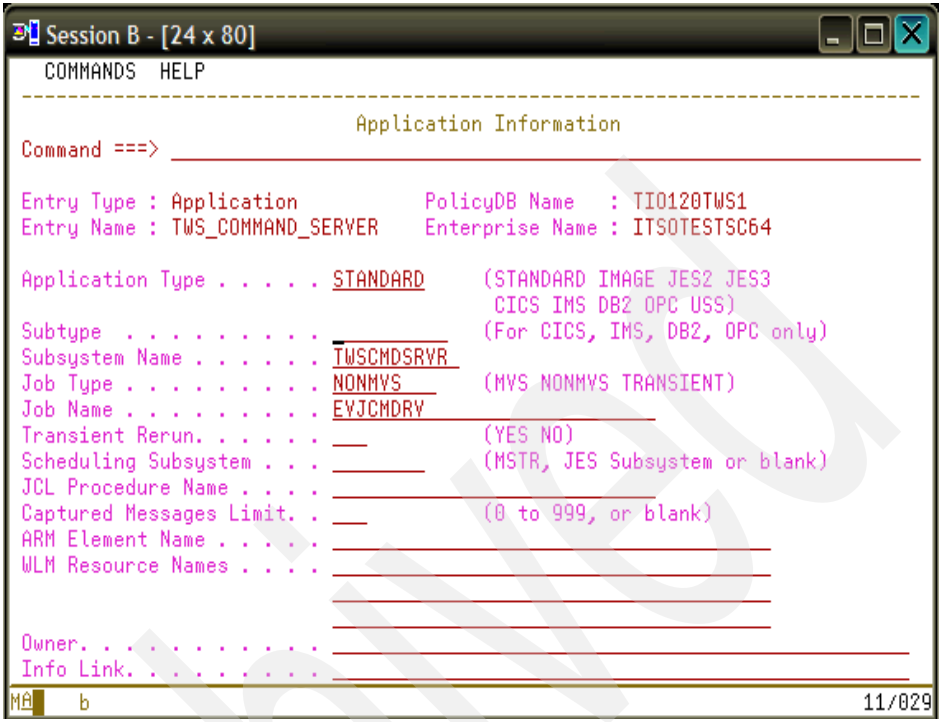

Figure 4-33 Application Information panel

You can see on the Application Information panel that the Job Type is NONMVS, where as both the TWS for z/OS controller and tracker are MVS. The Job Name also differs from the Subsystem Name. Press **PF3** to display the Policy Selection panel.

There are three policy items on the Policy Selection panel that are used by the TWS for z/OS command server:

- STARTUP
- SHUTDOWN
- WHERE USED

To define the new TWS for z/OS batch command server:

1. Place an s in the Action column next to STARTUP on the Policy Selection panel. The Subsystem Startup Processing panel that is shown in Figure 4-34 displays.

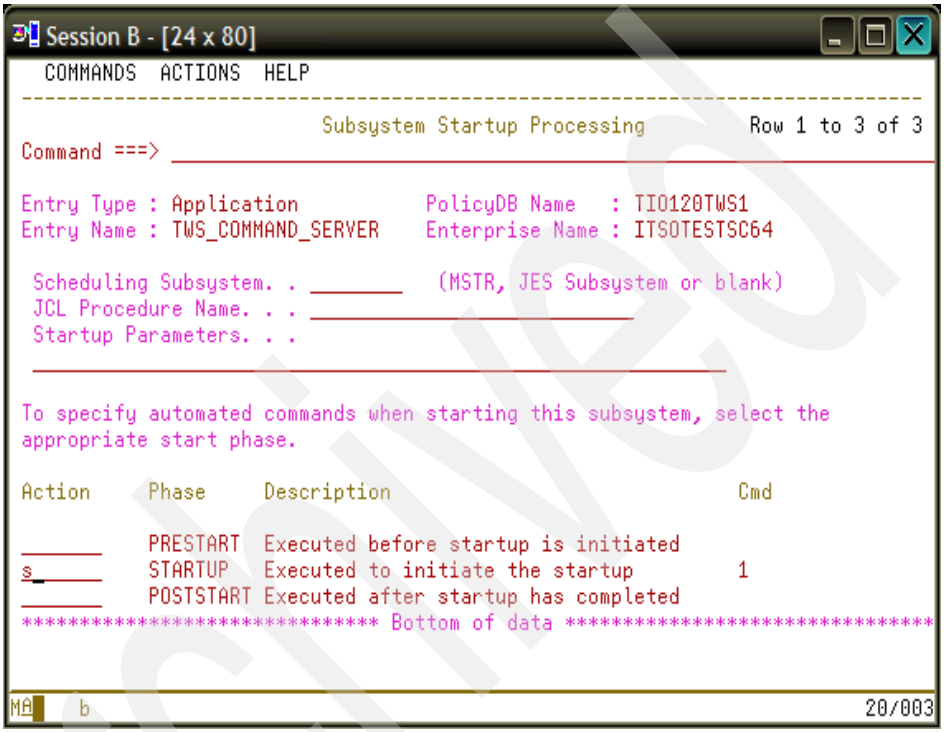

Figure 4-34 Subsystem Startup Processing panel for TWS for z/OS command server

2. There is no PRESTART or POSTSTART Phase for the TWS for z/OS command server. Place an s in the Action column next to STARTUP. The Startup Command Processing panel that is shown in Figure 4-35 on page 140 displays.

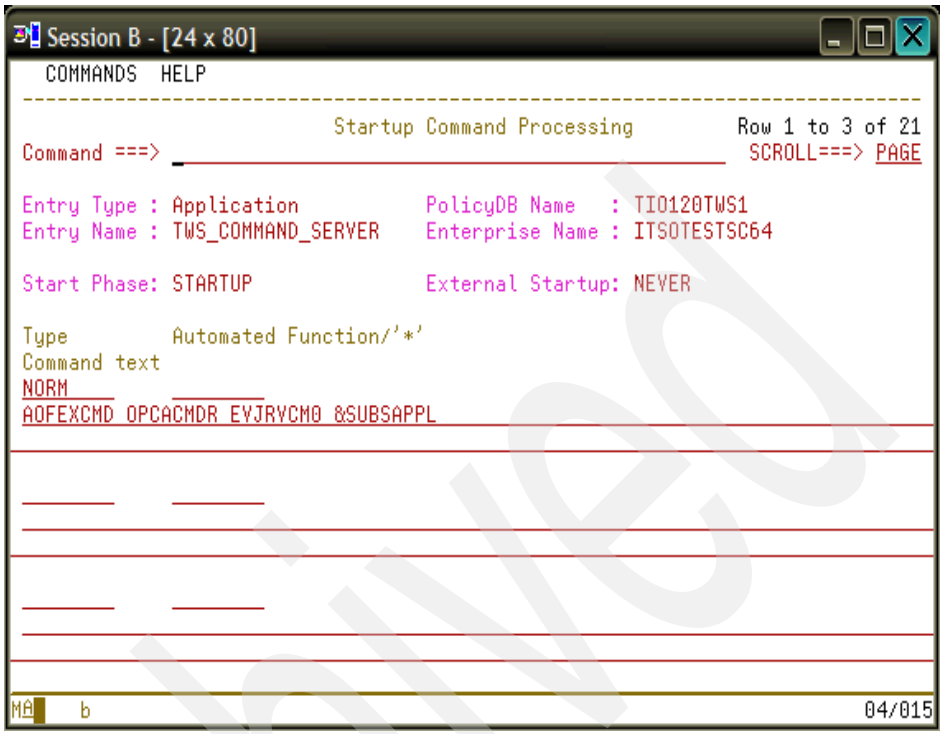

Figure 4-35 Startup Command Processing panel

3. This panel shows the startup command that used by System Automation to start the Tivoli Workload Scheduler command server. Press **PF3** twice to return to the Policy Selection panel.
4. Place an s in the Action column next to SHUTDOWN on the Policy Selection panel. The Subsystem Shutdown Processing panel that is shown in Figure 4-36 on page 141 displays.

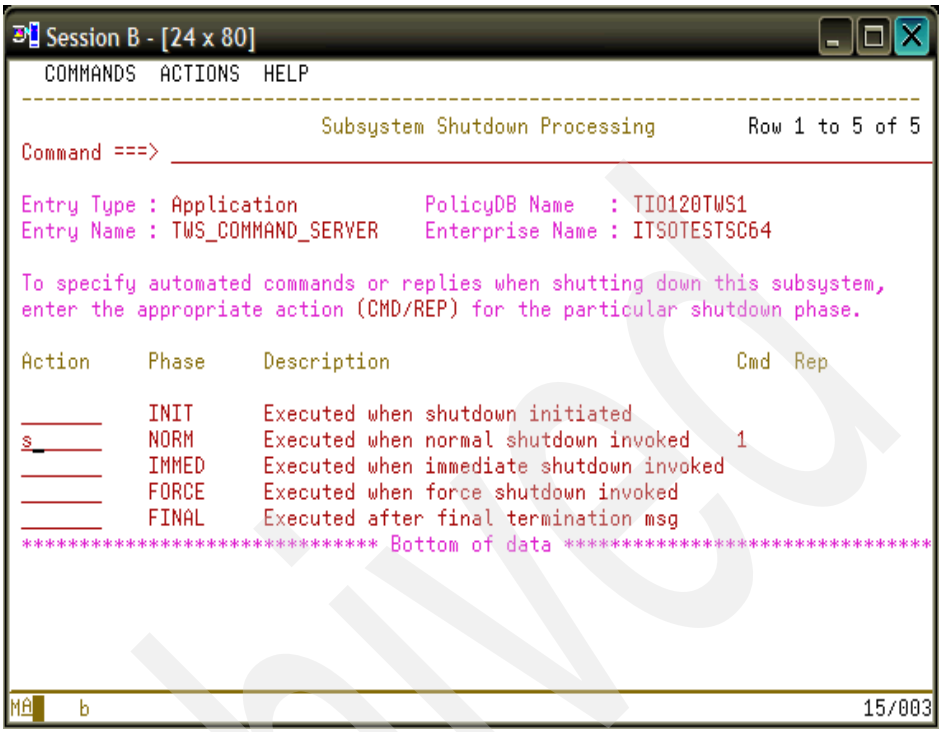

Figure 4-36 Subsystem Shutdown Processing panel

5. There is no INIT, IMMED, FORCE, or FINAL Phase for the TWS for z/OS command server. Place an s in the Action column next to NORM. The Shutdown Command Processing panel that is shown in Figure 4-37 on page 142 display.

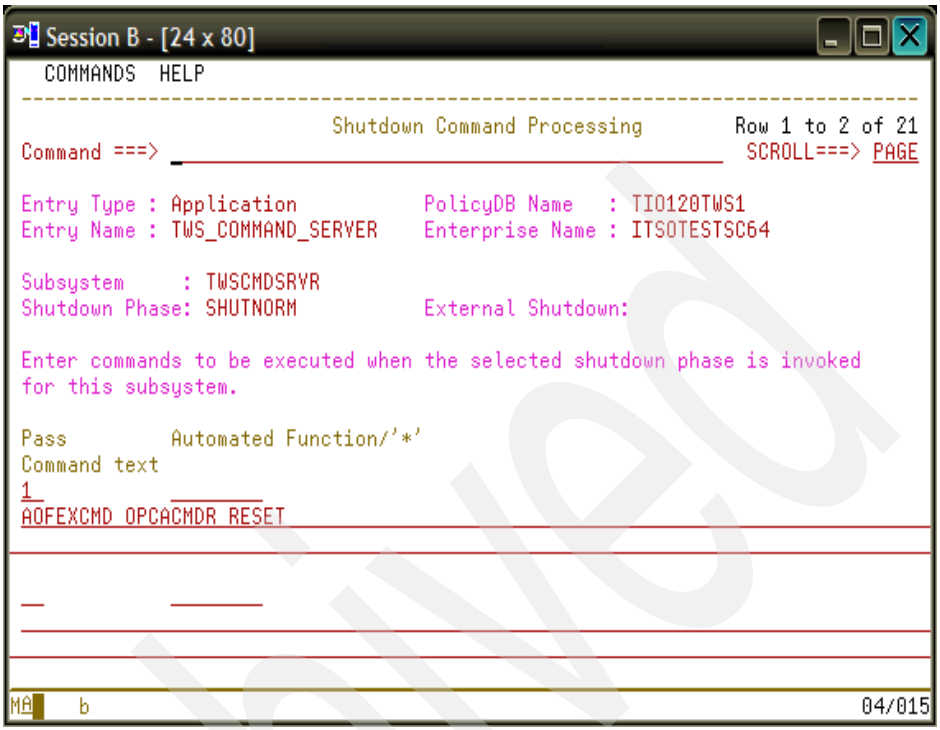

Figure 4-37 Shutdown Command Processing

6. This panel shows the shutdown command that is used by System Automation to stop the TWS for z/OS command server. Press **PF3** twice to return to the Policy Selection panel.
7. Place an s in the Action column next to WHERE USED on the Policy Selection panel. The Where Used panel displays. Complete the Where Used panel in the same manner as you did for the TWS for z/OS controller.
8. This completes the definition for the TWS for z/OS command server. Press **PF3** until you return to the Entry Name Selection panel.

Defining the z/OS status observer subsystem

The z/OS status observer subsystem is a non-MVS subsystem which reflects the status of an System Automation resource in TWS for z/OS special resources. At present, the status can be one of the following:

- Available
- Unavailable

System Automation generates two TWS for z/OS special resources per System Automation resource. Each TWS for z/OS special resource has a specific name. The application that corresponds to the special resource must be defined in the automation entry for OPCAO. This definition is described in “Special resources” on page 166.

This section only looks at the definition of the z/OS status observer subsystem. To define the z/OS status observer subsystem:

1. Type **new** on the command line of the Entry Name Selection panel that is shown in Figure 4-19 on page 121 and press Enter. The Define New Entry panel displays. Table 4-5 shows the data that you need to enter in this panel.

Table 4-5 Define New Entry data to enter

In this field	Enter this data
Name	No specific name, so use TWS_OBSERVER.
Subsystem Name	TWSOBSERVER
Object Type	INSTANCE
Application Type	STANDARD
Job Type	NONMVS
Job Name	OPCOBSVR
Short description	(Optional) Enter a short description.

The Define New Entry panel is shown in Figure 4-38 on page 144.

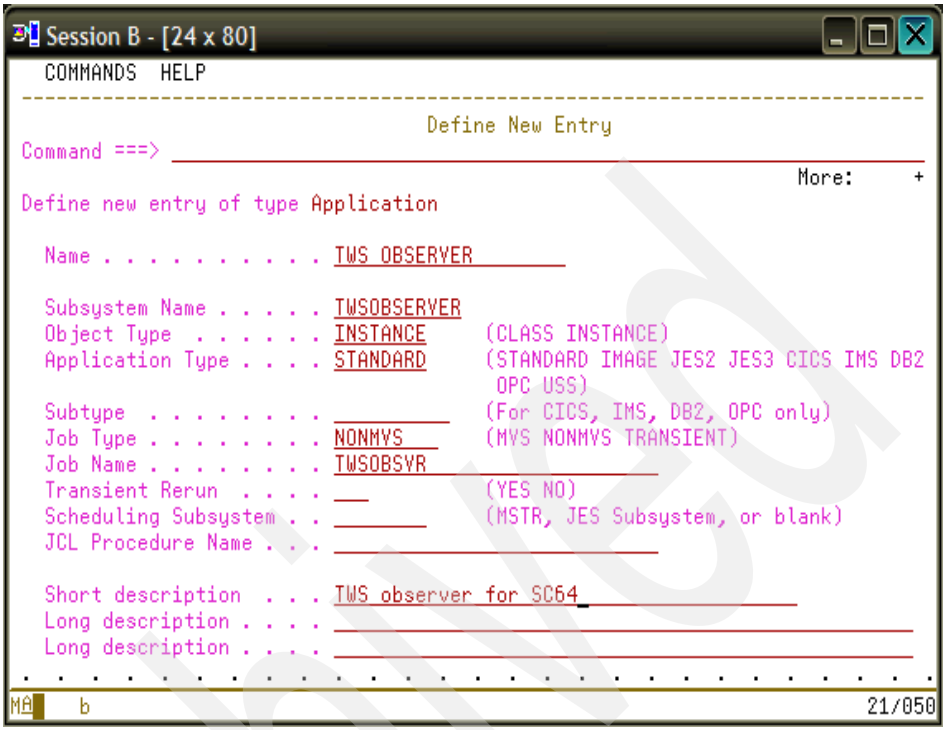

Figure 4-38 Define New Entry panel for z/OS status observer

2. After you have completed the fields, Press **PF3** to display the Policy Selection panel.

   The following policy items on the Policy Selection panel are used by the TWS for z/OS command server:

   – AUTOMATION INFO
   – RELATIONSHIPS
   – STARTUP
   – SHUTDOWN
   – WHERE USED

3. Place an s in the Action column next to AUTOMATION INFO on the Policy Selection panel. The Subsystem Startup Processing panel that is shown in Figure 4-39 on page 145 displays.

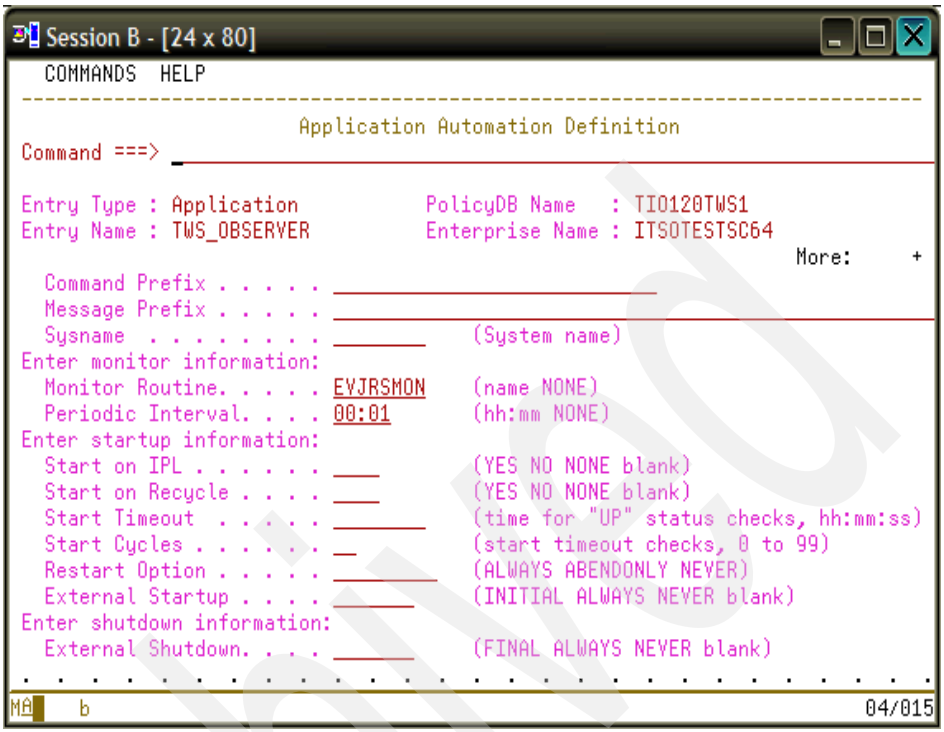

Figure 4-39 Application Automation Definition panel for z/OS status observer

4. Place an s in the Action column next to RELATIONSHIPS on the Policy Selection panel. The Selection List panel that is shown in Figure 4-40 on page 146 displays.

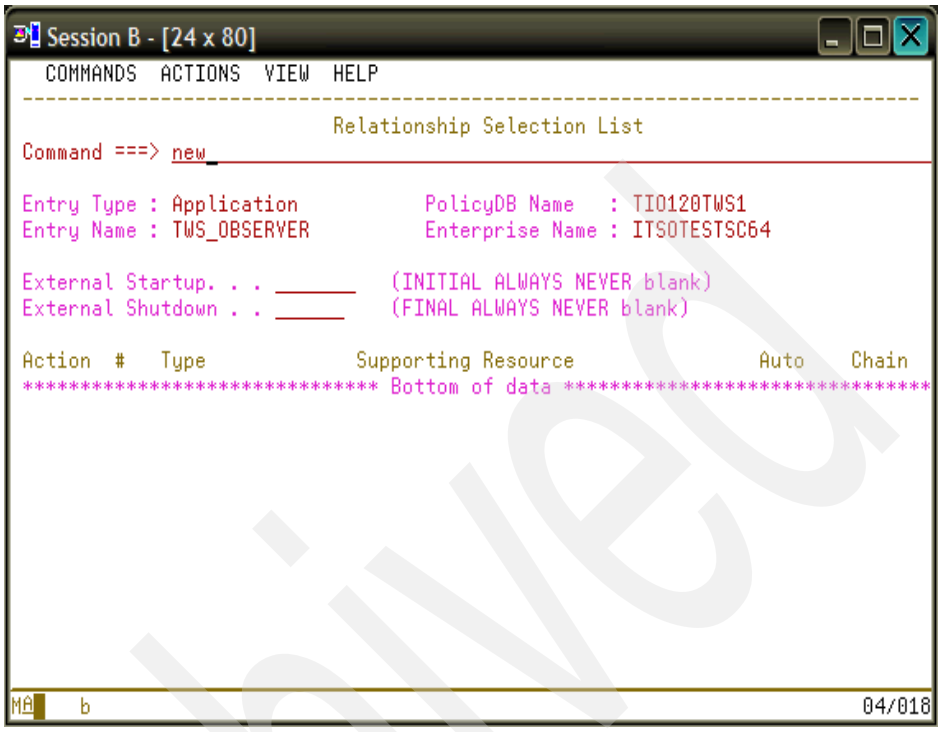

Figure 4-40 Relationship Selection List for z/OS status observer

5. For each entry, type **new** on the command line and press Enter. The Define Relationship panel displays.

   You need to create three relationship entries:

   – `FORCEDOWN(WhenObservedWasAvailable) to supporting resource sanetview/APL/=`

     In this entry, `sanetview` is the name of the System Automation NetView subsystem. The Define Relationship panel for this entry is shown in Figure 4-41.

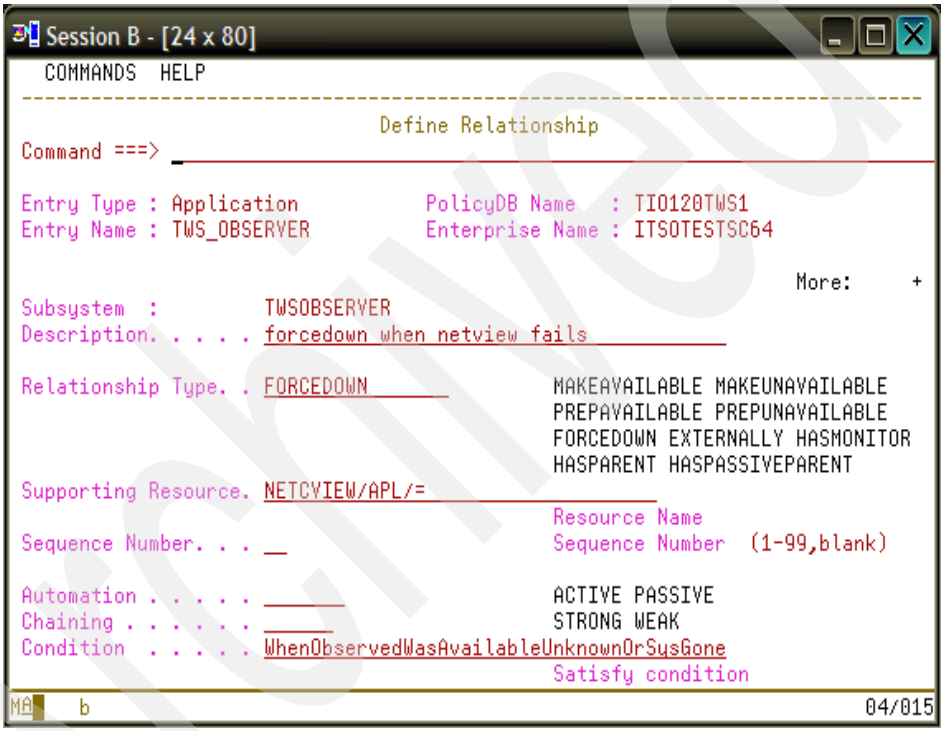

Figure 4-41 Define Relationship panel for first entry for z/OS status observer

Placing a question mark (?) in the Condition field displays a list of conditions that you can select.

After you have completed the fields, Press **PF3** which returns you to the Relationship Selection List.

– FORCEDOWN(WhenObservedHardDown) to supporting resource twsController/APL/=

In this entry, opcController is the name of your TWS for z/OS controller subsystem. In our scenario, this entry is TWSC.

The Define Relationship panel for this entry is shown in Figure 4-42.

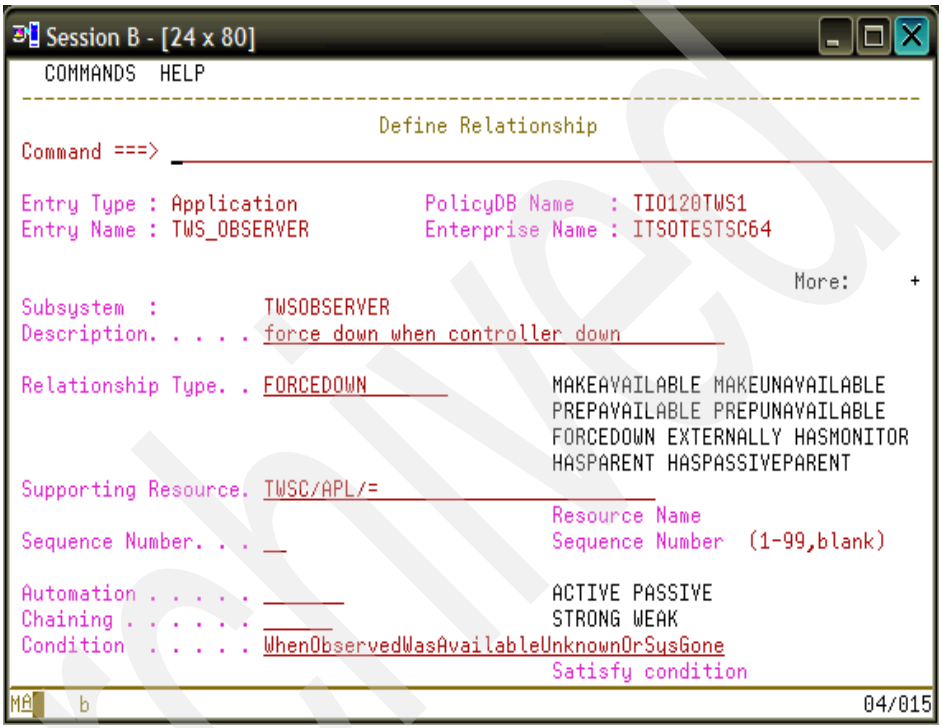

Figure 4-42 Define Relationship panel for second entry for z/OS status observer

Placing a question mark (?) in the Condition field displays a list of conditions that you can select.

After you have completed the fields, Press **PF3** to return to the Relationship Selection List.

– HASPARENT to supporting resource twsController/APL/=

In this entry, opcController is the name of your TWS for z/OS controller subsystem. In our scenario, this entry is TWSC.

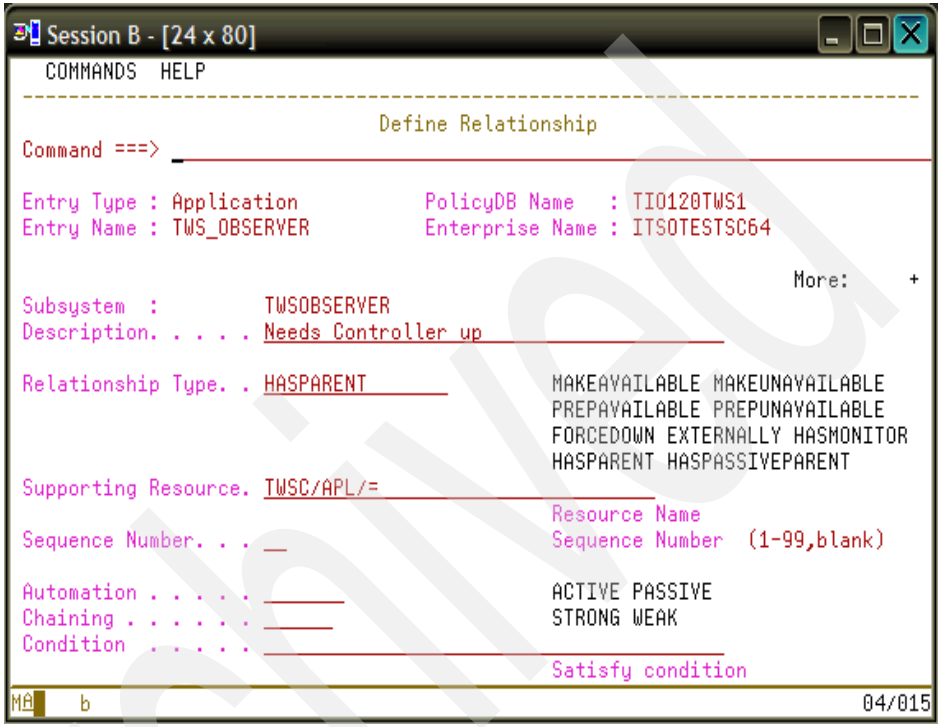

Figure 4-43 Define Relationship panel for third entry for z/OS status observer

After you have completed the fields, Press **PF3** twice to return to the Policy Selection panel.

6. Place an s in the Action column next to STARTUP on the Policy Selection panel. The Subsystem Startup Processing panel that is shown in Figure 4-44 on page 150 displays.

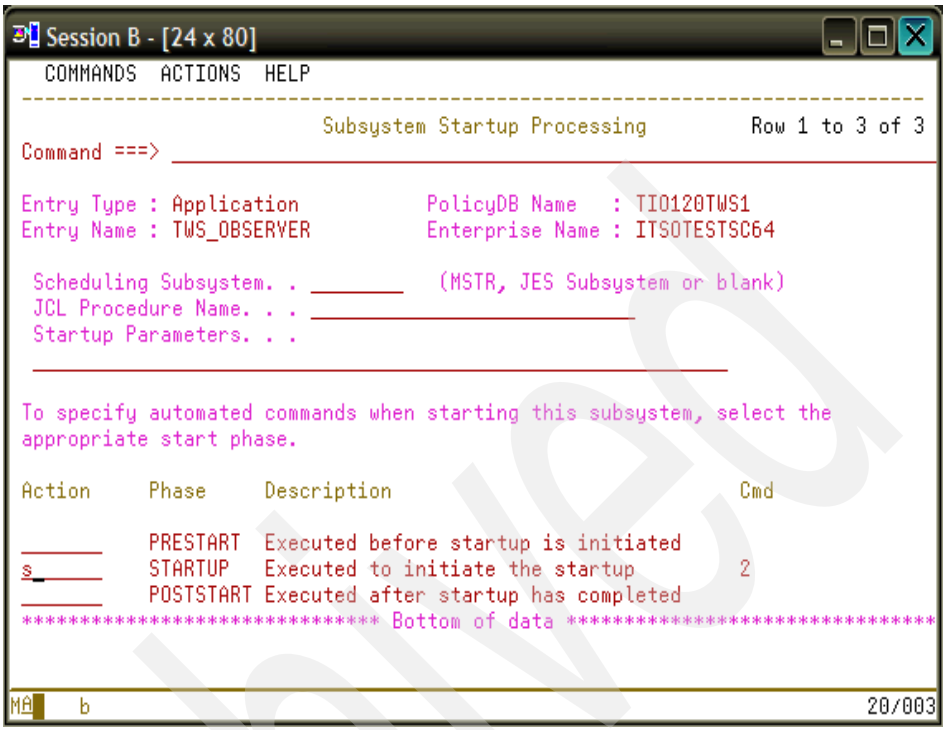

Figure 4-44 Subsystem Startup panel for z/OS status observer

7. There is no PRESTART or POSTSTART Phase for the z/OS status observer. Place an s in the Action column next to STARTUP. The Startup Command Processing panel that is shown in Figure 4-45 on page 151 displays.

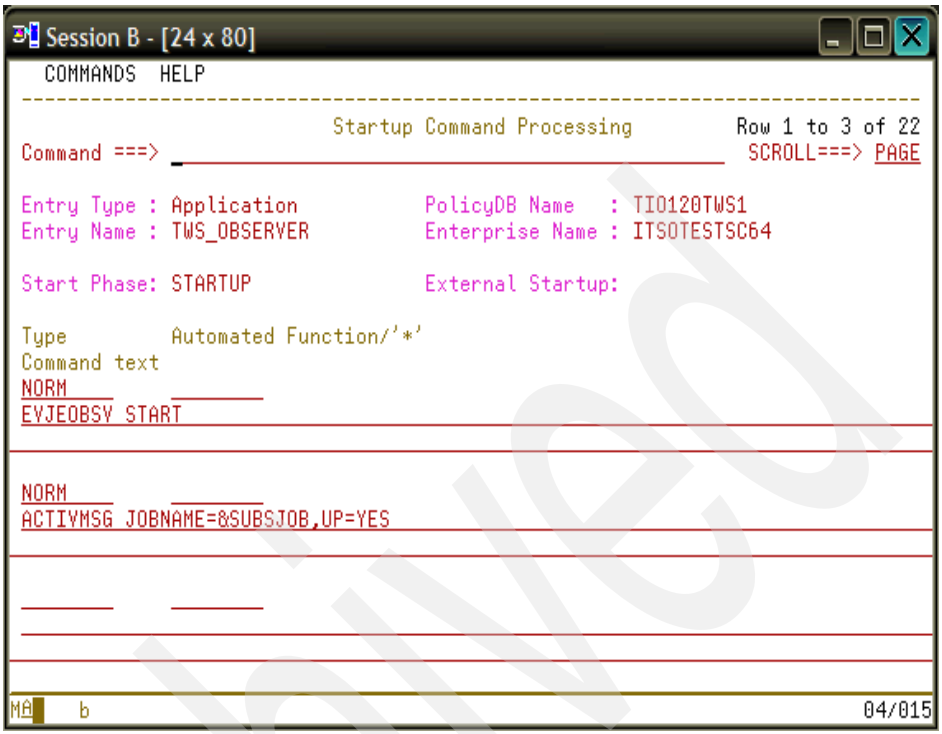

Figure 4-45 Startup Command Processing panel for z/OS status observer

8. This panel shows the startup commands that are used by System Automation to start the z/OS status observer. Press **PF3** twice to return to the Policy Selection panel.
9. Place an s in the Action column next to SHUTDOWN on the Policy Selection panel. The Subsystem Shutdown Processing panel that is shown in Figure 4-46 on page 152 displays.

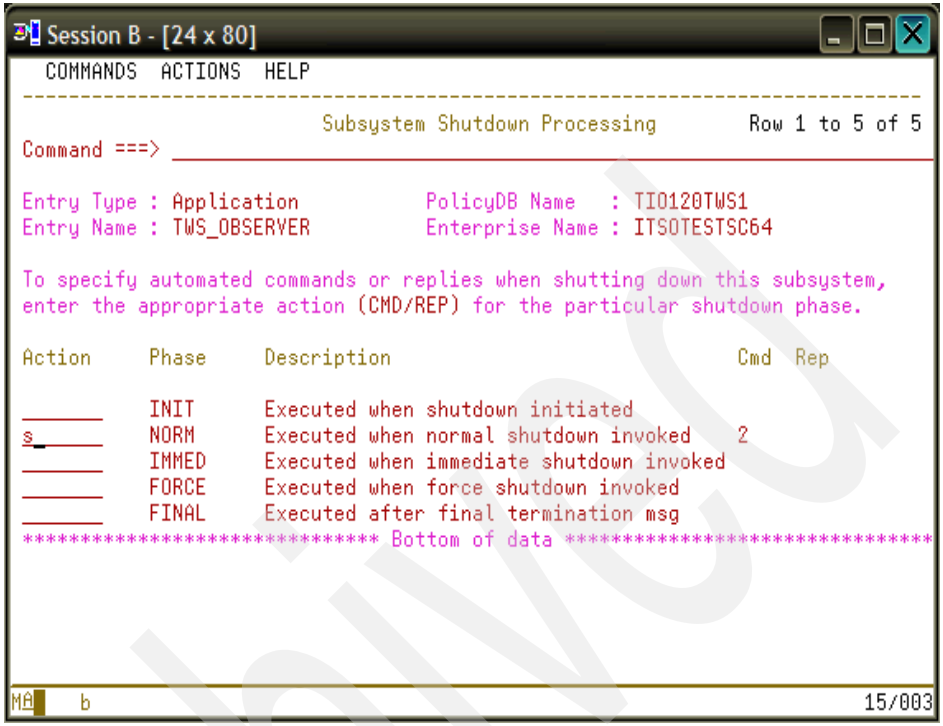

Figure 4-46 Subsystem Shutdown Processing panel for z/OS status observer

10. There is no INIT, IMMED, FORCE, or FINAL Phase for the z/OS status observer. Place an s in the Action column next to NORM. The Shutdown Command Processing panel that is shown in Figure 4-47 on page 153 displays.

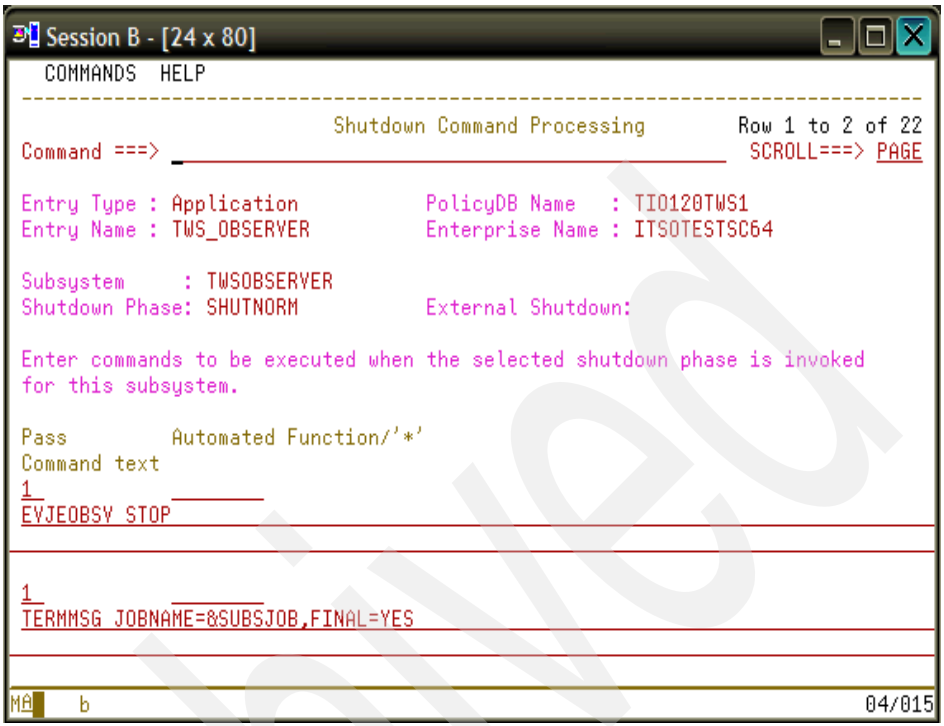

Figure 4-47 Shutdown Command Processing for z/OS status observer

11. This panel shows the shutdown commands that are used by System Automation to stop the z/OS status observer. Press **PF3** twice to return to the Policy Selection panel.
12. Place an s in the Action column next to WHERE USED on the Policy Selection panel. The Where Used panel is displayed. Complete the Where Used panel in the same manner as you did for the TWS for z/OS controller.
13. This completes the definition for the z/OS status observer. Press **PF3** until you return to the Entry Name Selection panel.

All the application entries required for the TWS for z/OS interfaces are now complete.

Defining the Product Automation entries for OPCAO

Product automation entries are specific to particular products. TWS for z/OS is one of these products. The product automation entries are additional to entries which are normally created for a product. Many of these entries still use the old product name of OPC. These entries work normally for TWS for z/OS.

To define the Product Automation entries for OPCAO select **option 20 PRD** Product Automation from the Entry Type Selection menu that is shown in Figure 4-13 on page 115. The Entry type selections for Product Automation panel that is shown in Figure 4-48 displays.

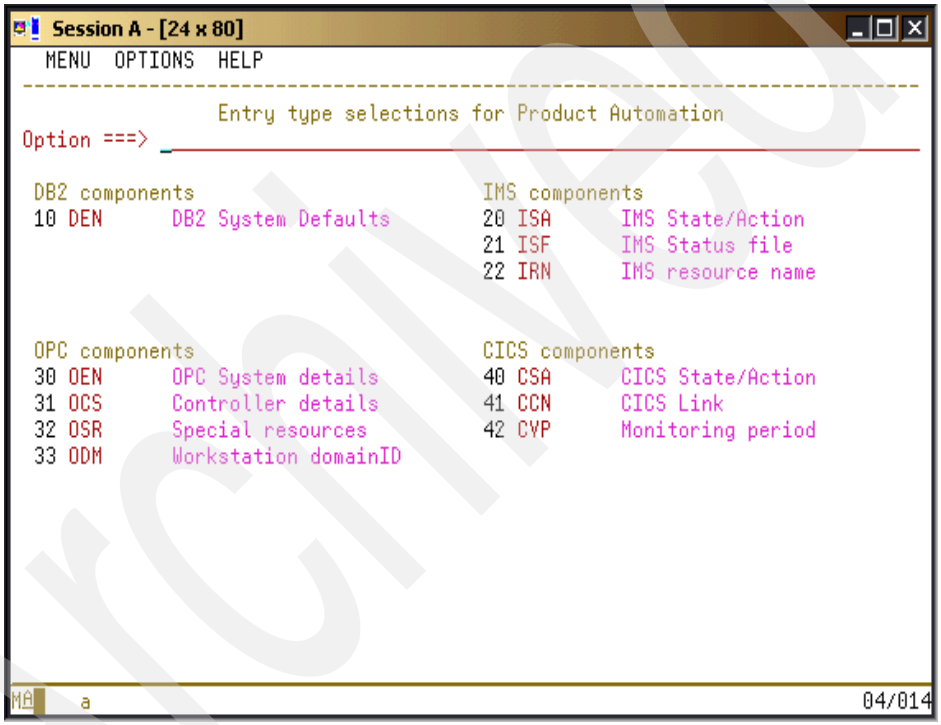

Figure 4-48 Entry type selections for Product Automation panel

The following sections discuss the options that are listed under OPC components. The definitions for these sections all begin from the Entry type selections for Product Automation panel.

OPC System details

The OPC System details entry provides control information for OPCAO. To define this entry:

1. Select **option 30 - OEN** OPC System details from the Entry type selections for Product Automation panel and press Enter. The Entry Name Selection panel that is shown in Figure 4-49 displays.

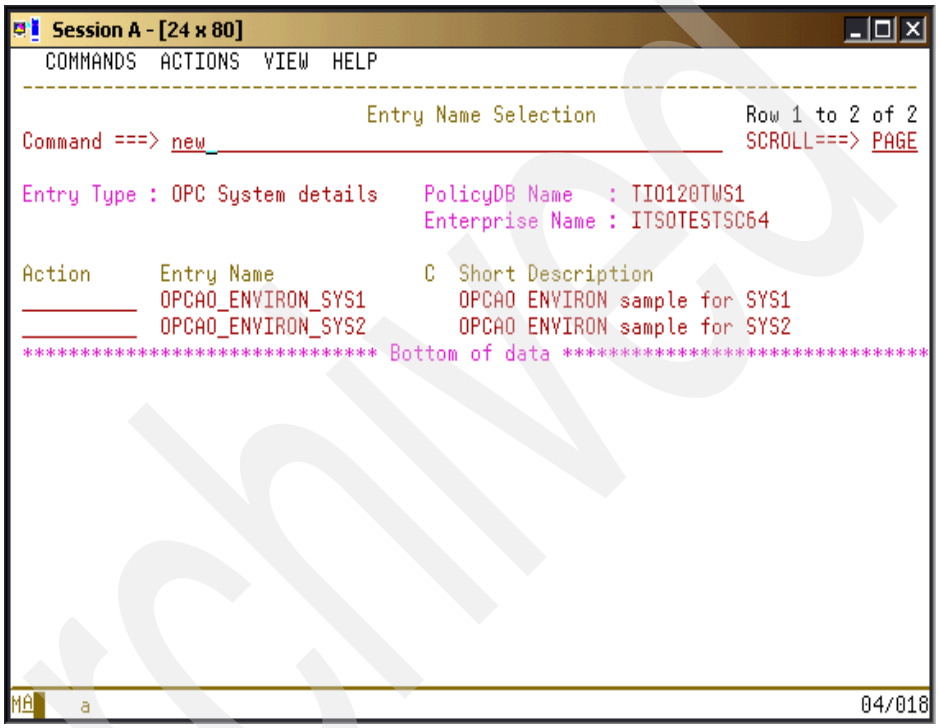

Figure 4-49 OPC System Details Entry Name Selection panel

2. Type **new** on the command line and press Enter. The Define New Entry panel that is shown in Figure 4-50 on page 156 displays.

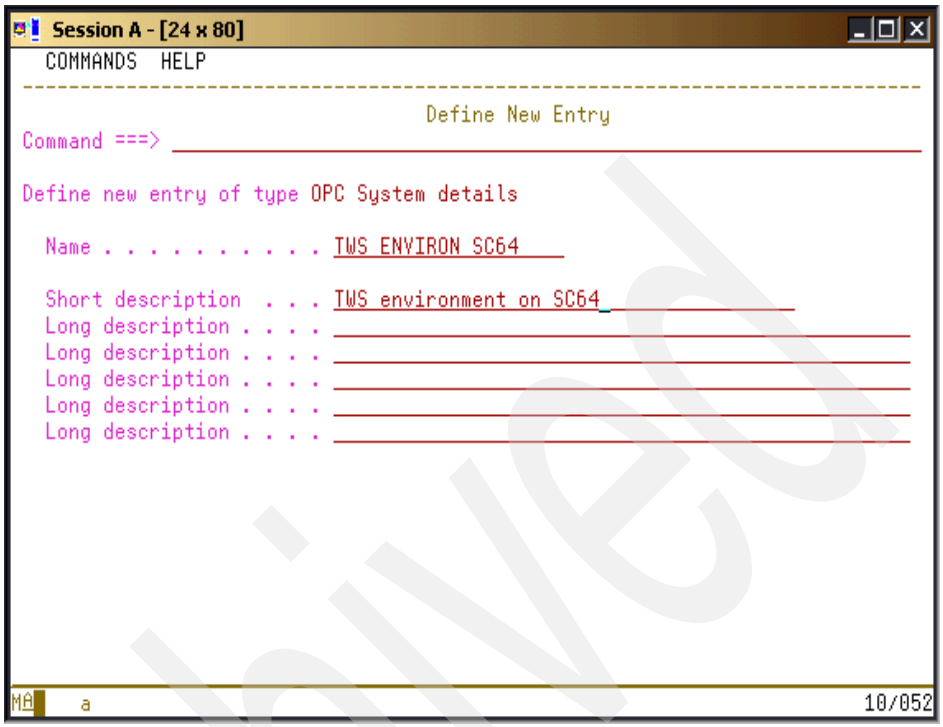

Figure 4-50 OPC Systems details Define New Entry panel

3. Enter a name for the entry and a short description. For this scenario, we used:
   - `TWS_ENVIRON_SC64` for the name for our entry. Choose a name that is relevant to your site.
   - `TWS environment on SC64` as the short description for our entry.
4. Press **PF3** to save the data. The Policy Selection panel that is shown in Figure 4-51 on page 157 displays.

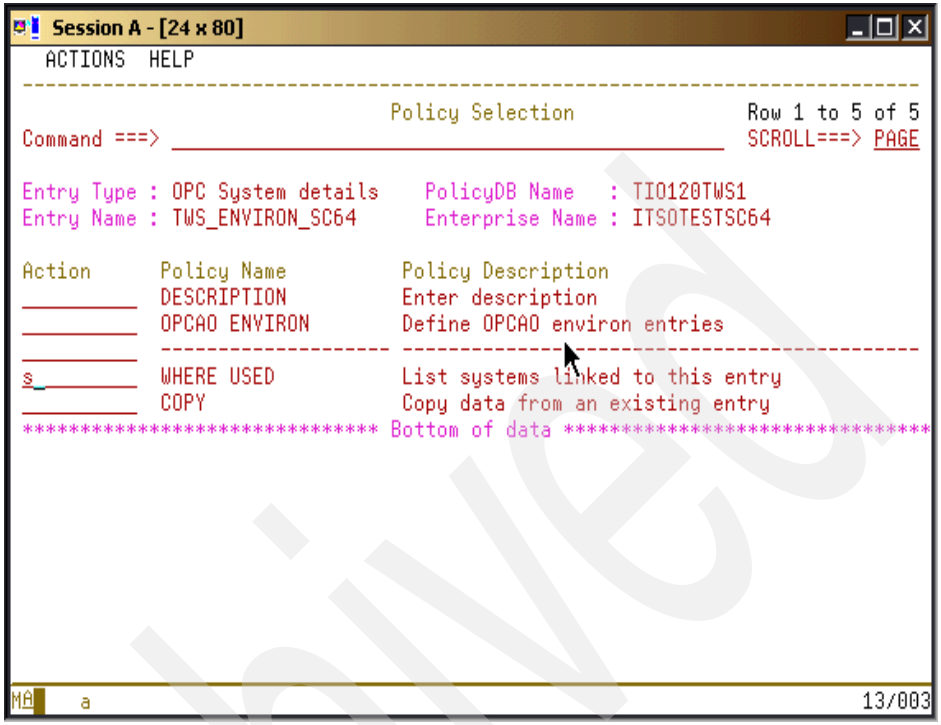

Figure 4-51 OPC System details Policy Selection panel

5. Place an s in the Action field for the WHERE USED. The Where Used panel that is shown in Figure 4-52 on page 158 displays.

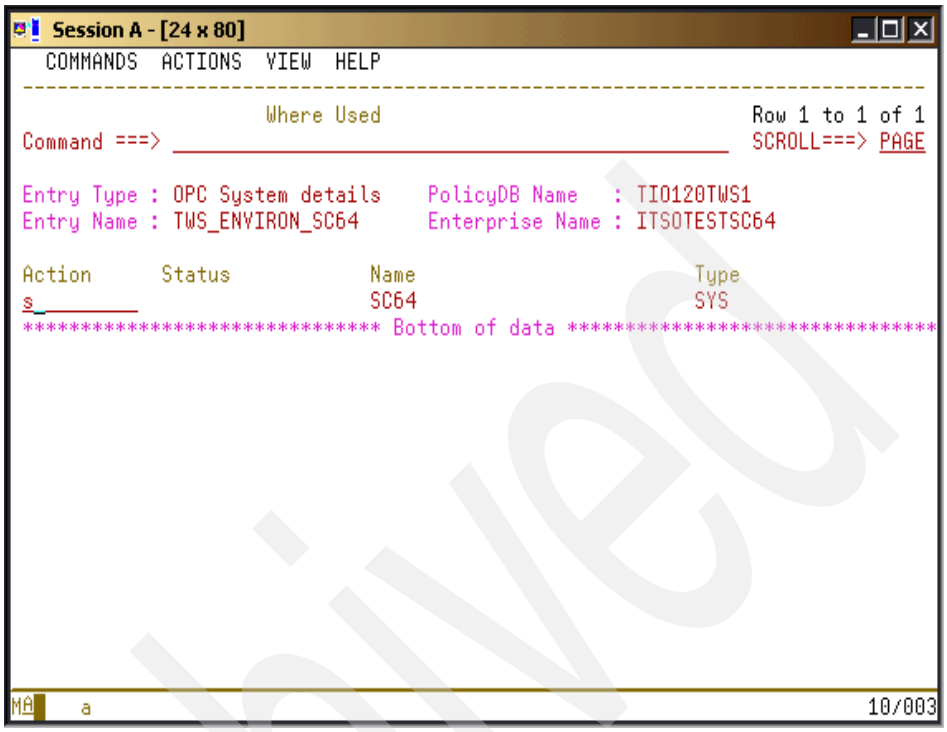

Figure 4-52 OPC System Details Where Used panel

6. This panel should show all systems that are defined in the SYSPLEX PDB. Select the system on which this TWS for z/OS controller is used by placing an s in the Action column next to the system and pressing Enter. (We only have one system in our SYSPLEX PDB which is SC64.) The Status column changes to SELECTED as shown in Figure 4-53 on page 159.

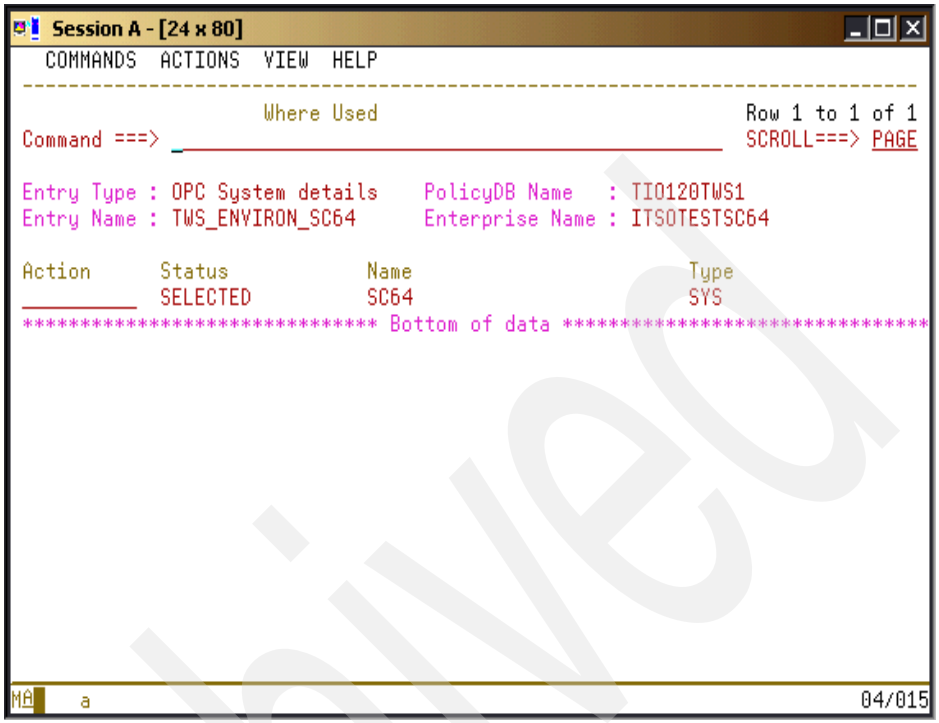

Figure 4-53 OPC System details Where Used panel after s action

7. This completes the OEN entry. Press **PF3** until you return to the Entry type selections for Product Automation panel.

Controller details

This entry provides System Automation with the information that it requires to determine:

- Controller subsystem names. You can define more than one controller.
- System Automation resources that have corresponding TWS for z/OS special resources.
- NetView domain on which this controller runs.

To define this entry:

1. Select option **31 - OCS** Controller details from the Entry type selections for Product Automation panel and press Enter. The Entry Name Selection panel that is shown in Figure 4-54 displays.

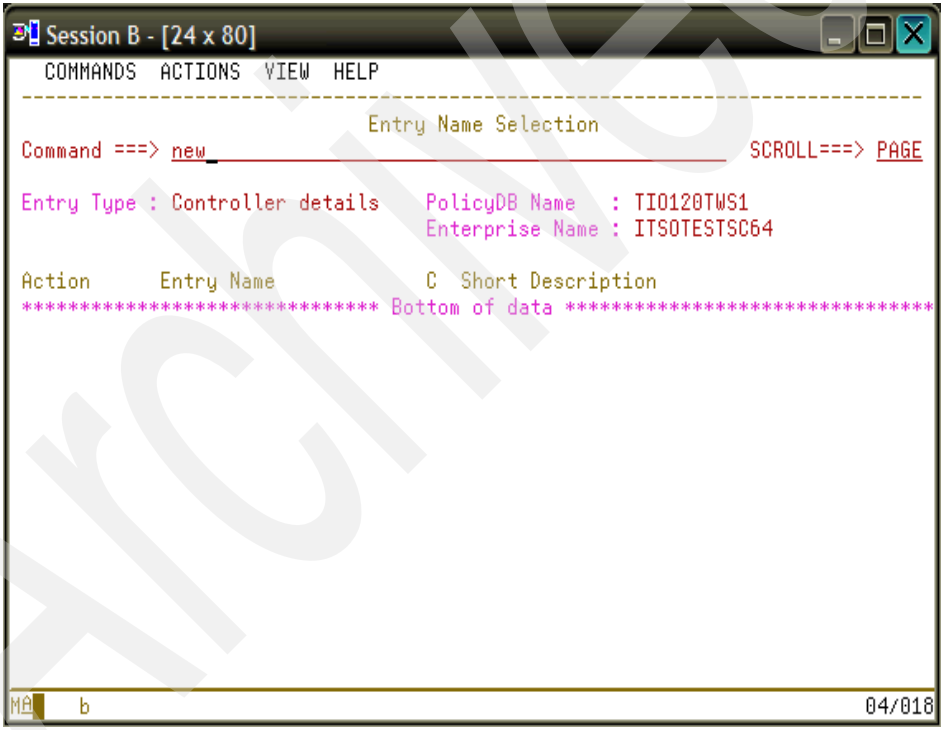

Figure 4-54 Entry Name Selection panel for OCS entry

2. Type **new** on the command line and press Enter. The Define New Entry panel that is shown in Figure 4-55 on page 161 displays.

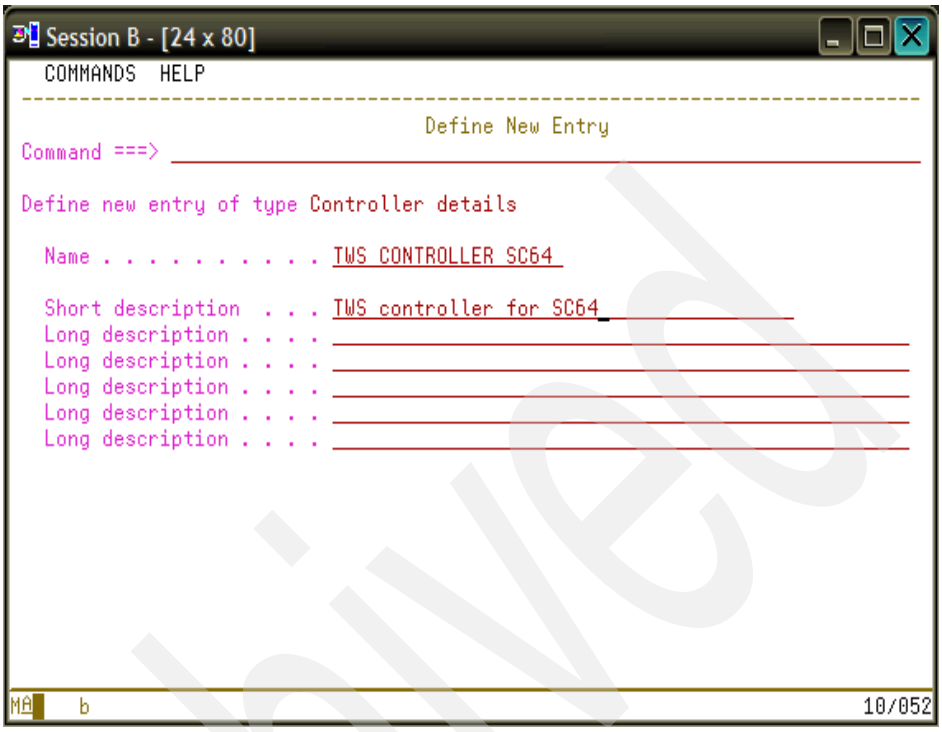

Figure 4-55 Define New Entry panel for OCS entry

3. Enter a name for the entry and a short description. For this scenario, we used:
   - `TWS_CONTROLLER_SC64` as the name for our entry. Choose a name that is relevant to your site.
   - `TWS controller for SC64` as the short description for our entry.
4. Press **PF3** to save your data. The Policy Selection panel that is shown in Figure 4-56 on page 162 displays.

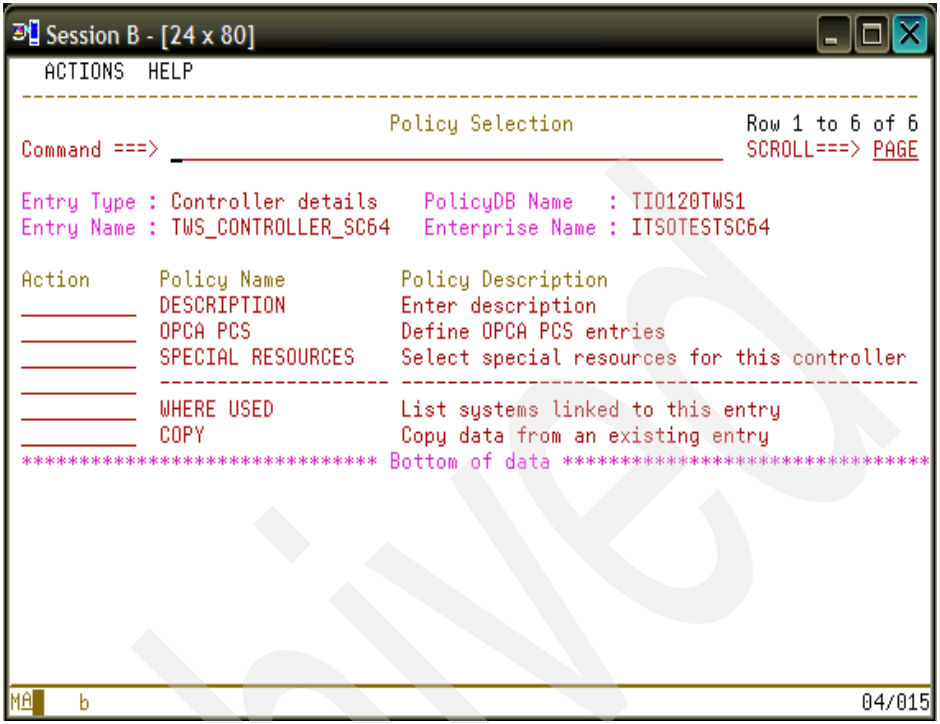

Figure 4-56 Policy Selection panel for OCS entry

There are three policy items on the Policy Selection panel that are used by the TWS for z/OS controller OCS entry:

- OPCA PCS
- SPECIAL RESOURCES

  Only used if you specify YES in the OPCA PCS policy entry.
- WHERE USED

5. Place an s in the Action field for OPCA PCS and press Enter. The OPC Controller details panel that is shown in Figure 4-57 on page 163 displays.

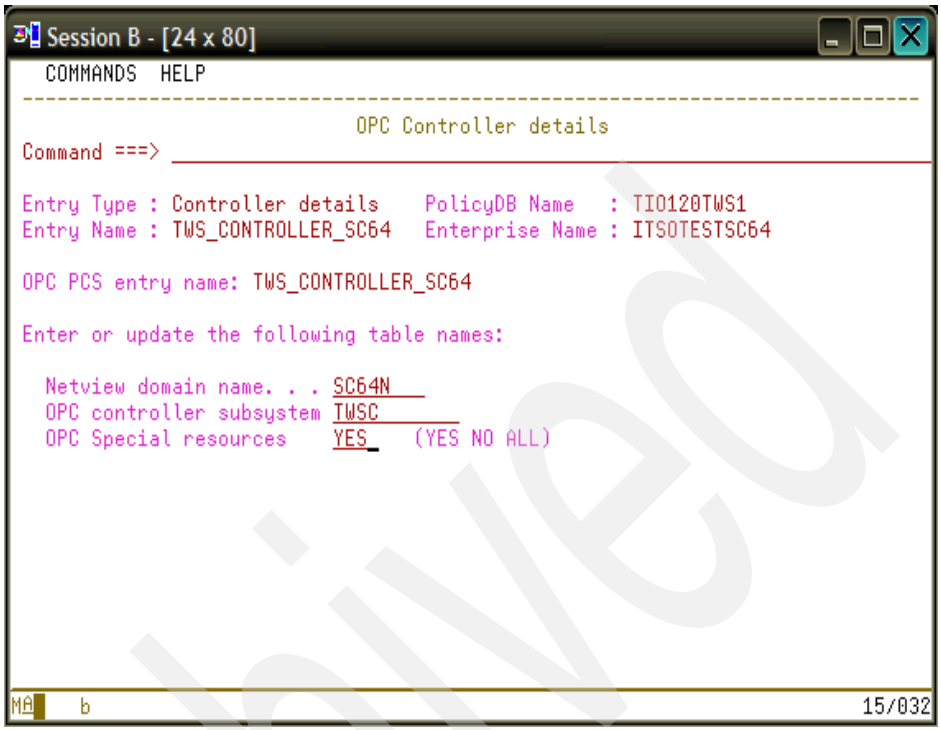

Figure 4-57 OPC Controller details panel

6. Complete the fields on the OPC controller details panel as follows:
   - NetView domain name

     This specifies which NetView domain the TWS for z/OS controller runs. Another option here is to specify `SYSPLEX`, meaning that the controller can run on any system in the sysplex.

     Our NetView domain name is SC64N.
   - OPC controller subsystem

     This is the name of your TWS for z/OS controller. In our scenario this name is TWSC.

– OPC Special resources

Defines how System Automation resources and corresponding TWS for z/OS special resources are handled. There are three options:

- YES

  Only System Automation resources set up in a special resources list as defined in “Special resources” on page 166 have a status reflected in TWS for z/OS. You must then select SPECIAL RESOURCES.

- NO

  No System Automation resources have a TWS for z/OS special resource set.

- ALL

  All System Automation resources have a TWS for z/OS resource set. There is no need to define any special resource lists.

We use YES in our scenario. Therefore, we must also update the SPECIAL RESOURCES policy item.

7. Press **PF3** to return to the Policy Selection panel.
8. Because we specified YES for OPC Special resources in this entry, we must now define which special resource lists are used by this entry. Place an s in the Action field for SPECIAL RESOURCES and press Enter. The Special resources for Controller details panel that is shown in Figure 4-58 on page 165 displays.

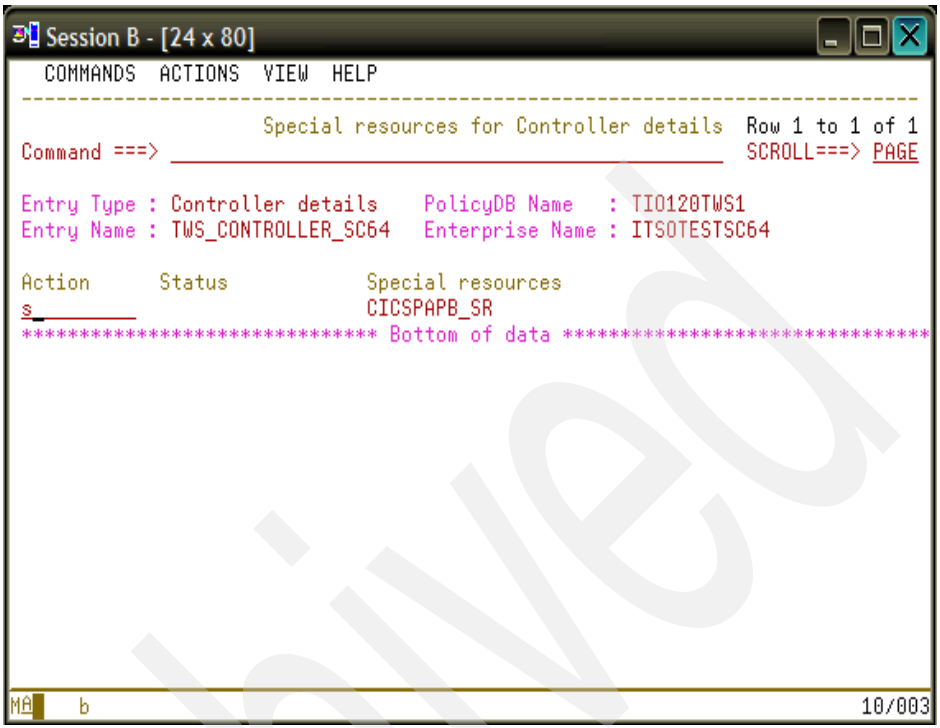

Figure 4-58 Special resources for Controller details panel

This panel shows a list of all OSR entries. We only have one list in our scenario.

9. Place an s in the Action column next to the required Special resources and press Enter. The status field changes to SELECTED. Press **PF3** to return to the Policy Selection panel.
10. Place an s in the Action field for WHERE USED and press Enter. The Where Used panel that is shown in Figure 4-59 on page 166 displays.

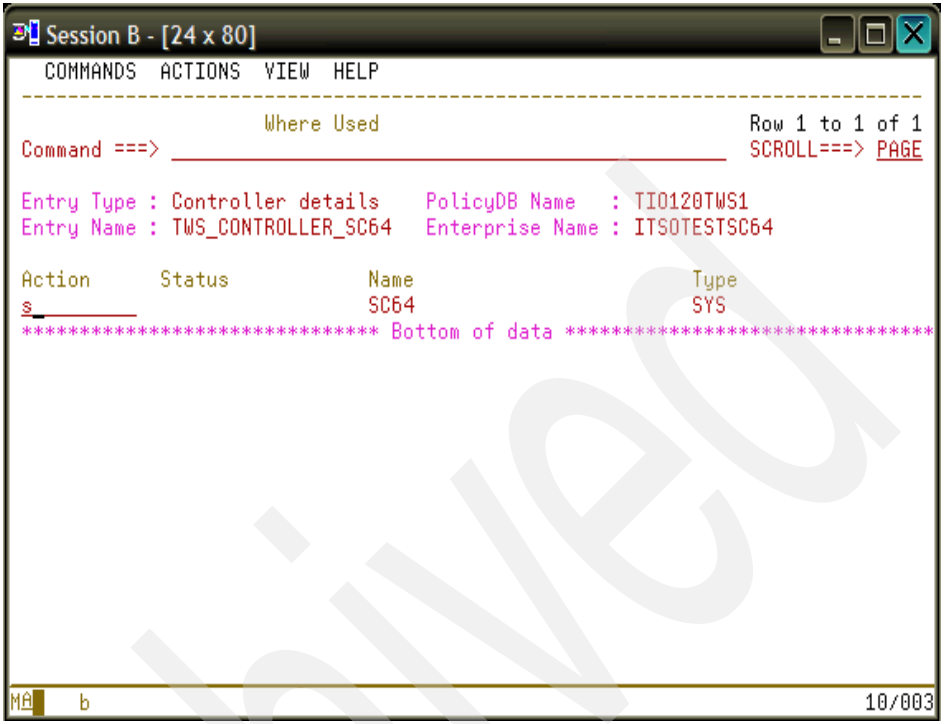

Figure 4-59 Where Used panel for OCS entry

11. This panel shows a list of all systems that are defined in the SYSPLEX PDB. We only have one system in our SYSPLEX PDB. Place an `s` in the Action column for the system where this controller runs and press Enter. The Status field changes to `SELECTED`. Press **PF3** three times to return to the Entry type selections for Product Automation panel.

This completes the definition of the OCS entry.

Special resources

This entry provides a specific list of System Automation resources for which a TWS for z/OS special resource can be updated with a status from System Automation. One entry can contain a number of System Automation resources.

System Automation contains many resources. However, you might not want a corresponding TWS for z/OS special resource for every System Automation resource. By defining System Automation resources using this option, you can choose which System Automation resources have a corresponding TWS for z/OS special resource. To enable this list to be used, this entry must then be referenced in the OCS policy for the TWS for z/OS controller. A special resource

list is only used if the OPC Special resources policy specifies YES. (For more information, see “Controller details” on page 160.)

To define the Special resources:

1. Select option **32 - OSR** Special resources from the Entry type selections for Product Automation panel and press Enter. The Entry Name Selection panel that is shown in Figure 4-60 displays.

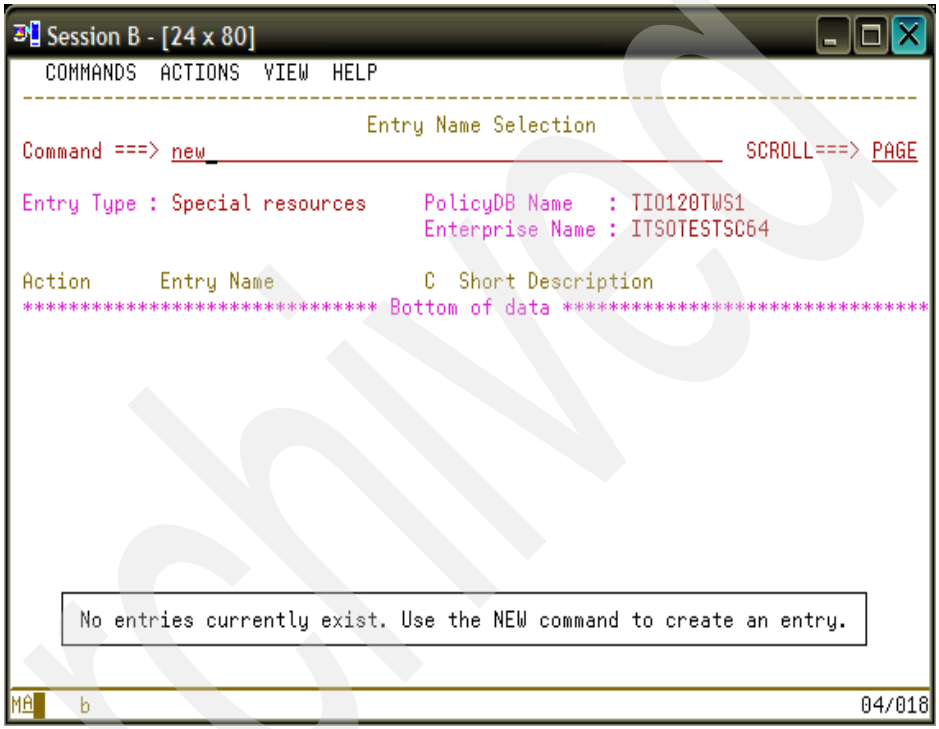

Figure 4-60 Entry Name Selection for OSR entry

2. Type **new** on the command line and press Enter. The Define New Entry panel that is shown in Figure 4-61 on page 168 displays.

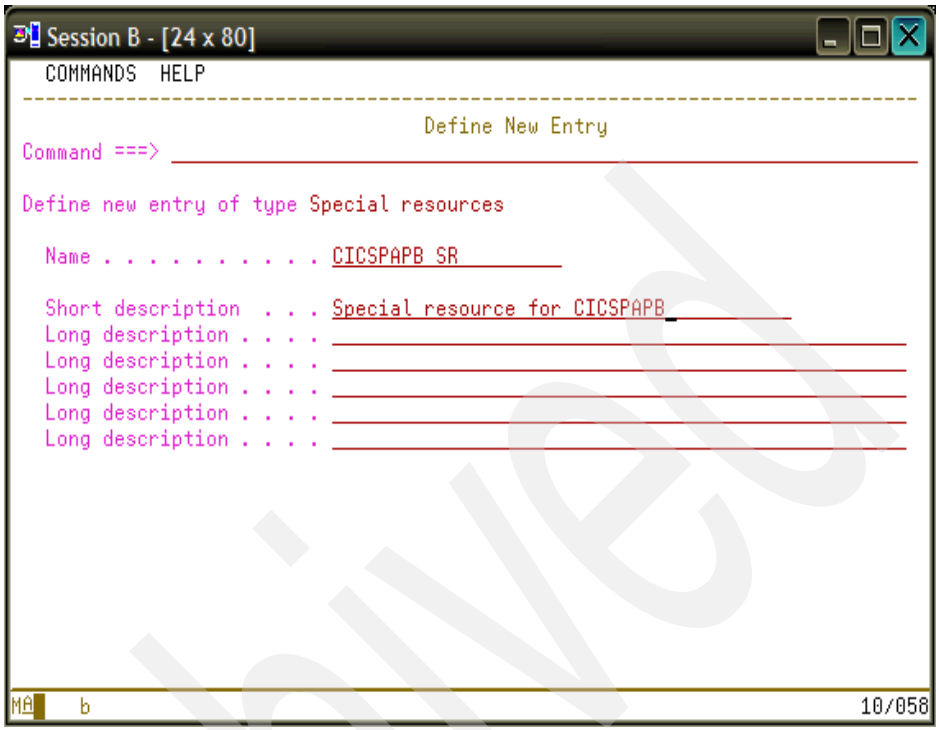

Figure 4-61 Define New Entry panel for OSR entry

3. Enter a name for the entry and a short description. For this scenario, we used:
   - `CICSPAPB_SR`

     We use the name of the System Automation resource as part of the name as this list only represents the one System Automation resource. If this entry represented a number of System Automation resources we would have used a more generic name.
   - `Special resource for CICSPAPB`

     This is the short description for our entry.
4. Press **PF3** to save your data. The Policy Selection panel that is shown in Figure 4-56 on page 162 displays.

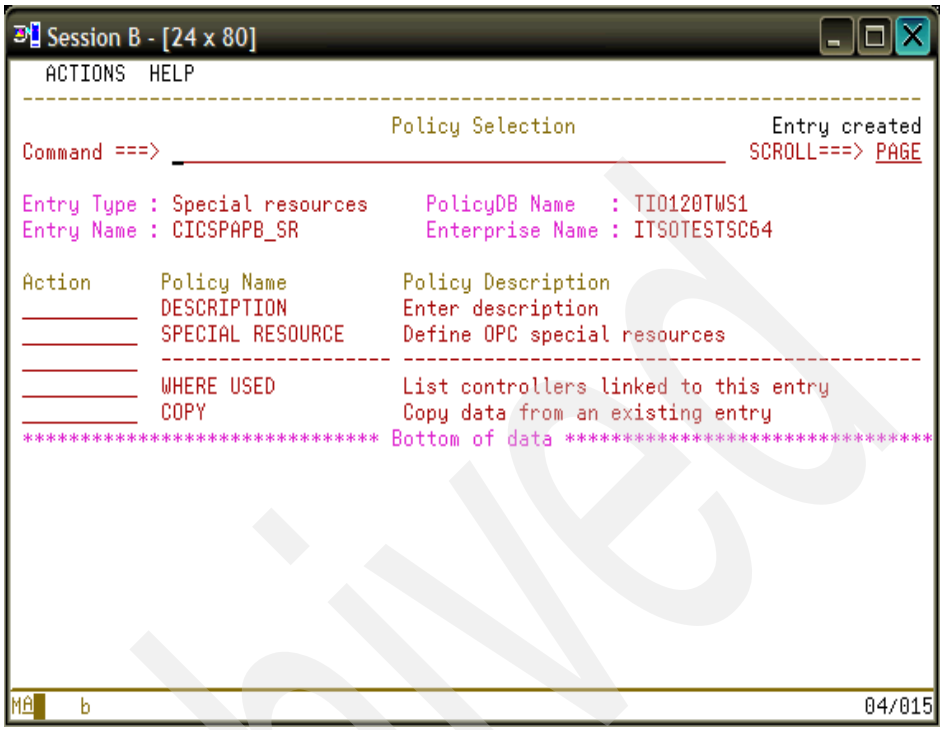

Figure 4-62 Policy Selection panel for OSR entry

5. There are two policy items on the Policy Selection panel that are used for special resources:
   - SPECIAL RESOURCES
   - WHERE USED
6. Place an s in the Action field for SPECIAL RESOURCES and press Enter. The OPC special resources definition panel that is shown in Figure 4-63 on page 170 displays.

c

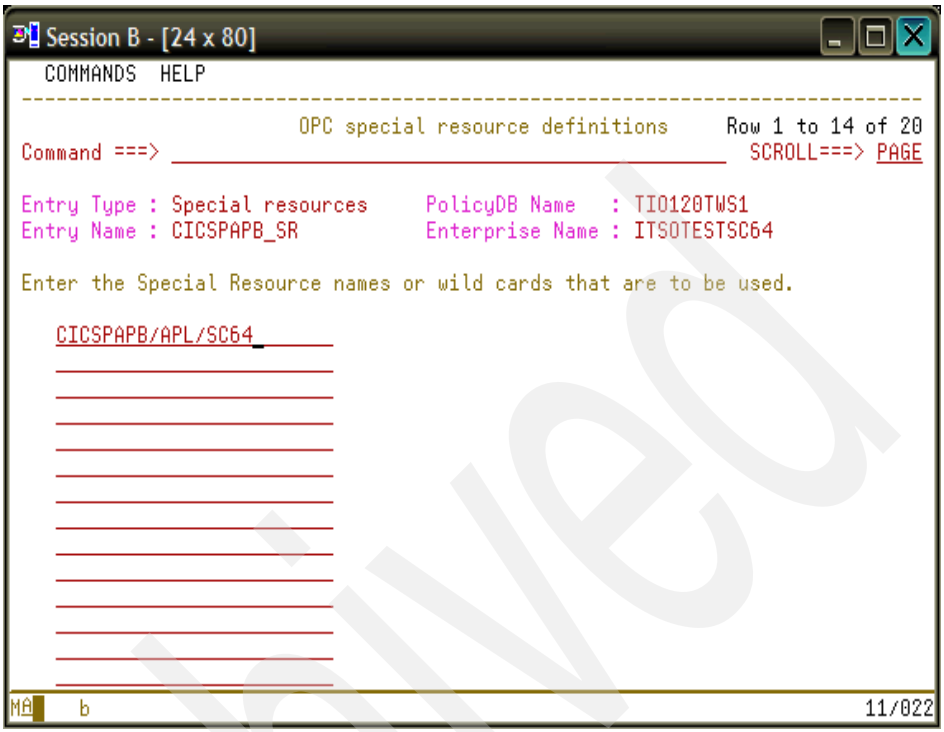

Figure 4-63 OPC special resource definitions for OSR entry

7. Enter the name of the System Automation resource which is to have a corresponding TWS for z/OS special resource. The System Automation resource name for our CICSPAPB system is `CICSPAPB/APL/SC64`. This name is converted into two TWS for z/OS special resource names by System Automation:

   – ING.SC64.APL.CICSPAPB.DOWN

     The availability flag of this TWS for z/OS special resource is `Y` when CICSPAPB is down and `N` when it is up.

   – ING.SC64.APL.CICSPAPB.UP

     The availability flag of this TWS for z/OS special resource is `N` when CICSPAPB is down and `Y` when it is up.

   The availability flag for each TWS for z/OS special resource should be opposite to each other because one represents CICSPAPB being up and the CICSPAPB being down.

8. One complete press **PF3** to return to the Policy Selection panel.

9. Place an s in the Action field for WHERE USED. The Where Used panel that is shown in Figure 4-64 displays.

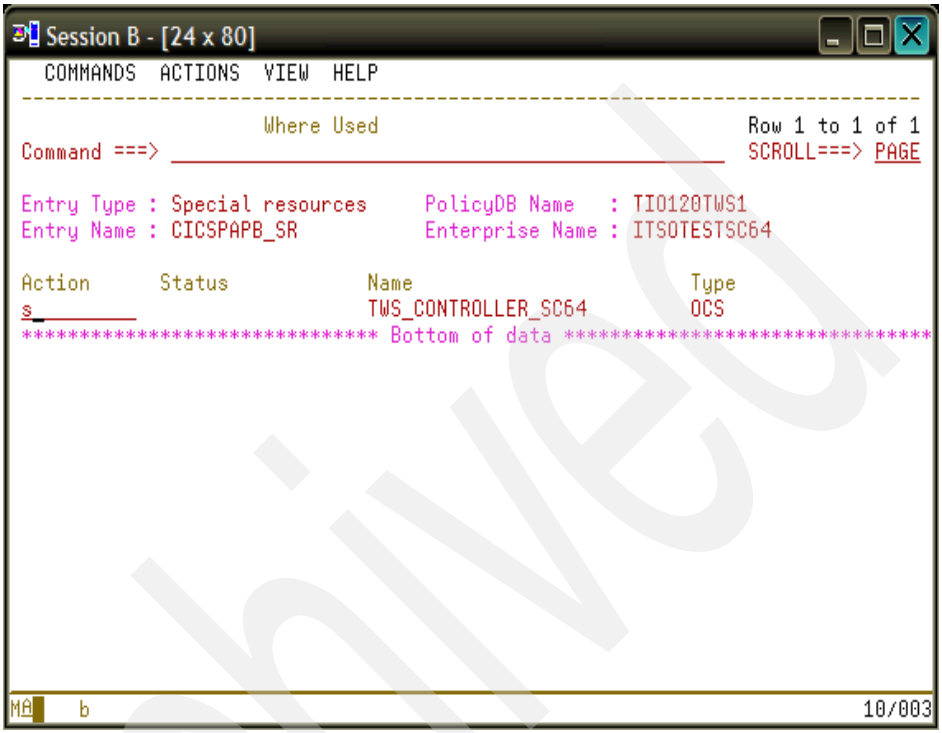

Figure 4-64 Where used panel for OSR entry

This panel shows a list of available TWS for z/OS controllers (OCS entries) with which to associate this OSR entry. We only have one entry in our scenario.

10. Place a s in the Action column next to the required TWS for z/OS controller entry and press Enter. The status field changes to SELECTED. Press **PF3** three times to return to the Entry type selections for Product Automation panel.

11. This completes the definition of an OSR entry. Repeat this procedure if you need to create more entries.

Workstation domainID

This entry provides System Automation with the information that it requires to determine which System Automation on which system a particular NV*xx* TWS for z/OS workstation represents. This entry is called the Workstation domainID entry (ODM).

To define the Workstation domainID:

1. Select option **33 - ODM** Workstation domainID from the Entry type selections for Product Automation panel. The Entry Name Selection panel that is shown in Figure 4-65 displays.

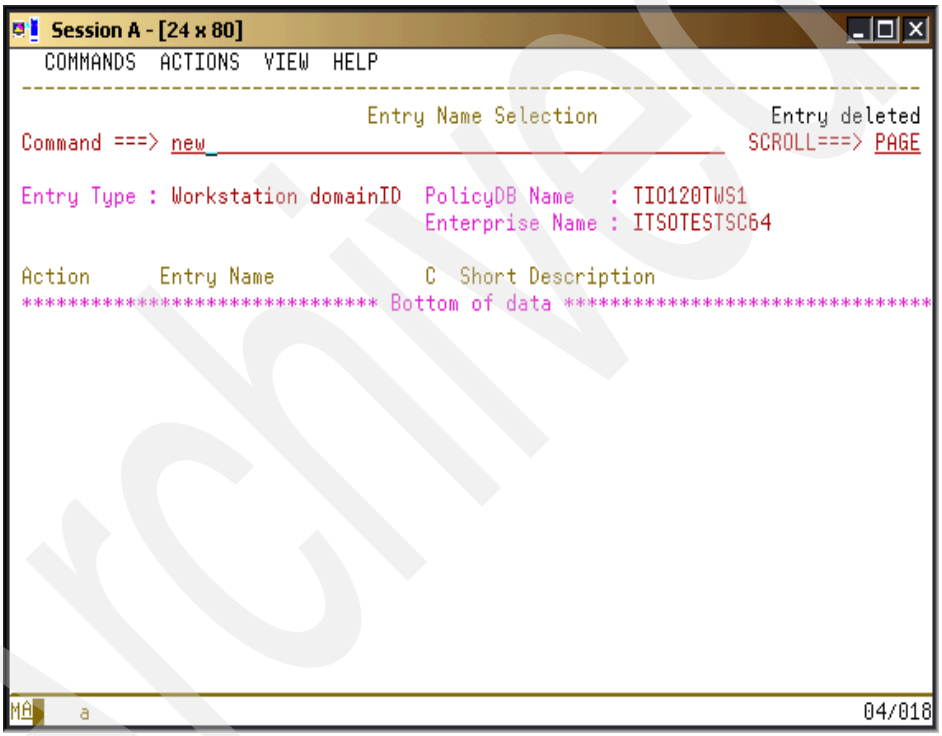

Figure 4-65 Workstation domainID Entry Name Selection panel

2. Type **new** on the command line and press Enter. The Define New Entry panel that is shown in Figure 4-66 on page 173 displays.

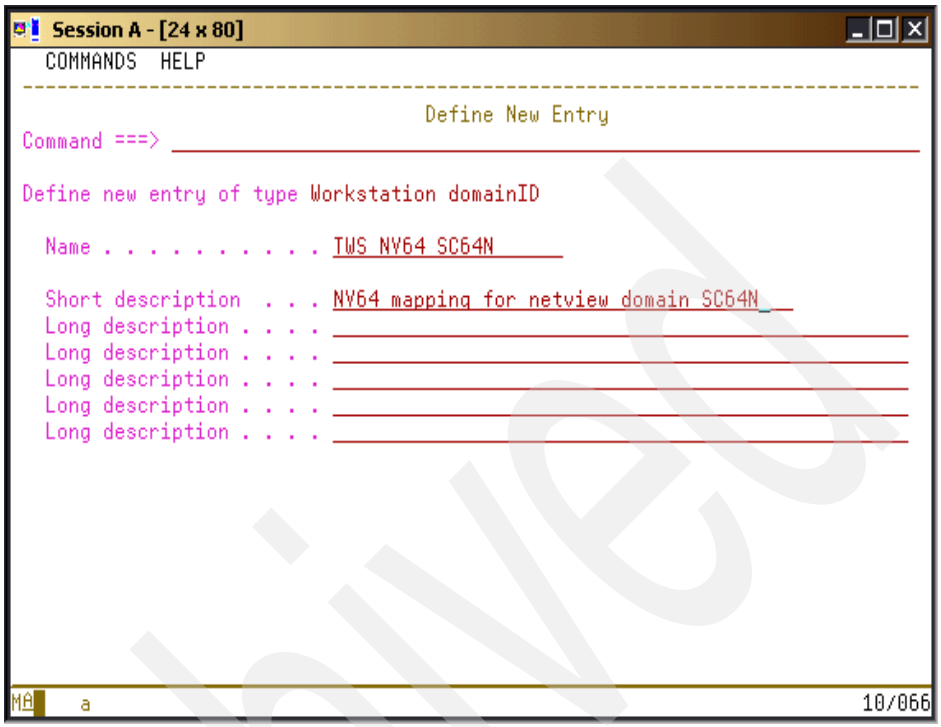

Figure 4-66 Workstation domainID Create New Entry panel

3. Enter a name for the entry and a short description. For our scenario, we used:
   - `TWS_NV64_SC64N` as the name for our entry.
   - `NV64 mapping for netview domain SC64N` as the short description for our entry.
4. Press **PF3** to save your data. The Policy Selection panel that is shown in Figure 4-67 on page 174 displays.

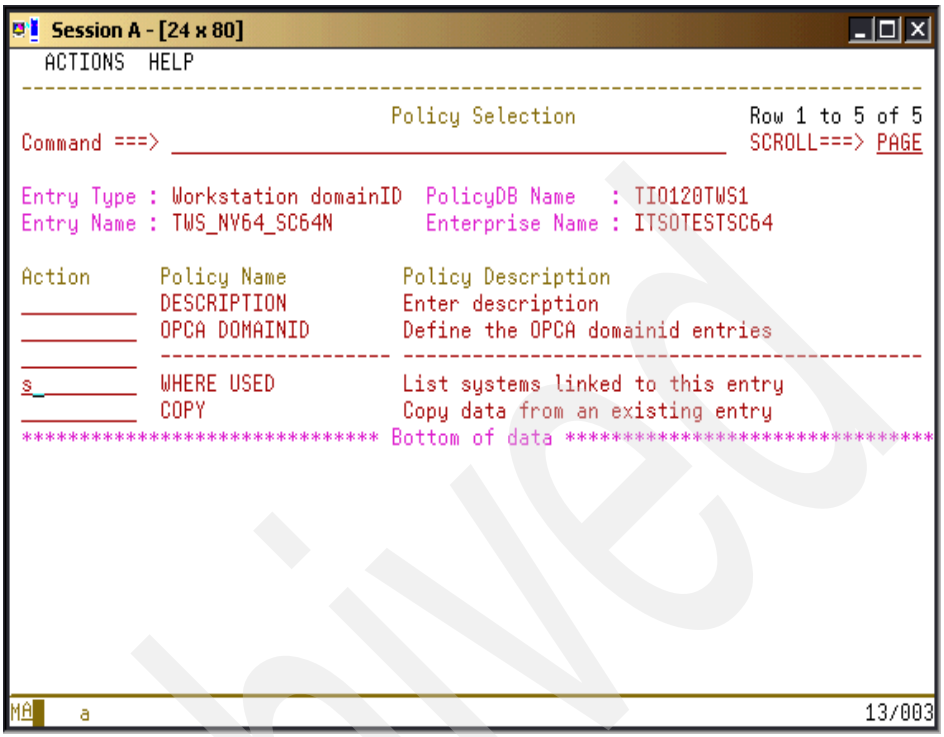

Figure 4-67 Workstation domainID Policy Selection panel

5. Place an s in the Action field for the OPCA DOMAINID. The Code Processing panel that is shown in Figure 4-68 on page 175 displays.

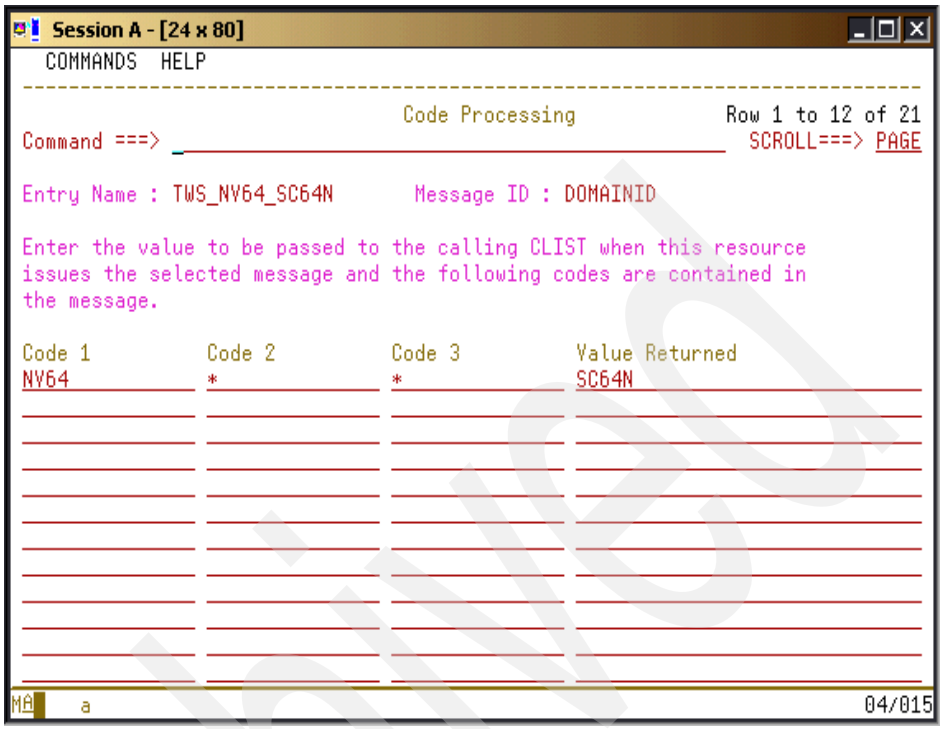

Figure 4-68 Code Processing panel for ODM entry

6. Complete the Code processing panel as follows:
   - Place the name of your NVxx TWS for z/OS workstation in Code 1 column. Our workstation name is NV64.
   - Place an asterisk (*) in Code 2 and Code 3 columns.
   - Place the name of your NetView domain in Value Returned column. Our NetView domain is SC64N
7. Press **PF3** to return to the Entry Name Selection panel. Place an s in the action column next to Where Used. The Where Used panel that is shown in Figure 4-69 on page 176 displays.

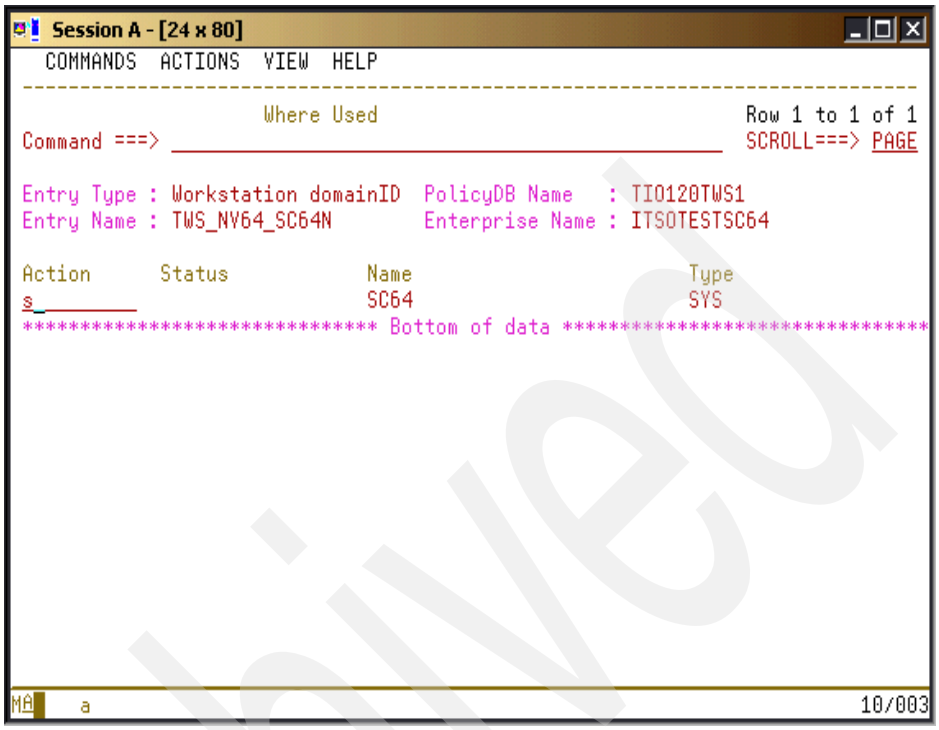

Figure 4-69 Workstation domainID Where Used panel

8. This panel should show all systems that are defined in the SYSPLEX PDB. Select the system on which this workstation is used by placing an `s` in the Action column next to the system and pressing Enter. (We only have one system in our SYSPLEX PDB which is SC64.) The Status column changes to `SELECTED` as shown in Figure 4-70 on page 177.

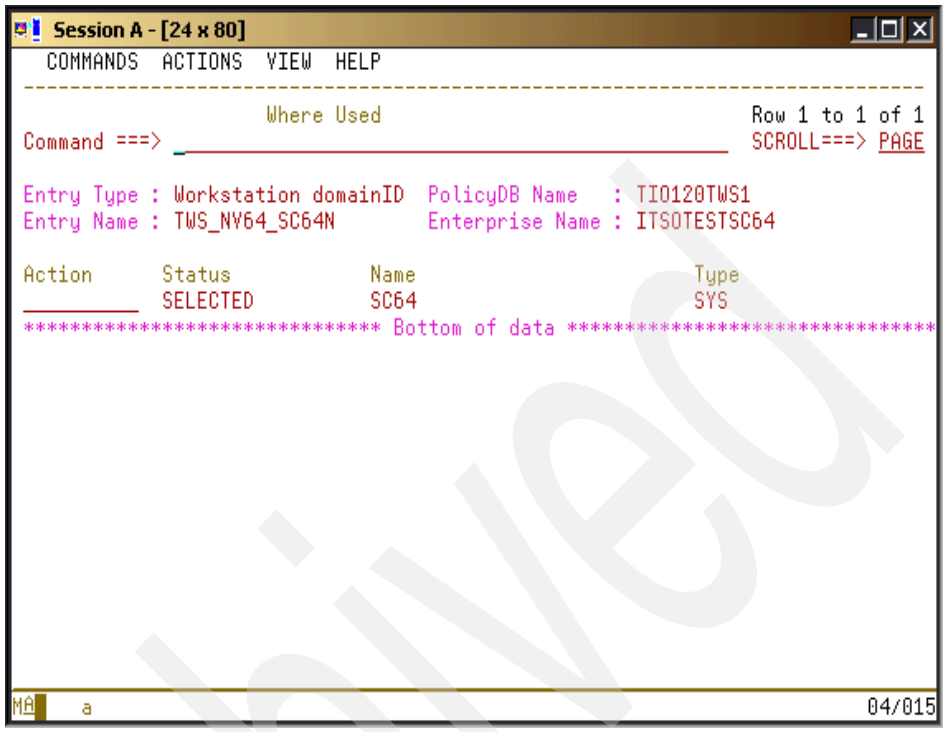

Figure 4-70 Workstation domainID Where Used panel after 's' action

9. This completes the ODM entry. Press **PF3** until you return to the Entry type selections for Product Automation panel.

This completes the System Automation part of the TWS for z/OS to System Automation interface.

4.1.4 Defining work that uses the Tivoli Workload Scheduler for z/OS NV64 workstation

We used a simple scenario to test the TWS for z/OS to OPC automation option of System Automation interface. It consists of three TWS for z/OS applications:

- The first application contains an operation which stops CICSPAPB.

  To prevent this operation from executing if CICSPAPB is already down, the TWS for z/OS special resource ING.SC64.APL.CICSPAPB.UP must be available.

  Figure 4-71 shows the operation which shuts down CICSPAPB.

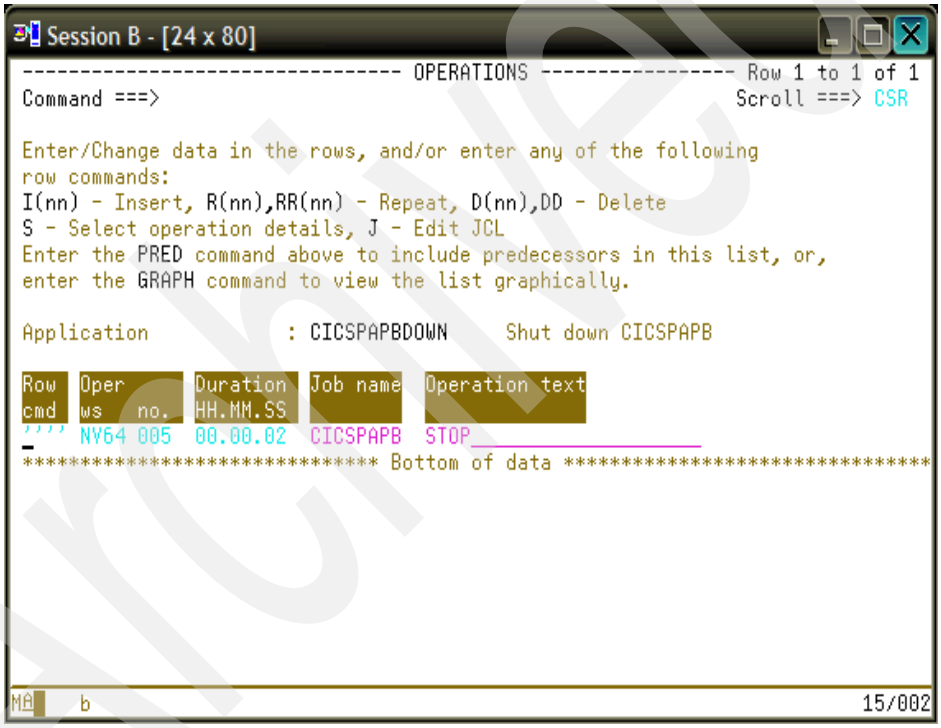

Figure 4-71 Operation to shutdown CICSPAPB

This operation uses the NV64 workstation. The operator text `STOP` is passed to System Automation. System Automation looks for an application in the SYSPLEX PDB called CICSPAPB. After it finds this entry, it checks the shut down policy. Figure 4-72 on page 179 shows the shut down policy for CICSPAPB.

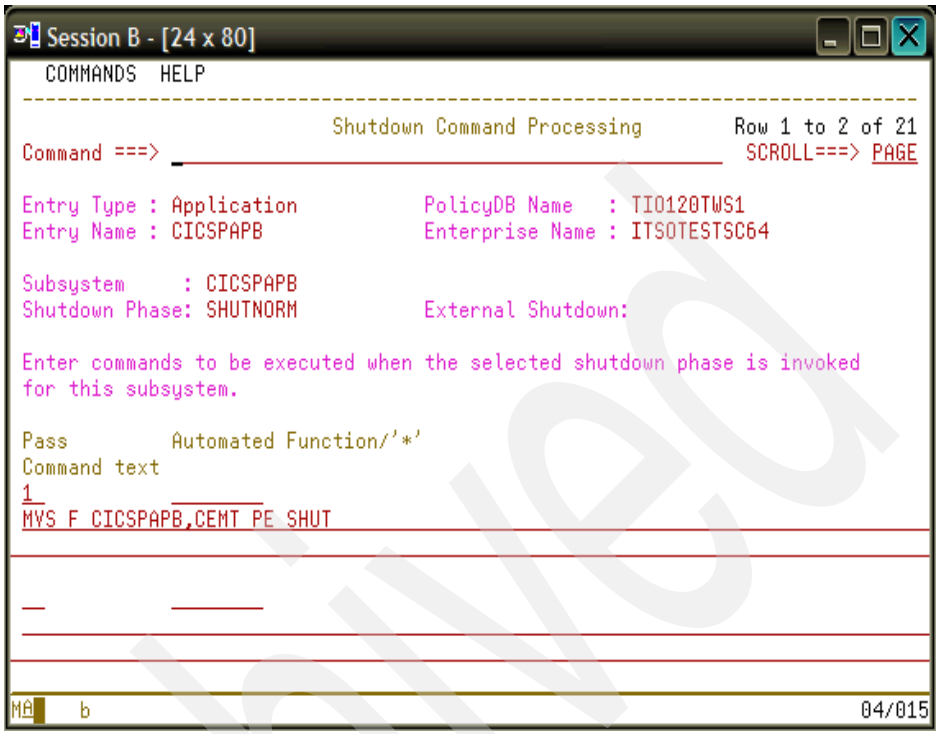

Figure 4-72 Shutdown Command Processing for CICSPAPB

System Automation issues the following z/OS command:

```
F CICSPAPB,CEMT PE SHUT
```

After CICSPAPB successfully stops, the operation in TWS for z/OS is marked complete. The z/OS status observer then changes the TWS for z/OS special resource ING.SC64.APL.CICSPAPB.DOWN to available and ING.SC64.APL.CICSPAPB.UP to unavailable.

- The second application has a dummy batch job.

  CICSPAPB must be down for this job to run. For this job, the TWS for z/OS special resource ING.SC64.APL.CICSPAPB.DOWN must be available.

  This TWS for z/OS special resource is marked available by the z/OS status observer after CICSPAPB has successfully stopped.

- The third application contains an operation which starts CICSPAPB.

  To prevent this operation from executing if CICSPAPB is already down, the TWS for z/OS special resource ING.SC64.APL.CICSPAPB.DOWN must be available.

Figure 4-73 shows the operation which starts CICSPAPB.

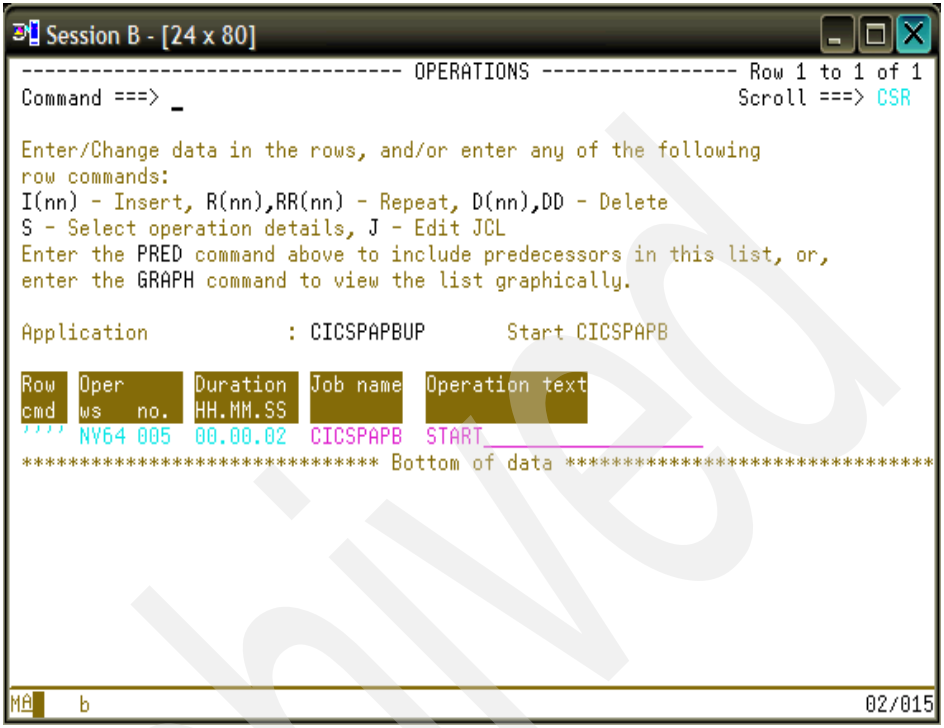

Figure 4-73 Operation to start CICSPAPB

This operation uses the NV64 workstation. The operator text `START` is passed to System Automation. System Automation looks for an application in the SYSPLEX PDB called CICSPAPB. After it finds this entry, it checks the start up policy. Figure 4-74 on page 181 shows the start up policy for CICSPAPB.

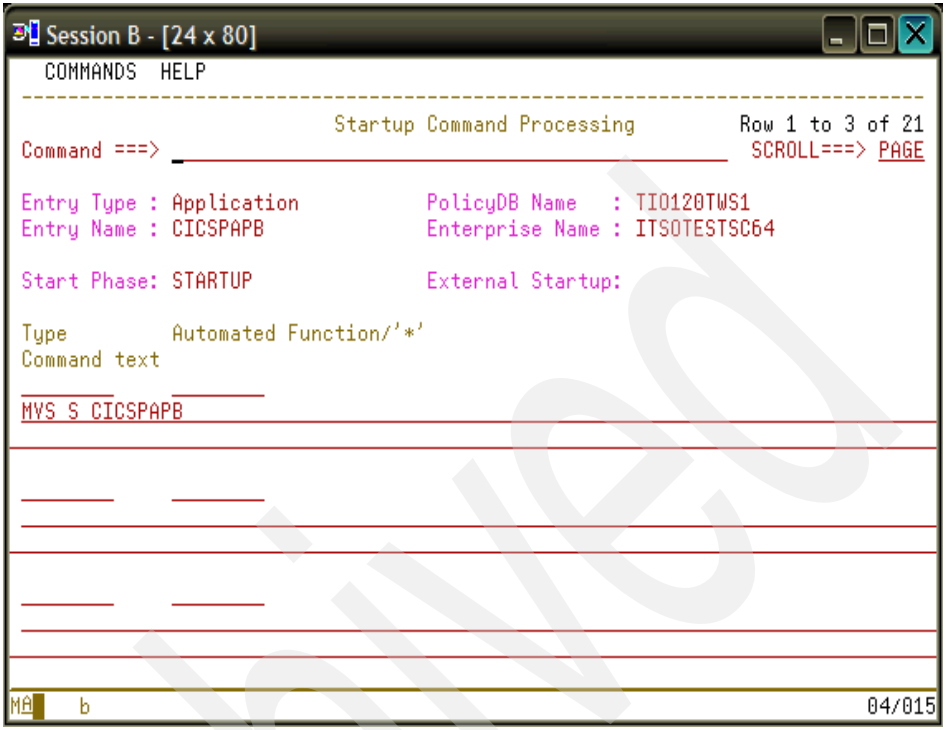

Figure 4-74 Startup Command Processing for CICSPAPB

System Automation issues the following z/OS command:

```
S CICSPAPB
```

After CICSPAPB successfully starts, the operation in TWS for z/OS is marked complete. The z/OS status observer then changes the TWS for z/OS special resource ING.SC64.APL.CICSPAPB.UP to available and ING.SC64.APL.CICSPAPB.DOWN to unavailable.

This scenario is very simple. It uses a simple start and stop command and two special resources. System Automation is capable of much more complex processing by using the TEXT field of a TWS for z/OS operation.

For more information see:

- *OPC Automation Programmer's Guide and Operator's Reference*, SC33-7046
- *Defining Automation Policy*, SC33-7039

4.2 Issues encountered during the interface activation

We experienced one issue due to an error in our automation policy definition. This error is described in the *OPC Automation Programmer's Guide and Operator's Reference manual*, SC33-7046.

The error was the TWS for z/OS operation ending in error with a code of 003. The description from the manual is:

> If OPC Automation does not find the NV*xx* index then OPC Automation issues a message, posts the operation status to E (ended-in-error, U003), and logs the results. No communications can occur with this workstation until the definition is corrected. On the domain where the OPC controller is running, the workstation must be defined in the WORKSTATION DOMAINS policy object (ODM entry type).

Our ODM entry was incorrect. It did not specify the workstation name NV64. This issue was corrected, and the interface functioned normally.

4.3 Successful implementation of the Tivoli Workload Scheduler for z/OS and System Automation interface

This chapter has shown the basic implementation of both the TWS for z/OS part and System Automation part of the interface. Our test scenario was used to test the basic part of the interface. We did not test any complex scenarios. We also did not test the TWS Batch Command server.

A good knowledge of System Automation is required to implement this interface. What we have shown in this chapter allows someone with little System Automation knowledge to use the System Automation ISPF dialog to define the required TWS definitions. However, there are naming standards to consider when defining entries in System Automation. The System Automation administrator should be involved at this level.

We found it very useful to be able to logon to NetView where System Automation was active and browse the NETLOG (BR NETLOGA command). This enabled us to see any commands being triggered by TWS for z/OS and any problems with the interface. This was particularly important for the invalid entry we had created.

We did not show the building or activation of the System Automation policy because it is beyond the scope of this chapter.

5

Integrating Tivoli Business Systems Manager

This chapter discusses the integration of Tivoli Workload Scheduler for z/OS (TWS for z/OS) with Tivoli Business Systems Manager. Tivoli Business Systems Manager is used to monitor critical business systems, allowing support organizations, application owners, and management to view system components according to organizational priority or as they relate to overall business needs. Business systems can include resources ranging from z/OS to distributed system resources. Tivoli Business Systems Manager integrates with the resources from TWS for z/OS to build business systems that reflect the current status of critical jobs and resources belonging to TWS for z/OS.

Tivoli Business Systems Manager inputs the applications from the TWS for z/OS daily plan into batch schedule resources. By using available pattern-matching rules, resources can be grouped into business systems that reflect the logical relationship of applications, the physical locations, resources, or other business needs of your organization. The Batch Management Summary view and the Event Viewer display the status of current operations with each application.

This chapter provides details on configuring TWS for z/OS V8.2 to work with Tivoli Business Systems Manager 3.1.

5.1 Description of the environment and systems

This section describes the systems and configuration that we used for integrating TWS for z/OS with Tivoli Business Systems Manager. Figure 5-1 shows the systems in our environment.

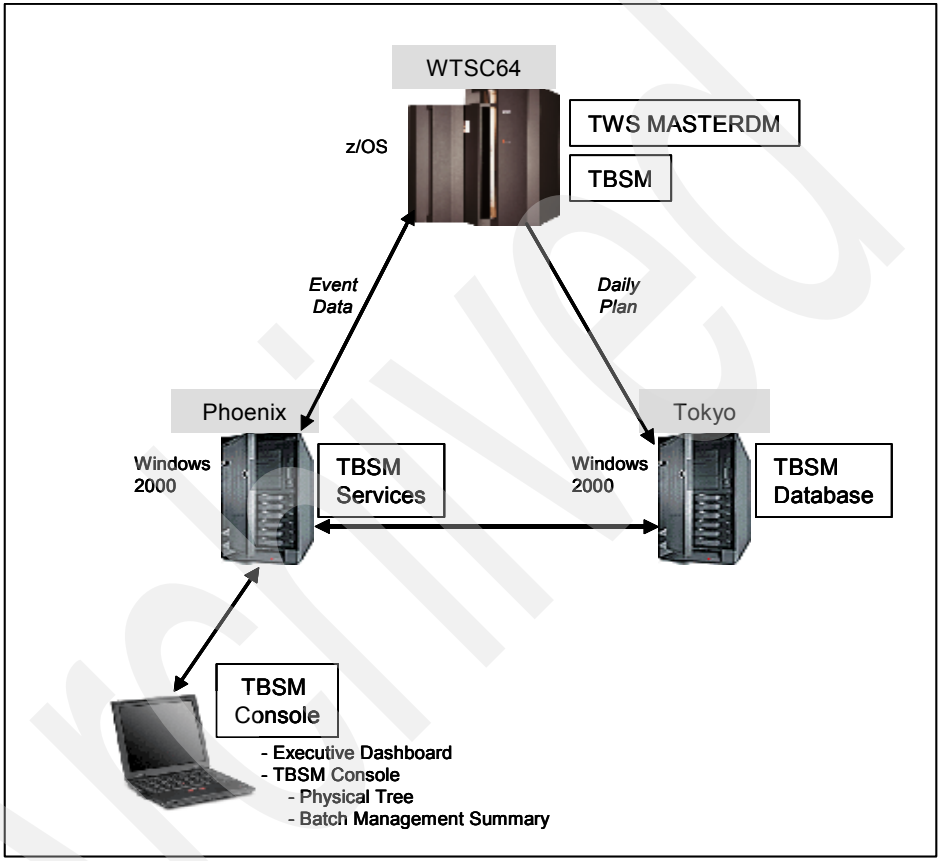

Figure 5-1 Systems and configuration that were used for this chapter

The following started task procedures are defined in z/OS for both products:

TWSC	TWS for z/OS Controller
GTMDSPC	Tivoli Business Systems Manager Dataspace
GTMSRVR	Tivoli Business Systems Manager Object Server
GTMPUMP	Tivoli Business Systems Manager Object Pump

Server Tokyo contains the Microsoft® SQL Database for Tivoli Business Systems Manager. Server Phoenix is configured with services that communicate with the mainframe and the console server for Tivoli Business Systems Manager. You can find instructions for configuring Tivoli Business Systems Manager in the *Tivoli Business Systems Manager V3.1 Installation and Configuration Guide,* SC32-9089.

5.2 Forwarding events from Tivoli Workload Scheduler for z/OS to Tivoli Business Systems Manager

Configuring TWS for z/OS and Tivoli Business Systems Manager consists of two steps. The first step is to configure both products to communicate with each other. The second step is to mark jobs in TWS for z/OS to forward events to Tivoli Business Systems Manager, resulting in an event-driven view. The details for configuration of loading a daily plan and for creating lines of business are discussed in 5.3, “Creating lines of business using Daily Plan data” on page 190.

5.2.1 Configuring communications

To establish communication between the mainframe components of Tivoli Business Systems Manager on SC64 with server Phoenix, you need to turn on the TWS for z/OS communication with Tivoli Business Systems Manager on the mainframe as follows:

1. In the OPCOPTS initialization statement for TWS for z/OS, add the following to turn on external monitoring of jobs.

   ```
   EXTMON(YES)
   ```

2. In the Tivoli Business Systems Manager source/390 object pump start up parameters, add the following statement to point to the TWS for z/OS controller task, where the controller task is named TWSC:

   ```
   OPC_JOBNAME=TWSC
   ```

After the tasks are started, the Tivoli Business Systems Manager exit module AOPEDI is called by TWS for z/OS. If the communication fails, an EQQZ232 message would be expected in the TWS for z/OS EQQMLOG. The lack of such a message indicates that communication is established. Status events for jobs that are marked for external monitoring are received by Tivoli Business Systems Monitor.

5.2.2 Turning on external monitoring for jobs

Individual operations in TWS for z/OS are not externally monitored by default. External monitoring must be turned on for each job that is monitored by Tivoli Business Systems Monitor. There are two ways to turn on external monitoring in TWS for z/OS:

- Jobs are marked using the automatic options of the operation details of the job definition accessed through option 1.4.3 from the TWS for z/OS primary options menu (panel EQQAMJBP).

  Jobs that are monitored by Tivoli Business Systems Manager should have the setting for “EXTERNAL MONITOR” set to Y from the default of N. This setting is used for individual jobs. Example 5-1 shows an example of this setting in panel EQQAMJBP.

Example 5-1 Panel EQQAMJBP

```
EQQAMJBP ---------------- JOB, WTO, AND PRINT OPTIONS ----------------------
 Command ===>
 Enter/Change data below:
 Application            : BPIB4WORKDAY
 Operation              : CPU1 005
 Job name               : BPIJOBA
 JOB CLASS          ===> _            Job class of corresponding job
 ERROR TRACKING     ===> Y            Y means error automatically tracked
 HIGHEST RETURNCODE ===> ____         Highest return code not in error
 EXTERNAL MONITOR   ===> Y            Job monitored by external product (Y/N)
 CENTRALIZED SCRIPT ===> N            Centralized script Y/N (for FTW only)
 CRITICAL  ===> N   POLICY  ===> _    WLM critical job (Y/N) and assist policy
 Job release options:                 Answer Y or N for options below:
  SUBMIT            ===> Y            Automatically submitted
  HOLD/RELEASE      ===> Y            Automatically held and released
  TIME DEPENDENT    ===> N            Run the job at specified time
  SUPPRESS IF LATE  ===> N            Suppress the job if not on time
  DEADLINE WTO      ===> N            Deadline WTO, Y or N
 WS fail options:
  RESTARTABLE       ===> _            Operation is restartable
  REROUTEABLE       ===> _            Operation is eligible for reroute
 Print options:
  FORM NUMBER       ===> ________
  SYSOUT CLASS      ===> _
```

- The mass update function is available using 1.4.5 from the primary TWS for z/OS panel.

  On panel EQQAUPDL there is an OPR - EXTERNAL MONITOR option that updates a number of jobs at one time. Example 5-2 shows this panel scrolled down to display the OPR - EXTERNAL MONITOR option.

Example 5-2 Panel EQQAUPDL

```
EQQAUPDL --------- MASS UPDATING OF APPLICATION DESCRIPTION Row 37 to 48 of 57
 Command ===>                                                 Scroll ===> PAGE

 Enter the row command S to create pending updates for a data item.
 Enter the CLEAR command above to delete all pending updates.

 Number of pending updates in this session: 0

 TYPE OF BATCH JOB  ===> T           T - Trial run, U - Updating run

 Row
 cmd Data item
 '   OPR - INPUT ARRIVAL TIME
 '   OPR - DEADLINE DAY
 '   OPR - DEADLINE TIME
 '   OPR - DEADLINE WTO
 '   OPR - CLEAN UP TYPE
 '   OPR - USER SYSOUT
 '   OPR - EXPANDED JCL
 S   OPR - EXTERNAL MONITOR
 '   OPR - CENTRALIZED SCRIPT
 '   OPR - EXTENDED NAME USED
 '   OPR - EXTENDED NAME
 '   OPR - RESTARTABLE
```

5.2.3 Event-driven view on Tivoli Business Systems Manager

Proper configuration allows events to flow, populating the Tivoli Business Systems Manager Console as shown in Figure 5-2. The Batch Schedule Set: SC64 object appears in the All Resources business object container under the defined physical tree for the Operating System: SC64 object. The only batch schedules and jobs that appear in this event-driven view are the externally monitored jobs that have sent an update to Tivoli Business Systems Manager.

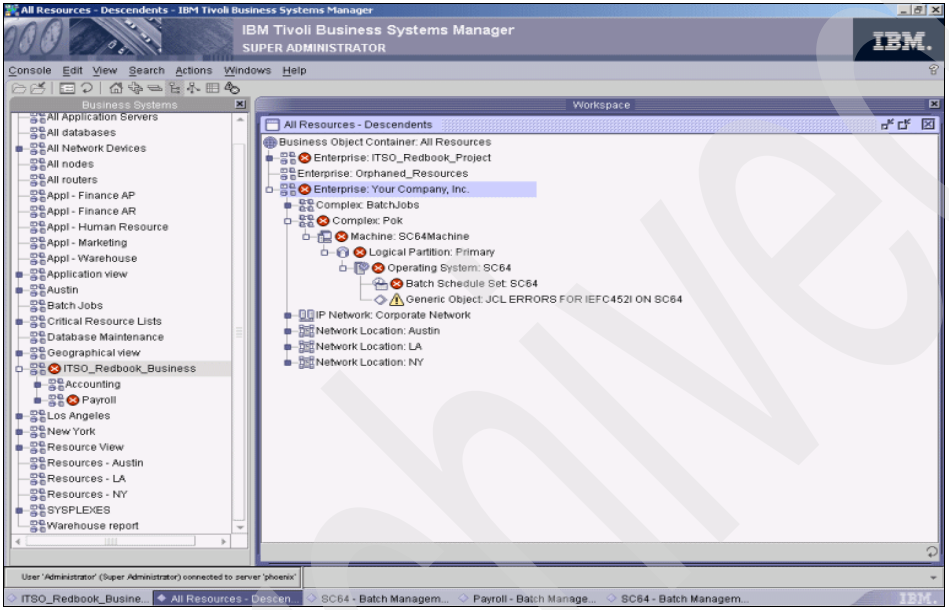

Figure 5-2 "All Resources" view of events from TWS for z/OS

Tivoli Business Systems Manager represents applications from TWS for z/OS as batch schedule sets on the All Resources console view. To see the batch schedule set contents, open the Batch Management Summary view for the batch schedule set, as shown in Figure 5-3 on page 189.

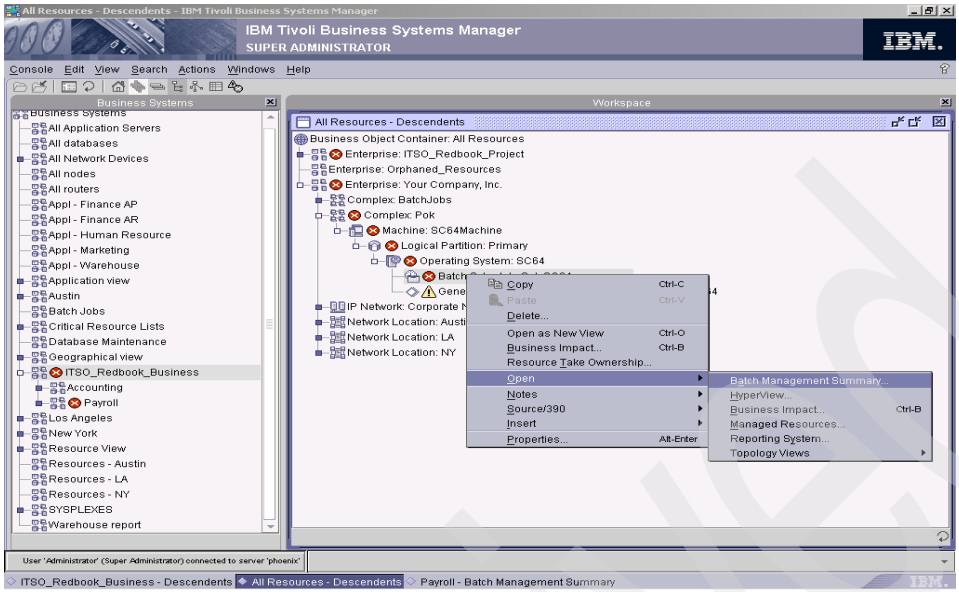

Figure 5-3 Opening the Batch Management Summary view

The Batch Management Summary view contains three horizontal panes. The top pane lists Batch Schedule Sets or the daily plans to which jobs that have events belong. The second pane shows the Batch Schedules, or applications, that own the events received in Tivoli Business Systems Manager. The third pane contains jobs that have had events.

When you choose a batch schedule set, the batch schedules for that set are listed in the second pane. When you choose a batch schedule in the second pane, jobs that belong to that schedule and have had events are displayed in the third pane. In Figure 5-4 on page 190, the batch schedule set and batch schedule that are viewed are highlighted.

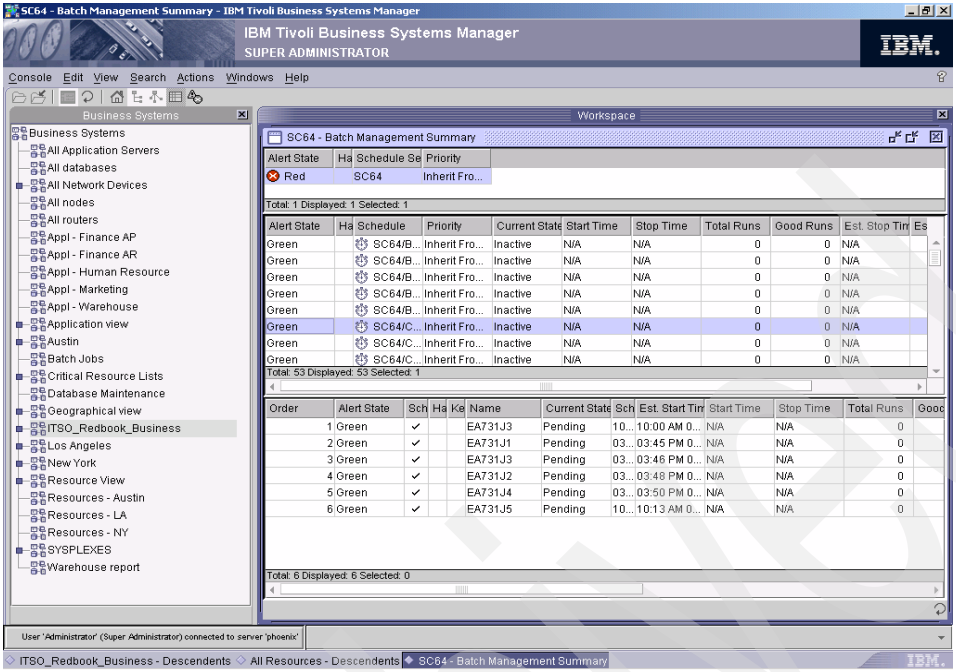

Figure 5-4 Batch Management Summary view with Batch Schedule opened

5.3 Creating lines of business using Daily Plan data

Tivoli Business Systems Manager allows resources in the business environment to be represented and organized into lines of business. A line of business contains resources that are organized in any logical manner, including application-specific resources, all resources supported by an organization, or resources in a regional area. The line of business can tie together a variety of resources, including resources from mainframe databases, UNIX servers, and Windows application servers. For TWS for z/OS resources, Tivoli Business Systems Manager can be used to organize batch schedule sets (daily plans), batch schedules (applications), and jobs in meaningful ways, either as lines of business only for TWS for z/OS or in lines of business that contain other types of resources.

Proper planning of lines of business is one of the most important administrative tasks for Tivoli Business Systems Manager. It is helpful to outline prospective lines of business and to review their usage before constructing the lines of business in the database. Further information about planning and creating lines of business is available in *Tivoli Business Systems Manager V3.1 Planning Guide*, SC32-9088.

In general, lines of business are normally created by either using “Drag and Drop” of objects into a line of business or by using the Automated Business Systems method. For TWS for z/OS, a third method is available, that allows lines of business to be built with objects from the daily plan. The strength of both Automated Business Systems and building lines of business from the daily plan is that both methods allow the Tivoli Business Systems Manager administrator to plan for future objects by using pattern-matching of resource names. Thus, similarly named resources can be added “on the fly” as they are discovered. “Drag and Drop” allows for quick creation of lines of business, but only uses resources that have been created.

Tip: Proper naming conventions allow for easy creation of lines of business when using Automated Business Systems or creating lines of business from the daily plan.

A recommended method of building lines of business is to create small lines of business and to use the small lines of business to build larger lines of business, as though they were modules. A use of this method can be seen in Example 5-3 on page 193, where both East_Coast and West_Coast lines of business are created for the Payroll line of business. The same East_Coast and West_Coast lines of business could also be incorporated into separate regional lines of business without being redefined. Similarly, the Payroll line of business or its children could be used in another line of business for yet another view.

Important: You should create lines of business using the daily plan data method or using the Automated Business Systems method before the daily plan is loaded. You should create “Drag and Drop” lines of business after the daily plan is loaded.

5.3.1 Creating the .sqi file to define lines of business

For the test environment in our lab, we created a single parent line of business named ITSO_Redbook_Business, with child lines of business named Accounting and Payroll. The Payroll line of business has two children lines of business named East_Coast and West_Coast. Criteria were defined to populate the lines of business. The population of TWS for z/OS objects into lines of business is defined using a dynamic resource hierarchy definition file. Basic instructions about how to use this file are found in the *IBM Tivoli Business Systems Manager V3.1 Administrator's Guide*, SC32-9085.

The .sqi script that we used to define our lines of business is described in Example 5-3 on page 193. A discussion of the script follows. Our sample script is named hierarchy.sqi, but any .sqi name is acceptable.

Note: The scripts in this chapter are indented by level of complexity. We recommend that you do the same in your scripts.

Example 5-3 Sample file hierarchy.sqi

```
include(BUsinessObject.sqi)

--hierarchy_3.sqi

BEGIN_DYNA_OBJ_PATH(TWS, TWS Path for applications)
  DYNA_OBJ_PATH(BUSC, BUSC)
    DYNA_OBJ_PATH(ENT, ITSO_Redbook_Project)
      DYNA_OBJ_PATH(COMP, SC64_Computer)
        DYNA_OBJ_PATH(MACH, SC64_Machine)
         DYNA_OBJ_PATH(LPAR, SC64_LPAR)
          DYNA_OBJ_PATH(OS, SC64)
           DYNA_OBJ_PATH(BCYS, SC64,  SC64_DailyPlan)
END_DYNA_OBJ_PATH(TWS)

--Define the mapping of TWS
OPC_APP_PATTERN(TWS, dailyplan, %, %, %)

BEGIN_DYNA_OBJ_PATH(ACCOUNTING, Show all General Ledger applications)
  DYNA_OBJ_PATH(LOBC, LOBC)
    DYNA_OBJ_PATH(LOB,ITSO_Redbook_Business)
      DYNA_OBJ_PATH(LOB, Accounting)
        DYNA_OBJ_PATH(LOB, General_Ledger)
END_DYNA_OBJ_PATH(ACCOUNTING)

--Define the mapping of ACCOUNTING
OPC_APP_PATTERN(ACCOUNTING, dailyplan, GLDLY%)


BEGIN_DYNA_OBJ_PATH(East_Coast, Show all Payroll applications)
  DYNA_OBJ_PATH(LOBC, LOBC)
    DYNA_OBJ_PATH(LOB,ITSO_Redbook_Business)
      DYNA_OBJ_PATH(LOB, Payroll)
        DYNA_OBJ_PATH(LOB, East_Coast)
END_DYNA_OBJ_PATH(East_Coast)

BEGIN_DYNA_OBJ_PATH(West_Coast, Show all Payroll applications)
  DYNA_OBJ_PATH(LOBC, LOBC)
    DYNA_OBJ_PATH(LOB,ITSO_Redbook_Business)
      DYNA_OBJ_PATH(LOB, Payroll)
        DYNA_OBJ_PATH(LOB, West_Coast)
END_DYNA_OBJ_PATH(West_Coast)

--Define the mapping of PAYROLL
OPC_APP_PATTERN(East_Coast, dailyplan, CS%)
OPC_APP_PATTERN(West_Coast, dailyplan, U%)
```

Each hierarchy in the sample file is specified by a beginning line and an ending line. The lines between define the objects that will appear in the line of business that is being defined.

```
BEGIN_DYNA_OBJ_PATH(<Name>, <Description>)
.
.
END_DYNA_OBJ_PATH(<Name>)
```

Hierarchies can contain objects of the classes that are listed in Table 5-1. You can find more information about Tivoli Business Systems Manager object types in the *IBM Tivoli Business Systems Manager V3.1 Administrator's Guide*, SC32-9085.

Restriction: Individual jobs are not defined in hierarchy files. They are displayed in lines of business as children of the appropriate batch schedule (BTCY), when the batch schedule is defined in the line of business.

Table 5-1 Objects that are used in dynamic resource hierarchical definition file

Class ID	Class Name
BUSC	BusinessObjectContainer
ENT	Enterprise
COMP	Complex
BCYS	BatchScheduleSet
BTCY	BatchSchedule
LOBC	LineOfBusinessContainer
LOB	LineOfBusiness

Objects are defined in the physical tree with the section of the hierarchy script that is in Example 5-4 on page 195. The resulting physical tree is seen in Figure 5-5 on page 195.

Important: When defining the physical tree, we strongly recommend that the batch schedule set name and the OS name are the same, as in our example. This aids in autodiscovery of new objects during event processing. Because additional applications might be added after the daily plan has been uploaded, the TWS environment can continue to correctly display status on the Tivoli Business Systems Manager console.

Example 5-4 Physical Tree definition from hierarchy.sqi

```
BEGIN_DYNA_OBJ_PATH(TWS, TWS Path for applications)
  DYNA_OBJ_PATH(BUSC, BUSC)
    DYNA_OBJ_PATH(ENT, ITSO_Redbook_Project)
      DYNA_OBJ_PATH(COMP, SC64_Computer)
        DYNA_OBJ_PATH(MACH, SC64_Machine)
         DYNA_OBJ_PATH(LPAR, SC64_LPAR)
          DYNA_OBJ_PATH(OS, SC64)
           DYNA_OBJ_PATH(BCYS, SC64,  SC64_DailyPlan)
END_DYNA_OBJ_PATH(TWS)

--Define the mapping of TWS
OPC_APP_PATTERN(TWS, dailyplan, %, %, %)
```

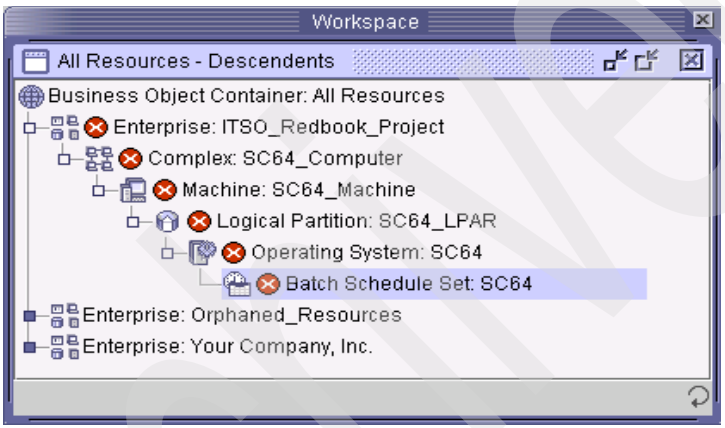

Figure 5-5 Physical Tree view from Hierarchy.sqi

At the bottom of the defined hierarchy, the mapping is defined. The mapping, OPC_APP_PATTERN, defines the patterns from the daily plan(s) that are used to match objects on the Tivoli Business Systems Manager console. Mapping can match on the following criteria:

- OPC_PATH_NAME, name of the dynamic object path to aggregate matching applications.
- OPC_PATH_PATTERN, wildcard expression matched to the names of the defining daily plan.
- OPC_APP_PATTERN, wildcard expression matched to the application IDs.
- OPC_OWNER_PATTERN, wildcard expression matched to the application owner fields.
- OPC_DESC_PATTERN, wildcard expression matched to the application description fields.

Wildcard substitution percent character (%) and expression [..] are used according to SQL rules. Wildcard expressions are discussed further in the *Tivoli Business Systems Manager V3.1 Administrator's Guide*, SC32-9085.

Lines of business are also defined in the hierarchy.sqi script. The Accounting line of business hierarchy is shown in Example 5-5 and the resulting line of business view is seen in Figure 5-6. Important differences between the physical view and line of business views include the use of LOBC and LOB objects in the line of business. The OPC_APP_PATTERN used in Example 5-5 matches on three variables, including matching with all application patterns that begin with GLDLY. The line of business in Figure 5-6 shows this section of the line of business expanded to display the jobs that are children of the batch schedule (application).

Example 5-5 Line of Business Accounting definition from Hierarchy.sqi

```
BEGIN_DYNA_OBJ_PATH(ACCOUNTING, Show all General Ledger applications)
  DYNA_OBJ_PATH(LOBC, LOBC)
    DYNA_OBJ_PATH(LOB,ITSO_Redbook_Business)
      DYNA_OBJ_PATH(LOB, Accounting)
        DYNA_OBJ_PATH(LOB, General_Ledger)
END_DYNA_OBJ_PATH(ACCOUNTING)

--Define the mapping of ACCOUNTING
OPC_APP_PATTERN(ACCOUNTING, dailyplan, GLDLY%)
```

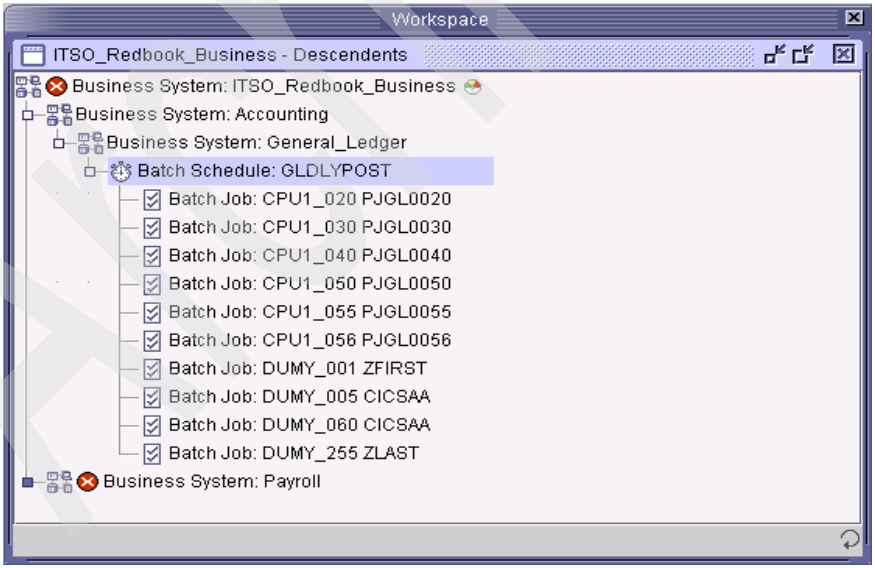

Figure 5-6 Line of Business Accounting view from hierarchy.sqi

A more complicated line of business example is also in the hierarchy.sqi file. The Payroll line of business is defined with two children, East_Coast and West_Coast as shown in Example 5-6. Note that Payroll does not have a separate definition but is included in the definition of both children line of businesses. Figure 5-7 shows the view of the line of business.

Example 5-6 Line of Business Payroll definition from Hierarchy.sqi

```
BEGIN_DYNA_OBJ_PATH(East_Coast, Show all Payroll applications)
  DYNA_OBJ_PATH(LOBC, LOBC)
    DYNA_OBJ_PATH(LOB,ITSO_Redbook_Business)
      DYNA_OBJ_PATH(LOB, Payroll)
        DYNA_OBJ_PATH(LOB, East_Coast)
END_DYNA_OBJ_PATH(East_Coast)

BEGIN_DYNA_OBJ_PATH(West_Coast, Show all Payroll applications)
  DYNA_OBJ_PATH(LOBC, LOBC)
    DYNA_OBJ_PATH(LOB,ITSO_Redbook_Business)
      DYNA_OBJ_PATH(LOB, Payroll)
        DYNA_OBJ_PATH(LOB, West_Coast)
END_DYNA_OBJ_PATH(West_Coast)

--Define the mapping of PAYROLL
OPC_APP_PATTERN(East_Coast, dailyplan, CS%)
OPC_APP_PATTERN(West_Coast, dailyplan, U%)
```

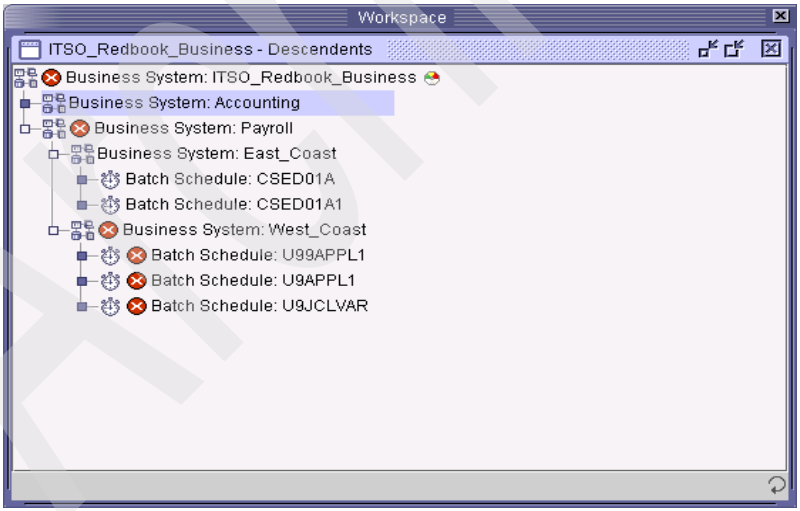

Figure 5-7 Line of Business Payroll view from hierarchy.sqi

5.3.2 Compiling the .sqi file

After the .sqi file that defines the hierarchy for the lines of business definitions is completed, you must compile the file to create an .sql file to upload to the Tivoli Business Systems Manager database as shown in Example 5-7. To compile the file, use this command:

```
clsql <hierarchyfile.sqi>
```

Tip: We recommend saving copies of your hierarchy .sqi and .sql files in a safe place for future reference.

Example 5-7 Compiling hierarchy.sqi file

```
C:\tbsm\Data\OPC>dir Hierarchy.*

03/01/2005  10:21a               1,461 Hierarchy.sqi

C:\tbsm\Data\OPC>clsql Hierarchy.sqi

C:\tbsm\Data\OPC>dir Hierarchy.*

03/01/2005  10:21a               1,461 Hierarchy.sqi
03/01/2005  10:22a              10,529 Hierarchy.sql
```

5.3.3 Running the .sql file to create lines of business

After the .sqi file with the hierarchy is compiled to create a .sql file, run the .sql against the Tivoli Business Systems database to add the information to the database. This adds definitions to the database. A simple way to do this is to load the file, in our case the hierarchy.sql file, into the Query Analyzer and run it.

Important: The .sqi hierarchy file must be compiled and the created .sql file loaded into the database for the definitions to be created. This must be done *before* any daily plan information has been loaded into the database.

After the definitions have been loaded and the daily plan has been loaded, the lines of business are available for use in Tivoli Business Systems Manager.

5.3.4 Changing TWS for z/OS definitions

Two SQL resource management tools that are installed with Tivoli Business Systems Manager are particularly useful to clear current TWS for z/OS definitions before adding a changed hierarchy file. The two tools are

Object..ResetBatchClasses and Object..ResetOpcPatterns. More information about these objects is available in the *IBM Tivoli Business Systems Manager V3.1 Administrator's Guide*, SC32-9085.

5.4 Loading the TWS for z/OS Daily Plan

This section discusses how to load the daily plan from TWS for z/OS into Tivoli Business Systems Manager. The current plan contains all the information about jobs to be run in the next scheduling period. The daily plan job is run to extend the current plan, using data from the long-term plan. When the current plan is extended via execution of the daily plan job, a report is generated. This report is used to create objects that represent the plan in the Tivoli Business Systems Manager database. Instructions for both manual and automated processing of daily plans are found in *IBM Tivoli Business Systems Manager V3.1 Administrator's Guide*, SC32-9085. For our example, we used the automated processing. Using the daily plan to populate Tivoli Business Systems Manager views is an important step when creating lines of business to monitor TWS for z/OS.

5.4.1 Copying the daily plan report to the database server for processing

The daily plan application is scheduled with two jobs. One is the daily plan job, which extends the current plan. The second job calls an FTP script on system Tokyo to initiate the FTP of the daily plan report and uploads it into the Tivoli Business Systems Manager database. Both jobs are part of the same TWS for z/OS application, with the second job (FTP) depending on completion of the daily plan job.

The FTP job is run on server Tokyo, and it pulls the daily plan report from the SC64 mainframe to server Tokyo. The FTP script is executed by the following command:

```
ftp -s:<ftpscriptname>
```

The script is located in the /tbsm/Data/OPC directory on server Tokyo. In the script that we used, shown in Example 5-8 on page 200, the `userid` and `password` fields actually contain the user ID and password for a user on SC64 who had rights to FTP the file.

Important: Make sure that this file cannot be accessed by unauthorized users. One way to do this is to use "Centralized Scripts" and store this script on the mainframe.

Example 5-8 Sample FTP script

```
open 9.12.6.9
userid
password
ASCII
cd ..
get TWS.INST.TWSC.DPEXT.LIST dailyplan
bye
```

5.4.2 Creating the opc_autoload.ksh file

The opc_autoload.ksh file (Example 5-9) was created and placed in the /tbsm/bin directory on server Tokyo. This file is called by the job to load the daily plan report into the database.

Example 5-9 Sample opc_autoload.ksh file

```
cd C:/tbsm/Data/OPC
check_opc_plan.ksh -fcheck_opc_plan_1.cfg -lEN_US -v -Stokyo -Usa -Psa
```

5.4.3 Creating the check_opc_plan_1.cfg configuration file

The check_opc_plan_1.cfg file has a single entry, in the following format:

```
<Daily Operational Plan>|<DATE format string>
```

In this entry, <Daily Operational Plan> is the name of the file that the daily plan report is written into on the database server. For this configuration, the <Daily Operational Plan> is named *dailyplan*, as seen in the sixth line of Example 5-8. Our check_opc_plan_1.cfg is shown in Example 5-10.

Example 5-10 Sample check_opc_plan_1.cfg

```
dailyplan|YY/MM/DD
```

5.4.4 Scheduling an OPC Check for the new Daily Plan job

To schedule an OPC check for the new Daily Plan job:

1. On the Tivoli Business Systems Manager database server, open the Microsoft SQL Server Enterprise Manager window.
2. Change the job `OPC Check for new daily plan` to read `sh opc_autoload.ksh`, as shown in Figure 5-8.

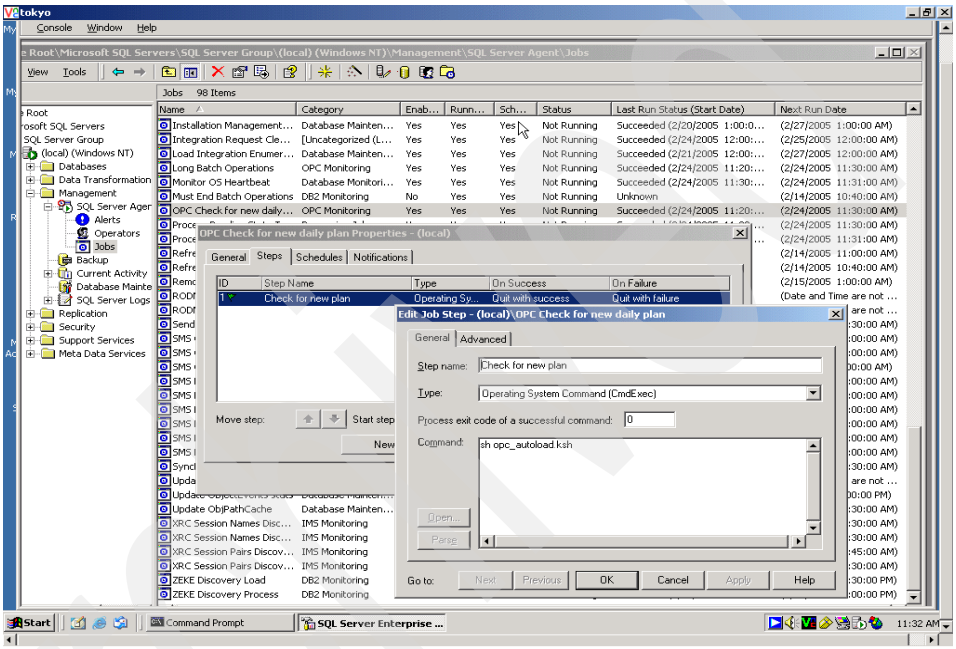

Figure 5-8 Changing text in job "OPC Check for new daily plan"

3. Enable the OPC Check for new daily plan job as shown in Figure 5-9 on page 202.

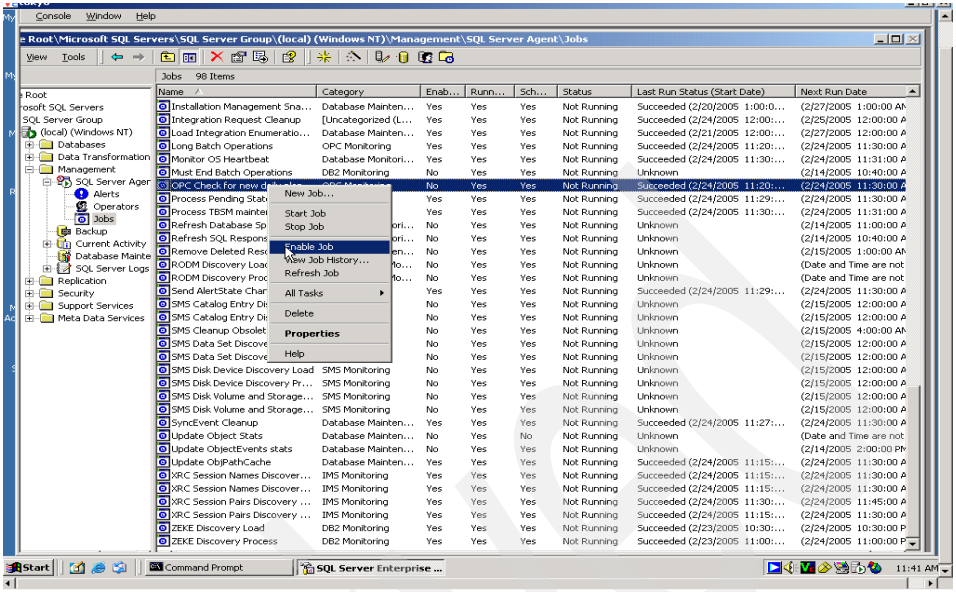

Figure 5-9 Enabling the "OPC Check for new daily plan" job

4. Schedule the job once per minute, as shown in Figure 5-10.

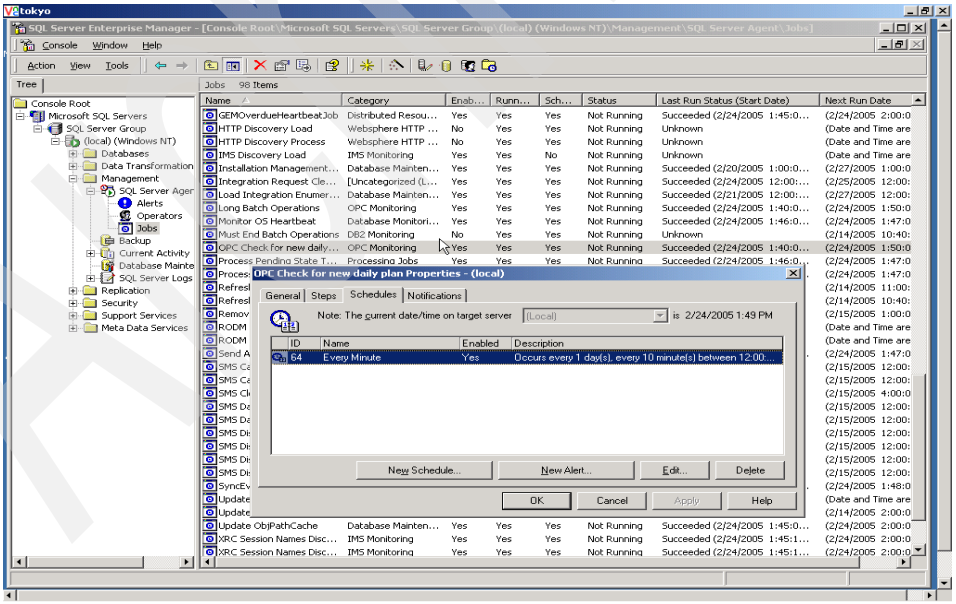

Figure 5-10 Scheduling the "OPC Check for new daily plan" job

Important: For each upload of the daily plan, changes populate to the consoles after both the “OPC Check for new daily plan” and the “UpdateObjPathCache” job have run, in that order. While troubleshooting, you can run the jobs manually to ensure that data has been loaded to the database.

5.5 Viewing TWS for z/OS objects in Tivoli Business Systems Manager

This section discusses the ways that TWS for z/OS objects can be viewed in Tivoli Business Systems Manager. Available views include the physical tree, the batch management view, the event management view, lines of business, and the executive dashboard. Each of these views has strengths that make them appropriate for certain audiences. This discussion focuses on views that are available after the daily plan has been loaded to the Tivoli Business Systems Manager database.

5.5.1 Viewing the physical tree on the All Resources view

The physical tree is in the All Resources view and begins with the enterprise object. You can drill down the tree to see the batch schedule sets for each daily plan that has been uploaded. Figure 5-11 on page 204 shows the batch schedule set for SC64 on the physical tree, using the All Resources view. You can open the properties of the batch schedule set from this view, review child events at the batch schedule level, and then drill further using the properties until the original event on the job is viewed. This method, however, is far from ideal when viewing events, because it takes time and effort to find the original event.

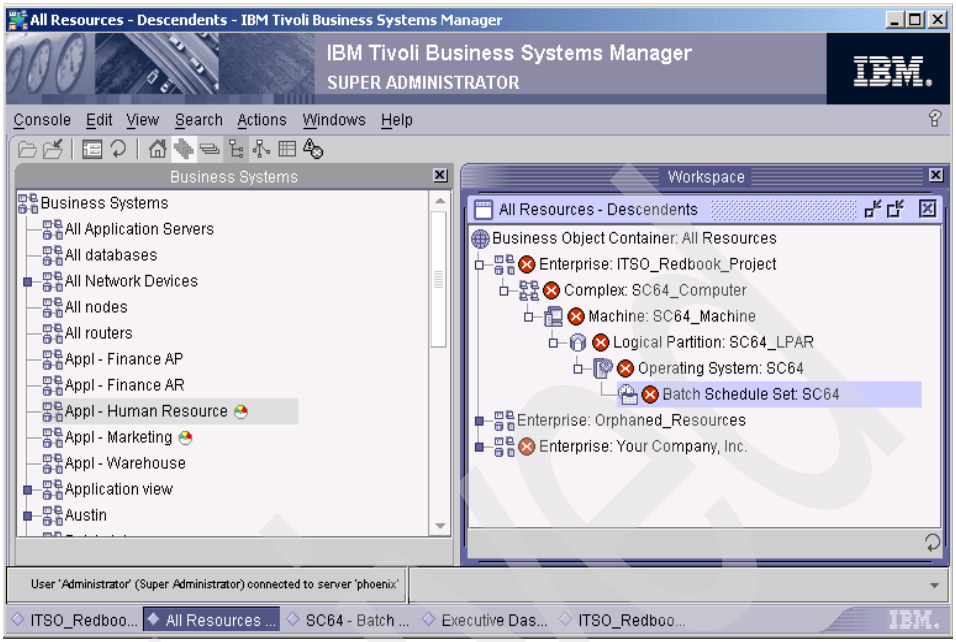

Figure 5-11 Physical Tree on All Resources view

5.5.2 The Batch Management Summary view

A better way to view events from the All Resources view is to use the Batch Management Summary view. Figure 5-12 on page 205 shows how to open the Batch Management Summary view. The Batch Management Summary provides a good view of the batch schedule set (daily plan), batch schedule (applications), and jobs that are scheduled in TWS for z/OS. Information about customizing this view is available in Appendix C of the *IBM Tivoli Business Systems Manager V3.1 Introducing the Consoles*, SC32-9086 manual.

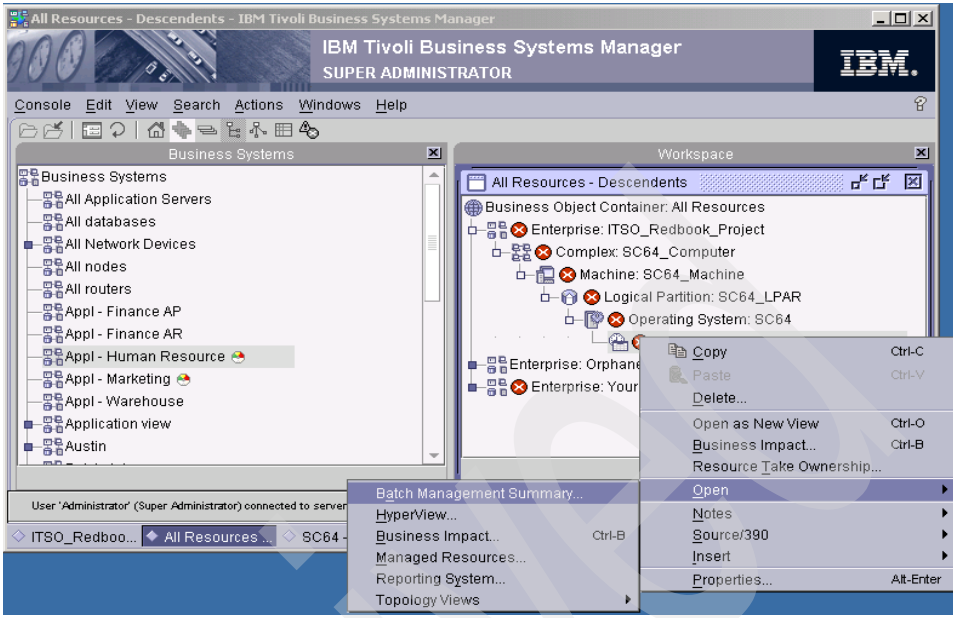

Figure 5-12 Opening the Batch Management Summary from the All Resources view

Figure 5-13 on page 206 shows a Batch Management Summary for a schedule set that is opened through the job level. The batch management summary view can be opened against any parent level of the physical tree, starting with the Enterprise that contains a batch schedule set or batch schedule. It can also be opened against any parent object in a line of business that contains a batch schedule set or batch schedule in Tivoli Business Systems Manager. The batch schedule set has three panes, as shown in Figure 5-13 on page 206 — the batch schedule set, child batch schedules, and their jobs. The batch schedule view has two panes, one for batch schedules and one for jobs.

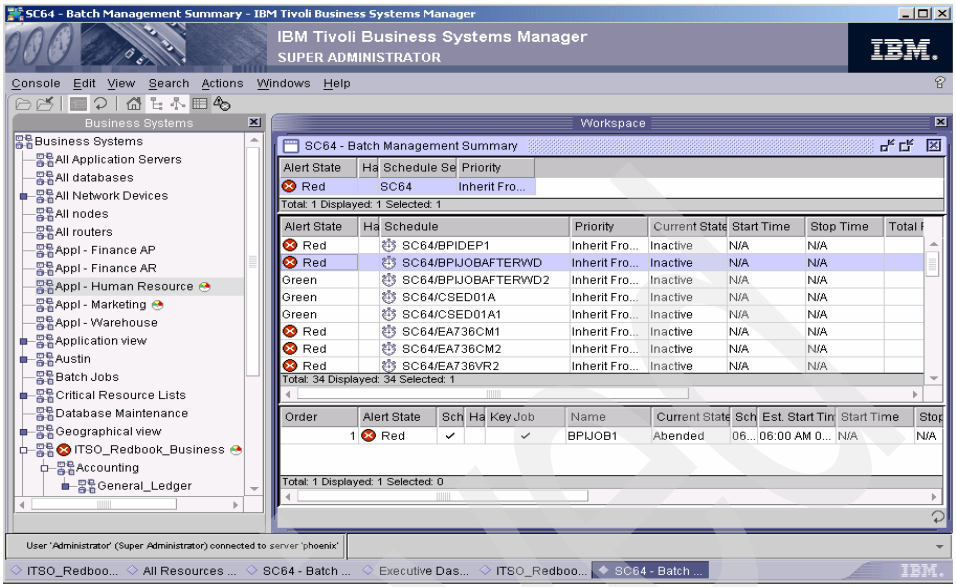

Figure 5-13 Batch Management Summary View

5.5.3 The Event Management view

The Event Management view adds a pane at the bottom of the Batch Management Summary view, allowing direct management of events as they appear in TWS for z/OS. This view can be added by opening the batch management summary and choosing **View** → **Events** from the main menu. The event view pane that is added at the bottom shows all events that meet the filtered criteria. Figure 5-14 on page 207 shows a close-up of an event viewer pane that was opened from the Batch Management Summary of the line of business ITSO_Redbook_Business. Note that the first filter, Date (which is set to the last 30 minutes) has been broken, so all events are shown.

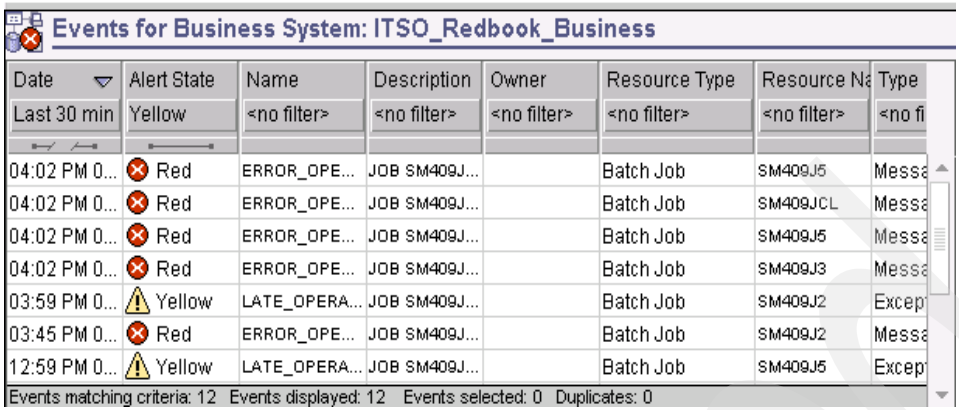

Figure 5-14 Event Viewer pane

5.5.4 Viewing lines of business

Tivoli Business Systems Manager allows users to create lines of business that show resources in the environment as they relate to one another. In our test environment, we created two lines of business that were related through a single parent line of business (see Figure 5-6 on page 196). After a line of business has been created, it can be used in its entirety as a building block of a larger line of business. Discussion of creating lines of business that contain TWS for z/OS is in 5.3, “Creating lines of business using Daily Plan data” on page 190. Further information about planning and creating lines of business is available in *Tivoli Business Systems Manager V3.1 Planning Guide*, SC32-9088.

5.5.5 The Executive Dashboard

The Executive Dashboard is a special feature of Tivoli Business Systems Manager. We included the line of business ITSO_Redbook_Business on an example dashboard. Events against the line of business turned it red (see Figure 5-15 on page 208). Drilling down on the red icon gives further information about recent events so that they may be followed until any issues are resolved and status is returned to normal (green), as shown in Figure 5-16 on page 208. Further instructions on configuring and using the executive dashboard are available in the *Tivoli Business Systems Manager V3.1 Introducing the Consoles*, SC32-9086 manual.

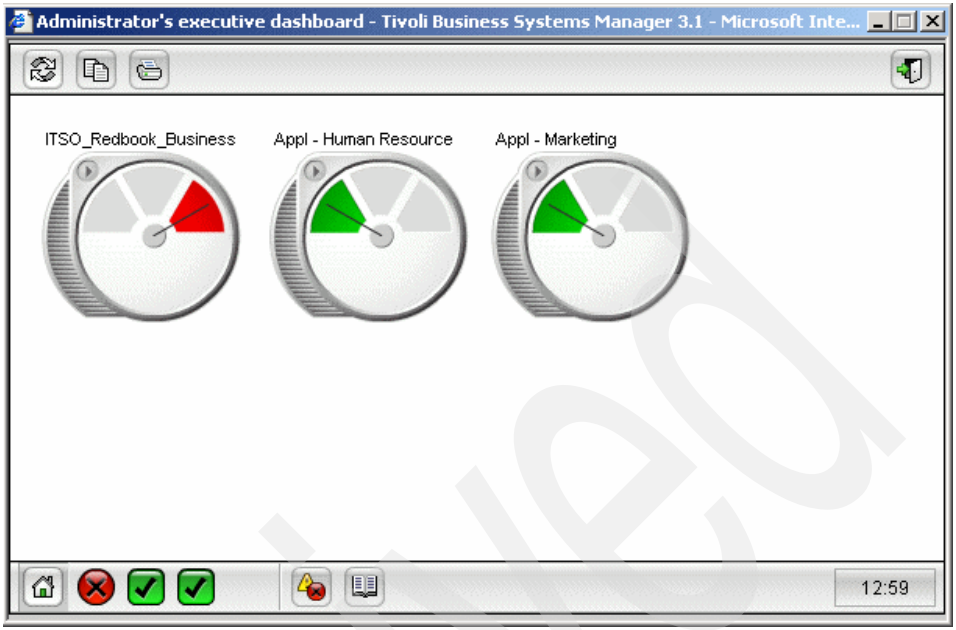

Figure 5-15 Executive Dashboard view

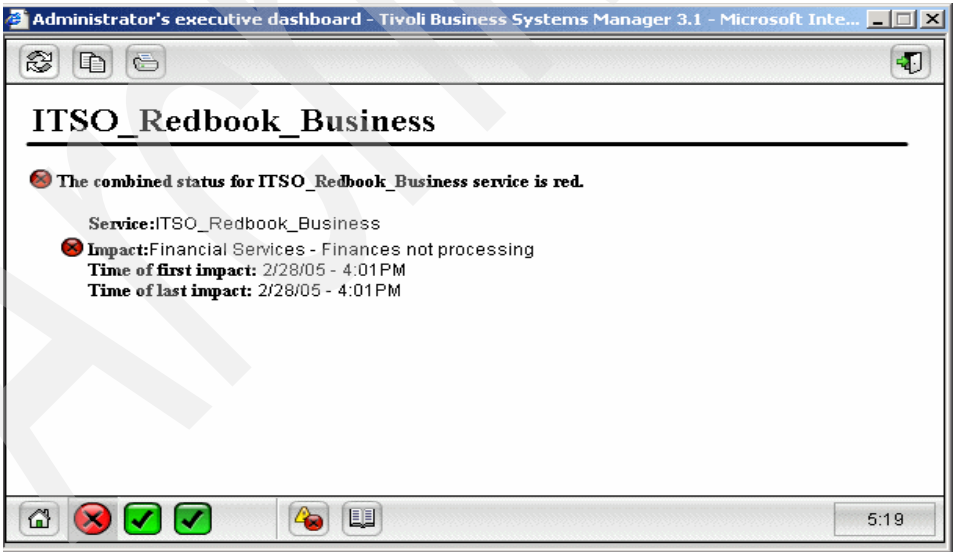

Figure 5-16 Executive Dashboard, drilled down on ITSO_Redbook_Business

6

Integrating Tivoli Decision Support for OS/390

This chapter describes the integration of Tivoli Workload Scheduler for z/OS (TWS for z/OS) and Tivoli Decision Support for OS/390. It discusses how to install the OPC Component in Tivoli Workload Scheduler for z/OS and produce reports, to create additional reports, and to host those reports on a Web site.

For our integration we installed Tivoli Decision Support for OS/390 Version 1.6 on the SC64 system, which is one of four systems in a Parallel Sysplex environment. The TWS for z/OS controller is also installed on this system.

Our installation of Tivoli Decision Support for OS/390 is using the suggested default values of DRL and DRLSYS for the prefix and system table prefix values respectively.

6.1 Installing the TWS for z/OS - OPC Component

This section describes the standard installation for the TWS for z/OS - OPC Component. If necessary, refer to Chapter 5 in *Tivoli Decision Support for OS/390 Administration Guide Version 1.6*, SH19-6816.

To install this component:

1. Select option **2** (Administration) from the TDS for OS390 Primary Menu panel and press Enter.
2. Select option **2** (Components) from the Administration panel and press Enter.
3. Select the **TWS for z/OS - OPC Component** from the Components panel as shown in Figure 6-1.

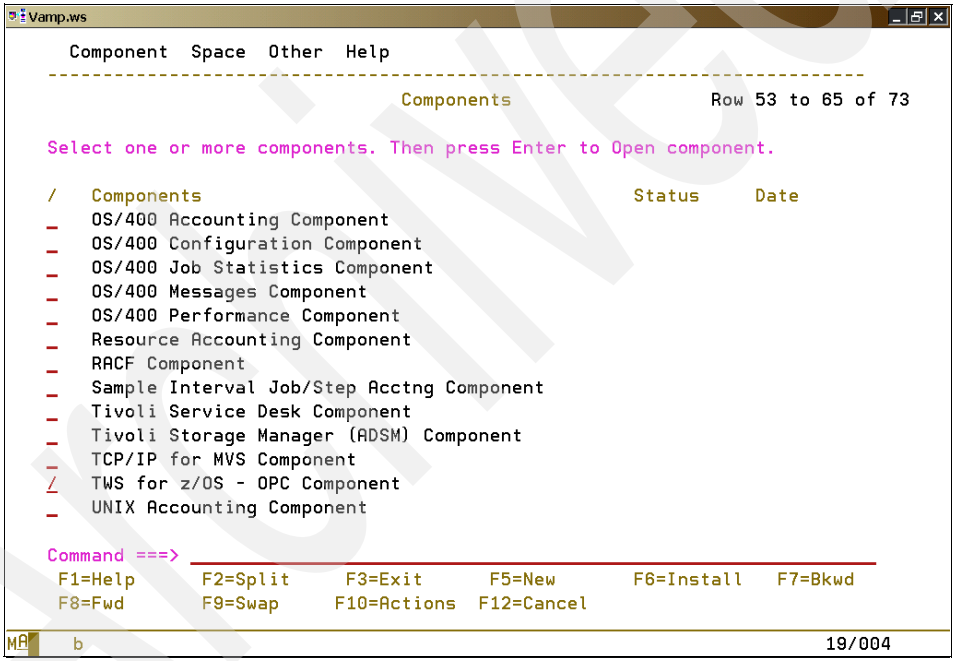

Figure 6-1 Selecting the component to install

4. Press F6 (Install) to install. The Installation Options appears, as shown in Figure 6-2 on page 211.

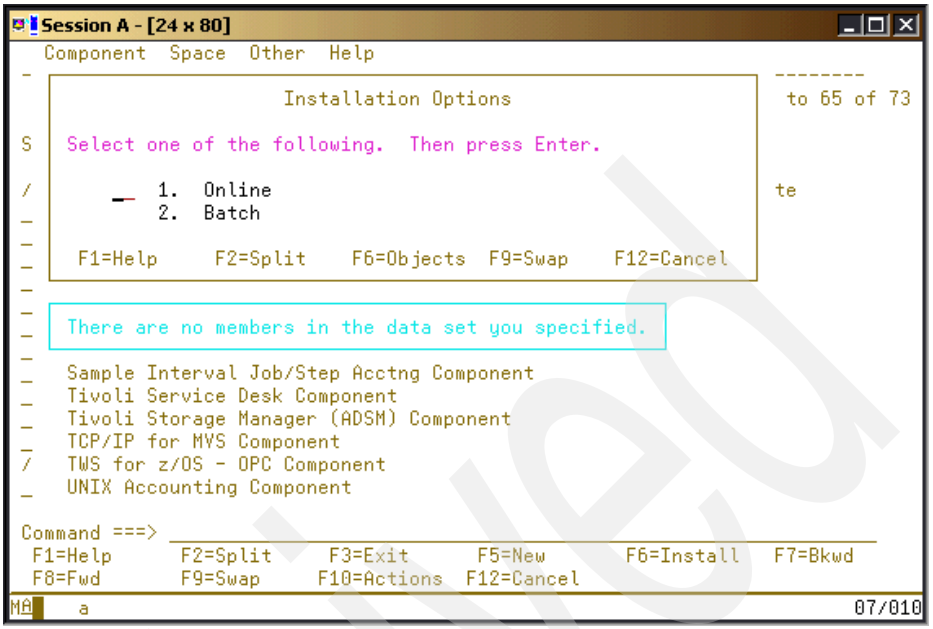

Figure 6-2 Selecting the installation method

Note: A text box that contains the following message only appears if the DRL.LOCAL.DEFS data set is empty.

```
There are no members in the data set you specified.
```

This is normal if you have no local modifications in place for any existing installed component.

5. Choose your preferred method of installation and press Enter.

Recommendation: We recommend saving the installation results for future reference. This is explained for both options.

Options for installation are:

– Online

Installing online is the easiest method. It means you do not have to code any JCL, and the end result is more identifiable. However, the installation is immediate, so you should consider whether the time is best to be performing the installation.

For example, do not execute this option while DB2 housekeeping for this Tivoli Decision Support for OS/390 database is running, because either a housekeeping job or this component installation will fail, or both, due to resource contention.

If you choose this option, SQL statements are executed, and you see the DATABASE STATUS PANEL that is shown in Figure 6-3.

Figure 6-3 Online installation in progress

A successful online installation results in a message as shown in Figure 6-4 on page 213.

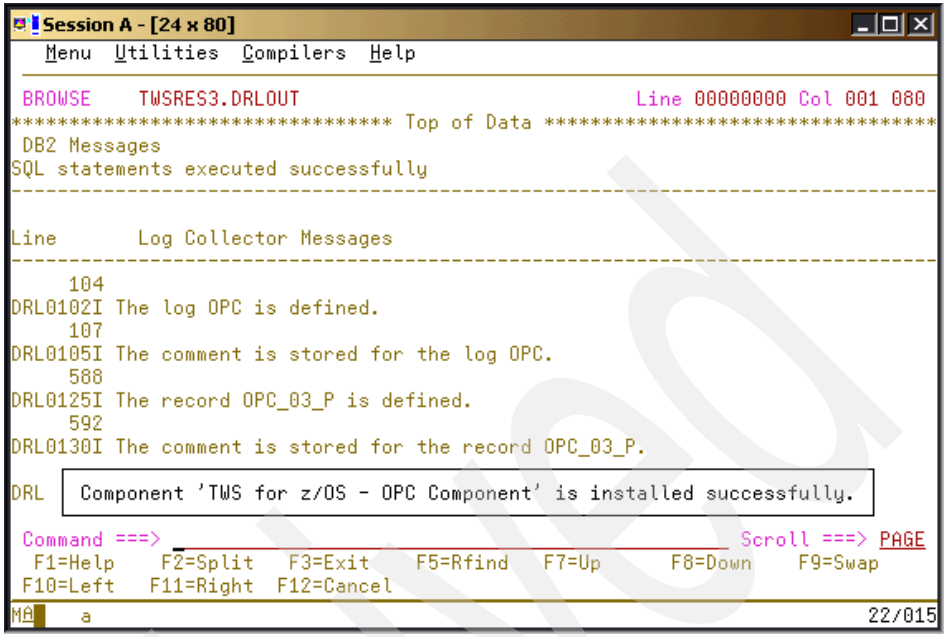

Figure 6-4 Successful online installation notification

Press F3 (Exit), and the Lookup Tables panel appears.

Recommendation: To save the installation results, press F2 (Split) to get another ISPF menu, and rename your '*userid*.DRLOUT' data set to something meaningful. For example, for our example, we renamed ours to DRL.DRLOUT.TDS16.TWSCOMP.INSTALL.

After you rename the data set, press F3 (Exit) to exit back to the Lookup Tables panel.

– Batch

Installing in batch gives you the flexibility to decide when the job runs.

For example, by coding TYPRUN=HOLD on the job card, you could include the instruction to release the job as part of a change record to ensure it is done outside core business hours or after DB2 housekeeping has finished.

If you choose this option, the necessary JCL is created for you automatically, as shown in Figure 6-5 on page 214. All you need is a valid job card. However, before you submit the job, you should save the installation results.

Important: If your system is part of a sysplex and you are installing via batch, remember to code the correct SAFF= value on the JOBPARM statement.

Recommendation: To save the installation results, change the DRLOUT DD statement on all of the RUNLOG, RUNRDEF, and COPYMSG steps to create your desired data set. Alternatively, you could leave the JCL as it is and save the whole job output.

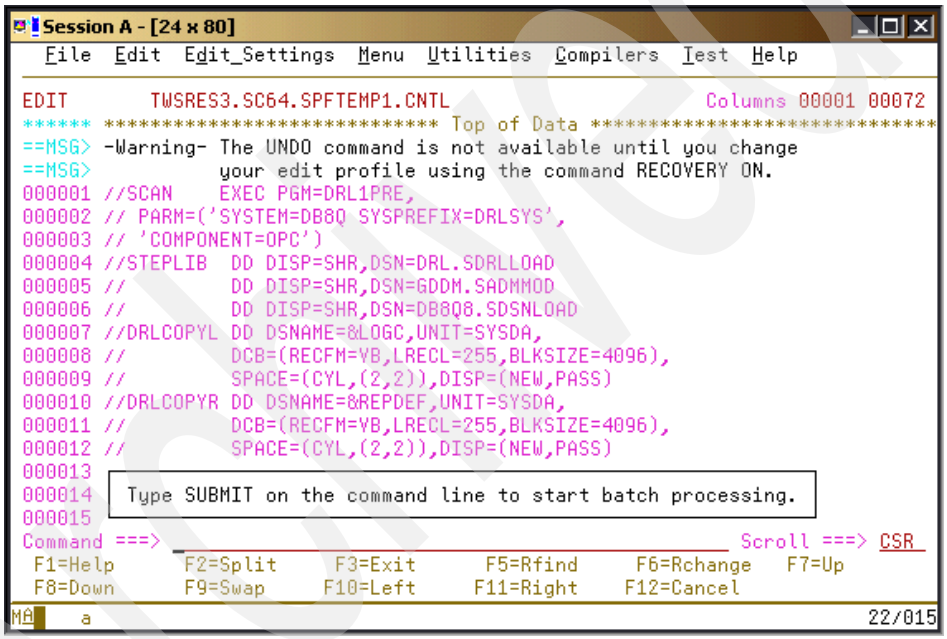

Figure 6-5 Installing via batch JCL

Submit the job and check the output for successful installation. Pressing F3 (Exit) returns you to the Components panel and the Status and Date fields now have values as shown in Figure 6-6 on page 215.

Note: If you install via batch, the RUNLOG step shows a return code of 08 because the SQL ALTER TABLE statements, in the <*hlq*>.SDRLDEFS members related to this install, result in several SQLCODE = -204 errors. This is nothing to be concerned about.

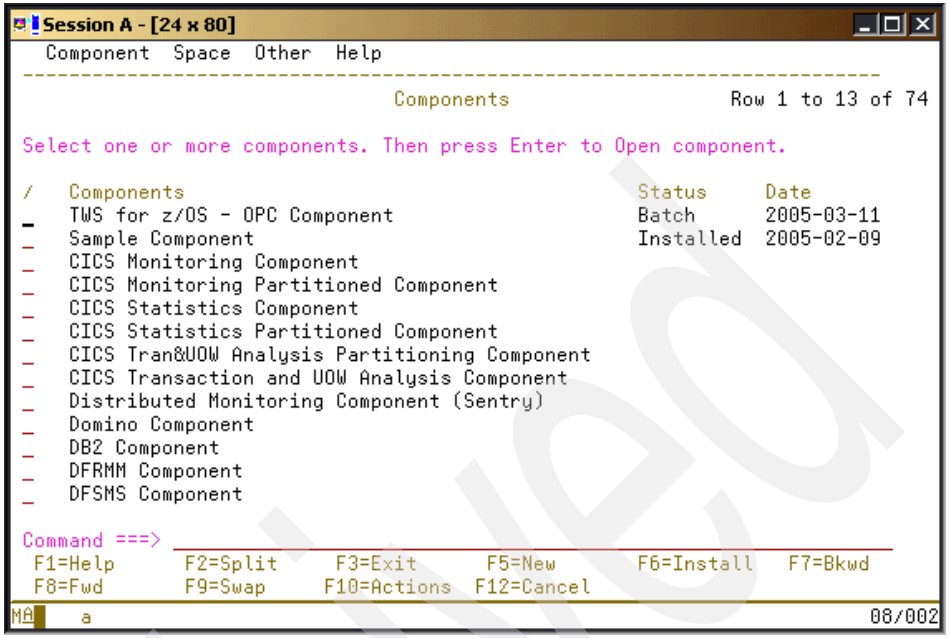

Figure 6-6 Installing via batch status

6. Populate the OPC_WORKSTATION lookup table with the details from the TWS for z/OS controller from which you will be collecting data and, subsequently, on which the data will be reported.

   How you choose to perform the install determines how you populate the table.

   – Online

     If you installed online, you are already in the right location. Select the OPC_WORKSTATION table and press Enter.

   – Batch

     If you installed via batch, you have to:

     i. Select option **2** (Administration) from the TDS for OS390 Primary Menu panel and press Enter.

     ii. Select option **4** (Tables) from the Administration menu and press Enter.

     iii. Select the OPC_WORKSTATION table and press F10 (Actions).

     iv. Tab across to the Edit menu and press Enter.

     v. Select option **3** (ISPF) and press Enter.

   Regardless of how you populate the OPC_WORKSTATION lookup table, you are presented with the panel that is shown in Figure 6-7 on page 216.

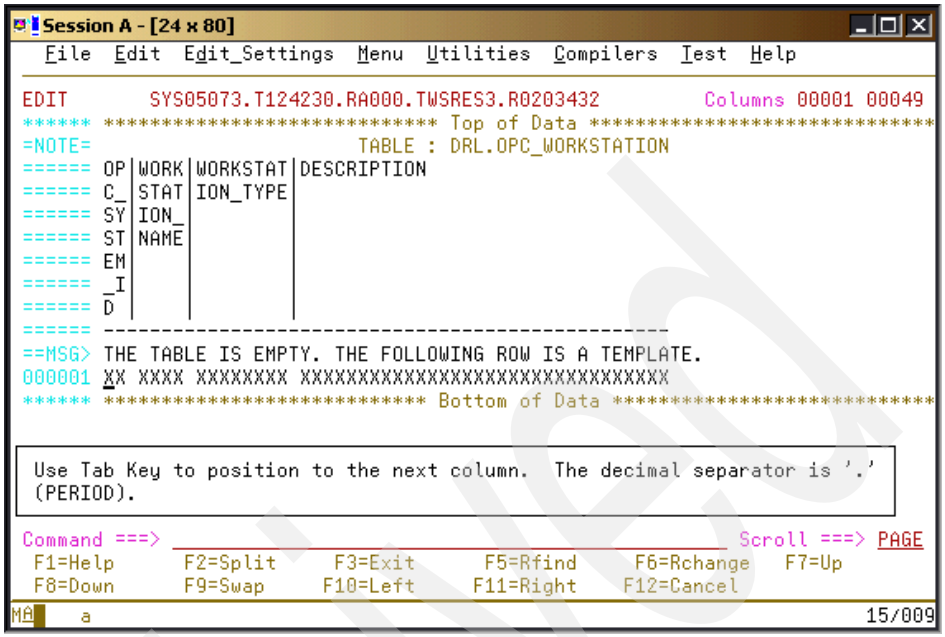

Figure 6-7 Empty lookup table

The OPC_SYSTEM_ID value to code should correspond to the value coded on the LOGID parameter of the BATCHOPT initialization statement. This statement is contained in the PDS member of the data set that is referenced by the EQQPARM DD statement in the current plan extend or replan batch job. Example 6-1 shows this code in our installation.

Example 6-1 Location of BATCHOPT

```
//EQQPARM  DD DISP=SHR,DSN=TWS.INST.PARM(BATCHOPT)
```

Note: If the LOGID parameter is not coded on the BATCHOPT initialization statement, the default value of 01 is being used. You should code this value in the OPC_WORKSTATION table.

The WORKSTATION_NAME, WORKSTATION_TYPE, and DESCRIPTION fields should be populated with the values that are defined in the Workstation descriptions option of the TWS for z/OS database (option 1.1 from the OPERATIONS PLANNING AND CONTROL panel).

Important: The workstation names and types *must* use uppercase letters. The workstation type values C, G, and P should be coded as COMPUTER, GENERAL, and PRINTER respectively. The description field can be a mix of both upper and lowercase letters.

Figure 6-8 shows part of our populated table.

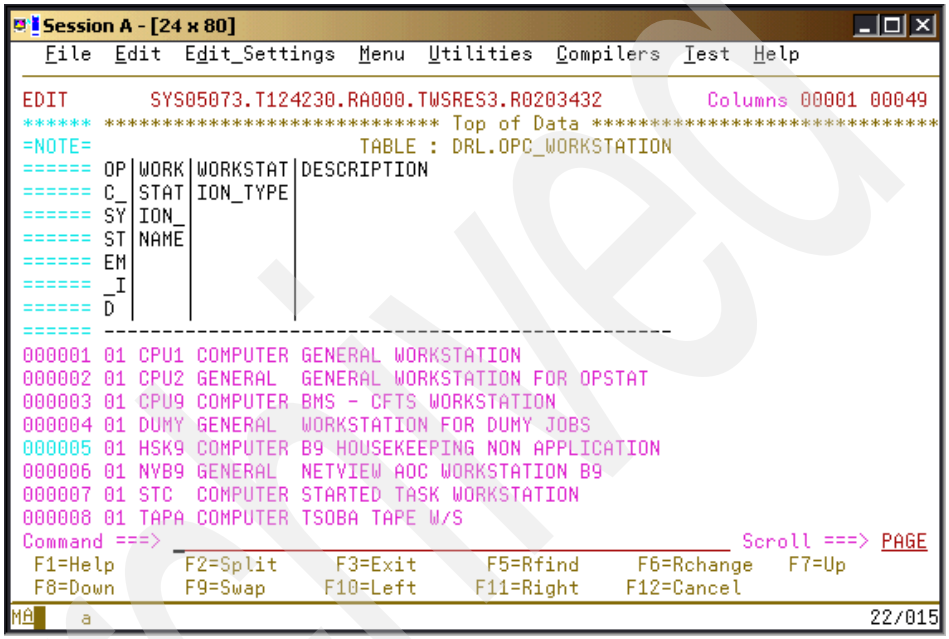

Figure 6-8 Coding the lookup table

Code a row for *every* workstation that is defined to the Tivoli Workload Scheduler for z/OS controller to ensure that they can all be reported upon.

Press F3 (Exit) to save the changes. A confirmation appears, as shown in Example 6-2.

Example 6-2 Successful table changes

```
The table changes are saved successfully.
```

If you installed online, you are returned to the Components panel, as shown in Figure 6-9 on page 218. Note the Status and Date fields are now populated.

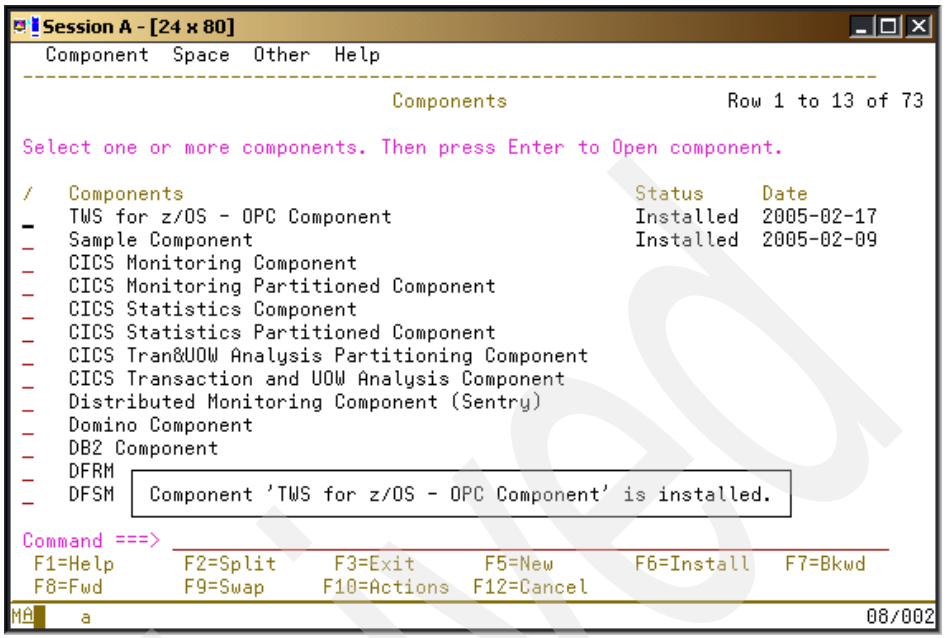

Figure 6-9 Component installed notification

If you installed via batch, you are returned to the Tables panel.

The “TWS for z/OS - OPC Component” is now fully installed and is ready for the collection of data. Information on collecting data is discussed in 6.4, “Collecting TWS for z/OS data” on page 233. However, before you collect data, you first need to identify the data.

6.2 Identifying the data to be collected

As part of the TWS for z/OS current plan extend or replan process, all the events that have occurred since the last planning process was run are written to the EQQTROUT DD statement in the JCL of the current plan extend or replan batch job. If the EQQTROUT DD statement does not specify a data set, you need to change it, because the event log data needs to be in a data set for it to be collected into the Tivoli Decision Support for OS/390 database.

Important: Remember to arrange for the TWS for z/OS dialog skeletons to be updated with the same data set as specified on the EQQTROUT DD statement in both the current plan extend and replan batch jobs to prevent event data loss if these functions are performed via the dialog.

6.2.1 BATCHOPT member considerations

We have already discussed the LOGID parameter value. There are two other parameters to consider:

NCPTROUT The values are either YES (the default) or NO. This parameter dictates whether records relating to the new current plan are recorded to the EQQTROUT data set.

OCPTROUT The values are either NO (the default) or YES to dictate whether records relating to the old current plan are recorded to the EQQTROUT data set, or CMP, which is a variation of the YES value that dictates what happens to the type 03 records for completed and deleted occurrences and the type 01 and type 04 records.

Note: In our installation, we used OCPTROUT(CMP) and NCPTROUT(YES) to ensure that as much event log data is available in Tivoli Decision Support for OS/390 as soon as possible.

For more information about the BATCHOPT initialization statement, refer to *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning Version 8.2*, SC32-1265 manual.

6.2.2 Handling the tracklog data

You need a process needs in place to handle the tracklog data so that it can be made available for collection into Tivoli Decision Support for OS/390 and any other use. Having a data set that is defined to the EQQTROUT DD statement is the main requirement, but how the data is managed also needs to be considered. As a current plan extend and one or more replan events can occur daily, you also need to plan for archiving the data.

In our installation, we coded the EQQTROUT DD statement to use the TWS.INST.TWSC.TRACKLOG sequential data set, as shown in Example 6-3.

Example 6-3 The EQQTROUT data set

```
//EQQTROUT DD DISP=OLD,DSN=TWS.INST.TWSC.TRACKLOG
```

Within the same daily planning process batch job, this data set is copied to (+1) of the 'TWS.INST.TWSC.PLAN.TRACKLOG GDG *only* if the daily planning process is successful, as defined by the COND parameter on the EXEC statement of the step name (see Example 6-4).

Example 6-4 Copying the EQQTROUT data set

```
//*********************************************************************
//* STEP4 COPY TRACKLOG TO GDG
//*********************************************************************
/*
//TRKCOPY  EXEC PGM=IEBGENER,COND=(8,LT)
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,DSN=TWS.INST.TWSC.TRACKLOG
//SYSUT2   DD DISP=(NEW,CATLG),DSN=TWS.INST.TWSC.PLAN.TRACKLOG(+1),
//         SPACE=(CYL,(1,1),RLSE),VOL=SER=SBOXA6,UNIT=SYSDA,
//         RECFM=VB,LRECL=32756,BLKSIZE=32760
//SYSIN    DD DUMMY
//*
```

The entire contents of the TWS.INST.TWSC.TRACKLOG GDG are then copied to (+1) of the TWS.INST.TWSC.DAILY.TRACKLOG GDG by a separate, daily job that provides a single data set containing all the day's events.

Example 6-5 on page 221 shows the complete TWSTRCPY batch job JCL with both the copy step and the delete step.

Example 6-5 The daily TWSTRCPY job

```
//TWSTRCPY JOB (0),'M FERGUSON',MSGLEVEL=(1,1),REGION=64M,
//         CLASS=A,MSGCLASS=X,TIME=1440,NOTIFY=&SYSUID
/*JOBPARM S=SC64
//*********************************************************************
//* STEP1 COPY TRACKLOG PLAN GDG TO DAILY GDG
//*********************************************************************
/*
//STEP1    EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSUT1   DD DISP=OLD,DSN=TWS.INST.TWSC.PLAN.TRACKLOG
//SYSUT2   DD DISP=(NEW,CATLG),DSN=TWS.INST.TWSC.DAILY.TRACKLOG(+1),
//         SPACE=(CYL,(1,1),RLSE),VOL=SER=SBOXA6,UNIT=SYSDA,
//         RECFM=VB,LRECL=32756,BLKSIZE=32760
//SYSIN    DD DUMMY
//*
//*********************************************************************
//* STEP2 DELETE PLAN GDG DATA SETS
//*********************************************************************
/*
//STEP2    EXEC PGM=IDCAMS,COND=(0,NE)
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DELETE TWS.INST.TWSC.PLAN.TRACKLOG.* PURGE
//*
```

We will collect gen(0) of the TWS.INST.TWSC.DAILY.TRACKLOG GDG into Tivoli Decision support for OS/390.

Note: The coding of the BATCHOPT initialization statement and the process of dealing with the tracklog data that are described in this section are not something over which the owner of the Tivoli Decision Support for OS/390 database usually has any control. If the daily planning process JCL is not suitable, you either have to arrange for it to be changed, as in our examples, or set up your own process of copying and archiving the tracklog data.

6.3 Creating the collect application

When you have set up the TWS for z/OS component and have discussed the tracklog data management, you are ready to collect the tracklog data. It is best to collect the data using a TWS for z/OS application so that it is automated and can be tied into the creation of the daily tracklog data set.

In our example installation, the tracklog data is the only data that was collected. However, if we were already collecting one or more other types of log data, we would need to plan how the tracklog data collect application would integrate with existing applications.

Some things to consider are:

- Existing application naming convention
- Existing job name naming convention
- Existing special resource naming convention

It is worth noting that there is no predefined way of setting up a TWS for z/OS environment for managing a Tivoli Decision Support for OS/390 database. So, we assume that there will be more than one Tivoli Decision Support for OS/390 database installed on the SC64 system at some point in the future and that more than one TWS for z/OS controller will be collected into our initial Tivoli Decision Support for OS/390 database.

6.3.1 Standardization considerations

Standardization of names is important if you are managing a multi Tivoli Decision Support for OS/390 database environment that collects several different types of data from several different systems. It introduces economies of scale and enables you to easily tell from either the application or job name what is being collected from where and into which database. Additionally, it also enables you to automate the creation of the applications and job names, which you can then load into the TWS for z/OS database using the batch loader facility and into the relevant JCL library respectively.

With these standards in mind, we set up an application in TWS for z/OS called TDS#01#TWS#01#DC, where:

- *TDS* reflects that it is a Tivoli Decision Support for OS/390 application.
- *01* reflects that it is the first Tivoli Decision Support for OS/390 database on the system.
- *TWS* reflects that it is a TWS for z/OS related application.
- *01* reflects that it is processing data from the system with LOGID(01) coded.
- *DC* reflects that it is a daily data collection application.

The application contains a dummy start operation, a T01T01C1 collect job operation, and a dummy end operation. The collect job name is made up of the following characters:

- *T* reflects that it is a Tivoli Decision Support for OS/390 job.
- *01* reflects that it is the first Tivoli Decision Support for OS/390 database on the system.
- *T* reflects that it is a TWS for z/OS related job.
- *01* reflects that it is collecting LOGID(01) data.
- *C* reflects that it is a collect job.
- *1* gives it uniqueness.

If a second TWS for z/OS controller were collected, the application would be called TDS#01#TWS#02#DC with job name T01T02C1, and a third application would be called TDS#01#TWS#03#DC with job name T01T03C1.

These names were chosen so that we could also have the following applications and job names in our Tivoli Decision Support for OS/390 database:

- TDS#01#SMF#63#DC for the daily collect of SMF data from the SC63 system with a job name of T01S63C1.
- TDS#01#SMF#64#DC for the daily collect of SMF data from the SC64 system with a job name of T01S64C1.
- TDS#01#IMS#63#DC for the daily collect of IMS data from the SC63 system with a job name of T01I63C1.
- TDS#01#IMS#64#DC for the daily collect of IMS data from the SC64 system with a job name of T01I64C1.

We can then use similar names for collecting data into a second Tivoli Decision Support for OS/390 database:

- TDS#02#TWS#01#DC for the daily collect of TWS for z/OS data from a system with LOGID(01) coded with a job name of T02T01C1. This would be a different TWS for z/OS to that collected by the T01T01C1 job.
- TDS#02#TWS#02#DC for the daily collect of TWS for z/OS data from a system with LOGID(02) coded with a job name of T02T02C1. This would be a different TWS for z/OS to that collected by the T01T02C1 job.
- TDS#02#SMF#65#DC for the daily collect of SMF data from the SC65 system with a job name of T02S65C1.
- TDS#02#SMF#70#DC for the daily collect of SMF data from the SC70 system with a job name of T02S70C1.

- TDS#02#IMS#65#DC for the daily collect of IMS data from the SC65 system with a job name of T02I65C1.
- TDS#02#IMS#70#DC for the daily collect of IMS data from the SC70 system with a job name of T02I70C1.

6.3.2 Creating the tracklog data collection application

With the standardized application name and job name decided, we now create them in TWS for z/OS. Figure 6-10 shows the creation of our TDS#01#TWS#01#DC application.

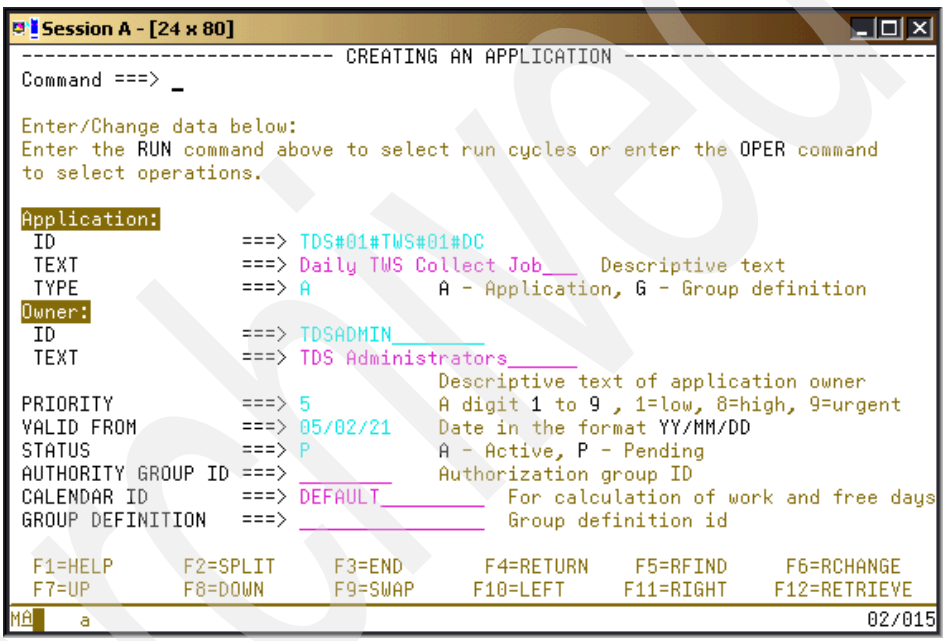

Figure 6-10 Creating the collect application

An application is made up of several parts so that the job runs when we want it to. These parts are the run cycle, operations, predecessors, and special resources.

Run cycle

The collect job needs to run on the same days as the application that creates the tracklog data set. So, our application needs to have the same run cycle. To set up the run cycle:

1. Enter **RUN** on the command line to open to the RUN CYCLES panel. Complete the fields in this panel with appropriate values according to your installation standards. Figure 6-11 shows the values that we coded.

Recommendation: We recommend that you use rule-based run cycles because they are easier to manage and are more flexible. Using an offset-based run cycle could require you to code interval origin and end dates, which, if not maintained, can result in the application not being scheduled. Rule-based run cycles also give you the opportunity to see exactly when the application will be scheduled as you are creating the application.

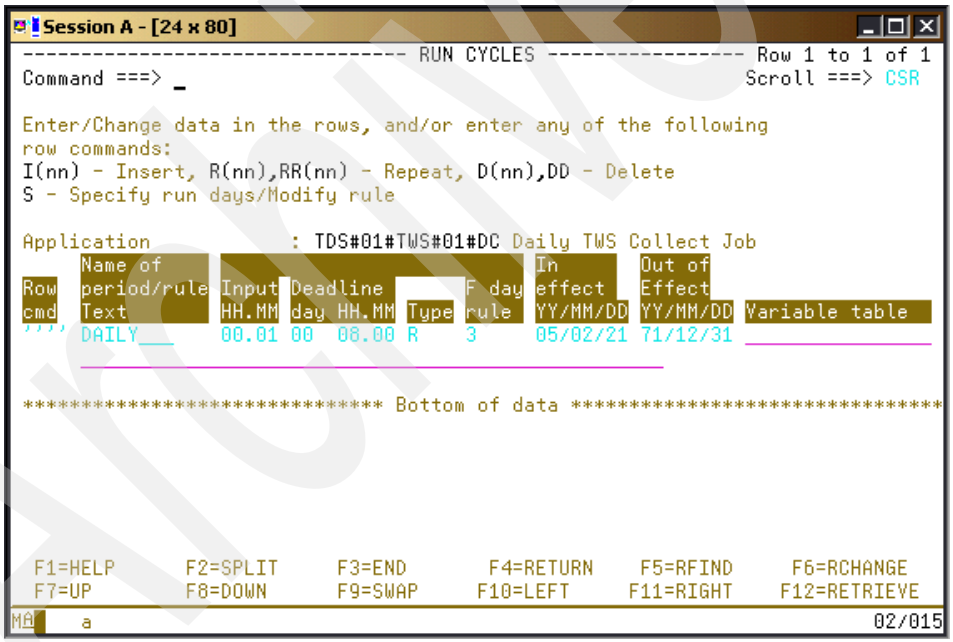

Figure 6-11 Creating the run cycle

2. Type S next to the rule name and press Enter to define exactly when the application will run. Figure 6-12 on page 226 shows the rule that we defined so that our application runs on every day of every week.

Important: The run cycle that you use should be the same as the run cycle that is used in the application that controls the creation of the daily tracklog data set. In our environment, this is TWSTRACKLOGCOPY. If the TWSTRACKLOGCOPY application ran and our application did not, then TWS.INST.TWSC.DAILY.TRACKLOG GDG would be created but not collected. If the TWSTRACKLOGCOPY application did not run and our application did, the T01T01C1 job would fail trying to collect the same data more than once.

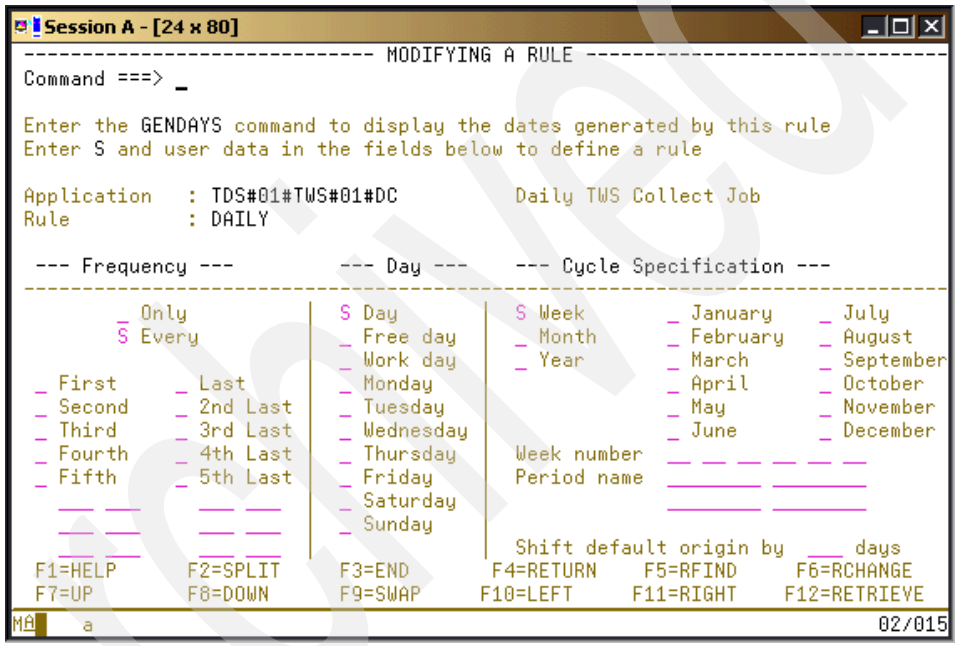

Figure 6-12 Creating the rule

3. Type **GENDAYS** on the command line and press Enter to confirm which days the application is scheduled in the current plan. Figure 6-13 on page 227 verifies our request.

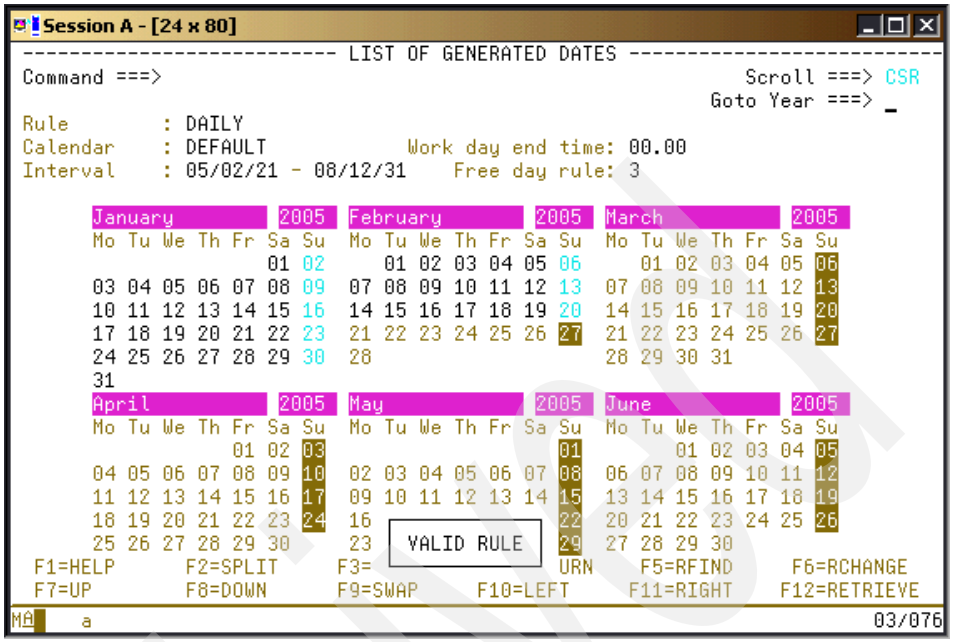

Figure 6-13 Verifying the dates

4. Press F3 (END) three times to return to the CREATING AN APPLICATION panel.

Operations

The operations list determines what runs and in what order. To set up the operations list:

1. Enter **OPER** on the command line to move to the OPERATIONS panel. The operations that we defined for this application are:
   - 001 - a dummy start to which the required predecessors can be defined.
   - 005 - the T01T01C1 collect job, dependant on operation 001.
   - 255 - a dummy end, dependant on operation 005.

   Figure 6-14 on page 228 shows these operations.

Note: Enter **PRED** on the command line if you see the Operation text field instead of the Internal predecessors field.

Recommendation: We recommend the use of dummy start and end operations for dependency resolution to allow you to easily add or delete jobs within an application without having to alter predecessors.

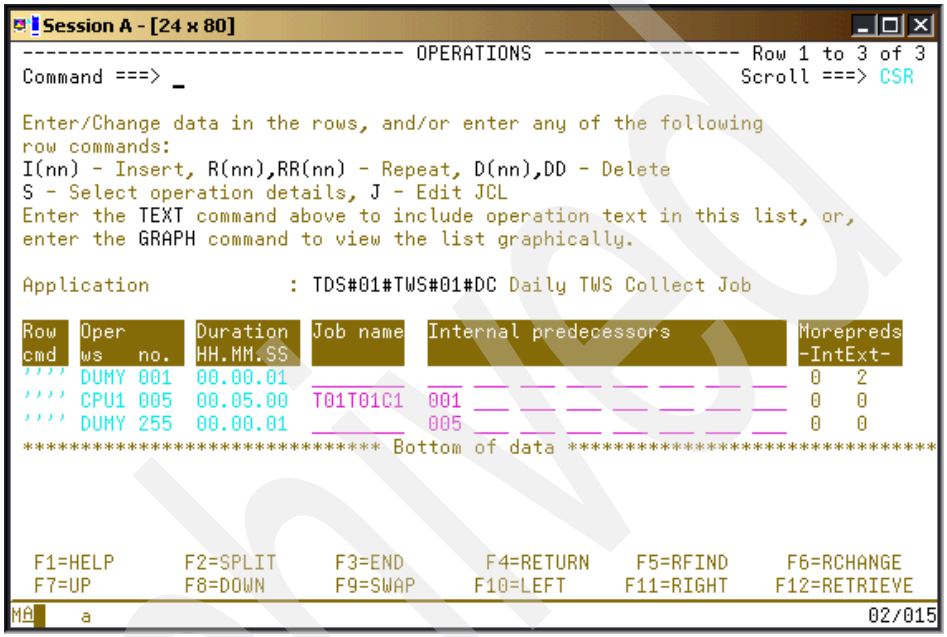

Figure 6-14 Creating the operations within the application

Tip: You do not need to code a job name for the dummy operations.

2. We code the operation text as shown in Example 6-6 on page 229. Enter **OPER** on the command line to get to the Operation text field. This information is not used in the running of the application. However, this information does give an indication of what each operation is and is good practice.

Example 6-6 Operation text

```
Application            : TDS#01#TWS#01#DC Daily TWS Collect Job

Row  Oper      Duration  Job name  Operation text
cmd  ws   no.  HH.MM.SS
'''' DUMY 001  00.00.01  ________  Dummy Start Job_________
'''' CPU1 005  00.05.00  T01T01C1  Daily TWS Collect Job___
'''' DUMY 255  00.00.01  ________  Dummy End Job___________
```

Predecessors

As well as the operations within the application being dependant on each other and being defined by the internal predecessors, we want to ensure that the application does not start until two specific external events have occurred:

- The DB2 housekeeping for our Tivoli Decision Support for OS/390 database has completed successfully. This is the TDS#01#DLY#DB2HK application that runs the evening before. So, we define the dummy end operation of that application to ensure that the database is fully available again before we collect more data.
- The creation of the daily TWS for z/OS tracklog data set. This is the TWSTRACKLOGCOPY application.

Figure 6-15 on page 230 shows these events defined as predecessors to our dummy start, operation 001.

Important: Remember that these predecessors are specific to our environment. You need to code what is appropriate for your environment.

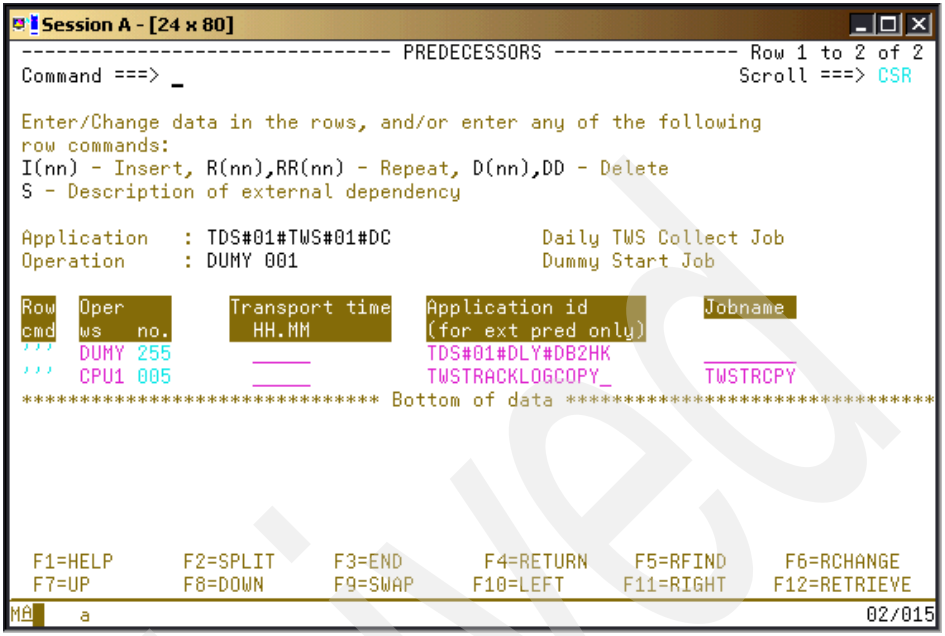

Figure 6-15 Predecessors to the application

3. Press F3 (END) to return to the OPERATIONS panel.

Special resources

TWS for z/OS special resources can be used to aid the flow of applications and prevent jobs from failing due to contention issues. This is particularly important in a DB2-based environment.

Once again, standardization is important, so we set up a special resource called T01.COLLECT.TWS for our Tivoli Decision Support for OS/390 environment. While still in the TDS#01#TWS#01#DC application, press F2 (SPLIT) and go to option 1.6.3 from the OPERATIONS PLANNING AND CONTROL panel to get to the CREATING A SPECIAL RESOURCE panel. Figure 6-16 on page 231 shows the values that we coded. We did not code anything for either option 1 (Intervals) or option 2 (WS) because that allows the special resource to be generic rather than specific to a workstation or interval.

We also need to set up special resources for other types of data collect if we start collecting them some time in the future (T01.COLLECT.SMF and T01.COLLECT.IMS, for example).

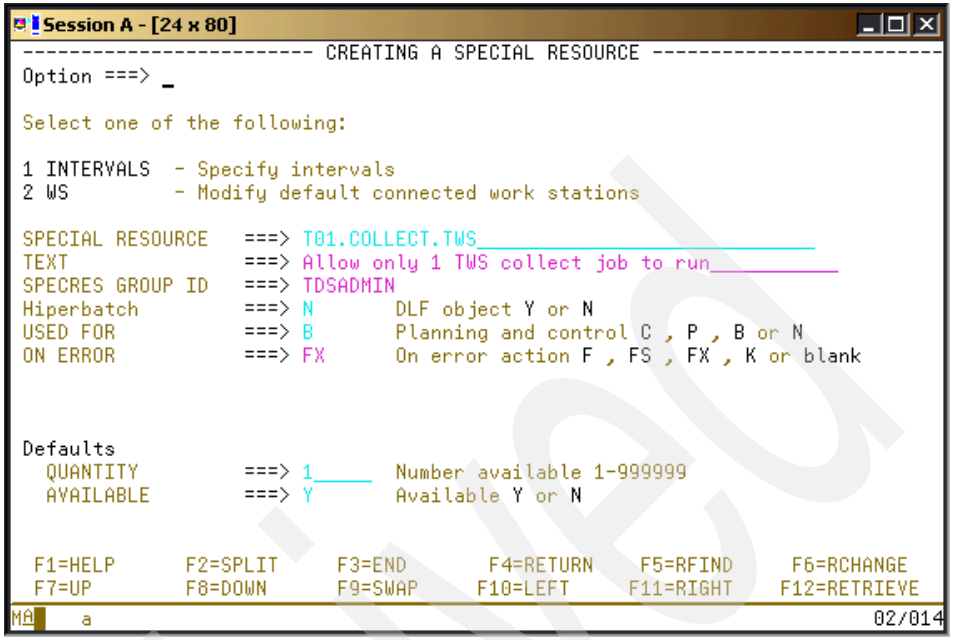

Figure 6-16 Creating a special resource

We define the T01.COLLECT.TWS special resource to the collect job, T01T01C1, to ensure that only one Tivoli Workload Scheduler data collect into our Tivoli Decision Support for OS/390 database can run at any one time. Even though this is the only Tivoli Workload Scheduler data collect defined so far, it still needs defining to prevent contention with the daily purge job.

Figure 6-17 on page 232 shows the T01.COLLECT.TWS special resource that was added to our collect job.

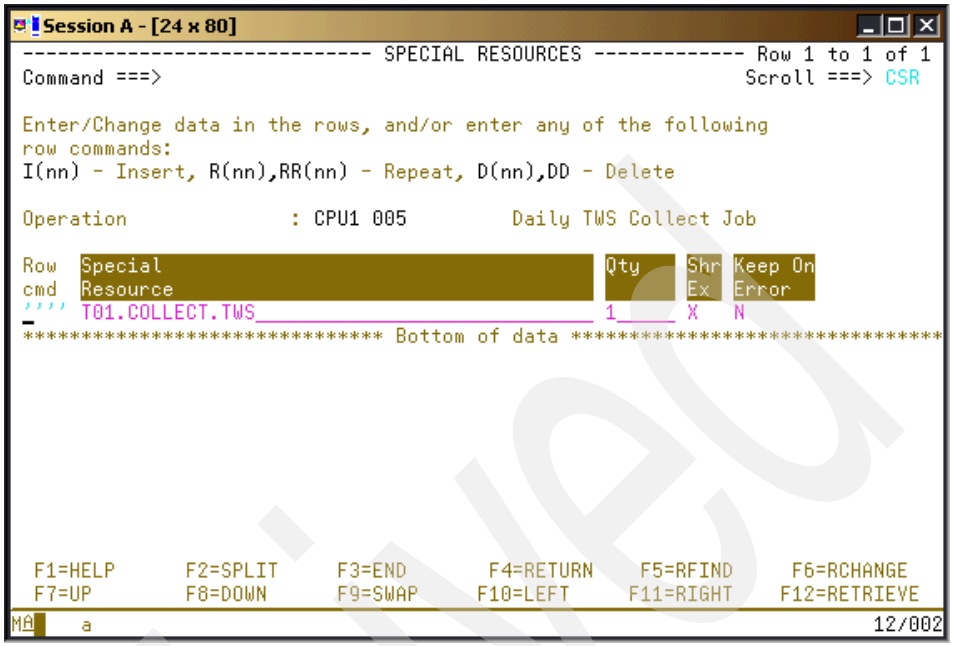

Figure 6-17 Defining the special resource to the collect job

The T01.COLLECT.TWS special resource would also be defined to any other Tivoli Workload Scheduler data collect jobs for this Tivoli Decision Support for OS/390 database, T01T02C1 and T01T03C1 for example, to allow them to run even if T01T01C1 fails.This is because the reason for a collect failure might not be Tivoli Decision Support for OS/390 database related, so it would be incorrect to stop the collection of data due to a single job failure.

Note: The T01.COLLECT.TWS special resource is also defined to the daily purge job in the TDS#01#DLY#PURGE application, along with any other special resources that are related to this Tivoli Decision Support for OS/390 database. This ensures there are no collect or report jobs running at the same time as the purge job and prevents a possible DB2 contention issue. Example 6-7 on page 233 shows all the special resources that were defined to the purge job.

Example 6-7 Special resources defined to the purge job

```
Operation              : CPU1 005         Daily Purge Job

Row  Special                                        Qty    Shr Keep On
cmd  Resource                                              Ex  Error
'''' T01.COLLECT.IMS_____________________________ 1_____ X   N
'''' T01.COLLECT.SMF_____________________________ 1_____ X   N
'''' T01.COLLECT.TWS_____________________________ 1_____ X   N
'''' T01.REPORTS.IMS_____________________________ 1_____ X   N
'''' T01.REPORTS.SMF_____________________________ 1_____ X   N
'''' T01.REPORTS.TWS_____________________________ 1_____ X   N
```

A similar special resource, T02.COLLECT.TWS, would be set up for collecting TWS for z/OS data into a second Tivoli Decision Support for OS/390 database, along with T02 versions of all the other T01 special resources.

The whole TDS#01#TWS#01#DC application has now been built, so press F3 (END) all the way out of the application and back to the LIST OF OPERATIONS panel.

Tip: Save the application with a status of P (pending) until you are ready to start collecting data. Otherwise, the application is scheduled and the T01T01C1 job fails OSUF because no JCL exists.

6.4 Collecting TWS for z/OS data

The collect job is defined in the TDS#01#TWS#01#DC application that we created in the “6.3, “Creating the collect application” on page 222” and its JCL needs to be placed in one of the TWS for z/OS job libraries that are defined to the EQQJBLIB DD statement of the TWS for z/OS controller started task. To do this, we create member T01T01C1 in TWS.INST.JOBLIB, copy the DRLJCOLL skeleton from <hlq>.SDRLCNTL into it, and modify it for our installation.

Important: The member name and the job name have to be identical. Otherwise, the job fails with an OSUF error code when submitted by TWS for z/OS.

Example 6-8 shows our T01T01C1 job after modification.

Example 6-8 Collect JCL

```
//TO1TO1C1 JOB (ACCT#),'TWS COLLECT',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(TO1TO1C1)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT           ON SC64
//*************************************************************
//*                                                           *
//*  LICENSED MATERIALS - PROPERTY OF IBM                     *
//*                                                           *
//*  5695-101 (C) COPYRIGHT IBM CORPORATION 1993, 2003        *
//*  SEE COPYRIGHT INSTRUCTIONS.                              *
//*                                                           *
//*************************************************************
//*                                                           *
//*  NAME: DRLJCOLL                                           *
//*                                                           *
//*  STATUS: Tivoli Decision Support for OS390 1.6.0          *
//*                                                           *
//*  FUNCTION:                                                *
//*    Tivoli Decision Support for OS390 COLLECT JOB.         *
//*                                                           *
//*************************************************************
//COLLECT  EXEC PGM=DRLPLC,PARM=('SYSTEM=DB8Q SYSPREFIX=DRLSYS')
//STEPLIB  DD DISP=SHR,DSN=DRL.SDRLLOAD
//         DD DISP=SHR,DSN=DB8Q8.SDSNLOAD
//DRLIN    DD *

COLLECT OPC
COMMIT AFTER BUFFER FULL ONLY
BUFFER SIZE 180M;

//DRLLOG   DD DISP=SHR,DSN=TWS.INST.TWSC.DAILY.TRACKLOG(0)
//DRLOUT   DD SYSOUT=*,DCB=(RECFM=F,LRECL=80)
//DRLDUMP  DD SYSOUT=*,DCB=(RECFM=VB,LRECL=32756)
//
```

Note: We add the COMMIT and BUFFER operands to the COLLECT statement to improve collect performance.

To start collecting the TWS for z/OS tracklog data, we modify the TDS#01#TWS#01#DC application to a status of A (active) and let it get added to the current plan with the next daily current plan extend job.

6.5 Running the OPC reports

Now that we have successfully collected some TWS for z/OS data into our Tivoli Decision Support for OS/390 database, we can produce some reports. This section demonstrates how to produce reports and discusses some of the limitations. It also shows how to overcome these limitations using TWS for z/OS.

6.5.1 Creating the report batch JCL

First, we need to obtain the batch JCL that is required. You can use the sample JCL that is provided in <hlq>.SDRLCNTL(DRLJBATR). However, that JCL requires that you code all of the data sets correctly. So, in our scenario, we let the product do this for us in the following way:

1. Select option **1** (Reports) from the Tivoli Decision Support for OS390 Primary Menu panel and press Enter.
2. Select a report and press F10 (Actions). We choose the OPC09 report.
3. Tab across to the Batch menu and press Enter.
4. Select option **2** (Invoke batch...) and press Enter.
5. Enter the report variables as requested and press Enter. Example 6-9 shows our selection.

Important: The quotes are required to prevent an error when running the job later.

Example 6-9 Entering report data selection values

```
Batch Reports Data Selection          Row 1 to 3 of 3

 Type information.  Then press Enter to edit JCL.

 Report ID  . : OPC09

 Variable          Value                                  Oper  Req
 DATE              '2005-02-01'                        >  =     Yes
 PERIOD_NAME       'PRIME'                             >  =     No
 OPC_SYSTEM_ID     '01'                                >  =     Yes
***************************** Bottom of data ******************************
```

6. We can now copy the JCL to a data set. Figure 6-18 on page 236 shows the BATCHJCL member in our local library.

   After pressing Enter, the column information in the top right-hand corner of the panel changes to `Member BATCHJCL created`.

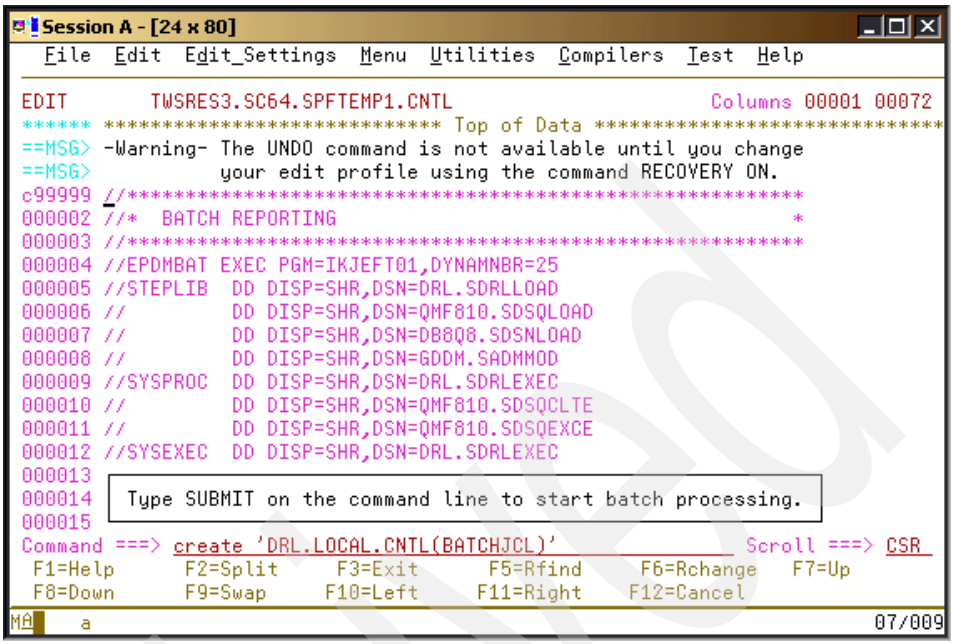

Figure 6-18 Creating report batch JCL

7. Either press F3 (Exit) or F12 (Cancel) to go back to the Reports panel.

6.5.2 Running the batch job

Adding a valid job card and submitting the JCL that you just created results in a JCL error because the ADMDEFS DD statement is coded with `DSN=` . So, you comment out that statement and submit the job again. Then, a report is produced. The destination of the actual report output is defined by the coding of the DSQPRINT DD statement. Example 6-10 on page 237 shows our JCL.

Note: To demonstrate the reports, we collected some historic tracklog data from another installation. It was a week's worth of data that covered the dates from January 31 to February 6. Thus, we used that date in our example.

Example 6-10 Report JCL as a batch job

```
//TWSRES3R JOB (ACCT#),'TWS REPORT09',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*****************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*****************************************************************
//*****************************************************************
//*  BATCH REPORTING                                              *
//*****************************************************************
//EPDMBAT EXEC PGM=IKJEFT01,DYNAMNBR=25
//STEPLIB  DD DISP=SHR,DSN=DRL.SDRLLOAD
//         DD DISP=SHR,DSN=QMF810.SDSQLOAD
//         DD DISP=SHR,DSN=DB8Q8.SDSNLOAD
//         DD DISP=SHR,DSN=GDDM.SADMMOD
//SYSPROC  DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF810.SDSQCLTE
//         DD DISP=SHR,DSN=QMF810.SDSQEXCE
//SYSEXEC  DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF810.SDSQCLTE
//         DD DISP=SHR,DSN=QMF810.SDSQEXCE
//*****************************************************************
//*  MESSAGES                                                     *
//*****************************************************************
//DRLOUT   DD SYSOUT=*
//*
//*****************************************************************
//*  SAVED TABULAR AND GRAPHIC REPORTS TO                         *
//*****************************************************************
//DRLREP   DD DISP=SHR,DSN=DRL.LOCAL.REPORTS
//ADMGDF   DD DISP=SHR,DSN=DRL.LOCAL.CHARTS
//*
//*****************************************************************
//*  PRINT REPORTS TO:                                            *
//*****************************************************************
//DSQPRINT DD SYSOUT=T,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//*
//*****************************************************************
//*  GDDM LIBRARIES                                               *
//*****************************************************************
//ADMGGMAP DD DISP=SHR,DSN=QMF810.SDSQMAPE
//ADMCFORM DD DISP=SHR,DSN=DRL.SDRLFENU
//         DD DISP=SHR,DSN=QMF810.SDSQCHRT
//ADMSYMBL DD DISP=SHR,DSN=GDDM.SADMSYM
//*ADMDEFS  DD DISP=SHR,DSN=
//DSQUCFRM DD DISP=SHR,DSN=DRL.LOCAL.ADMCFORM
//*****************************************************************
//*  QMF LIBRARIES                                                *
//*****************************************************************
```

```
//DSQDEBUG DD DUMMY
//DSQUDUMP DD DUMMY
//DSQPNLE  DD DISP=SHR,DSN=QMF810.DSQPNLE
//DSQSPILL DD DSN=&SPILL,DISP=(NEW,DELETE),UNIT=SYSDA,
//  SPACE=(CYL,(1,1),RLSE),DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
//DSQEDIT  DD DSN=&EDIT,UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//  DCB=(RECFM=FBA,LRECL=79,BLKSIZE=4029)
//DRLFORM  DD DSN=TWSRES3.DRLFORM,UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),
//  DCB=(RECFM=VB,LRECL=255,BLKSIZE=2600),DISP=(NEW,DELETE)
//*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  %DRLEBATR +
    SYSTEM=DB8Q +
    SYSPREFIX=DRLSYS +
    PREFIX=DRL +
    QMF=YES +
    GDDM=YES +
    REPORT=+
OPC09 +
    &DATE='''2005-02-01''' +
    &PERIOD_NAME='''PRIME   ''' +
    &OPC_SYSTEM_ID='''01''' +
     DIALLANG=1 +
     PRODNAME=Tivoli Decision Support +

/*
```

Remember: If your system is part of a sysplex, you need to code the correct SAFF= value on the JOBPARM statement.

This JCL is specific to the OPC09 report that was selected with the variables. To run the same report with different variables, you need to update the SYSTSIN DD statement before submitting the job again.

If you wanted to run a different report, you would need to change the SYSTSIN DD statement accordingly. It is best to change this statement as described in 6.5.1, “Creating the report batch JCL” on page 235 to ensure that everything is coded correctly.

Note: Unless you have a printer that is defined to GDDM®, graphical reports cannot be run this way in batch jobs. You get a message similar to the following:

```
The graphic report OPC07 cannot be printed
```

We explain how to manage graphical reports in “Graphical reports” on page 251.

So, there are several limitations to running batch reports on a regular basis in this way. The workaround to this limitation is to automate the reports, as shown in the next section.

6.5.3 Using TWS for z/OS to run the report job

You can now use the JCL that you created in DRL.LOCAL.CNTL(BATCHJCL) as a model for the TWS for z/OS report job. Create member T01T01D9 in the same TWS for z/OS job library in which the T01T01C1 job was created (TWS.INST.JOBLIB in our installation), copy the JCL from DRL.LOCAL.CNTL(BATCHJCL), and change the job name to T01T01D9.

Adding date variables

To run the report on a daily basis, you need to use a TWS for z/OS date variable to substitute the date. So, add the three lines of JCL, as shown in Example 6-11, just after the JOBPARM statement and change the &DATE=”’yyyy-mm-dd”’ variable on the SYSTSIN DD statement to &DATE=”’&TVAR1”’.

Example 6-11 Adding date variable information

```
//*%OPC SCAN
//*%OPC SETFORM OCDATE=(CCYY-MM-DD)
//*%OPC SETVAR TVAR1=(OCDATE-1CD)
```

Note: The DRLFORM DD statement is originally coded with `DSN=TWSRES3.DRLFORM`. So, we changed it to `DSN=&FORM` because the user ID that TWS for z/OS is using to submit batch does not have access to the TWSRES3 hlq.

Example 6-12 shows the modified JCL that is ready to submit to TWS for z/OS.

Example 6-12 Report JCL for Tivoli Workload Scheduler for z/OS submission

```
//TO1TO1D9 JOB (ACCT#),'TWS REPORTO9',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=144O,REGION=0M
//*****************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*****************************************************************
//*%OPC SCAN
//*%OPC SETFORM OCDATE=(CCYY-MM-DD)
//*%OPC SETVAR TVAR1=(OCDATE-1CD)
//*****************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(TO1TO1D9)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT           ON SC64
//*****************************************************************
//*                                                               *
//*  LICENSED MATERIALS - PROPERTY OF IBM                         *
//*                                                               *
//*  5695-1O1 (C) COPYRIGHT IBM CORPORATION 1993, 2OO3            *
//*  SEE COPYRIGHT INSTRUCTIONS.                                  *
//*                                                               *
//*****************************************************************
//*                                                               *
//*  Name:   DRLJBATR                                             *
//*                                                               *
//*  Status: Tivoli Decision Support for OS39O 1.6.0              *
//*                                                               *
//*  Function:                                                    *
//*    TDS for OS39O batch report job.                            *
//*                                                               *
//*****************************************************************
//EPDMBAT EXEC PGM=IKJEFTO1,DYNAMNBR=25
//STEPLIB  DD DISP=SHR,DSN=DRL.SDRLLOAD
//         DD DISP=SHR,DSN=QMF81O.SDSQLOAD
//         DD DISP=SHR,DSN=DB8Q8.SDSNLOAD
//         DD DISP=SHR,DSN=GDDM.SADMMOD
//SYSPROC  DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF81O.SDSQCLTE
//         DD DISP=SHR,DSN=QMF81O.SDSQEXCE
//SYSEXEC  DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF81O.SDSQCLTE
//         DD DISP=SHR,DSN=QMF81O.SDSQEXCE
//*************************************************************
//*  MESSAGES                                                 *
//*************************************************************
//DRLOUT   DD SYSOUT=*
//*
//*************************************************************
```

```
//*  SAVED TABULAR AND GRAPHIC REPORTS TO                  *
//*********************************************************
//DRLREP   DD DISP=SHR,DSN=DRL.LOCAL.REPORTS
//ADMGDF   DD DISP=SHR,DSN=DRL.LOCAL.CHARTS
//*
//*********************************************************
//*  PRINT REPORTS TO:                                    *
//*********************************************************
//DSQPRINT DD SYSOUT=T,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//*
//*********************************************************
//*  GDDM LIBRARIES                                       *
//*********************************************************
//ADMGGMAP DD DISP=SHR,DSN=QMF810.SDSQMAPE
//ADMCFORM DD DISP=SHR,DSN=DRL.SDRLFENU
//         DD DISP=SHR,DSN=QMF810.SDSQCHRT
//ADMSYMBL DD DISP=SHR,DSN=GDDM.SADMSYM
//*ADMDEFS  DD DISP=SHR,DSN=
//DSQUCFRM DD DISP=SHR,DSN=DRL.LOCAL.ADMCFORM
//*********************************************************
//*  QMF LIBRARIES                                        *
//*********************************************************
//DSQDEBUG DD DUMMY
//DSQUDUMP DD DUMMY
//DSQPNLE  DD DISP=SHR,DSN=QMF810.DSQPNLE
//DSQSPILL DD DSN=&SPILL,DISP=(NEW,DELETE),UNIT=SYSDA,
//  SPACE=(CYL,(1,1),RLSE),DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
//DSQEDIT  DD DSN=&EDIT,UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//  DCB=(RECFM=FBA,LRECL=79,BLKSIZE=4029)
//DRLFORM  DD DSN=&FORM,UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),
//  DCB=(RECFM=VB,LRECL=255,BLKSIZE=2600),DISP=(NEW,DELETE)
//*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
  %DRLEBATR +
    SYSTEM=DB8Q +
    SYSPREFIX=DRLSYS +
    PREFIX=DRL +
    QMF=YES +
    GDDM=YES +
    REPORT=+
OPC09 +
    &DATE='''&TVAR1''' +
    &PERIOD_NAME='''PRIME   ''' +
    &OPC_SYSTEM_ID='''01''' +
     DIALLANG=1 +
     PRODNAME=Tivoli Decision Support +
/*
//
```

Creating the report application

With the JCL created, you need to create a new application in TWS for z/OS for the report job by doing the following:

1. Copy the TDS#01#TWS#01#DC application.
2. Change the name to TDS#01#TWS#01#DR (the R standing for reports).
3. Change the application text to Daily TWS Report Jobs.
4. Change the predecessor of operation 001 to be operation 255 of the TDS#01#TWS#01#DC application to ensure that the report does not run until a successful collect of the tracklog data.
5. Change the job name of operation 005 to T01T01D9 to reflect that the job will run report OPC09.
6. Change the name of the special resource to T01.REPORTS.TWS on operation 005.

Note: You can change the run cycle to suit your requirements. For example, you might not want to run the job on the weekends. We decided to leave ours the same.

Running the batch job with in-stream JCL

We then added this new TDS#01#TWS#01#DR application to the current plan with an input arrival date of 02 February 2005 so that it produces a report for February 1. Remember that we have collected data from another installation that covers the dates from January 31 to February 6.

The batch job fails with an OJCV error code, a variable substitution failure, because the temporary data sets that are defined by the product JCL are a single ampersand (&) that is seen by TWS for z/OS to be a variable substitution character. You can see this error by editing the JCL on the error handler and paging down to the highlighted JCL after the actual JCL. Example 6-13 shows the section of JCL that identifies the problem.

Example 6-13 OJCV failure

```
=NOTE= //*>EQQJ535E 02/22 10.44.31
=NOTE= //*>         UNDEFINED VARIABLE SPILL LINE 00071 OF ORIG JCL
====== //DSQSPILL DD DSN=&SPILL,DISP=(NEW,DELETE),UNIT=SYSDA,
====== //  SPACE=(CYL,(1,1),RLSE),DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
====== //DSQEDIT  DD DSN=&EDIT,UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
====== //  DCB=(RECFM=FBA,LRECL=79,BLKSIZE=4029)
====== //DRLFORM  DD DSN=&FORM,UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),
====== //  DCB=(RECFM=VB,LRECL=255,BLKSIZE=2600),DISP=(NEW,DELETE)
```

Example 6-13 on page 242 only refers to the DSN=&SPILL on the DSQSPILL DD statement because that was the first failure. If we change this statement to DSN=&&SPILL and rerun the job, it will fail again for the DSN=&EDIT on the DSQEDIT DD statement. So, we changed all three of statements to be double ampersand (&&) in the T01T01D9 member of the TWS.INST.JOBLIB JCL library, delete the JCL within the failed job on the error handler to refresh it from the updated JCL library version, and rerun the job.

The job fails again with another OJCV error code, this time due to the %DRLEBATR on the SYSTSIN DD statement because a single percent symbol (%) is also a TWS for z/OS variable substitution character.

Tip: Other option to solve these issues are:

- If you have problems with the DSN=, then just leave out the DSN completely, and the system generates the temporary DSN.
- If you have problems with the percent symbol (%) in front of the REXX program name, you can leave that out as well.

These issues demonstrate that in-stream JCL with report variables cannot work due to TWS for z/OS date variable substitution restrictions. Not even the use of the NOSCAN statements helps because placing these statements around the values on the SYSTSIN DD statement results in the addition of a SYSIN DD statement and nothing is produced. Example 6-14 shows our attempt to work around these issues.

Example 6-14 Suppressing variable substitution attempt

```
//SYSTSIN  DD *
//*%OPC BEGIN ACTION=NOSCAN
  %DRLEBATR +
    SYSTEM=DB8Q +
    SYSPREFIX=DRLSYS +
    PREFIX=DRL +
    QMF=YES +
    GDDM=YES +
    REPORT=+
OPC09 +
    &DATE='''&TVAR1''' +
    &PERIOD_NAME='''PRIME   ''' +
    &OPC_SYSTEM_ID='''01''' +
     DIALLANG=1 +
     PRODNAME=Tivoli Decision Support +

/*
//*%OPC END ACTION=NOSCAN
```

Example 6-15 on page 244 shows the result of the attempt to use NOSCAN.

Example 6-15 Suppressing variable substitution result

```
//SYSTSIN  DD *
//*%OPC BEGIN ACTION=NOSCAN
//SYSIN     DD *               GENERATED STATEMENT
//*%OPC END ACTION=NOSCAN
//
```

No matter what we try, it will not work because the //*%OPC BEGIN ACTION=NOSCAN statement within the SYSTSIN DD statement results in a generated //SYSIN DD statement and the job fails.

To get the report to work successfully, we need to move the report request out of the JCL and away from the date variable substitution. Creating a procedure from the core JCL is also a good idea. It allows you to set up many different reports and to maintain only one copy of the JCL.

Because you cannot have in-stream statements within a procedure and cannot pass variables through a procedure into another data set, such as a report definition library, the best solution is to set up some REXX executable to execute the reports. This allows you to pass the date and other variables through the procedure to obtain the desired results.

The following sections describe the creation of the REXX executable for the OPC09 report and the JCL procedure.

6.5.4 Creating a REXX executable for the report

We created a REXX executable library, called DRL.LOCAL.REXX as a PDS with DCB attributes of (RECFM=VB,LRECL=255,BLKSIZE=27998) and created a member called OPC09. We then created the REXX code, as shown in Example 6-16, to execute the SQL statements for the OPC09 report.

Example 6-16 REXX code for the OPC09 report

```
/*rexx*/
Arg parm1 parm2 parm3 .

 exitrc = 0
 If SUBSTR(parm1,1,6) = 'DATE1=' then indate1 = SUBSTR(parm1,7)
 If SUBSTR(parm2,1,7) = 'PERIOD=' then period1= SUBSTR(parm2,8)
 If SUBSTR(parm3,1,4) = 'OSI=' then osi1= SUBSTR(parm3,5)
    datea = "&DATE='''"indate1"'''"
    perb  = "&PERIOD_NAME='''"period1"'''"
    osic  = "&OPC_SYSTEM_ID='''"osi1"'''"

repgen= " %DRLEBATR",
```

```
          " SYSTEM=DB8Q ",
          "  SYSPREFIX=DRLSYS ",
          "  PREFIX=DRL ",
          "  QMF=YES ",
          "  DUALSAVE=YES ",
          "  GDDM=NO" datea perb osic ,
          "  REPORT=OPC09 DIALLANG=1 ",
          "  PRODNAME=Tivoli Decision Support"
   ADDRESS TSO
        repgen
    rep_rc = rc
    If rep_rc ^=0 then
    Say "OPC09 report for "osi1" return code:"rep_rc
exit rep_rc
```

6.5.5 Creating a JCL procedure

We created a procedure library, called DRL.LOCAL.PROCLIB as a PDS (partitioned data set) with DCB attributes of (RECFM=FB, LRECL=80,BLKSIZE=23440) and created a member called TDSREP1R. Into this member, we copied the EPDMBAT step JCL, without the SYSTSIN DD statement, from our T01T01D9 job, as shown in Example 6-17.

Example 6-17 TDSREP1R procedure

```
//TDSREP PROC
//*************************************************************
//*  BATCH REPORTING                                          *
//*************************************************************
//EPDMBAT EXEC PGM=IKJEFT01,DYNAMNBR=25
//STEPLIB  DD DISP=SHR,DSN=DRL.SDRLLOAD
//         DD DISP=SHR,DSN=QMF810.SDSQLOAD
//         DD DISP=SHR,DSN=DB8Q8.SDSNLOAD
//         DD DISP=SHR,DSN=GDDM.SADMMOD
//SYSPROC  DD DISP=SHR,DSN=DRL.LOCAL.REXX
//         DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF810.SDSQCLTE
//         DD DISP=SHR,DSN=QMF810.SDSQEXCE
//SYSEXEC  DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF810.SDSQCLTE
//         DD DISP=SHR,DSN=QMF810.SDSQEXCE
//*************************************************************
//*  MESSAGES                                                 *
//*************************************************************
//DRLOUT   DD SYSOUT=*
//*
//*************************************************************
//*  SAVED TABULAR AND GRAPHIC REPORTS TO                     *
```

```
//*********************************************************
//DRLREP   DD DISP=SHR,DSN=DRL.LOCAL.REPORTS
//ADMGDF   DD DISP=SHR,DSN=DRL.LOCAL.CHARTS
//*
//*********************************************************
//*  PRINT REPORTS TO:                                    *
//*********************************************************
//DSQPRINT DD SYSOUT=T,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//*
//*********************************************************
//*  GDDM LIBRARIES                                       *
//*********************************************************
//ADMGGMAP DD DISP=SHR,DSN=QMF810.SDSQMAPE
//ADMCFORM DD DISP=SHR,DSN=DRL.SDRLFENU
//         DD DISP=SHR,DSN=QMF810.SDSQCHRT
//ADMSYMBL DD DISP=SHR,DSN=GDDM.SADMSYM
//*ADMDEFS  DD DISP=SHR,DSN=
//DSQUCFRM DD DISP=SHR,DSN=DRL.LOCAL.ADMCFORM
//*********************************************************
//*  QMF LIBRARIES                                        *
//*********************************************************
//DSQDEBUG DD DUMMY
//DSQUDUMP DD DUMMY
//DSQPNLE  DD DISP=SHR,DSN=QMF810.DSQPNLE
//DSQSPILL DD DSN=&&SPILL,DISP=(NEW,DELETE),UNIT=SYSDA,
//  SPACE=(CYL,(1,1),RLSE),DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
//DSQEDIT  DD DSN=&&EDIT,UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//  DCB=(RECFM=FBA,LRECL=79,BLKSIZE=4029)
//DRLFORM  DD DSN=&&FORM,UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),
//  DCB=(RECFM=VB,LRECL=255,BLKSIZE=2600),DISP=(NEW,DELETE)
//*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//*
```

Note: We concatenate the DRL.LOCAL.REXX data set as the first entry on the SYSPROC DD statement, ahead of the ones that were already there, to ensure that our OPC09 REXX executable code gets picked up properly.

6.5.6 Setting the report batch settings

With the REXX executable and the procedure created, the last thing you need to do is set the batch settings for the report to enable the output to be directed to a data set rather than wherever the DSQPRINT DD statement directs it. Refer to *Tivoli Decision Support for OS/390 Guide to the Reporting Dialog Version 1.6*, SH19-6842 for more information.

To set the batch settings:

1. Select option **1 (**Reports**)** from the Tivoli Decision Support for OS390 Primary Menu panel and press Enter.
2. Select **OPC09 report** and press F10 (Actions).
3. Tab across to the Batch menu and press Enter.
4. Select option **1** (Set batch options for report...) and press Enter.
5. Complete the fields as shown in Figure 6-19 and press Enter.

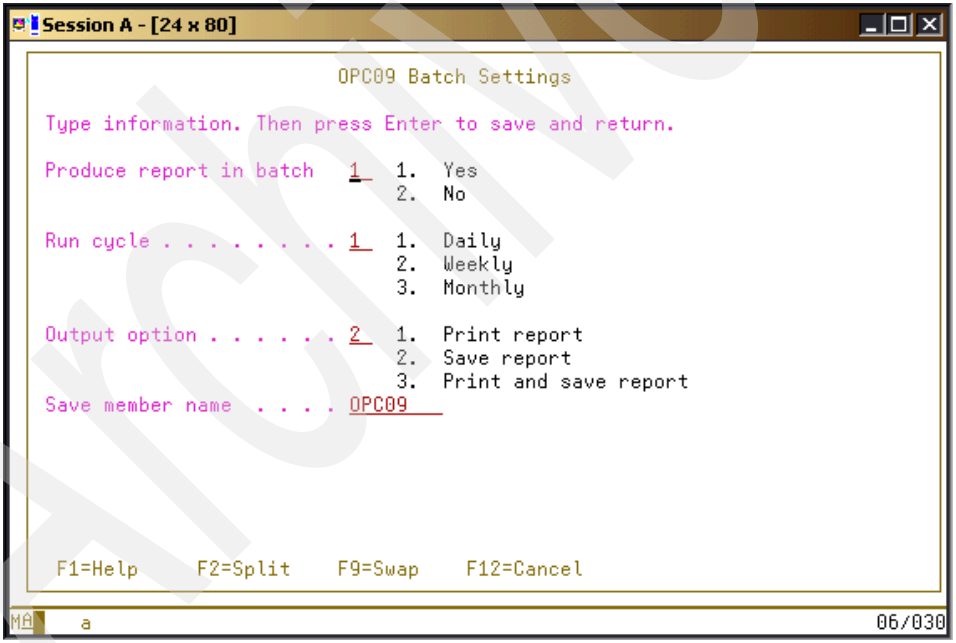

Figure 6-19 Batch settings for OPC09

When complete, you will receive the following message:

```
The batch setting is saved successfully.
```

This results in the actual report that is placed in member OPC09 of DRL.LOCAL.REPORTS when you run the batch job.

Note: This section concentrates on the OPC09 report, but you can select multiple reports before going to the Batch menu. Thus, you can set the batch settings for all selected reports one after the other.

6.5.7 Running the batch job with the procedure and REXX

Now that you have created a procedure and some REXX and have set the batch settings for the report, you need to alter the T01T01D9 JCL in TWS.INST.JOBLIB so that it executes the TDSREP1R procedure with an override for the SYSTSIN DD statement that provides the actual report request, as shown in Example 6-18.

Example 6-18 Report JCL executing the REXX code

```
//T01T01D9 JOB (ACCT#),'TWS REPORT',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*****************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*****************************************************************
//*%OPC SCAN
//*%OPC SETFORM OCDATE=(CCYY-MM-DD)
//*%OPC SETVAR TVAR1=(OCDATE-1CD)
//*****************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(T01T01D9)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT            ON SC64
//*****************************************************************
//*                                                               *
//*  LICENSED MATERIALS - PROPERTY OF IBM                         *
//*                                                               *
//*  5695-101 (C) COPYRIGHT IBM CORPORATION 1993, 2003            *
//*  SEE COPYRIGHT INSTRUCTIONS.                                  *
//*                                                               *
//*****************************************************************
//*                                                               *
//*  Name:   DRLJBATR                                             *
//*                                                               *
//*  Status: Tivoli Decision Support for OS390 1.6.0              *
//*                                                               *
//*  Function:                                                    *
//*    TDS for OS390 batch report job.                            *
//*                                                               *
//*****************************************************************
//JOBPROC  JCLLIB ORDER=(DRL.LOCAL.PROCLIB)
//*
```

```
//STEP01   EXEC TDSREP1R
//EPDMBAT.SYSTSIN  DD *
  EX 'DRL.LOCAL.REXX(OPC09)' +
   'DATE1=&TVAR1 PERIOD=PRIME OSI=01'
/*
//
```

Rerunning the T01T01D9 job from the TWS for z/OS error handler after refreshing the JCL, results in the report being produced in member OPC09 of the DRL.LOCAL.REPORTS data set. Figure 6-20 shows the top part of the report.

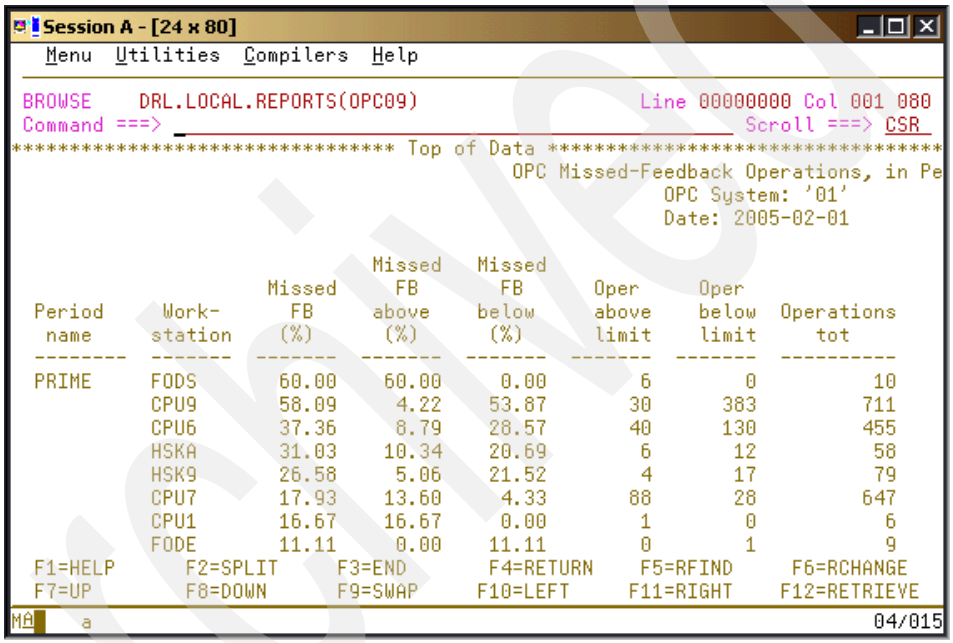

Figure 6-20 Result of the OPC09 report

As well as viewing the report in the data set member, it can also be sent somewhere, if required, using a file transfer process such as Xmit, NetView File Transfer Program, or TCP/IP File Transfer Program.

Note: If the TWS for z/OS JCL library member that contains the report JCL has NUMBER ON as part of its profile, the job will fail with the OJCV error code, and the message shown in Example 6-19 on page 250 is generated. Change the profile to NUMBER OFF in the JCL library, refresh the JCL on the error handler again, and rerun.

Example 6-19 Effect of NUMBER ON when executing the REXX code

```
//STEP01   EXEC TDSREP1R
//EPDMBAT.SYSTSIN  DD *
  EX 'DRL.LOCAL.REXX(OPC09)' +
//*>EQQJ530E 02/23 15.16.05
//*>         SCANNING LENGTH EXCEEDED LINE 00034 OF ORIG JCL
   'DATE1=&TVAR1 PERIOD=PRIME OSI=01'
/*
//
```

This particular example produced a report that was based on the period name of PRIME. To run the same report for a different period name, create a new job with the same JCL except for the PERIOD= part.

6.5.8 Other report information

So far, our discussion has all been based on the OPC09 report. For any of the other reports, you would need to perform similar actions. The report job(s) would either be added to the TDS#01#TWS#01#DR, TDS#01#TWS#01#WR or TDS#01#TWS#01#MR application for daily, weekly, or monthly processing as required.

Note: The TDS#01#TWS#01#WR and TDS#01#TWS#01#MR applications can be created by copying the TDS#01#TWS#01#DR application and making the necessary changes, such as the run cycle and job names.

Tabular reports

Figure 6-21 on page 251 shows an example of another tabular report, the OPC05 report. This report requires two dates variables and a specific job name as well as the OSI value.

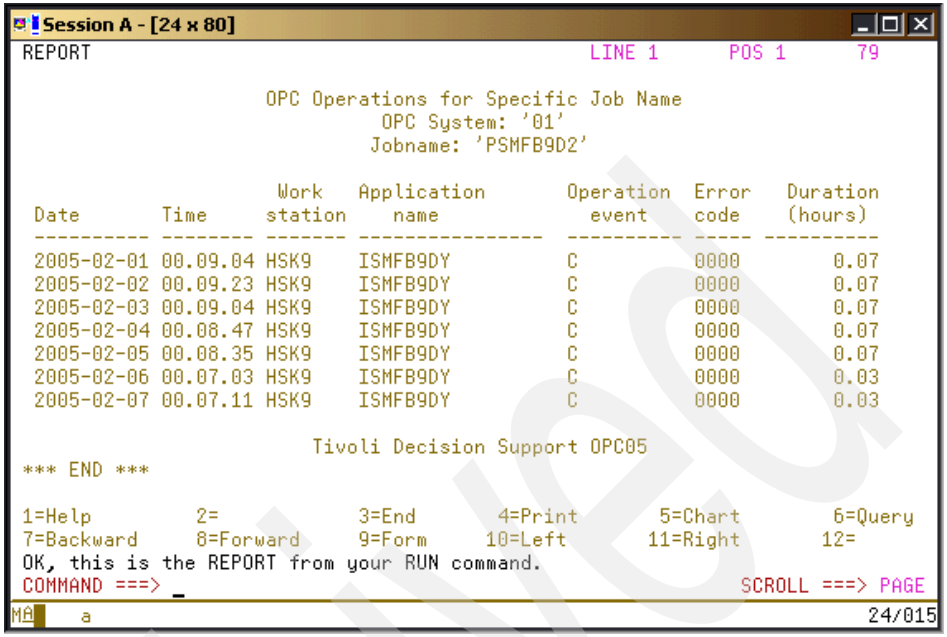

Figure 6-21 The OPC05 report

Graphical reports

For the graphical reports OPC07, OPC10, and OPC11, a corresponding member is also created in the DRL.LOCAL.CHARTS data set. However, this member needs to be converted into a GIF file first if it is to be transmitted for viewing. Example 6-20 shows the JCL that we used for the OPC11 report. The resultant DRL.MEUB9.OPC11.GIFBIN data set is the one that we would transmit.

Example 6-20 JCL for GIF conversion

```
//STEP02  EXEC PGM=IKJEFT01,REGION=3M,COND=(0,NE)
//ADMSYMBL DD DISP=SHR,DSN=GDDM.SADMSYM
//ADMGDF   DD DISP=SHR,DSN=DRL.LOCAL.CHARTS
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN  DD *
 PROFILE PREFIX(DRL)
  CALL 'GDDM.SADMMOD(ADMUGIF)' 'FROM(OPC11) BACKCOL(-2) +
    W(1024) D(600) A(1) T(-1) +
 TO(MEUB9.OPC11.GIFBIN) '
/*
```

Note: For the graphical reports you can also use the *Viewer*. It is delivered on AS-IS basis with TDS 1.7. The Viewer is a Java™ application running in Windows or UNIX. It has a "batch" facility to produce GIF files.

OPC10

The OPC10 report will not work unless you have the MVS Component installed, because it shows a comparison between the number of jobs that are processed by OPC and the total number of jobs that are processed. The OPC10 report needs information from the MVS_ADDRSPACE_M table. You get the panel shown in Figure 6-22 if you try to run this report online.

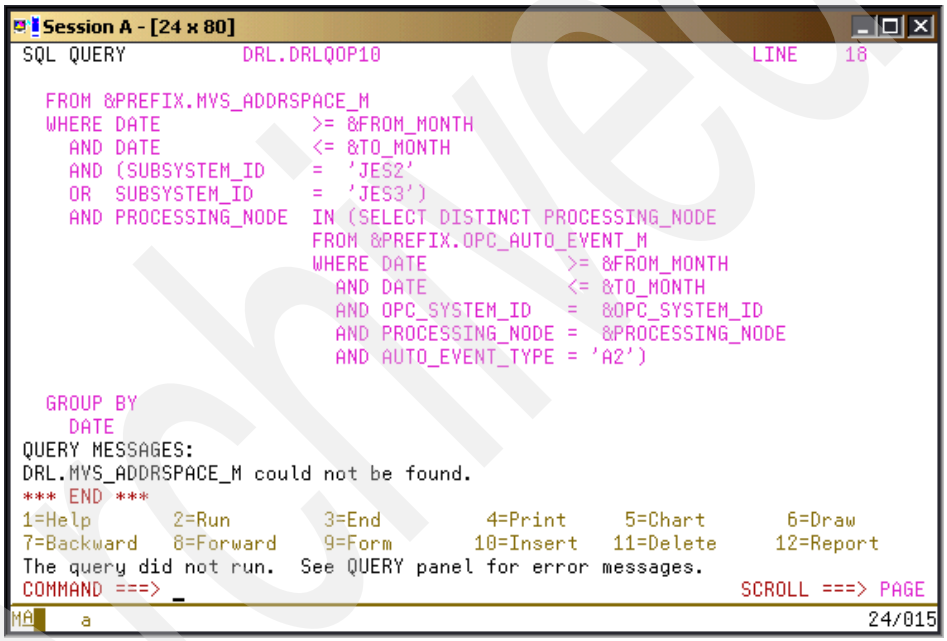

Figure 6-22 OPC10 report failure

In addition, the OPC10 report requires the value of the processing node. You find this value by browsing the TWS for z/OS controller started task that is running in the system and by searching for the EQQZ073I message. The node name at the end of the second line is the value needed. Figure 6-23 on page 253 shows our installation value of WTSCPLX2.

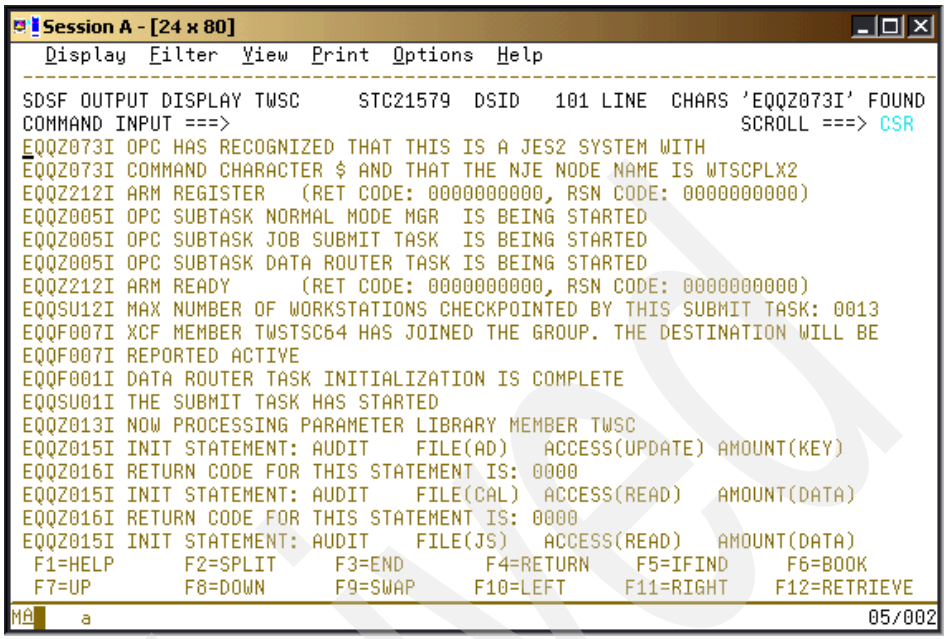

Figure 6-23 Finding the PSN value

An example of the OPC10 report is shown in Figure 6-24 on page 254. This image was captured from another installation due to the lack of historical data in our installation.

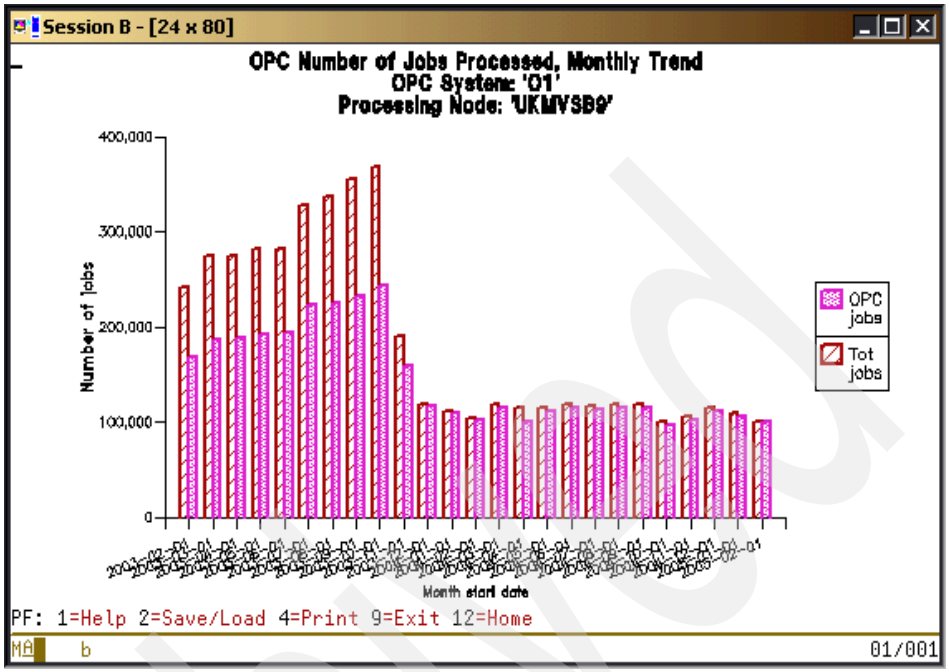

Figure 6-24 OPC10 report example

6.5.9 A daily TWS for z/OS view of the Tivoli Decision Support for OS/390 applications

With the data collect application being scheduled on a daily basis, along with the Tivoli Decision Support for OS/390 database purge application and DB2 housekeeping application, Figure 6-25 on page 255 shows a view of the current plan.

Note: The TDS#01#DLY#DB2HK application has an input arrival time of one minute later than the others, because it is dependent on both of the applications completing and the dependent applications, in turn, are dependant on it completing for their occurrences the following day. If the TDS#01#DLY#DB2HK application had the same input arrival time, it would cause a dependency loop in the current plan extend process.

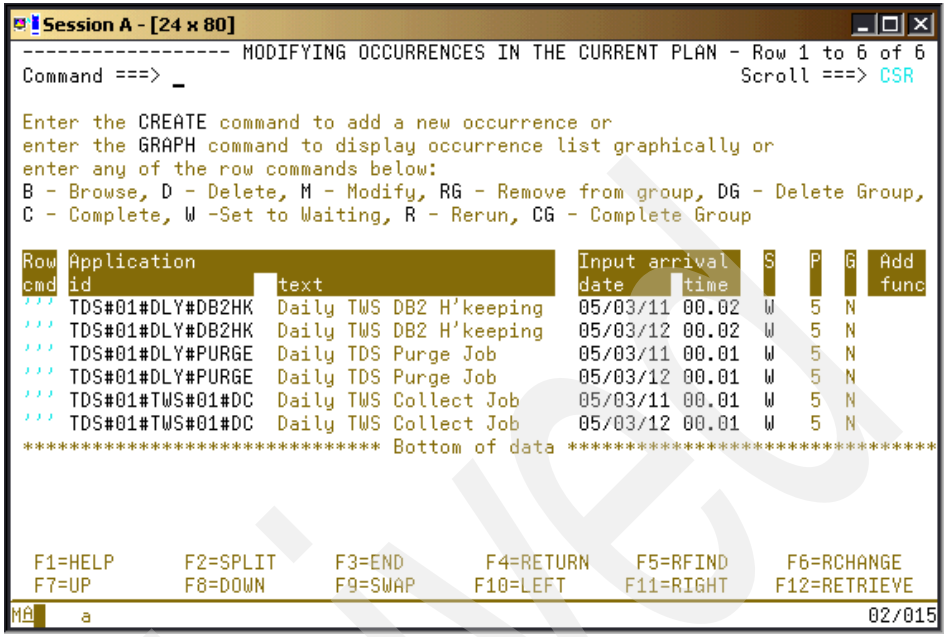

Figure 6-25 A daily view in the current plan

Entering b.3 alongside the TDS#01#TWS#01#DC application for 12 March 2005 shows us the dependencies that exist. These are the TDS#01#DLY#DB2HK application and the TWSTRACKLOGCOPY application from 11 March 2005 as predecessors, and the TDS#01#DLY#DB2HK for 12 March 2005 as a successor. Figure 6-26 on page 256 shows these dependencies.

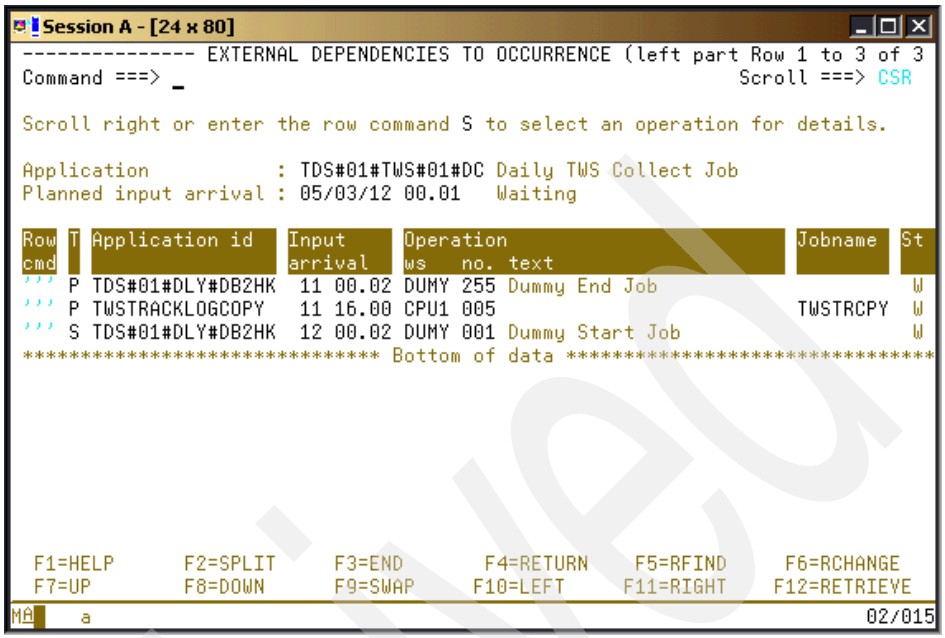

Figure 6-26 Daily dependencies

6.5.10 Overcoming the reporting restrictions

The preceding sections demonstrated how restrictive the supplied reports can be for automating. Although an alternative is suggested in *Tivoli Decision Support for OS/390 Guide to the Reporting Dialog Version 1.6*, SH19-6842, practical experience illustrates that the suggested alternative is also restrictive because the dates are different every day the job runs. For example, if a batch report job that is scheduled to run on a Monday for the previous seven days, as suggested in the manual, does not run until Tuesday for some reason, the resulting report will be based on the wrong seven days.

In addition, most of the supplied reports are not ideal for running on a daily basis because they are either date range or are monthly based. So, we decided what reports we wanted to run, on both a weekly and monthly basis, and created our own version of them with fewer variables that need to be specified.

To do this, we actually created our own component called Extra OPC Component. The next section explains how to define and install this component and the changes that you need to make to use it. The code is available for your use. To download the code, refer to Appendix D, “Additional material” on page 687.

6.6 The Extra OPC Component

The Extra OPC Component is basically a rewrite of the standard OPC reports with changes made to the selection criteria to make them more flexible for automation. The OPC_OPER_EVNTJ_D/M tables have also been created as copies of the standard OPC_OPER_EVENT_D/M tables with the addition of the JOB_NAME column and a change to the purge criteria, from 30 to 35 days, for the daily table.

Note: The MVS_SYSTEM_ID field does not exist in the OPC_OPER_EVNTJ_D/M tables because they were created before that field was added to the OPC_OPER_EVENT_D/M tables.

The associated views have also been rewritten for the Extra OPC Component. The new OPC16T report (OPC Jobs Ended in error by Error Code) uses these views and is a useful addition to the report set.

6.6.1 Defining the extra component

We created the following members in the DRL.LOCAL.DEFS data set, which are the extra component definition statements:

DRLIOPCE	Inserts into the DRLCOMPONENTS and DRLCOMP_OBJECTS tables
DRLOOPCE	Report group and report definitions
DRLTOPCJ	Additional table definition
DRLVOPCJ	Additional view definitions
TFOPnn	Forms for reports OPCnnT
TQOPnn	Queries for reports OPCnnT

6.6.2 Adding the extra component to the Components list

To add this extra component to the list of components in the Components panel:

1. Select option **2** (Administration) from the Tivoli Decision Support for OS390 Primary Menu panel and press Enter.
2. Press F10 (Actions), go to the Other menu, and press Enter.
3. Select option **5** (Process TDS for OS390 statements...) and press Enter.
4. Complete the panel as shown in Figure 6-27 on page 258 and press F5 (Execute) for the Extra OPC Component to appear in the list of components. The results of the SQL statements execution are returned to our *userid*.DRLOUT data set.

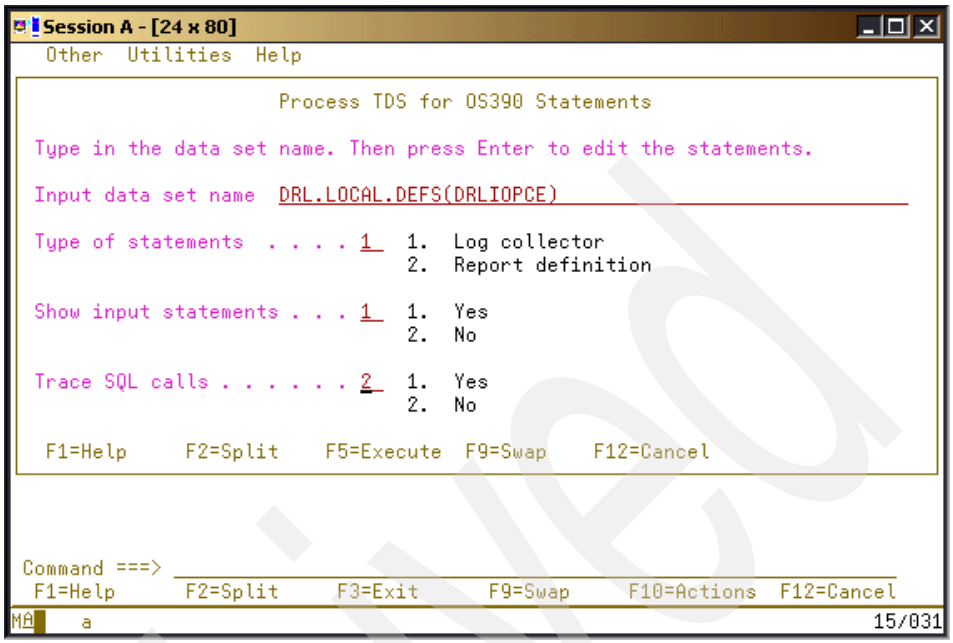

Figure 6-27 Defining the extra component

Recommendation: To save the installation results for future reference, press F2 (Split) to get another ISPF menu and rename the *userid*.DRLOUT data set. For example, we renamed ours to the following:

DRL.DRLOUT.TDS16.OPCEXTRA.COMP.DEFINED.

The Extra OPC Component is now listed with the other components. Example 6-21 shows part of the list.

Example 6-21 Extra component listed

```
Component  Space  Other  Help
------------------------------------------------------------------------------
                                Components                  Row 13 to 25 of
Select one or more components. Then press Enter to Open component.

/   Components                                          Status     Date
_   DFSMS Component
_   Extra OPC Component
_   EREP Component
_   Internet Connection Secure Server Component
_   IMS 5.1 Collect Component
_   IMS 5.1 Log Records Component
```

6.6.3 Installing the extra component

With the Extra OPC Component now listed, you can install it by selecting it, pressing F6 (Install), and then selecting the preferred method.

If you install online, the results are returned to the *userid*.DRLOUT data set with the following message overlaid:

```
Component 'Extra OPC Component' is installed successfully.
```

Pressing F3 (Exit) returns you to the Components panel where you can now see that the Extra OPC Component is installed (see Figure 6-28).

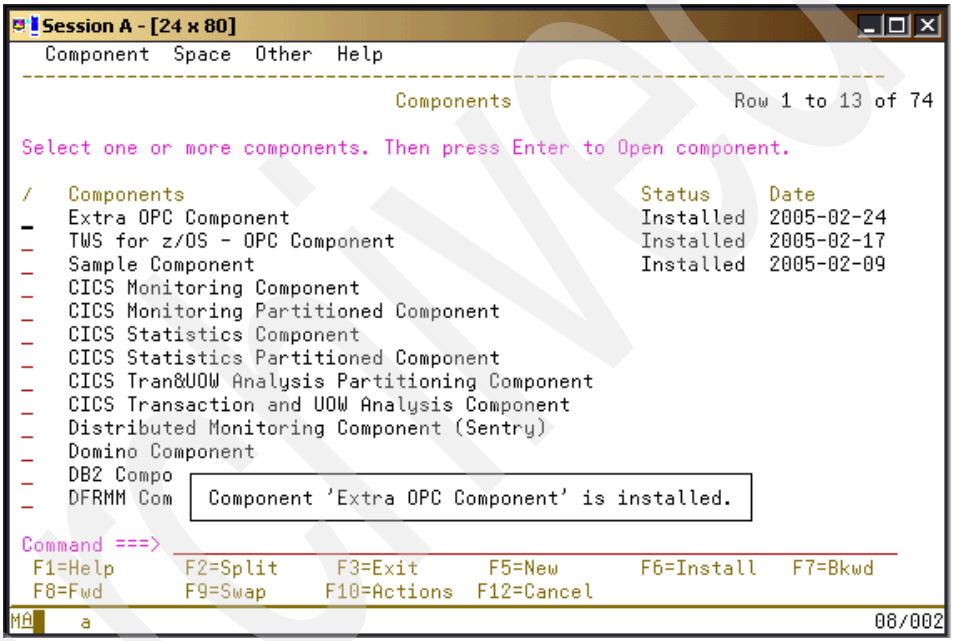

Figure 6-28 Component list with the extra component installed

Recommendation: To save the installation results for future reference, press F2 (Split) to get another ISPF menu and rename the *userid*.DRLOUT data set. For example, we renamed ours to DRL.DRLOUT.TDS16.OPCEXTRA.COMP.INSTALL.

If you install via batch, you are presented with the JCL for submission and the status on the Components panel shows as Batch.

Important: If your system is part of a sysplex, remember to code the correct SAFF= value on the JOBPARM statement.

Recommendation: To save the installation results, change the DRLOUT DD statement on all of the RUNLOG, RUNRDEF, and COPYMSG steps to create your desired data set. Alternatively, you could leave the JCL as it is and save the whole job output.

As well as seeing that the Extra OPC Component is installed, you can see that the extra reports are in place by doing the following:

1. Press F3 (Exit) from the Components panel.
2. Either select option **5** (Reports) from the Administration panel and press Enter, or press F3 (Exit) once more and select option **1** (Reports) from the Tivoli Decision Support for OS390 Primary Menu and press Enter. The list of reports shown are the standard component ones.
3. Press F4 (Groups) to see the list of report groups. The OPC Reports Extra group is listed as shown in Figure 6-29.

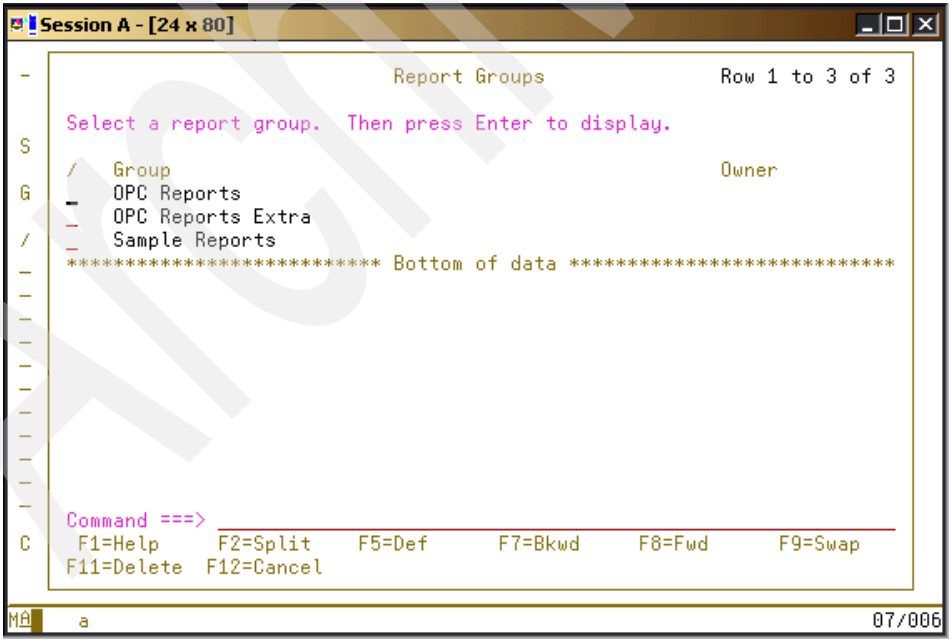

Figure 6-29 Extra component report group

4. Select this group and press Enter produces the list of extra reports, as shown in Figure 6-30, which are very similar to the standard reports. In fact, OPC16T and OPC15T are the only new reports and all the others are re-writes of the regular reports that are required.

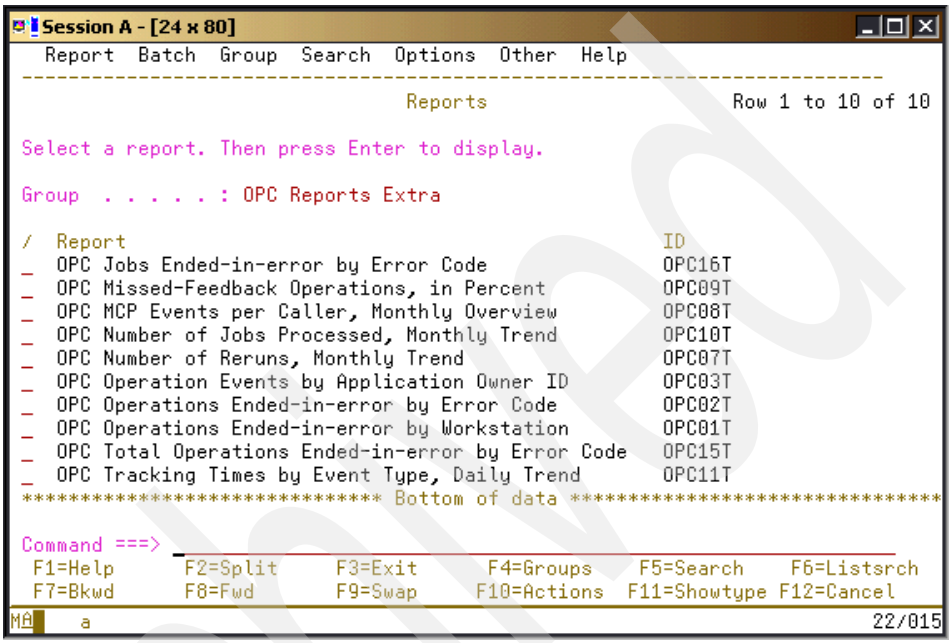

Figure 6-30 Extra component report list

The batch settings for all these new reports have been set as part of the install because the definition of the reports in the DRLOOPCE member have the words `BATCH SAVE FILE OPC`*nn* (where *nn* is the report number) coded.

Note: If you have not already done so in your installation, you need to set the batch settings for the standard OPC06 and OPC14 reports because they are included in the extra jobs.

6.6.4 REXX executable changes

The REXX execs that are provided are coded for the modified reports. Example 6-22 shows how different the code looks when choosing the OPC09T report.

Example 6-22 REXX for the OPC09T report

```
/*rexx*/
Arg parm1 parm2 parm3  .

 exitrc = 0
 If SUBSTR(parm1,1,6) = 'DATE1=' then indate1 = SUBSTR(parm1,7)
 If SUBSTR(parm2,1,6) = 'DATE2=' then indate2 = SUBSTR(parm2,7)
 If SUBSTR(parm3,1,4) = 'OSI=' then osi2 = SUBSTR(parm3,5)
    datea = "&TO_DATE='''"indate1"''''"
    dateb = "&FROM_DATE='''"indate2"''''"
    osic  = "&OPC_SYSTEM_ID='''"osi2"''''"

repgen= " %DRLEBATR",
          " SYSTEM=DB8Q PREFIX=DRL",
          "  SYSPREFIX=DRLSYS QMF=YES ",
          "  DUALSAVE=YES ",
          "  GDDM=NO" datea dateb osic ,
          "  REPORT=OPC09T DIALLANG=1 ",
          "  PRODNAME=Tivoli Decision Support"
   ADDRESS TSO
        repgen
    rep_rc = rc
    If rep_rc ^=0 then
    Say "OPC09T report for "osi2" return code:"rep_rc
exit rep_rc
```

Compare this example with the original OPC09 REXX executable in Example 6-16 on page 244. We have provided the ability to supply two dates and produce a report between them and have removed the need for the period parameter.

6.6.5 JCL procedure changes

The step name within the TDSREP1R procedure has been changed from EPDMBAT to STEP01.

Because three of the new batch jobs create data sets to be transmitted, you need a separate procedure to make job failure reruns easier. We created the TDSREP2R procedure, as a copy of the TDSREP1R procedure. We then added

STEP00 ahead of STEP01 and added STEP02 after STEP01. Example 6-23 shows the TDSREP2R procedure.

Example 6-23 TDSREP2R procedure

```
//TDSREP PROC
//*************************************************************
//*  BATCH REPORTING                                          *
//*************************************************************
//STEP00  EXEC PGM=IEFBR14
//*
//STEP01  EXEC PGM=IKJEFT01,DYNAMNBR=25
//STEPLIB  DD DISP=SHR,DSN=DRL.SDRLLOAD
//         DD DISP=SHR,DSN=QMF810.SDSQLOAD
//         DD DISP=SHR,DSN=DB8Q8.SDSNLOAD
//         DD DISP=SHR,DSN=GDDM.SADMMOD
//SYSPROC  DD DISP=SHR,DSN=DRL.LOCAL.REXX
//         DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF810.SDSQCLTE
//         DD DISP=SHR,DSN=QMF810.SDSQEXCE
//SYSEXEC  DD DISP=SHR,DSN=DRL.SDRLEXEC
//         DD DISP=SHR,DSN=QMF810.SDSQCLTE
//         DD DISP=SHR,DSN=QMF810.SDSQEXCE
//*************************************************************
//*  MESSAGES                                                 *
//*************************************************************
//DRLOUT   DD SYSOUT=*
//*
//*************************************************************
//*  SAVED TABULAR AND GRAPHIC REPORTS TO                     *
//*************************************************************
//DRLREP   DD DISP=SHR,DSN=DRL.LOCAL.REPORTS
//ADMGDF   DD DISP=SHR,DSN=DRL.LOCAL.CHARTS
//*
//*************************************************************
//*  PRINT REPORTS TO:                                        *
//*************************************************************
//DSQPRINT DD SYSOUT=T,DCB=(RECFM=FBA,LRECL=133,BLKSIZE=1330)
//*
//*************************************************************
//*  GDDM LIBRARIES                                           *
//*************************************************************
//ADMGGMAP DD DISP=SHR,DSN=QMF810.SDSQMAPE
//ADMCFORM DD DISP=SHR,DSN=DRL.SDRLFENU
//         DD DISP=SHR,DSN=QMF810.SDSQCHRT
//ADMSYMBL DD DISP=SHR,DSN=GDDM.SADMSYM
//*ADMDEFS  DD DISP=SHR,DSN=
//DSQUCFRM DD DISP=SHR,DSN=DRL.LOCAL.ADMCFORM
//*************************************************************
```

```
//*  QMF LIBRARIES                                              *
//*************************************************************
//DSQDEBUG DD DUMMY
//DSQUDUMP DD DUMMY
//DSQPNLE  DD DISP=SHR,DSN=QMF810.DSQPNLE
//DSQSPILL DD DSN=&&SPILL,DISP=(NEW,DELETE),UNIT=SYSDA,
//  SPACE=(CYL,(1,1),RLSE),DCB=(RECFM=F,LRECL=4096,BLKSIZE=4096)
//DSQEDIT  DD DSN=&&EDIT,UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
//  DCB=(RECFM=FBA,LRECL=79,BLKSIZE=4029)
//DRLFORM  DD DSN=&&FORM,UNIT=SYSDA,SPACE=(TRK,(5,5),RLSE),
//  DCB=(RECFM=VB,LRECL=255,BLKSIZE=2600),DISP=(NEW,DELETE)
//*
//SYSPRINT DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//*
//STEP02  EXEC PGM=IKJEFT01,REGION=3M,COND=(0,NE)
//ADMSYMBL DD DISP=SHR,DSN=GDDM.SADMSYM
//ADMGDF   DD DISP=SHR,DSN=DRL.LOCAL.CHARTS
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//*
```

6.6.6 TWS for z/OS environment

With the Extra OPC Component installed, we only run weekly and monthly report batch jobs. We do not require any reports on a daily basis, so the TDS#01#TWS#01#DR application has been pended.

Weekly jobs

All the reports that we want on a weekly basis are scheduled using the TDS#01#TWS#01#WR application. The run cycle is defined for every Monday of every week so that we produce reports based on the previous Monday to Sunday inclusive. Figure 6-31 on page 265 shows the operations list.

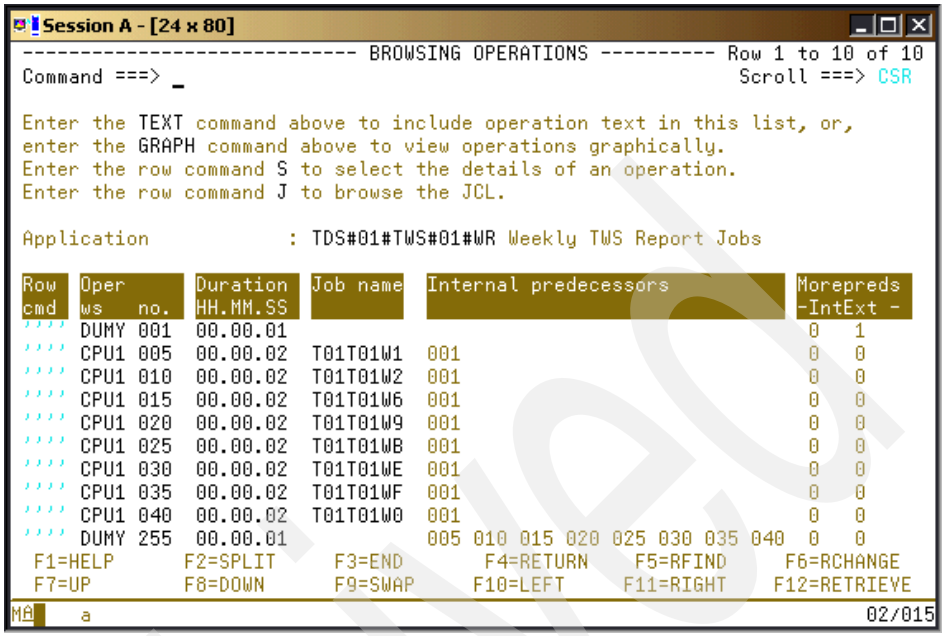

Figure 6-31 The weekly application

The predecessor to the 001 dummy start operation is the TDS#01#TWS#01#DC daily collect application. This ensures that all of Sunday's tracklog data has been collected into Tivoli Decision Support for OS/390 before running the reports. Each report operation has the T01.REPORTS.TWS special resource defined to it. This allows all of them to run independently, one at a time, even if one fails.

Monthly jobs

All the reports that we want on a monthly basis are scheduled using the TDS#01#TWS#01#MR application. The run cycle is defined for only the 2nd day of every month so that we produce reports based on the previous month. Figure 6-32 on page 266 shows the operations list.

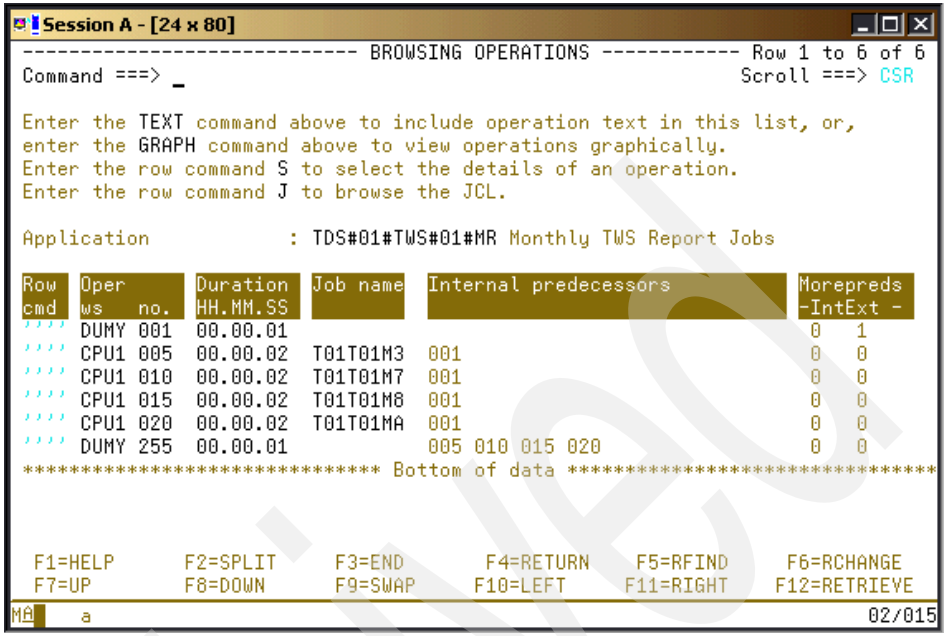

Figure 6-32 The monthly application

The predecessor to the 001 dummy start operation is the TDS#01#TWS#01#DC daily collect application. This ensures that all of the tracklog data for the previous month has been collected into Tivoli Decision Support for OS/390 before running the reports. Each report operation has the T01.REPORTS.TWS special resource defined to it. This allows all of them to run independently, one at a time, even if one fails.

Running our example job

Continuing with the OPC09 report example, Example 6-24 shows the JCL for the weekly OPC09T report.

Example 6-24 T01T01W9 JCL

```
//T01T01W9 JOB (ACCT#),'TWS REPORT OPC09T',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*****************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*****************************************************************
//*%OPC RECOVER JOBCODE=(S*,U*,5-4095),RESTART=Y
//*%OPC SCAN
//*%OPC SETFORM OCDATE=(CCYY-MM-DD)
//*%OPC SETVAR TVAR1=(OCDATE-1CD)
//*%OPC SETVAR TVAR2=(OCDATE-7CD)
//*****************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(T01T01W9)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT            ON SC64
//*****************************************************************
//*                                                               *
//*  LICENSED MATERIALS - PROPERTY OF IBM                         *
//*                                                               *
//*  5695-101 (C) COPYRIGHT IBM CORPORATION 1993, 2003            *
//*  SEE COPYRIGHT INSTRUCTIONS.                                  *
//*                                                               *
//*****************************************************************
//*                                                               *
//*  Name:   DRLJBATR                                             *
//*                                                               *
//*  Status: Tivoli Decision Support for OS390 1.6.0              *
//*                                                               *
//*  Function:                                                    *
//*    TDS for OS390 batch report job.                            *
//*                                                               *
//*****************************************************************
//JOBPROC  JCLLIB ORDER=(DRL.LOCAL.PROCLIB)
//*
//STEP01  EXEC TDSREP1R
//*
//STEP01.SYSTSIN  DD *
 EX 'DRL.LOCAL.REXX(OPC09T)' +
 'DATE1=&TVAR1 DATE2=&TVAR2 OSI=01'
/*
//
```

Compare this JCL with the daily JCL that we created in Example 6-18 on page 248. We now have two date variables and no PERIOD= value to match the modified REXX executable.

We added the TDS#01#TWS#01#WR application to the current plan with an input arrival date of 07 February 2005, for Monday February 7, and only the T01T01W9 job within. The report produced, shown in Figure 6-33, can be compared with the report that we created earlier in Figure 6-20 on page 249.

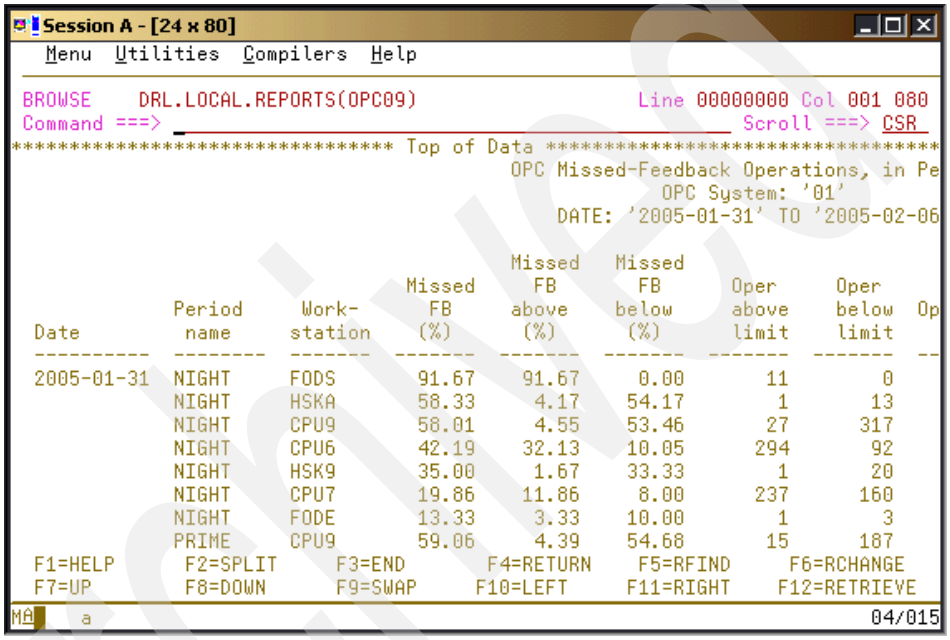

Figure 6-33 Result of the OPC09T report

6.6.7 Retaining report data

Now that you are producing the data that you want on both a weekly and monthly basis, you might want to keep it for a certain amount of time. You can use TWS for z/OS date variables to help retain report data.

Graphical

You need to convert graphical reports to GIF files if you want to transmit them somewhere (as discussed in “Graphical reports” on page 251). If you need to retain the data, it would be better to retain the GIF data set rather than the raw GDDM data. Example 6-25 on page 269 shows how the week number can be added to the output data set name for this purpose.

Example 6-25 Weekly graphical report JCL with week number in the data set name

```
//TO1TO1WB JOB (ACCT#),'TWS REPORT OPC11T',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*****************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*****************************************************************
//*%OPC RECOVER JOBCODE=(S*,U*,5-4095),RESTART=Y
//*%OPC SCAN
//*%OPC SETFORM OCDATE=(CCYY-MM-DD)
//*%OPC SETVAR TVAR1=(OCDATE-1CD)
//*%OPC SETVAR TVAR2=(OCDATE-7CD)
//*%OPC SETVAR TVARW=(OWW-1CD)
//*****************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(TO1TO1WB)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT           ON SC64
//*****************************************************************
//*                                                               *
//*  LICENSED MATERIALS - PROPERTY OF IBM                         *
//*                                                               *
//*  5695-101 (C) COPYRIGHT IBM CORPORATION 1993, 2003            *
//*  SEE COPYRIGHT INSTRUCTIONS.                                  *
//*                                                               *
//*****************************************************************
//*                                                               *
//*  Name:   DRLJBATR                                             *
//*                                                               *
//*  Status: Tivoli Decision Support for OS390 1.6.0              *
//*                                                               *
//*  Function:                                                    *
//*    TDS for OS390 batch report job.                            *
//*                                                               *
//*****************************************************************
//JOBPROC  JCLLIB ORDER=(DRL.LOCAL.PROCLIB)
//*
//STEP01  EXEC TDSREP2R
//STEP00.DDD DD DSN=DRL.MEUB9.OPC11&TVARW..GIFBIN,
//      DISP=(MOD,DELETE,DELETE),SPACE=(TRK,(0))
//*
//STEP01.SYSTSIN  DD *
 EX 'DRL.LOCAL.REXX(OPC11T)' +
 'DATE1=&TVAR1 DATE2=&TVAR2 OSI=01'
/*
//*
//STEP02.SYSTSIN  DD *
 PROFILE PREFIX(DRL)
  CALL 'GDDM.SADMMOD(ADMUGIF)' 'FROM(OPC11) BACKCOL(-2) +
    W(1024) D(600) A(1) T(-1) +
 TO(MEUB9.OPC11&TVARW..GIFBIN) '
/*
//
```

Compare the STEP02 SYSTSIN coding to that in Example 6-20 on page 251. The resultant data set name is DRL.MEUB9.OPC11nn.GIFBIN, where *nn* is the week number that is derived from the date variable.

Tabular

The tabular reports are within the DRL.LOCAL.REPORTS data set. So, you can copy the entire data set to DRL.LOCAL.REPORTS.Dyymmdd, for example, but it contains both weekly and monthly related members. So, we created separate weekly data sets for the weekly related members and monthly data sets for the monthly related members.

Weekly

Example 6-26 shows the JCL that creates a weekly data set for week 09 of the year 2005 called DRL.MEUB9.OPC.W092005. This JCL requires the input arrival date of the application to be 7 March 2005.

Example 6-26 Creating a weekly copy of the tabular reports

```
//TO1TO1WK JOB (ACCT#),'TWS WEEKLY ARCHIVE',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*******************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*******************************************************************
//*%OPC RECOVER JOBCODE=(S*,U*,5-4095),RESTART=Y
//*%OPC SCAN
//*%OPC SETVAR TVARW=(0WW-1CD)
//*%OPC SETVAR TVARY=(0YYYY-1CD)
//*******************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(TO1TO1WK)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT           ON SC64
//*******************************************************************
//* THIS JOB CREATES A WEEKLY COPY OF THE 'DRL.LOCAL.REPORTS'
//* DATA SET FOR ARCHIVING TO1 RELATED REPORTS.
//*******************************************************************
//*
//DELETE   EXEC PGM=IEFBR14
//OUT       DD DSN=DRL.MEUB9.OPC.W&TVARW.&TVARY,
//             DISP=(MOD,DELETE,DELETE),SPACE=(TRK,(0)),UNIT=SYSDA
/*
//*
//COPY     EXEC PGM=IEBCOPY,COND=(0,NE)
//SYSPRINT  DD SYSOUT=*
//IN        DD DISP=SHR,DSN=DRL.LOCAL.REPORTS
//OUT       DD DSN=DRL.MEUB9.OPC.W&TVARW.&TVARY,
//             DISP=(,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(TRK,(15,15,10),RLSE),
//             DCB=DRL.LOCAL.REPORTS
//SYSIN     DD *
    COPY INDD=IN,OUTDD=OUT
    SELECT MEMBER=(OPC01,OPC02,OPC06,OPC09,OPC11,OPC14,OPC15,OPC16)
/*
//
```

Monthly

Example 6-27 shows the JCL that creates a monthly data set for February of the year 2005 called DRL.MEUB9.OPC.MFEB2005. This JCL requires the input arrival date of the application to be 1 March 2005.

Example 6-27 Creating a monthly copy of the tabular reports

```
//TO1TO1MT JOB (ACCT#),'TWS MONTHLY ARCHIVE',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*************************************************************
//*%OPC RECOVER JOBCODE=(S*,U*,5-4095),RESTART=Y
//*%OPC SCAN
//*%OPC SEARCH NAME=(CALMNTHNUM)
//*%OPC SETVAR TVARM=(OMM-1CD)
//*%OPC SETVAR TVARY=(OYYYY-1CD)
//*
//* NOTE: THE TVARM VARIABLE IS USED AS THE INDEPENDANT VARIABLE
//* WHEN DEFINING THE CALMNTH DEPENDANT VARIABLE.
//*************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(TO1TO1MT)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT           ON SC64
//*************************************************************
//* THIS JOB CREATES A MONTHLY COPY OF THE 'DRL.LOCAL.REPORTS'
//* DATA SET FOR ARCHIVING TO1 RELATED REPORTS.
//*************************************************************
//*
//DELETE   EXEC PGM=IEFBR14
//OUT       DD DSN=DRL.MEUB9.OPC.M&CALMNTH.&TVARY,
//             DISP=(MOD,DELETE,DELETE),SPACE=(TRK,(0)),UNIT=SYSDA
/*
//*
//COPY     EXEC PGM=IEBCOPY,COND=(0,NE)
//SYSPRINT  DD SYSOUT=*
//IN        DD DISP=SHR,DSN=DRL.LOCAL.REPORTS
//OUT       DD DSN=DRL.MEUB9.OPC.M&CALMNTH.&TVARY,
//             DISP=(,CATLG,DELETE),UNIT=SYSDA,
//             SPACE=(TRK,(15,15,10),RLSE),
//             DCB=DRL.LOCAL.REPORTS
//SYSIN     DD *
    COPY INDD=IN,OUTDD=OUT
    SELECT MEMBER=(OPC03,OPC07,OPC08,OPC10)
/*
//
```

JCL variable table

The batch job in Example 6-27 on page 271 uses a JCL variable table called CALMNTHNUM to create the three character month name part of the data set name. Figure 6-34 shows the initial creation panel.

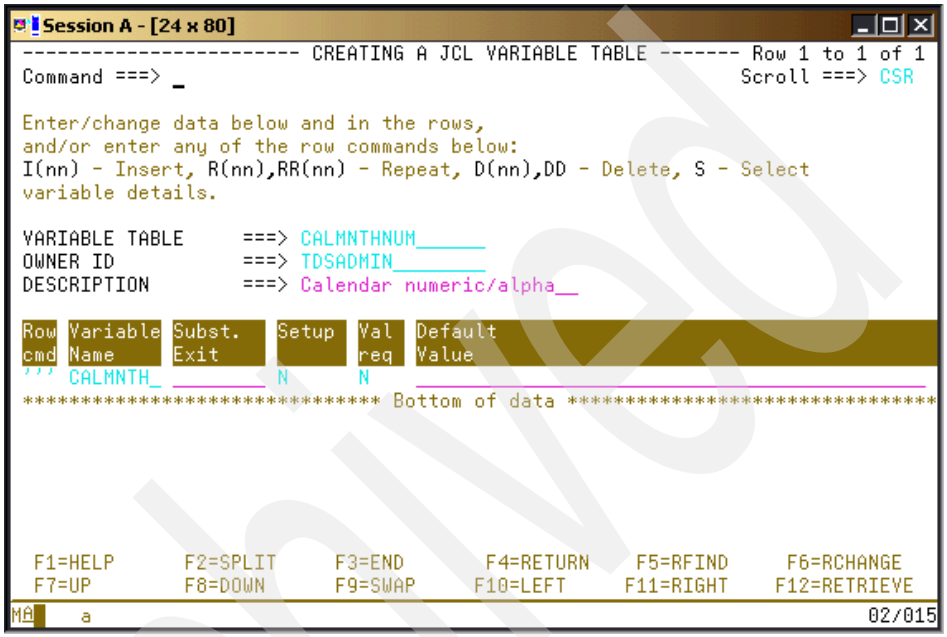

Figure 6-34 Creating a JCL variable table called CALMNTHNUM

Selecting the variable takes us to the MODIFYING A JCL VARIABLE panel, which we code as shown in Figure 6-35 on page 273.

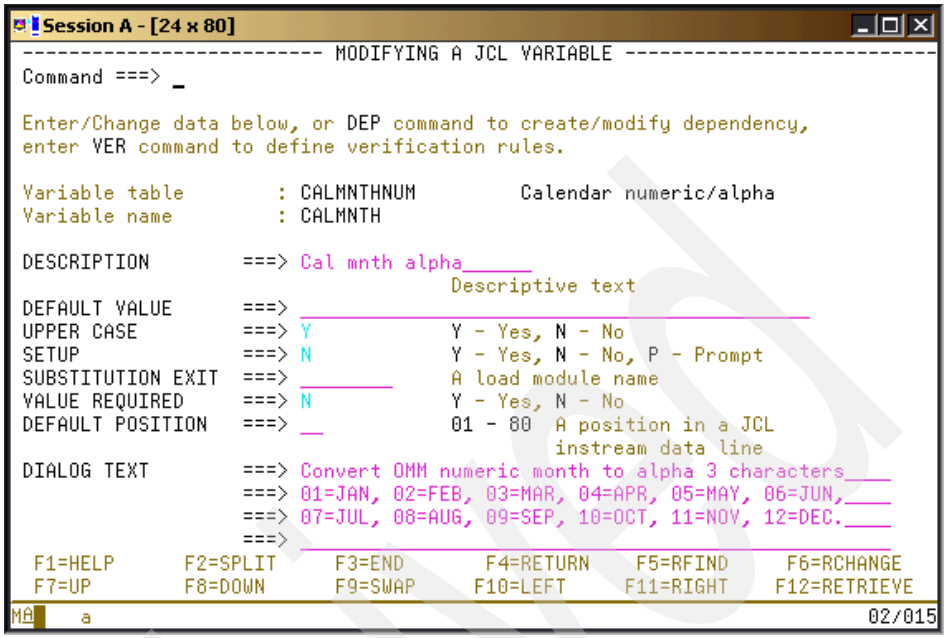

Figure 6-35 Modifying the JCL variable table

Entering the **DEP** command takes us to the JCL VARIABLE DEPENDENCY VALUE LIST panel where we code the independent variable TVARM and values for all the months of the year. Figure 6-36 on page 274 shows this with just the first two months visible. Paging down five times allows you to code the remaining ten values as listed against the DIALOG TEXT in Figure 6-35.

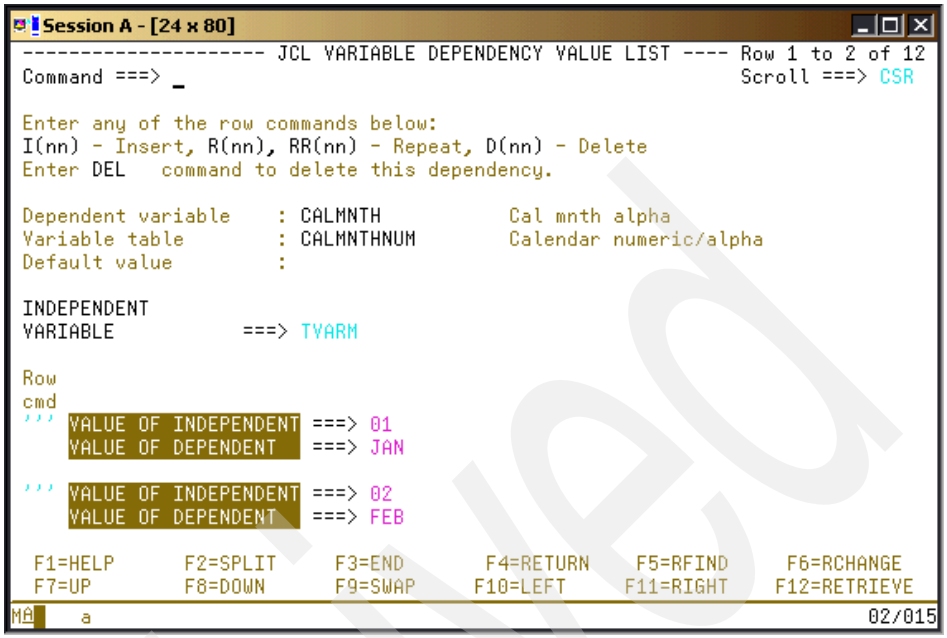

Figure 6-36 Coding the JCL variable dependency values

TWS for z/OS considerations

If you were to use the process in the examples above, you would need to add the T01T01WK weekly job to the end of the TDS#01#TWS#01#WR application with all the jobs in the application as predecessors. This ensures that all reports have been successfully produced before you take a copy of them. Likewise, you would need to add the T01T01MT monthly job to the end of the TDS#01#TWS#01#MR application with all the jobs in the application as predecessors.

If you are reporting on more than one TWS for z/OS controller, it would be advisable to ensure that the report applications for one controller have completed before the next one starts. This prevents the members of the DRL.LOCAL.REPORTS data set being a mix of reports for different TWS for z/OS controllers. This would possibly require dependencies between the applications which could be restrictive.

An option could be to setup separate data sets for the reports for each TWS for z/OS controller being reported on, and pass the data set name as an override to the procedure. Example 6-28 shows a suggestion.

Example 6-28 Example JCL for separate report data sets

```
//JOBPROC  JCLLIB ORDER=(DRL.LOCAL.PROCLIB)
//*
//STEP01  EXEC TDSREP1R
//*
//STEP01.DRLREP   DD DISP=SHR,DSN=DRL.LOCAL.REPORTS.T01
//STEP01.SYSTSIN  DD *
 EX 'DRL.LOCAL.REXX(OPC01T)' +
 'DATE1=&TVAR1 DATE2=&TVAR2 OSI=01'
/*
```

Graphical reports would also require the addition of the override as shown in Example 6-29.

Example 6-29 Override statement for graphical reports

```
//STEP01.ADMGDF   DD DISP=SHR,DSN=DRL.LOCAL.CHARTS.T01
```

6.7 Hosting reports on a Web site

Hosting the reports on a Web site is a good way of making them available to lots of people without them having to access Tivoli Decision Support for OS/390. We are not going to describe how to set a Web site up because it can be done in so many different ways. However, we do show examples of converting the standard report output to Hierarchical File System (HFS) format. Refer to *z/OS MVS JCL Reference*, SA22-7597 and *z/OS UNIX System Services User's Guide*, SA22-7801 for more information.

6.7.1 Converting tabular reports to HFS format

An example of converting a tabular report into HFS format is shown in Example 6-30 on page 276.

Note: The date variables are coded for the job to run on a Monday so that you always get the preceding week, even when the year changes.

Example 6-30 Converting a tabular report

```
//TO1TO1W2 JOB (ACCT#),'FILE TO OMVS',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*********************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*********************************************************************
//*%OPC SCAN
//*%OPC SETVAR TVY=(OYY-4CD)
//*%OPC SETVAR TVW=(OWW-2CD)
//*********************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(TO1TO1W2)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT           ON SC64
//*********************************************************************
//** This job converts normal text to OMVS format so it is visible    **
//** from the WEB.                                                    **
//*********************************************************************
//*
//BATCH   EXEC PGM=IKJEFT01
//INFILE   DD DISP=SHR,DSN=DRL.FTP.MEUB9.OPCO2&TVW..TEXT
//OUTF DD PATH='/etc/perfdata/meub9/meub9_OPCO2_yr&TVY.wk&TVW..txt',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRWXU,SIRWXG,SIROTH,SIXOTH)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 OCOPY INDD(INFILE) OUTDD(OUTF)
/*
//*
//DELETE  EXEC PGM=IEFBR14,COND=(0,NE)
//INFILE   DD DSN=DRL.FTP.MEUB9.OPCO2&TVW..TEXT,
//            DISP=(OLD,DELETE,DELETE)
/*
//
```

6.7.2 Converting graphical reports to HFS format

An example of converting a graphical report into HFS format is shown in Example 6-31 on page 277.

Note: The date variables are coded so that you always get the preceding month, even when the year changes.

Example 6-31 Converting a graphical report

```
//T01T01M7 JOB (ACCT#),'FILE TO OMVS',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*********************************************************************
/*JOBPARM LINES=999999,CARDS=999999,SYSAFF=SC64
//*********************************************************************
//*%OPC SCAN
//*%OPC SETVAR TVY=(OYY-1MO)
//*%OPC SETVAR TVM=(OMM-1MO)
//*********************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(T01T01M7)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT           ON SC64
//*********************************************************************
//** This job converts normal gifbin to OMVS format so it is visible  **
//** from the WEB.                                                     **
//*********************************************************************
//*
//BATCH   EXEC PGM=IKJEFT01
//INFILE   DD DISP=SHR,DSN=DRL.FTP.MEUB9.OPC07&TVM..GIFBIN
//OUTF DD PATH='/etc/perfdata/meub9/meub9_OPC07_yr&TVY.mon&TVM..gif',
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),
// PATHMODE=(SIRWXU,SIRWXG,SIROTH,SIXOTH)
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 OCOPY INDD(INFILE) OUTDD(OUTF) BINARY
/*
//*
//DELETE  EXEC PGM=IEFBR14,COND=(0,NE)
//INFILE   DD DSN=DRL.FTP.MEUB9.OPC07&TVM..GIFBIN,
//            DISP=(OLD,DELETE,DELETE)
/*
//
```

6.7.3 Using NetView FTP to transfer data sets to Web hosting system

If the Web hosting system is not the same system that has Tivoli Decision Support for OS/390 installed or the DASD is not visible from the Web hosting system, you need to transfer the reports. You can transfer reports using an application that contains FTP jobs, with each job dependant on the report job that creates the data to be sent via FTP.

The applications could be called TDS#01#TWS#01#WF and TDS#01#TWS#01#MF, for example, with job names of T01T01Fx where *x* ties in with the last character of the report job names, because these are unique. Alternatively, the FTP jobs could be added to the report applications, each one a successor to the corresponding report job.

If the Web hosting system is not controlled by the same TWS for z/OS controller as the Tivoli Decision Support for OS/390 system, the FTP jobs need to run post-transfer jobs, using the RPTJOBLB and RPTJOBOK parameters. These jobs would then be in an application with SUBMIT=N coded under the Automatic Options of the operation details and would be predecessors to the conversion jobs. This means that when the data arrives, the post-transfer job is tracked by the TWS for z/OS controller allowing the conversion job to run knowing that the data set to be converted exists. These tracking jobs could be in a separate application or in the same one as the conversion jobs.

Suggestion: In another installation, we coded these remote tracking jobs with the same job names as the FTP jobs that send the data from the Tivoli Decision Support for OS/390 system, and the conversion jobs have the same job names as the original report creation jobs. The application names on the two different TWS for z/OS controllers are also the same.

Example 6-32 shows a sample job for a tabular report with these parameters.

Example 6-32 Example FTP JCL for tabular reports

```
//TO1TO1F1 JOB (ACCT#),'FTP REPORT OPCO1T',MSGLEVEL=(1,1),
//         CLASS=A,MSGCLASS=X,TIME=1440,REGION=0M
//*****************************************************************
/*JOBPARM LINES=999999,CARDS=999999
//*****************************************************************
//*%OPC SCAN
//*%OPC SETVAR TVARW=(OWW-1CD)
//*****************************************************************
//* THIS JCL RESIDES IN ===> 'TWS.INST.JOBLIB(TO1TO1F1)' ON SC64
//* JOB OUTPUT GOES TO  ===> SDSF HELD OUTPUT            ON SC64
//*****************************************************************
//*
//STEPO1  EXEC PGM=DVGIFBI
//SYSUDUMP DD SYSOUT=*
//DVGLOG   DD SYSOUT=*
//SYSIN    DD *
 FUNCTION=ADD                          /* ADD REQUEST TO QUEUE
 XMODE=TO                              /* TRANSFER MODE
 RMTNODE=remote node name              /* REMOTE NODEID
 WAIT=YES                              /* JOB WAITS FOR XFER RESULT
 WAITTIME=10                           /* ONLY WAIT 10S - DEFAULT=180S
 SFILEID='DRL.LOCAL.REPORTS(OPCO1)'    /* SENDING FILEID
 SFILEORG=PO                           /* SENDING FILE TYPE
 RFILEID='DRL.FTP.MEUB9.OPCO1&TVARW..TEXT' /* RECEIVING FILEID
 RFILEORG=PS                           /* REMOTE FILE TYPE
 RPTJOBLB='DRL.FTP.JOBLIB'             /* REMOTE JOB LIBRARY
 RPTJOBOK=TO1TO1F1                     /* REMOTE JOB
```

```
 SSECURP=('*','*')                      /* SENDING SECURITY PARAMETERS
 RSECURP=('*','*')                      /* RECEIVING SECURITY PARAMETERS
/*
//
```

If a graphical report was being sent via FTP, you need to use the parameters shown in Example 6-33 instead.

Example 6-33 Different FTP JCL for graphical reports

```
SFILEID='DRL.MEUB9.OPC11&TVARW..GIFBIN      /* SENDING FILEID
SFILEORG=PS                              /* SENDING FILE TYPE
RFILEID='DRL.FTP.MEUB9.OPC01&TVARW..GIFBIN' /* RECEIVING FILEID
RFILEORG=PS                              /* REMOTE FILE TYPE
```

The application to track the ETTs and run the conversion jobs could either be a single application or could easily be two separate applications linked by dependencies.

You can see that the RFILEID data set names tie in with the INFILE names shown in the examples for HFS conversion, in the previous sections.

6.8 Dealing with multiple TWS for z/OS controllers

Collecting multiple TWS for z/OS tracklog data into a single Tivoli Decision Support for OS/390 database is perfectly acceptable and makes sense because it means that you only have one database to manage. For each additional TWS for z/OS controller that you want to collect into your Tivoli Decision Support for OS/390 database, do the following:

1. Identify the PDS member of the data set referenced by the EQQPARM DD statement in the current plan extend or replan batch job.
2. Make arrangements for the LOGID value on the BATCHOPT initialization statement, within that member, to be updated to a unique value from the Tivoli Decision Support for OS/390 perspective. For example, LOGID(02) for the second TWS for z/OS controller being collected, LOGID(03) for the third, and so forth.

   If the LOGID parameter is not coded it is because LOGID(01) is the default value. If this is the case, you should add the parameter and code appropriately.

Important: If the LOGID value is already something other than 01, then do not change it because it could be this value for a good reason.

If this other LOGID value is already in use in Tivoli Decision Support for OS/390, you might have to arrange for the TWS for z/OS corresponding to this other value to be changed to a new value, or decide to collect this TWS for z/OS data into a different Tivoli Decision Support for OS/390 database.

Restriction: The LOGID parameter value limit of 99 is, therefore, a restriction to the number of TWS for z/OS controllers that you could collect into a single Tivoli Decision Support for OS/390 database.

3. Make arrangements for the TWS for z/OS controller to be recycled to pick up the new LOGID value or wait until the next scheduled recycle.
4. Update the OPC_WORKSTATION lookup table with the workstation information for this additional TWS for z/OS controller.

Note: This step can be done in advance of the preceding steps as long as you ensure that you code the correct OPC_SYSTEM_ID value to correspond with the new LOGID value.

5. Create a new Tivoli Decision Support for OS/390 collect job in the relevant TWS for z/OS JCL library. T01T02C1 for the LOGID(02) related controller, for example, in our installation. This is best done by copying an existing job and updating as necessary.
6. Create the new TWS for z/OS application to collect the data into the Tivoli Decision Support for OS/390 database. TDS#01#TWS#02#DC for the LOGID(02) related controller, for example, in our installation. This is best done by copying an existing application and updating as necessary. Save it as pending until it is required.

Attention: Do not start collecting this new tracklog data into your Tivoli Decision Support for OS/390 database until you have a full day's worth of data with the updated LOGID value. This ensures that you get the correct reports for the new controller and prevents erroneous data in existing reports.

7. Create the new TWS for z/OS applications for reporting on the new controller. TDS#01#TWS#02#WR and TDS#01#TWS#02#MR, for example, in our installation. This is best done by copying existing applications and updating as necessary. Save these as pending until they are required.
8. Create any other applications necessary, such as the FTP ones, if required.
9. The tracklog data will need to be sent to the system with Tivoli Decision Support for OS/390 installed. This can be done several ways, but we prefer the use of NetView FTP because of the post-transfer ability. This allows for the submission of a remote job that can be tracked by the TWS for z/OS controller that is running the Tivoli Decision Support for OS/390 workload.

   The post-transfer job could be added to the relevant TDS#01#TWS#nn#DC application as a predecessor operation to the corresponding T01TnnC1 job. This requires SUBMIT=N to be coded under the Automatic Options of the operation details. Alternatively, the job could be within its own application that would then be a predecessor to the collect application.

7

Integrating JOB/SCAN

This chapter discusses the integration of Tivoli Workload Scheduler for z/OS (TWS for z/OS) with JOB/SCAN. JOB/SCAN provides a variety of features to ensure that production z/OS batch processes are as error free as possible, that the system is ready to execute them, and they are consistent that with your company's standards.

This chapter explains how to customize TWS for z/OS and JOB/SCAN installations and how to prepare your JCL for scheduling. It also describes how to validate dependencies between operations, how to assess the impact of changing variables, and how to avoid production problems. Finally, it discusses how to tune JOB/SCAN for special situations.

7.1 Customizing for JOB/SCAN installations

In order to use JOB/SCAN as demonstrated in this book, you should have the following products installed:

- Tivoli Workload Scheduler 8.1 or later
- JOB/SCAN release 6.1.2F or later
- JOB/SCAN for TWS function pack

When you have installed these products, there will be some customization needed to ensure that the JOB/SCAN functions are available within TWS for z/OS. This section described those necessary customizations.

7.1.1 Product library allocation

You must pre-allocate the JOB/SCAN CLIST and ISPF libraries (ISPPLIB, ISPSLIB, ISPLLIB and ISPMLIB) so that they are available for TWS for z/OS users.You can allocate these libraries:

- Within your TSO JCL.
- As part of your TSO initialization CLISTs or REXX programs.
- Within individual product startup programs using ISPF LIBDEF commands.

We recommend that you allocate the JOB/SCAN libraries ahead of the TWS for z/OS libraries so that you can place modified TWS for z/OS panels in the JOB/SCAN library. Should you need to remove JOB/SCAN, TWS for z/OS libraries then only require reallocation.

Attention: The modifications described in this book assume that you have followed this recommendation.

7.1.2 Changing the TWS for z/OS command table

JOB/SCAN for Tivoli Workload Scheduler adds a new command called **JVAL**. This command allows you to validate JCL from anywhere within TWS for z/OS. To make this command available to TWS for z/OS users, copy and update the TWS for z/OS ISPF command table into the JOB/SCAN table library.

The JOB/SCAN REXX program **JOOCCMDT** updates the command table into the JOB/SCAN for TWS ISPF command table library. The syntax for the command is:

```
JOOCCMDT mode [<table-input-library> [<table> [<table-output-library>]]]
```

To perform the command update, issue the following command and substitute your data set names for the input and output data sets listed in this example:

```
TSO JOOCCMDT ADD TWS.V8R2M0.SEQQTBL0 EQQACMDS DSSI.JTWS10.ISPTLIB
```

In this command, `TWS.V8R2M0.SEQQTBL0` is the input data set and `DSSI.JTWS10.ISPTLIB` is the output data set

Ensure that this command is included in the concatenation as with 7.1.1, “Product library allocation” on page 284.

Note: This step is done once for all users.

Tip: If you do not modify the command table, these features are still available. Use the **`TSO JOOCTWSC`** command from the command line instead of **`JVAL`**.

7.1.3 Modifying the Optional functions panel

JOB/SCAN is available at any time in TWS for z/OS via the **`JVAL`** command. To assist users in finding JOB/SCAN, you can make this command available on the Tivoli Workload Scheduler Optional functions menu, which is accessible as option 10 from the TWS for z/OS primary panel.

To modify the Option functions panel:

1. Copy the TWS for z/OS ISPF panel EQQFUADP to the JOB/SCAN ISPF panel library.
2. Make the following changes to the copied panel to update the visual **`)BODY`** area of the panel and to update the process **`)PROC`** area.

Tip: This step is done once for all users.

   a. Add the JOB/SCAN option to the panel definition. Example 7-1 on page 286 shows the extra lines as lines 000048 and 000049.

Example 7-1 Inserting the description for JOB/SCAN

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
 ISREDDE2   EQQ.SEQQPENU(EQQFUADP) - 01.01                Columns 00001 00072
 Command ===>                                               Scroll ===> CSR
 000041 +
 000042 %1<AUDIT/DEBUG  +- Produce formatted report of tracklog events
 000043 +                  (Interactive invocation)
 000044 +
 000045 %2<AUDIT BATCH  +- Produce formatted report of tracklog events
 000046 +                  (BATCH submission)
 000047 +
 000048 %JV<JOB/SCAN    +- Validate JCL from TWS in execution sequence
 000049 +                  (BATCH submission)
 000050 +
```

b. To execute the JV option from the menu, add an extra line to the setting of &zsel. In Example 7-2 the extra line is numbered **000077**.

Example 7-2 Adding the JV option

```
File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
 ISREDDE2   EQQ.SEQQPENU(EQQFUADP) - 01.01                Columns 00001 00072
 Command ===>                                               Scroll ===> CSR
 000072   If (&zcmd ¬= CANCEL,RESET)
 000073    If (.msg = ' ')
 000074      &zsel = Trans(&selno
 000075           1,'CMD(%EQQAUDNS)'
 000076           2,'MOD(EQQXDSPX,EQQXSUBP)'
 000077          JV,'CMD(%J00CTS20 &xnxtsel)'
 000078          ' ',' '
 000079          '*','?')
```

7.1.4 Setting the JCL edit tool

TWS for z/OS includes a plug-in point for a JCL edit tool. This plug-in point is displayed when the edit/browse JCL function is invoked by the AD dialogs. Each user can specify their own tool. JOB/SCAN includes a panel for this purpose.

To set the JCL edit tools:

1. From the TWS for z/OS main menu, go to option 0.6.
2. Change the panel name to J00PTS00 as shown in Example 7-3 on page 287.

Note: Each TWS for z/OS user must set the JCL Edit Tool Information.

Example 7-3 Setting the JCL edit tool

```
EQQXJCLP ----------- SETTING JCL EDIT TOOL INFORMATION --------------------
 Command ===>

 Enter/change data below:

 PANEL NAME      ===> JOOPTSOO  Specify the name of the panel to
                                be displayed when editing JCL from
                                AD data base.

                                This panel name is used to provide
                                a way to call a user tool to edit JCL.

                                Default value is EQQAJCLE which is a dummy
                                panel.
```

It is possible that you already have another JCL Tool installed in option 0.6 for JCL source management. This will be obvious if this panel does not refer to EQQAJCLE. Fortunately, JOB/SCAN includes a feature for just such an occasion.

Note: Each TWS user must perform this procedure to retain access to the previous JCL edit tool.

You can access the pre-existing tool from JOB/SCAN using the following procedure:

1. Invoke the JOB/SCAN JCL edit tool.
   a. Browse the application list using the JOB/SCAN for Tivoli Workload Scheduler option 1.4.1.
   b. Select an application.
   c. Enter the **OPER** command to list the operations.
   d. Against any of the JCL operations, enter the **J** command. This command displays the JOB/SCAN JCL edit tool panel, as shown in Figure 7-1 on page 288.

```
JOOPTSOO------------ JOB/SCAN FOR TWS - VIEW JCL -----------------------
Option ===>


  Subsystem Name: TWSC
  Application   : GLDLYPOST
  Workstation   : CPU1
  Operation No  : 20
  Job Name      : PJGLOO20


  0 OPTIONS     - Specify Alternate JCL Edit TOOL

  1 VIEW        - VIEW JCL from EQQJBLIB

  X EXIT


  NOTE          : From VIEW use command JTWS to validate PJGLOO20
                  with resolved variables using JOB/SCAN from
                  Diversified Software, Inc.
```

Figure 7-1 JOB/SCAN for TWS JCL Edit Tool

2. Add the pre-existing JCL edit tool using option 0.
   a. Enter option **0** and set the details of the pre-existing JCL edit tool.
   b. Enter the **PANEL NAME** of your alternate JCL tool.
   c. Enter the **OPTION NAME** and **OPTION TEXT** fields, as shown in Figure 7-2 on page 289.

```
JOOPTS08 ------- JOB/SCAN FOR TWS - ALTERNATE JCL EDITING TOOL -------------
Command ===>

Enter/Change data below: Enter END to save or CANCEL to exit without saving

PANEL NAME      ===> MYPANEL      Specify the name of the panel to
                                  be displayed when editing JCL from
                                  within the  TWS plug-in.

OPTION NAME     ===> MY EDIT      Specify the option name to appear
                                  on the  plug-in menu.

OPTION TEXT     ===> My JCL Editor
                                 Specify the description to appear
                                   on the  plug-in menu.
```

Figure 7-2 Specifying an alternate JCL editing tool

These extra fields now appear on the JOB/SCAN panel, as shown in Figure 7-3.

```
JOOPTS00 --------------- JOB/SCAN FOR TWS - VIEW JCL ------------------
Option ===>


  Subsystem Name: TWSC
  Application   : GLDLYPOST
  Workstation   : CPU1
  Operation No  : 20
  Job Name      : PJGL0020


  0 OPTIONS     - Specify Alternate JCL Edit TOOL

  1 VIEW        - VIEW JCL from EQQJBLIB
  2 MY EDIT     - My JCL Editor
  X EXIT


  NOTE          : From VIEW use command JTWS to validate PJGL0020
                  with resolved variables using JOB/SCAN from
                  Diversified Software, Inc.
```

Figure 7-3 JOB/SCAN edit tool with alternate edit tool installed

Setting the JCL edit tool automatically

The way the TWS for z/OS dialogs are designed, each user must enter the panel names themselves because the information is stored in individual user ISPF profiles. However, there is a technique that you can use to select the edit tools automatically for everyone.

To select the edit tool automatically, include code that is similar to the code shown in Example 7-4 into the **)INIT** section of the TWS for z/OS EQQOPCAP primary panel.

Example 7-4 Code to automatically use the JOB/SCAN JCL edit tool

```
&XJCLPAN = 'JOOPTSOO'       /* JOB/SCAN JCL edit tool panel */
&XMANPAN = 'MYPANEL'        /* Name of additional edit tool */
&XMANOPT = 'MY EDIT'        /* Option name for edit tool    */
&XMANTXT = 'My JCL Editor'  /* Description for edit tool    */
VPUT (XJCLPAN XMANPAN XMANOPT XMANTXT) PROFILE
```

The first line sets the JOB/SCAN edit tool as the primary option for TWS for z/OS. The next three lines are equivalent to filling in the fields on the panel shown in Figure 7-2 on page 289. If you do not need to specify an alternate tool, set the XMANPAN variable to null (for example, `&XMANPAN = ‘’`).

7.1.5 Configuring JOB/SCAN to match your environment

JOB/SCAN needs to know the data sets that contain your production JCL. This information is configured in one place for all users. This section describes the process to configure JOB/SCAN.

Note: The *JOB/SCAN TWS for z/OS Scheduler Interface Handbook* contains the details for this process.

Determine the data sets that are allocated to EQQJBLIB

Look at the currently running TWS for z/OS controller started task. Note that the data sets that are allocated to DDNAME EQQJBLIB. In Figure 7-4 on page 291, you can see that one data set is allocated for statement 33.

```
 Display  Filter  View  Print  Options  Help
-------------------------------------------------------------------------------
SDSF OUTPUT DISPLAY TWSC     STC01343  DSID     3 LINE 24      COLUMNS 02- 81
COMMAND INPUT ===>                                           SCROLL ===> CSR

      31 XXEQQJT04   DD  DISP=SHR,DSN=TWS.V8R2M0.JT4
      32 XXEQQJT05   DD  DISP=SHR,DSN=TWS.V8R2M0.JT5
      33 XXEQQJBLIB  DD  DISP=SHR,DSN=TWS.V8R2M0.JOBLIB
      34 XXEQQPRLIB  DD  DISP=SHR,DSN=TWS.V8R2M0.JOBLIB
      35 XXEQQJCLIB  DD  DISP=SHR,DSN=TWS.V8R2M0.JCLIB
      36 XXEQQINCWK  DD  DISP=SHR,DSN=TWS.V8R2M0.INCWORK
      37 XXEQQSTC    DD  DISP=SHR,DSN=TWS.V8R2M0.STC
```

Figure 7-4 SDSF display of TWS task showing EQQJBLIB concatenation

Update J00CTWS

Edit the JOB/SCAN REXX executable J00CTWS to define your TWS for z/OS configuration to JOB/SCAN. In addition to defining other installation parameters, this REXX executable defines the TWS for z/OS subsystem names, a default subsystem name, the EQQJBLIB libraries for each, default application attributes, message library DSN, subsystem attributes such as LU Name, the TWS for z/OS product data set name prefix (for example, TWS.V8R2M0), and the TWS for z/OS database data set name prefix (for example, TWS.V*R2M0). To integrate JOB/SCAN and TWS for z/OS, edit J00CTWS from the JOB/SCAN CLIST library.

Note: The default subsystem name is only to provide a default name for JOB/SCAN users who are working outside of TWS for z/OS. The panels allow each user to override this default name.

For example, if you had two TWS for z/OS subsystems and you would like TWSC to be the default name, the J00CTWS would look like Example 7-5.

Example 7-5 Setting the subsystem names in J00CTWS

```
Command ===>                                                  Scroll ===> CSR
000008 /*********************************************************************
000009 * Set SUBSYS variables                                               *
000010 *********************************************************************/
000011 SUBSYS_DEFAULT = "TWSC"
000012 SUBSYS_LIST    = "TWSC TWSD"
000013
```

To set the job JCL data set names to represent what was shown in Figure 7-4 on page 291, J00CTWS would look like Example 7-6. In this example, TWSC has a single library concatenation, and TWSD has a concatenation of two libraries.

Example 7-6 Updated member J00CTWS

```
Command ===>                                                  Scroll ===> CSR
000028 DSN_LIST:
000029 SELECT
000030   WHEN SUBSYS="TWSC" THEN
000031     RETURN "TWS.V8R2M0.JOBLIB"
000032
000033   WHEN SUBSYS="TWSD" THEN
000034     RETURN "TWS.V8R2M0.FIRST.JOBLIB",
000035            "TWS.V8R2M0.SECOND.JOBLIB"
000036
000037   OTHERWISE
000038     RETURN "NONE"
000039 END
******************************** Bottom of Data ***************************
```

To assist in determining the proper data set name list, J00CTWS is also passed the application name (ADID), job name (JOBNAME), workstation name (WSNAME), and operation number (OPNO). You can use this information, in addition to SUBSYS, when determining which data sets names to return. This is available to account for any decisions made by the Job-Library-Read exit.

Set your default application attributes as depicted in Example 7-7. These values can be of your choosing and are used to associate a job with an application ID. The values are only used when you do not specify an application ID on the JTWS Edit macro panel and the job is not contained within a currently defined application.

Example 7-7 Setting the default application attribute values

```
000039   DEFAULT_JOB          = "DSSI01" /* <-- CHANGE TO APPROPRIATE VALUE*/
000040   DEFAULT_APPLICATION  = "APPLTST" /* <-- CHANGE TO APPROPRIATE VALUE*/
000041   DEFAULT_WORK_STATION = "IBM1" /* <-- CHANGE TO APPROPRIATE VALUE*/
000042   DEFAULT_OPERATION    = "001" /* <-- CHANGE TO APPROPRIATE VALUE*/
```

Example 7-8 on page 293 shows the default application attribute values. Note that all positions must contain values.

Example 7-8 Setting the default application attribute values

```
000090 SUBSYS_DETAIL:
000091 SELECT
/*  TWS subsystem detail          */
000092   WHEN SUBSYS="TWS1" THEN /*  <=== change subsystem name     */
000093     RETURN "IS1MEOPV TWS.V8R2MO TWS.V8R2MO" /* <=== CHANGE  */
000094
/*                                */
000095   WHEN SUBSYS="TWS2" THEN /*  <=== change subsystem name     */
000096     RETURN "NOT-USED TWSPRODUCTDSN TWSDATABASEDSN" /* <=== CHANGE  */
000097
/*  <=== add additional WHEN stmts.*/
000098   OTHERWISE
000099     RETURN "NONE"
000100 END
```

Important: Remember to use a comma at the end of each continued line that contains a data set name.

7.2 Preparing the JCL for scheduling

Getting the JCL ready for scheduling is a difficult process. Introducing the appropriate job tailoring variables, for the appropriate use, is a skill. It requires detailed knowledge of the scheduling and operational environment. Having a tool-set that resolves variables and validates the results is also essential.

This section discusses how to use JOB/SCAN with TWS for z/OS to help bring primitive JCL to full operational maturity.

7.2.1 Preparing and validating JCL within the ISPF editor

When it comes to the point of involving TWS for z/OS, first choose which job library in which to place the JCL. Next, create the TWS for z/OS elements that will eventually schedule the job. The more elements that you have in place that reflect the eventual production environment, the more accurate of an evaluation JOB/SCAN can do for you.

Whether you are working in a standard TWS for z/OS PDS or in a work PDS, you can use JOB/SCAN to resolve job tailoring variables and validate the JCL. To check a single job in conjunction with TWS for z/OS, use the JTWS edit macro from EDIT or VIEW. Enter the **JTWS** command from the VIEW or EDIT command line.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
 ISREDDE2   TWS.INST.JOBLIB(PJGLO020) - 05.10               Columns 00001
00072
 Command ===> JTWS                                           Scroll ===> CSR
 ****************************** Top of Data ******************************
 000001 //PJGLO020 JOB (A,P,GL),'JONES',CLASS=A,MSGCLASS=X,  GL MASTER
UPDATE
 000002 //             MSGLEVEL=(1,1)
 000003 //      JCLLIB ORDER=TWS.INST.JOBLIB
 000004 //*%OPC SCAN
 000005 //*%OPC SETFORM OCDATE=(MM/DD/YY)
 000006 //*%OPC SETVAR TACC=(OCDATE - 3WK)
 000007 //*%OPC SETFORM OCDATE=(DD/MM/CCYY)
 000008 //*%OPC SETVAR TAUD=(OCDATE - 27WD)
 000009 //*%OPC TABLE NAME=&OADID
 000010 //*****************************************************************
 000011 //JS010    EXEC  PPGLO020,TYPE=WEEKLY
 000012 //SYSUT1  DD  DSN=FC.GLPROD.INPUT.FILE,DISP=OLD
 000013 //PS030.DATECARD DD *
 000014 070589    &OYMD1       1 1 3 NNN USRCODE=&USRCODE
 000015 //PS050.DATECARD DD *                                  DATE CARD
OVERRID
 000016 070589                          TYPE 5 CONTROL
 000017 ACCDATE=&TACC
 000018 AUDITDATE=&TAUD
 000019 //
```

Figure 7-5 Invoking JOB/SCAN from ISPF Edit

Ordinarily, JCL needs to be in a TWS for z/OS library for resolution to occur. The **JTWS** macro has a technique that allows it to validate JCL from an external source. It takes the JCL contained in the ISPF workspace and temporarily inserts this into the JS file, retrieves the resolved JCL from TWS for z/OS, and the resets the JS file to its former state. When you are using a simulated input arrival this has no impact, however if you are using a real input arrival from the current plan then, you need to be aware this is happening. **JTWS** warns you it is about to do this and gives you the opportunity to abort before it does so. You can stop this warning from being displayed in the future by entering X in the *Enter "X" to suppress this panel in the future* field. as shown in Figure 7-6 on page 295.

```
JOOPTS1A ----------------- JOB/SCAN FOR TWS ADVISORY -----------------------
 COMMAND ===>

        Y - to continue
        N - to return

          JTWS updates the occurrence JCL in the JS FILE.  After
          TWS resolves the variables, JTWS restores this Job
          to its original contents.

   _ Enter "X" to suppress this panel in the future.
```

Figure 7-6 JOB/SCAN warning that the JS file is about to be updated

The main **JTWS** panel allows you to specify the application against which you want to test the JCL. If the application that you specify exists, then the job must exist within it on the specified workstation and operation number. Otherwise, TWS for z/OS is not able to provide resolved JCL.

Instead of manually entering this information into this panel, you can let the **JTWS** macro find the job within occurrences on the current plan or applications in the database. Set the two search fields accordingly, and **JTWS** locates the job for you.

```
JOOPTS01 ---------------------- JOB/SCAN FOR TWS ---------------------------
COMMAND ===>

   Review values and press ENTER to continue or END to return.

   CURRENT PLAN ENDS  ===> 05/02/18 08.00        TWS Subsystem : TWSC
   JOB NAME           ===> PJGL0020              LU NAME  : ________________
   APPLICATION        ===> *                     Blank and wild cards OK
   INPUT ARRIVAL      ===> __/__/__  __.__       Blank or YY/MM/DD HH.MM
   OPERATION (WS NO.) ===> ____ ___              Blank or WWWW NNN
   VARIABLE TABLE     ===>
   SEARCH TWS CP      ===> YES                   Yes or No
   SEARCH TWS AD'S    ===> YES                   Yes or No
   PROVIDE NF ENTRY   ===> NO                    Yes or No (NF=Not Found)
```

Figure 7-7 JTWS input panel when executed outside of TWS

The following lists describes some of the possible search fields in Figure 7-7 on page 295:

VARIABLE TABLE — Normally, when TWS for z/OS runs an application, it attaches a variable table to the occurrence based on run cycles, periods, or being directly named as it adds the occurrence to the current plan. The VARIABLE TABLE field on this panel allows you to simulate this process when validating against the database. You can also use it to specify alternate tables or to determine the impact of changing variables. For more information, see 7.4, “Assessing the impact of changing variables” on page 307.

SEARCH TWS — If you decided to let JTWS find your application for you, then it will search the Current Plan or the database that you selected.

PROVIDE NF ENTRY — If the JOB NAME cannot be found, then you are returned to your JCL editor if you entered NO for PROVIDE NF ENTRY.

However, if you entered YES, then a temporary application is created with the name that you provided on the panel in Figure 7-13 on page 306 (assuming it does not conflict with an existing application) to allow the JCL to be resolved. If you left the Application field blank, then the default Application name specified during the installation is used if there are no naming conflicts. Either temporary application is removed automatically when the checking process is over.

If you have requested that JOB/SCAN search for you and it does find entries in either the Current Plan or the Database, then these entries are presented to you as shown in Figure 7-8 on page 297. You still have the option at this point to change the variable table if you want. Enter S next to the application against which you want to validate your JCL.

```
JOOPTS10 -------------------- JOB/SCAN FOR TWS ------------- Row 1 to 4 of 4
COMMAND ===>                                                 SCROLL ===> CSR

   Use S to select. Press ENTER to process or END to return.

   CURRENT PLAN ENDS  ===> 05/02/18 08.00        TWS SUBSYSTEM : TWSC
   JOB NAME           ===> PJGLO020              LU NAME  : ________________
   APPLICATION        ===> *
   VARIABLE TABLE     ===>                       Blank for default

Sel Application id    Operation Jobname  Input Arrival  Variable     Status
                      ws no.             Date  Time     Table          Srce

 _  GLDLYPOST         CPU1 020 PJGLO020 05/02/15 17.00                E    CP
 _  GLDLYPOST         CPU1 020 PJGLO020 05/02/16 17.00                E    CP
 _  GLDLYPOST         CPU1 020 PJGLO020 05/02/17 17.00                E    CP
 _  GLDLYPOST         CPU1 020 PJGLO020 05/02/17 17.22                A    AD
****************************** Bottom of data ******************************
```

Figure 7-8 Selecting an application or occurrence from JTWS

After you have selected which application or occurrence to use, **JTWS** calls the TWS for z/OS API to resolve the variables in the JCL. The API returns the JCL fully resolved with all variable values substituted and //*%OPC directives taken into consideration.

However, if the JCL contains variables that require prompts, **JTWS** will first ask you to provide values to complete the resolution as shown on Figure 7-9 on page 298. Simply over type the values on the panel that is provided, and select **END** to continue.

```
JOOPTS06 -------------------- JOB/SCAN FOR TWS ------------- Row 1 to 1 of 1
COMMAND ===>

      Enter values for promptable variables.

      END to leave this panel and continue processing.

 NAME    VALUE                                                TYPE

USRCODE   NONE                                                 &
LU        VUJAX                                                &
UNIT1     STH                                                  &
UNIT2     WPH                                                  &
****************************** Bottom of data ***************************
```

Figure 7-9 Entering values for variables that require prompts

When the JCL is fully resolved, it is passed to the JOB/SCAN validation engine. You see the results in ISPF VIEW. The output is the Structured JCL Listing from JOB/SCAN. It contains the executing JCL as it would be submitted by TWS for z/OS with all variables resolved, a merged JCL listing that contains procedure JCL and control cards, and all error, warning, and information messages from the scan. These messages are included both in place within the JCL and as a summary at the end. You can see a portion of a Structured JCL Listing in Example 7-9.

Example 7-9 Abbreviated example of a Structured JCL listing

```
JOB/SCAN          STRUCTURED JCL LISTING    6.2.4A     2005-02-17  17:51:34
------------------------------------------------------------------------------
//PJGLOO2O JOB (A,P,GL),'JONES',CLASS=A,MSGCLASS=X,
//           MSGLEVEL=(1,1)
//*%OPC SCAN
//*>OPC SETFORM OCDATE=(MM/DD/YY)
//*>OPC SETVAR TACC=(OCDATE - 3WK)
//*>OPC SETFORM OCDATE=(DD/MM/CCYY)
//*>OPC SETVAR TAUD=(OCDATE - 27WD)
//*>OPC TABLE NAME=GLDLYPOST
//*************************************************************
//JSO1O    EXEC  PPGLOO2O,TYPE=WEEKLY
$$SYSUT1  DD  DSN=FC.GLPROD.INPUT.FILE,DISP=OLD
$$PSO3O.DATECARD DD *
070589    050217       1 1 3 NNN USRCODE=GLEDGER
$$PSO5O.DATECARD DD *                                  DATE CARD OVERRID
070589                              TYPE 5 CONTROL
ACCDATE=01/27/05
AUDITDATE=17/01/2005
```

```
  P1//PPGL0020 PROC INDX=FC,TYPE=D,
  P1//             APPLIC=GLPROD,
  P1//             SPTRN=20,SPROLL=500,DVOL=VOLWEEK2,
  P1//             GEN3='+1',GEN4='+1',GEN5='+1'
  P1//PS010    EXEC  PGM=IEBGENER                  COPY THE KEYTAPE TO DISK
  P1//SYSPRINT DD  SYSOUT=A
  P1//SYSIN    DD  DUMMY
  P1//SYSUT1   DD  DUMMY                          (PJGLTR01) FROM DATA PREP
  P1$/SYSUT1  DD  DSN=FC.GLPROD.INPUT.FILE,DISP=OLD
**WARNING - DSS9101W -  TWS WILL WAIT FOR SPECIAL RESOURCE.
*ADVISORY - DSS9102A -  TWS RESNAME = "FC.GLPROD.INPUT.FILE"
  P1//SYSUT2   DD  DSN=&&TEMP,                     WORKING INPUT DATA FILE
  P1//             DISP=(,PASS),
<<lines omitted from this example>>
//
*** ERROR SUMMARY AND COUNTS ***
    3  ADVISORY LEVEL
    1  WARNING LEVEL
    0  ERROR LEVEL
    4  TOTAL ISSUED       4 SUPPRESSED.
     THE HIGHEST SEVERITY CODE ENCOUNTERED WAS    04.
     THE  STDS  PGM  CALLED FOR THIS RUN WAS "TWSSTD1".
REL LINESV MSG.NO.    ERROR MESSAGE
--------- --------   ------------------------------------------------------
    29   4 DSS9101W -  TWS WILL WAIT FOR SPECIAL RESOURCE.                P
    30   0 DSS9102A -  TWS RESNAME = "FC.GLPROD.INPUT.FILE"               P
   127   0 DSS3230A - EXEC COND CODE MATCH FOR: JS010 PS050               P
   128   0 DSS3231A - MATCHING COND CODE: 8 GT PS040                      P
XPJ60K  - END OF REPORT  JOB/SCAN   6.2.4A (C) DIVERSIFIED SOFTWARE SYSTEMS
```

Pressing **PF3** leaves the display of the Structured JCL Listing and returns you to the Operation List. Pressing **PF3** again returns you to ISPF EDIT. You can then correct any problems and, if necessary, start the process again.

Using a simple and iterative process, you can see and validate your TWS for z/OS JCL without having to submit any applications. This is a complex and lengthy process without JOB/SCAN. For more information, see “JOB/SCAN sample output” on page 683.

7.2.2 Viewing and editing JCL from within an application definition

Traditionally, creating your application and viewing or modifying the JCL for the jobs within it has been two separate tasks: one performed from within TWS for z/OS and the other using ISPF services, usually on another screen. The JOB/SCAN for TWS JCL Edit Tool brings the two processes together.

When either modifying or browsing an Application in the database, enter the **J** line command next to an operation, as shown in Example 7-10.

Example 7-10 Browsing JCL from the operation list within an application

```
EQQABOPL -------------------- BROWSING OPERATIONS ---------- Row 1 to 10 of 10
 Command ===>                                                  Scroll ===> CSR

 Enter the PRED command above to include predecessors in this list, or,
 enter the GRAPH command above to view operations graphically.
 Enter the row command S to select the details of an operation.
 Enter the row command J to browse the JCL

 Application            : GLDLYPOST        General Ledger DLY Post

 Row  Oper      Duration  Job name  Operation text
 cmd  ws   no.  HH.MM.SS
 '''' DUMY 001  00.00.01  ZFIRST    Start of Application
 '''' DUMY 005  00.00.01  CICSAA    Stop CICSAA
 J''' CPU1 020  01.30.34  PJGL0020  GL Master Update
 '''' CPU1 030  00.02.00  PJGL0030  Combine RMT Inputs
 '''' CPU1 040  00.05.00  PJGL0040  Combine RMT Backups
 '''' CPU1 050  00.03.15  PJGL0050  Combine RMT Backups
 '''' CPU1 055  00.07.02  PJGL0055  Load latest version
 '''' CPU1 056  00.02.00  PJGL0056  Delete RMT Input
 '''' DUMY 060  00.00.01  CICSAA    Restart CICSAA
 '''' DUMM 255  00.00.01  ZLAST     End of Application
 ****************************** Bottom of data ********************************
```

This change cause a the JOB/SCAN for TWS View or Edit JCL panel to display as shown in Figure 7-10 on page 301.

```
JOOPTSOO --------------- JOB/SCAN FOR TWS - EDIT JCL -----------------------
 Option ===>

   Subsystem Name: TWSC
   Application   : GLDLYPOST
   Workstation   : CPU1
   Operation No  : 20
   Job Name      : PJGL0020

   0 OPTIONS     - Specify Alternate JCL Edit TOOL

   1 EDIT        - EDIT JCL from EQQJBLIB

   X EXIT

   NOTE          : From EDIT use command JTWS to validate PJGL0020
                   with resolved variables using JOB/SCAN from
                   Diversified Software, Inc.
```

Figure 7-10 The JOB/SCAN for TWS JCL edit tool

Option 1 locates the JCL for you from within the relevant libraries and then presents the JCL. If you are browsing the Application, then JOB/SCAN invokes VIEW. If you are modifying the Application, then it invokes EDIT.

From there, you can EDIT the JCL directly. To validate the JCL, use the **JTWS** macro. You are prompted by JOB/SCAN to do this by an advisory message, as shown in Figure 7-11 on page 302.

Note: Because the unresolved JCL does not require any information from TWS for z/OS, the API is not called to obtain the JCL in this instance.

```
 File  Edit  Edit_Settings  Menu  Utilities  Compilers  Test  Help
-------------------------------------------------------------------------------
 ISREDDE2   TWS.INST.JOBLIB(PJGLO020) - 05.10            Columns 00001 00072
 Command ===>                                                Scroll ===> CSR
 ****************************** Top of Data ******************************
 ==MSG> -Warning- The UNDO command is not available until you change
 ==MSG>           your edit profile using the command RECOVERY ON.
 000001 //PJGLO020 JOB (A,P,GL),'ROBIN',CLASS=A,MSGCLASS=X,  GL MASTER
UPDATE
 000002 //             MSGLEVEL=(1,1)
 000003 //      JCLLIB ORDER=TWS.INST.JOBLIB
 000004 //*%OPC SCAN
 000005 //*%OPC SETFORM OCDATE=(MM/DD/YY)
 000006 //*%OPC SETVAR TACC=(OCDATE - 3WK)
 000007 //*%OPC SETFORM OCDATE=(DD/MM/CCYY)
 000008 //*%OPC SETVAR TAUD=(OCDATE - 27WD)
 000009 //*%OPC TABLE NAME=&OADID
 000010 //*************************************************************
 000011 //JS010    EXEC  PPGLO020,TYPE=WEEKLY
 000012 //SYSUT1  DD  DSN=FC.GLPROD.INPUT.FILE,DISP=OLD
 000013 //PS030.DATECARD DD *
 000014 +-----------------------------------------------------------+
 000015 | Issue JTWS command to resolve variables and Validate Job.| OVERRID
 000016 +-----------------------------------------------------------+
 000017 ACCDATE=&TACC
```

Figure 7-11 Calling JTWS from within the JCL edit tool

The **JTWS** command works fundamentally the same as before as discussed in 7.2.1, “Preparing and validating JCL within the ISPF editor” on page 293, with a few minor differences. Because the JCL is edited from within TWS for z/OS, it does not need to ask for the application name. This is preset for you, along with the workstation and operation number.

```
JOOPTSO4 ---------------------- JOB/SCAN FOR TWS --------------------------
 COMMAND ===>

    Review values and press ENTER to continue or END to return.

    CURRENT PLAN ENDS  ===> 05/02/18 08.00

    JOB NAME           ===> PJGL0020
    APPLICATION        ===> GLDLYPOST
    INPUT ARRIVAL      ===> 05/02/17  19.36
    OPERATION (WS NO.) ===> CPU1 020             Blank or WWWW NNN
    VARIABLE TABLE     ===>
```

Figure 7-12 The JTWS input panel when executed within the Application Database

From this point, the process for validating the JCL is the same as the previous scenario. You can still specify an alternate variable table. Equally, you might be prompted for variable values before the JCL is resolved and validated. **JTWS** presents the Structured JCL Listing as before.

7.2.3 Automating the conversion from test to production JCL

As your JCL moves through various phases of the development life cycle, inevitable changes need to be made. Typically the jobname, data set names, job classes, and so forth are different between phases. The changes that you require depend entirely on site standards and local requirements.

JOB/SCAN has an intelligent JCL change mechanism that allows you to automate many tedious tasks to get the test JCL ready for production using a set of change rules that are JCL-syntax aware. Not only does this mechanism speed the process, but it removes the errors that are created by human intervention.

As an example, you might have the following rules for production JCL:

- Run all production jobs in class P.
- Set the LEVEL symbolic to PROD.
- Have a comment after the JOB statement that includes the application name.

Example 7-11 shows the JOB/SCAN change rules that implement these standards.

Example 7-11 JOB/SCAN change rules

```
ON JOB  CHG CLASS FROM  *    TO P    END
ON EXEC CHG LEVEL FROM  TEST TO PROD END
ON JOB  ADD STMTS AFTER //
//*************************************************************
//*%OPC SCAN
//* APPLICATION IS &OADID
ENDADD
END
```

You can apply these rules against test JCL as shown in Example 7-12.

Example 7-12 Job PJGL0010 as it exists in test mode

```
//PJGL0010 JOB (A,P,GL),'ROBIN JONES',CLASS=A,MSGCLASS=X,  GL REPORT
//             MSGLEVEL=(1,1)
//*************************************************************
//JS010    EXEC  PPGL0010,
//         TYPE=WEEKLY,
//         LEVEL=TEST
```

These changes result in a changed JCL as shown in Example 7-13.

Example 7-13 Job PJGL0010 as updated by JOB/SCAN

```
//PJGL0010 JOB (A,P,GL),'ROBIN JONES',CLASS=P,MSGCLASS=X,  GL REPORT
//             MSGLEVEL=(1,1)
//*************************************************************
//*%OPC SCAN
//* APPLICATION IS &OADID
//*************************************************************
//JS010    EXEC  PPGL0010,
//         TYPE=WEEKLY,
//         LEVEL=PROD
```

You can incorporate this change mechanism into whatever process you have for JCL source code management. JOB/SCAN is frequently integrated with such products such as the Software Configuration Library Manager available from IBM. The JOB/SCAN Change Facility is available under ISPF, but for this purpose, it is generally added to a batch promotion process.

7.3 Validating the dependencies between operations

The scenarios so far have dealt with single jobs and their relationship with TWS for z/OS elements. However, that is only the first stage of ensuring that JCL is ready for production. The main purpose of TWS for z/OS is to run whole suites of jobs in predefined sequences. JOB/SCAN scans jobs in sequence and takes into account the impact that one job might have on the next, such as cataloging or deleting data sets.

JOB/SCAN automatically interrogates TWS for z/OS to determine the run sequence of jobs, which in turn is used to validate the JCL in dependency sequence. JOB/SCAN can do this in many ways. However, this scenario looks only at validating a single application.

Start with the JOB/SCAN for TWS JCL Validation menu, as shown in Figure 7-13 on page 306. You access this panel using the **JVAL** command or from TWS for z/OS main menu option 10.JV.

From this panel, select option **4** to perform JCL validation for an application. If you enter **JVAL** while browsing or modifying an application, it selects option **4** automatically and populates the Application ID.

Tip: If you want to access the full JCL validation menu from within an application, you can do so by entering the **JVAL MENU** command.

Tip: This process uses the currently saved version of the application. So, if you are modifying an application and have made changes, remember to exit the application to save it first.

Important: You cannot validate multiple applications or an application group from the database. To do so would need an understanding of the input arrival times that are being used for each application. Equally, some of the selected applications might have relationships to each other via applications that were not selected. This is not easily discernible from the database without replicating the planning functions. Options 1 to 3 on this menu allow you to scan multiple applications, taking into account input arrival and implicit relationships. Refer to 7.5.3, “Validating JCL beyond the current plan” on page 315 for more information about validating multiple applications.

```
JOOPMENU ------------- JOB/SCAN FOR TWS - JCL VALIDATION -------------------
 Option ===>

 Welcome to JOB/SCAN for TWS. You are communicating with TWSC

 To validate JCL in execution sequence with TWS variable resolution
 select one of the following options and press ENTER.

 1  CURRENT PLAN      - JCL Validation from CP
 2  LONG TERM PLAN    - JCL Validation from LTP
 3  EXTENDED PLAN     - JCL Validation from Extended LTP
 4  APPLICATION       - JCL Validation for an application
 X  EXIT
```

Figure 7-13 The JVAL command menu

The Validate Application panel, shown in Figure 7-14, allows you to enter an Application ID and Simulated Input Arrival Time. TWS for z/OS uses the Input Arrival date and time to tailor variables. It also uses the Input Arrival date to select the relevant version of the Application, should multiple versions exist. If you entered the **JVAL** command from within an application, the version is preselected.

```
JOOPTS4A --------- JOB/SCAN FOR TWS - VALIDATE APPLICATION -----------------
 Command ===>

         S to submit the validation JOB   E to edit the validation JOB

 Specify selection criteria below:

 TWS SUBSYSTEM NAME  ===> TWSC
 SIMULATED IA TIME   ===> 05/02/17   20.40   In format YY/MM/DD HH.MM

 APPLICATION ID      ===> ________________
 SELECTION   EXIT    ===> ________________
 STANDARDS NAME      ===> ________________
```

Figure 7-14 Validating an Application

This option prepares a job to perform the validation. You can either submit the job directly or edit the validation job by entering either `S` or `E` respectively. The output from this job is the JOB/SCAN Structured JCL Listing. It shows all the JCL operations in the application in dependency sequence.

7.4 Assessing the impact of changing variables

So far, we have discussed validating JCL against elements that are defined within the TWS for z/OS database or plans. In some cases, though, you might want to look at hypothetical situations to see what would happen if you changed TWS for z/OS definitions rather than the JCL.

TWS for z/OS only returns resolved JCL based on an existing current version of an application. However, there are some elements that you can manipulate to perform a "what if?" scan.

One such element is a variable within a variable table. You would not want to change the existing production table, because this would impact every job that referenced that table. However, you can use the **JTWS** macro to simulate JCL that is resolved with alternate values to user variables.

To do this, identify the variable table that contains the variable or variables that you want to change. Make a copy of this table using option 1.9.2 within TWS for z/OS, as shown in Figure 7-15.

```
---------------------- LIST OF JCL VARIABLE TABLES -------- Row 1 to 3 of 3
Command ===>                                              Scroll ===> PAGE

Enter the CREATE command above to create a new JCL variable table
or enter any of the following row commands
M - Modify, C - Copy, D - Delete, B - Browse

Row Variable        Owner           Table                          Last
cmd Table           Id              Description                    update
c   GLDLYPOST       GL              General Ledger Daily           TWSRES6
'   GLMTYPOST       GL              General Ledger Monthly         TWSRES6
'   GLWLYPOST        GL              General Ledger Weekly
TWSRES5
***************************** Bottom of data ********************************
```

Figure 7-15 Copying a JCL variable table

Give the copied table a name that does not conflict with any real table names. Then, from within the copied table, alter the values of any variables that you want to assess the impact of changing, as shown in Figure 7-16 on page 308.

```
---------------------- CREATING A JCL VARIABLE TABLE ------- Row 1 to 1 of 1
Command ===>                                                 Scroll ===> PAGE

Enter/change data below and in the rows,
and/or enter any of the row commands below:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete, S - Select
variable details.

VARIABLE TABLE     ===> COPYGLDLYPOST___
OWNER ID           ===> GL  ____________
DESCRIPTION        ===> General Ledger__________

Row Variable Subst.   Setup  Val  Default
cmd Name     Exit            req  Value
''' USRCODE_ ________ N       N  NEW_LEDGER_____________________________________
''' LU______ ________ N       N  VUJAX__________________________________________
''' UNIT1___ ________ N       N  STH____________________________________________
''' UNIT2___ ________ N       N  WPH____________________________________________
*************************** Bottom of data ********************************
```

Figure 7-16 Updating a copied JCL variable table

There are two ways to use this table within JOB/SCAN, depending on how the variable table is connected to the job or application. Neither involves making changes to your real JCL tables and, therefore, have no risk of compromising your live system.

If the variable table is ordinarily connected to the application using a Run Cycle, Period, or a direct setting in the plan. Then, just as before, find the job that you wish to validate, and enter ISPF VIEW either within TWS for z/OS or externally.

Enter the **JTWS** command. From there you can use the VARIABLE TABLE field on the JTWS panel, specifying the name of your copy of the table, as illustrated in Figure 7-17 on page 309.

```
JOOPTS01 -------------------- JOB/SCAN FOR TWS ----------------------------
COMMAND ===>

   Review values and press ENTER to continue or END to return.

   CURRENT PLAN ENDS  ===> 05/02/18 08.00        TWS Subsystem : TWSC
   JOB NAME           ===> PJGL0020              LU NAME  : _________________
   APPLICATION        ===> *                     Blank and wild cards OK
   INPUT ARRIVAL      ===> __/__/__  __.__       Blank or YY/MM/DD HH.MM
   OPERATION (WS NO.) ===> ____ ___              Blank or WWWW NNN
   VARIABLE TABLE     ===> COPYGLDLYPOST
   SEARCH TWS CP      ===> YES                   Yes or No
   SEARCH TWS AD'S    ===> YES                   Yes or No
   PROVIDE NF ENTRY   ===> NO                    Yes or No (NF=Not Found)
```

Figure 7-17 Specifying an alternate JCL variable table

TWS for z/OS then uses this alternate variable table to provide values instead of the usual one that would have been connected to the application.

If the variable table was referenced explicitly in the JCL, either by a `//*%OPC TABLE` or `//*%OPC SEARCH` directive, then you need to amend this in ISPF VIEW before running **JTWS**. For example, replace the following:

```
//*%OPC TABLE NAME=&OADID
```

with

```
//*%OPC TABLE NAME=COPYGLDLYPOST
```

Then run **JTWS** as before, but do not specify anything in the TABLE NAME field.

This second technique can also be used if you wanted to check JCL against altered user variables contained within more than one variable table, by overriding each directive and pointing it to an alternate table.

If you had a combination of variable tables connected to the application and ones referenced in the JCL, then you can insert extra directives to reference the application connected tables immediately following the `//*%OPC SCAN` statement or at the top of the JCL if automatic JCL tailoring is enabled within TWS for z/OS.

Important: If you have used this technique from within EDIT, then make sure you CANCEL out of the JCL if you are using a live JCL source.

7.5 Avoiding production problems

Even with the most rigorous preproduction and testing regimes and water tight change processes, there are always things that can go wrong, or a minor change can have much more impact than first imagined. For example, a data set might be deleted when it should not have been. This could be an unexpected side effect of a Storage Management product deleting a data set, it could be the result of a scheduling error creating something without the later deletion, or it could simply be accidental manual deletion. Checking each job as it enters production will not spot these things if they happen after the JCL had been checked and migrated.

7.5.1 Daily validation of jobs in the current plan

When the current plan has been extended, the sequence of jobs, and the elements that are referenced by them, are as close to the final execution conditions as can be expected. There will always be the possibility of last minute changes and dynamic scheduling actions, but this is a reasonably stable point to evaluate the expected success or failure of batch.

The **JVAL** command has an option to allow you to perform such a final stage validation, either against the entire current plan or a subset of it.

Enter the **JVAL** command and select option **1** to access the Current Plan Validation panel (Figure 7-18 on page 311).

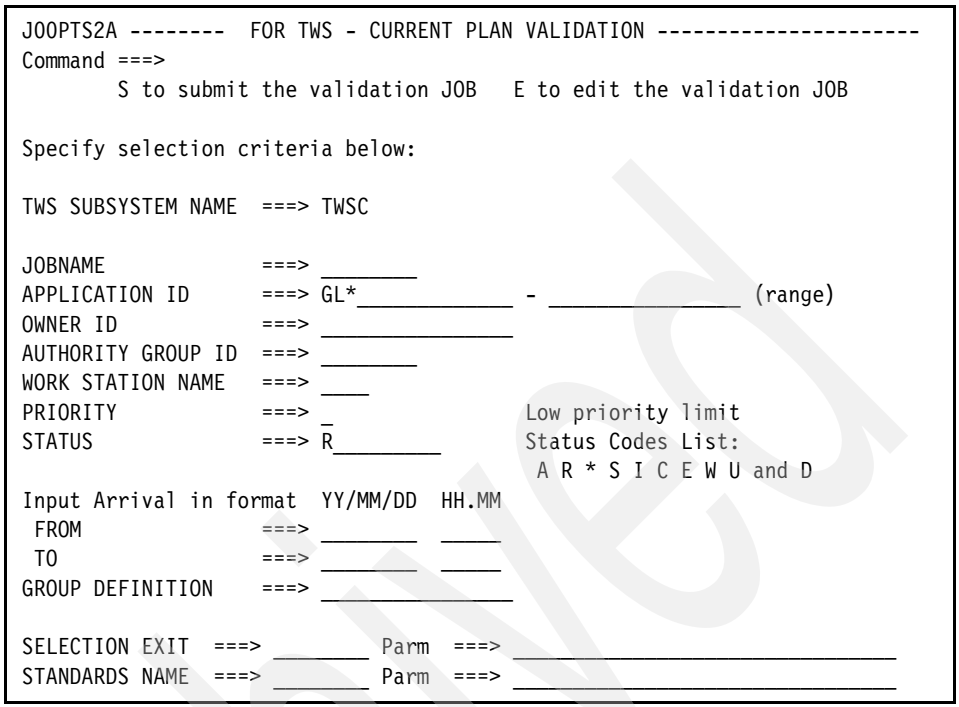

Figure 7-18 Validating jobs in the current plan

This panel allows you to specify exactly what you want like to check in the Current Plan.

To help choose the scope of the current plan to validate, set the appropriate filter fields. This can include Jobname, Application ID, Owner ID, Authority Group ID, Workstation Name, and Group Definition. With these fields, you can use the standard asterisk (*) and percentage (%) wildcards as used elsewhere in TWS for z/OS.

You can limit the lowest priority, and specify a list of operation statuses to further restrict the scope. If you do not set a status filter, then use **A**, **R**, *****, and **W**, effectively only validating jobs that have yet to run.

Finally, you can focus the validation on an input arrival range, checking only operations within the time period. Be aware that this can only be within the scope of the current plan.

You can make the current plan validation as focused or wide ranging as you wish. After you have entered your criteria for the check, enter S to submit the validation job or E to edit the validation job JCL before submission.

Use the Edit option to save the validation JCL to repeatedly run the same validation, possibly even schedule it. For example, you might have a system that is undergoing a frequent amount of changes for a particular period. Saving and scheduling a job to check the JCL scheduled for this system, every day would provide a valuable final safety net for the system during the period of change.

7.5.2 Daily planning process

The daily planning process itself is important to avoiding production problems, both from a TWS for z/OS and JOB/SCAN perspective. There are many factors to consider, including when you perform your daily planning and what you include within it. Integrating JOB/SCAN with some best practice planning processes can reduce your scheduling errors dramatically.

Consider such things as lead time between when you regularly extend your plan and when your current planning period ends. Ensure that there is time to recover any identified problems before TWS for z/OS runs out of planned workload. Equally consider when the majority of your support staff are on site for scheduling daily planning and any prevalidation. To avoid any unnecessary callout delays and resultant costs.

You might want to consider splitting your planning process in two — having a prevalidation stage that happens quite early to give staff ample time to find and fix problems and the actual extension stage, which also includes some final verification. Figure 7-19 on page 313 illustrates a possible daily planning process.

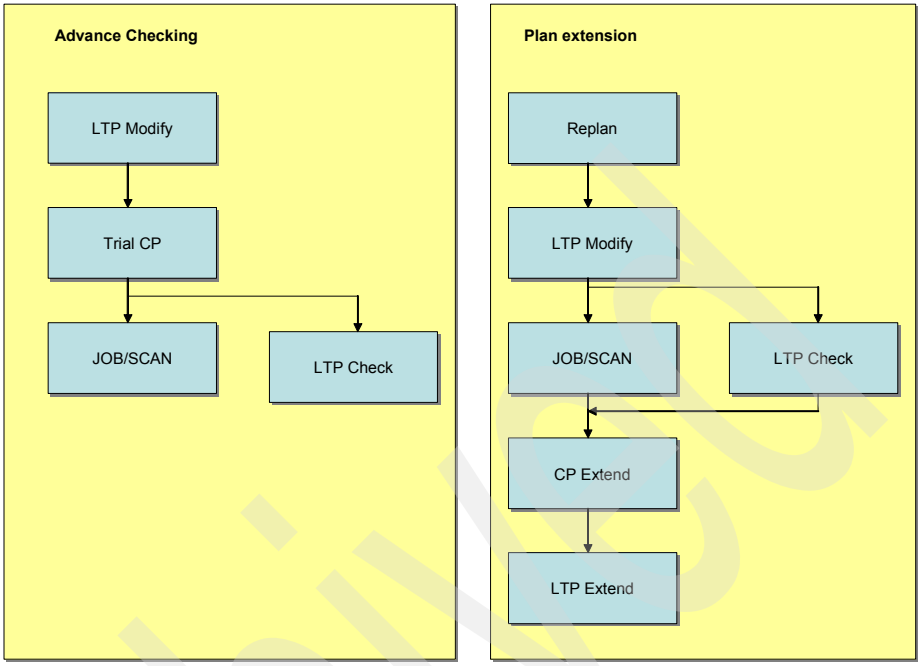

Figure 7-19 Daily planning processes

Advance checking

Consider scheduling your advance checking a few hours ahead of when you need to run the current plan extend process or during your main support hours if you usually extend when few staff are on site. This gives you time to spot any potential problems and correct them without having to delay the plan extend. Ideally, you should start advance checking with a Long Term Plan Modify All job. This ensures that everything that follows is working from the latest version. If you alter run cycles in an application, it has no bearing on when the application runs until the long-term plan has been modified. Whether this is done against the real Long Term Plan or a copy of it, depends on the circumstances. If you take a copy, ensure the rest of the advance checking runs from this copy.

Running a Trial Current Plan for the same period you are about to extend your Current Plan is the most effective way to spot problems. Especially dependency loops.

Assuming the planning is all in order, perform two further checks to ensure the safety of your next planning period:

- Run a scheduled version of the JOB/SCAN Current Plan Validation job. To create the JCL for this simply run option **1** of the **JVAL** command and use the **E** (Edit) command to save the JCL.
- Check is a little more complex to set up, but still vital. Check the Long Term Plan for occurrences with no predecessors and no time dependencies. This is vital to prevent the disaster of orphaned occurrences starting running as soon as the plan extends, often with disastrous consequences.

You can do this with a little bit of TWS for z/OS PIF programming. Execute a **LIST** of all LTOC entries (Long Term Plan occurrences) for the period you are about to extend the Current Plan. For each LTOC entry that is returned in which `LTOC#PRE` contains `0`, use the PIF to retrieve the application and search for an operation in which `ADOPTIM` is set to `Y`. If such an operation is not found, you have an occurrence that runs instantly, and you should flag this as an error.

Admittedly, TWS for z/OS PIF programming is not for everyone. However, this is a relatively simple process to create yourself and could be customized to perform other final checks on your Long Term Plan before the extension of the Current Plan.

Plan extension

Whether you extend once a day for 24 hours or use different planning periods, there are some considerations you should make with your plan extension process.

The first is to consider having a Replan job at the beginning of the planning suite. Though not the most obvious to consider, it does protect you from an unusual feature of TWS for z/OS planning. Most think that a Replan and a Plan Extend deletes completed occurrences from the Current Plan. This is not strictly true. It removes occurrences that have a status of D and changes any complete occurrences to have a status of D. So, even though it appears they have been deleted, in reality they take two planning jobs to be removed from the plan completely.

This can be particularly frustrating with the limit of 877 ETT occurrences of a single application in the plan. As the 877 limit includes ones in D status. So, you can hit the 877 limit with significantly less than 877 occurrences visible in the Current Plan. A replan job at the head of the planning suite ensures a clean start with every plan extend.

You should then do a Long Term Plan Modify All job, this time for real.

Follow this with the JOB/SCAN and LTP Check jobs. You might want to allow the plan to extend with some JCL errors, what level of severity is up to you, simply highlighting the problems for later investigation and correction. Bearing in mind that at the point your plan is extended, you would not normally expect your new workload to start running immediately.

When you are sure everything is safe, it is time to extend the Current Plan.

You might then also want to extend the Long Term Plan by a day. That way, you have a perpetual rolling Long Term Plan that is always the same length. Some sites might prefer to extend weekly by seven days or some other frequency, but this means that during the week the Long Term Plan length gets shorter and shorter until it is extended again.

7.5.3 Validating JCL beyond the current plan

Though validating the Current Plan when it is extended is a useful final safety net, if this is the only place where you are checking your JCL in the context and sequence, then expect errors to show up at the last minute.

To check multiple applications in sequence in advance, JOB/SCAN has interfaces to the Long Term Plan. One option allows you to validate jobs within the current extent of the Long Term Plan. Another allows you to validate beyond the Long Term Plan into the future. The TWS for z/OS API does not allow you to obtain when the Long Term Plan ends. Therefore, you have to choose the relevant option based on the length of your Long Term Plan. These are options 2 and 3, respectively, from the JVAL command menu.

Both options work with a copy of the Long Term Plan and a filtered copy of the database, allowing you to shorten the planning processes and understand the output. The only difference between the two options is that the extended option, not surprisingly, extends its copy of the Long Term Plan into the future.

Again, similar to the Current Plan option, you can filter which applications you wish to check. Due to the difference in nature between the Current and Long Term Plans, the filtering is not as wide ranging for this option, as can be seen in Figure 7-20 on page 316.

After you have decided on your filter criteria, you can either submit or edit the job as before.

```
JOOPTS2B --- JOB/SCAN FOR TWS - TRIAL PLAN VALIDATION (LONG TERM) ----------
 Command ===>

         S to submit the validation JOB   E to edit the validation JOB

 Specify selection criteria below:

 TWS SUBSYSTEM NAME  ===> TWSC
 WORK FILE PREFIX    ===> TWSRES6.TWSRES6.JTWS.TWSC_______________

 APPLICATION ID      ===> GL*_____________ - _______________ (range)
 OWNER ID            ===> _________________
 AUTHORITY GROUP ID  ===> ________
 PRIORITY            ===> _

 Input Arrival in format  YY/MM/DD  HH.MM
  FROM               ===> ________  _____
  TO                 ===> ________  _____
 GROUP DEFINITION    ===> _________________

SELECTION EXIT  ===> ________ Parm  ===> ____________________________________
STANDARDS NAME  ===> ________ Parm  ===> ____________________________________
```

Figure 7-20 Validating jobs in the long term plan

The EXTENDED version of the check looks almost identical to the LONG TERM version of the check, as shown in Figure 7-21 on page 317. Indeed it works in just the same way. However, you would use this to look at the impact of changes long into the future.

For example, you might want to look at what happens over extended holiday periods well in advance or check new jobs and schedules for your Year End process which will not run in the current scope of your Long Term Plan.

```
JOOPTS2B --- JOB/SCAN FOR TWS - TRIAL PLAN VALIDATION (EXTENDED) -----------
 Command ===>

         S to submit the validation JOB   E to edit the validation JOB

 Specify selection criteria below:

 TWS SUBSYSTEM NAME  ===> TWSC
 WORK FILE PREFIX    ===> TWSRES6.TWSRES6.JTWS.TWSC_______________

 APPLICATION ID      ===> GL*_____________ - _______________ (range)
 OWNER ID            ===> _______________
 AUTHORITY GROUP ID  ===> ________
 PRIORITY            ===> _

 Input Arrival in format  YY/MM/DD  HH.MM
  FROM               ===> ________  _____
  TO                 ===> ________  _____
 GROUP DEFINITION    ===> _______________

SELECTION EXIT  ===> ________ Parm  ===> ________________________________
STANDARDS NAME  ===> ________ Parm  ===> ________________________________
```

Figure 7-21 Validating jobs in an extended long term plan

7.5.4 Ad-hoc changes

Inevitably, there are last minute changes to production JCL. These changes fit into the following categories:

- Pre-emptive changes

  Typically these are changes to the JCL for a single run. This could be to accommodate slightly different processing over an extended holiday period, or initialize data as part of a migration to some new business process. These changes are known in advance but do not create a permanent change to the JCL.

- Job recovery

  This is when a job has failed and you need to make changes before restarting the job. This change could be something as simple as adding a RESTART card or as complex as adding overrides or new steps.

In both cases, the change is most likely going to be made directly through TWS for z/OS into the JS file, either within the Current Plan or the Long Term Plan. It is unlikely that the changes have been through any rigorous testing and even more

unlikely that the JCL has gone through any source management processes. So, validation at this stage is vital.

So, whether you are editing the JCL through the Current Plan, or Long Term plan, use the **JTWS** command before submitting the JCL. Figure 7-22 shows the **JTWS** command that is called from within the error handler.

```
EQQMJCLE ---------- EDITING JCL FOR A COMPUTER OPERATION ------------------
 Command ===> JTWS                                          Scroll ===> CSR

 Edit JCL below and press END to finish or CANCEL to reject:

 Application           : GLDLYPOST         General Ledger DLY Post
 Operation             : CPU1 20           GL Master Update
 Status of operation   : Ended in error    JCL
 Jobname               : PJGL0020          JCL last updated by: TWSRES6

 ******************************** Top of Data ****************************
 ==MSG> -Warning- The UNDO command is not available until you change
 ==MSG>           your edit profile using the command RECOVERY ON.
 000001 //PJGL0020 JOB (A,P,GL),'JONES',CLASS=A,MSGCLASS=X,  GL MASTER
UPDATE
 000002 //           MSGLEVEL=(1,1)
 000003 //      JCLLIB ORDER=TWS.INST.JOBLIB
 000004 //*%OPC SCAN
 000005 //*%OPC SETFORM OCDATE=(MM/DD/YY)
 000006 //*%OPC SETVAR TACC=(OCDATE - 3WK)
 000007 //*%OPC SETFORM OCDATE=(DD/MM/CCYY)
 000008 //*%OPC SETVAR TAUD=(OCDATE - 27WD)
 000009 //*%OPC TABLE NAME=&OADID
 000010 //*****************************************************************
 000011 //JS010    EXEC  PPGL0020,TYPE=WEEKLY
```

Figure 7-22 Invoking JTWS from the error handler

When used in this context, JTWS sets the Application, Input Arrival, and Operation number for you. However, when you are setting up an ad-hoc job, it is possible that there is no entry in the JS file. JOB/SCAN must take the JCL in edit and insert it into the JS file before TWS for z/OS can resolve the variables. Ordinarily, JTWS would restore the contents of the JS file to its former state. However, because you are in edit on the JCL, the TWS for z/OS API does not allow JTWS to delete it. This, in effect, prestages the JCL. JTWS warns you when this condition arises as shown in Figure 7-23 on page 319.

```
JOOPTS12 --------------- JOB/SCAN FOR TWS ADVISORY -----------------------
 COMMAND ===>

  Enter Y to continue processing.  JCL will be left in JS file.
  Enter N to terminate processing. JS file will remain unchanged.

  Warning: If processing continues JOB/SCAN will write records to
             the JS file that it cannot delete.  This will have the
             effect of prestaging the JCL from the editor.
```

Figure 7-23 JS File Update Warning

Because you are preparing JCL, it is likely that you want this to happen. If not, simply exit from editing the job, edit the job again, delete all the lines, and exit to delete the entry from the JS file.

Note: For more information about this process, see *IBM Tivoli Workload Scheduler for z/OS Managing the Workload*, SC32-1263.

7.6 Tuning JOB/SCAN for special situations

There are special situations within a TWS for z/OS application that might appear to be a JCL error at first, that in reality are not an error. For example, you might have a job that requires a special data set. The JCL refers to the data set as `DISP=OLD` or `DISP=SHR`, and yet the data set does not exist, and there is no visible JCL to create it.

This situation occurs with data sets that are transmitted from other locations and are not controlled by the same TWS for z/OS controller. With TWS for z/OS you would account for this situation by defining a *Special Resource* of the same name as the data set and making it a job requirement. The resource could be defined as unavailable by default and set to available by data set triggering, at which point the job would start and process the data set.

```
---------------------- BROWSING SPECIAL RESOURCES ------- Row 7 to 22 of 22
 Command ===>                                              Scroll ===> CSR

 Enter the row command S to select a resource.

 Row Special                                        Specres  A Qty    Num
 cmd Resource                                       group ID          Ivl
 '   EA733TAPES                                             Y 10     5
 '   FC.GLPROD.INPUT.FILE                                   Y 1      0
 '   HOLD.U9APPL                                            Y 1      0
```

Figure 7-24 Special Resources List

Note: For details about how to set up Data Set Triggering within TWS for z/OS, see *Implementing Support for Data Set Triggering in IBM Tivoli Workload Scheduler for z/OS Installation Guide*, SC32-1264.

This is not an error condition. However, by all appearances, it would be as shown in Figure 7-25. A possible solution to this situation is given in Appendix C, “JOB/SCAN examples” on page 677, with source code examples. Similar situations can be resolved in a similar fashion.

```
*** ERROR SUMMARY AND COUNTS ***
    1  ADVISORY LEVEL
    0  WARNING LEVEL
    1  ERROR LEVEL
    2  TOTAL ISSUED       3 SUPPRESSED.
     THE HIGHEST SEVERITY CODE ENCOUNTERED WAS    08.
REL LINESV MSG.NO.    ERROR MESSAGE
---------- --------  ------------------------------------------------------
    28   8 DSS4050E - DATA SET NOT FOUND IN CATALOG                         P
    29   0 DSS8900A - DSN = "FC.GLPROD.INPUT.FILE "                         P
XPJ60K  - END OF REPORT  JOB/SCAN    6.2.4A (C) DIVERSIFIED SOFTWARE SYSTEMS

****************************** Bottom of Data ******************************
```

Figure 7-25 Results of scan without interface to TWS for z/OS in place

Extending JOB/SCAN for this purpose is a simple task using a JOB/SCAN exit and a call to TWS for z/OS within it. In this case, the example exit is invoked as a JOB/SCAN standards program.

This example program is written in COBOL. You can find the source for this program in Appendix C, “JOB/SCAN examples” on page 677.

Note: JOB/SCAN has various exit points to help customize the validation process. This way, you can re-interpret errors or extend validation to enforce site standards and policies.

Here is what happens in the example program:

As JOB/SCAN is validating the JCL, it calls the standards program. Part of the information that is passed to the program includes any error messages. In this case, the program looks for the message `Data Set Not Found.` When this message is encountered, the exit calls TWS for z/OS to see if there is a special resource that is defined in the database with the same name as the data set. If there is, the exit tells JOB/SCAN not to issue that message. Instead, JOB/SCAN issues the message `TWS WILL WAIT FOR SPECIAL RESOURCE` and lists the resource name on the following line as shown in Figure 7-26.

```
*** ERROR SUMMARY AND COUNTS ***
    1  ADVISORY LEVEL
    1  WARNING LEVEL
    0  ERROR LEVEL
    2  TOTAL ISSUED        4 SUPPRESSED.
     THE HIGHEST SEVERITY CODE ENCOUNTERED WAS    04.
     THE  STDS  PGM  CALLED FOR THIS RUN WAS "TWSSTD1".
REL LINESV MSG.NO.    ERROR MESSAGE
---------- --------   ------------------------------------------------------
    28   4 DSS9101W -  TWS WILL WAIT FOR SPECIAL RESOURCE.                  P
    29   0 DSS9102A -  TWS RESNAME = "FC.GLPROD.INPUT.FILE"                 P
XPJ60K  - END OF REPORT  JOB/SCAN    6.2.4A (C) DIVERSIFIED SOFTWARE SYSTEMS
```

Figure 7-26 TWS WILL WAIT FOR SPECIAL RESOURCE message

7.6.1 Extending this example

This example shows a simple version of what is possible. You could extend this in a variety of ways. For example, by applying the test only to data sets of a particular naming standard or by restricting the search to certain job names or steps.

This example only looks to see if a named Special Resource exists in the database. You can take this further and check if the resource is defined against the current operation in the current application by placing the application name and operation number in comments in the JCL or passing them as parameters to

the JOB/SCAN Standards Program. The former is quite simple to achieve by including the following lines in your job JCL:

```
//*%OPC SCAN
//* TWS INFO &OADID &OOPNO &OJOBNAME &OYMD1 &OHHMM
```

The exit could then use the TWS for z/OS INFO comment from the resolved JCL to call the TWS for z/OS API with the exact context of the job.

You could also use the exit to check standards comparing the application to the JCL. For example, a standard might insist that certain jobs or programs execute from specific workstations. You can extend JOB/SCAN to check anything in the JCL that has relevance to the application.

Part 3

TWS Distributed integrations

This part of the book discusses product integrations with Tivoli Workload Scheduler Distributed (TWS) and includes the following chapters:

- Chapter 8, “Integrating Tivoli Enterprise Console” on page 325
- Chapter 9, “Integrating Tivoli Monitoring” on page 395
- Chapter 10, “Integrating Tivoli Business Systems Manager” on page 553
- Chapter 11, “Integrating Tivoli Intelligent Orchestrator” on page 579
- Chapter 12, “Integrating Tivoli Data Warehouse and Tivoli Service Level Advisor” on page 585
- Chapter 13, “Integrating Tivoli NetView Distributed” on page 615

8

Integrating Tivoli Enterprise Console

This chapter describes the integration between Tivoli Workload Scheduler 8.2 and Tivoli Enterprise Console 3.9. This integration creates a scheduling solution that is proactively monitored for some system level and all scheduler level problems.

This chapter discusses the benefits of this integration and includes information about the Tivoli Enterprise Console terminology and solution components. It also describes the configuration of both Tivoli Enterprise Console and Tivoli Workload Scheduler and discusses the design and implementation of this integration. Finally, this chapter describes how to use the integration and possible enhancements that can help you to identify needs in your environment that can be addressed by deploying Tivoli solutions.

If you are primarily a Tivoli Workload Scheduler (TWS) administrator who is responsible for implementing Tivoli Management Framework and Tivoli Enterprise Console, you should thoroughly read the documentation for those products. For information about the documentation that is available, see Appendix D, “Additional material” on page 687.

8.1 Benefits of integration

Tivoli Enterprise Console is a powerful, rule-based event management application that enables network, application, system, and database monitoring. Tivoli Enterprise Console collects, processes, and displays events that are gathered from various applications and systems within the managed environment. It can also respond to events automatically so that minimal user interaction is required.

TWS displays whether any jobs or job streams fail or abend and shows if any of the agents are unlinked or down. However, these error indications are not always the root of the problem, nor is it possible to monitor all resources in the scheduling network using just TWS. Tivoli Enterprise Console lets you leverage all the data that is collected by monitoring all possible aspects of TWS and the entire system environment, starting from the entire network (using Tivoli NetView) and down to the smallest application processes and application internals.

Tivoli Enterprise Console helps you effectively process the high volume of events in an IT environment. The essential functions include:

- Prioritizing events based on importance.
- Filtering redundant or low-priority events.
- Correlating events with other events from different sources.
- Initiating automatic corrective actions, when appropriate, such as escalation, notification, and the opening of trouble tickets.

Tivoli Enterprise Console integrates with TWS to provide an event management solution for the following TWS events.

- All abended or failed jobs and job streams.
- Unanswered prompts.
- Jobs potentially missing the deadline warnings (Late Jobs).
- TWS Fault Tolerant Agents that are unlinked or down.

The following shows the major Tivoli Enterprise Console-related components that are needed, along with their functions, for integration with TWS. We assume that TWS and Tivoli Enterprise Console are already installed in your

environment. See your Tivoli Framework administrator or IBM service provider for assistance with installing these products.

- TWS Plus Module

  Provides a set of predefined message formats, event classes, rules, and actions that can be taken if triggering events are generated, and Tivoli tasks to run against the scheduling environment.

- Tivoli Enterprise Console Adapter

  Formats the event information at the source to a form that is understood by the event server and sends it to the Tivoli Enterprise Console event server.

- Adapter Configuration Facility

  Provides a graphical user interface (GUI) to configure and distribute Tivoli Enterprise Console adapters.

The following sections provide the details of the integration.

8.2 Tivoli Enterprise Console terminology

Before discussing the entire integration environment and the use of Tivoli Enterprise Console, this section covers some basic terminology. These terms are the main components of the Tivoli Enterprise Console:

- Event adapters

  Processes that run on remote or local systems and monitor their resources (such as database application, physical disk, memory, and so forth.). Adapters detect reportable events, format them, and send them to the Tivoli Enterprise Console server. The event server then further processes the event. Adapters can monitor remote systems in the two ways:

  - They can receive events from any source that actively produces them
  - They can monitor ASCII log files if a source actively updates them with messages.

- Tivoli Enterprise Console event server

  The central server that handles all events in the management system environment. Tivoli Enterprise Console Server does event processing, evaluates the events against a set of predefined rules, and decides what actions, if any, to take on those events.

- Event consoles

  GUIs that allow users to see relevant events in different levels of details. Users can also respond to events, for example close or acknowledge them or initiate a task, depending on their administration roles.

- Events

  Units of information that are detected by event adapters and sent to the event server.

- Event definitions

  Definitions of type of events together with their properties. They are stored in BAROC files (Basic Recorder of Objects in C). Events and their properties have to be defined and then compiled into the rule base.

- Rules

  A set of preprogrammed logic steps, based on prolog, which process events that come through to the event server. Rules process the events and take action based on the attributes of those events. Just like event definitions, rules must be compiled into the rule base before they can be applied to incoming events.

When an event adapter detects an event that is generated from a source, it formats the event and sends it to the event server. The event server processes the event and performs any predefined tasks automatically. An adapter can be configured to discard certain type of events before they are even sent to Tivoli Enterprise Console, so that Tivoli Enterprise Console performs no unnecessary processing.

This chapter takes TWS as a source of the events.

8.3 Solution components

The following related IBM products are considered essential parts of the integration:

- Tivoli Management Framework 4.1.1. Tivoli Management Framework is a prerequisite for Tivoli Enterprise Console. It supplies the foundation necessary for Tivoli Enterprise Console to operate.
- Relational Database Management System (RDBMS). For this book, we used IBM DB2 Version 8.1. IBM DB2 is used in the integration that we show to store events that are generated by IBM Enterprise Console.
- IBM WebSphere® Application Server. IBM WebSphere is used by Tivoli Enterprise Console to present a Web interface as an alternative to the Java-based client console.

We assume that these components are installed and configured along with Tivoli Enterprise Console.

This chapter takes a stock installation of Tivoli Enterprise Console and integrates it with TWS. For details about installing and configuring Tivoli Enterprise Console, see your IBM service provider.

8.4 Tivoli Enterprise Console configuration

Before proceeding with the integration, we ensured that the following configuration requirements for Tivoli Enterprise Console were met:

- Tivoli Management Framework 4.1.1 with at least Fix Pack 2 installed. The fix pack is available for download at:

  `ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.1.1/4.1.1-TMF-FP02/`

- Tivoli Enterprise Console 3.9 with at least Fix Pack 2 installed. The fix pack is available for download at:

  `ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_3.9.0/3.9.0-TEC-FP02/`

- Tivoli Enterprise Console Adapter Configuration Facility (ACF) 3.9 is installed and patched to the same level as Tivoli Enterprise Console.
- If you intend to perform the configuration activities that require working within the Tivoli Management Framework Tivoli Desktop client application, you need Tivoli Desktop either installed on a Windows PC or accessible from your UNIX user account through the X Window System (by default the program name of `tivoli`), typically from the primary Tivoli Management Region (TMR) server.

Tivoli Enterprise 3.9 Fix Pack 2 contains important bug fixes for proper operation of the integration. If a later fix pack is available for Tivoli Enterprise Console, we recommend that you review it carefully and consider applying it as well (or in place of Fix Pack 2, if the later fix pack supersedes Fix Pack 2).

While Tivoli Enterprise Console can work without the Tivoli Enterprise Console Adapter Configuration Facility, integration with TWS as we show in this book requires interaction with components of the Tivoli Enterprise Console Adapter Configuration Facility. These components are called TEC logfile adapters and are the primary mechanism that conveys the state of TWS to Tivoli Enterprise Console.

8.5 TWS configuration

The Master Domain Manager (MDM) of each scheduling network that requires integration with Tivoli Enterprise Console must be installed as both a Managed Node and an endpoint in Tivoli Management Framework before performing the integration. See your site's Tivoli Framework administrator or your IBM service provider for details about how to accomplish this prerequisite.

Attention: Do not skip creating endpoints. Even if Managed Nodes already exist on the servers, endpoints are mandatory for the integration to succeed.

There are other benefits that are available by installing endpoints in Tivoli Management Framework on all Fault Tolerant Agents and Domain Manager in the scheduling network. These benefits are not directly related to the integration with Tivoli Enterprise Console. However, it is helpful to create these endpoints at the same time as you install the endpoint for Tivoli Management Framework.

You should also have written down the names of the *TWSuser* user account and its *TWSHome* home directory, as they is configured on the MDM. In the lab for this book, we used m82 as the *TWSuser* user account and /opt/tws/m82 as the *TWSHome* home directory of TWS on milan, the MDM in our architecture.

It is recommended, but not required, that you install the Job Scheduling Console (JSC) on UNIX MDMs to allow administrators to start the JSC from within the Tivoli Desktop GUI for Tivoli Management Framework. While convenient, it is not mandatory to configure the integration with Tivoli Enterprise Console. If you do have the JSC installed on the MDM, write down the directory into which it is installed. In our lab for this book, we installed the JSC into the default /opt/JSC_1_3 directory.

If the MDM is running on a Windows server, then the home directory of the JSC is not relevant. Any directory path is sufficient. It is not used by the integration because launching the JSC from the Tivoli Desktop depends upon the presence of the X Window System. In our lab for this book, we installed and configured Tivoli Workload Scheduler 8.2 at Fix Pack 6 level. We recommend that you apply the latest available fix pack that is applicable for your site.

8.6 Server architecture

In our lab, the following machines were relevant to the architecture of this solution scenario.

- milan
- paris

These machines ran the application servers for the solution scenario, as shown in Figure 8-1.

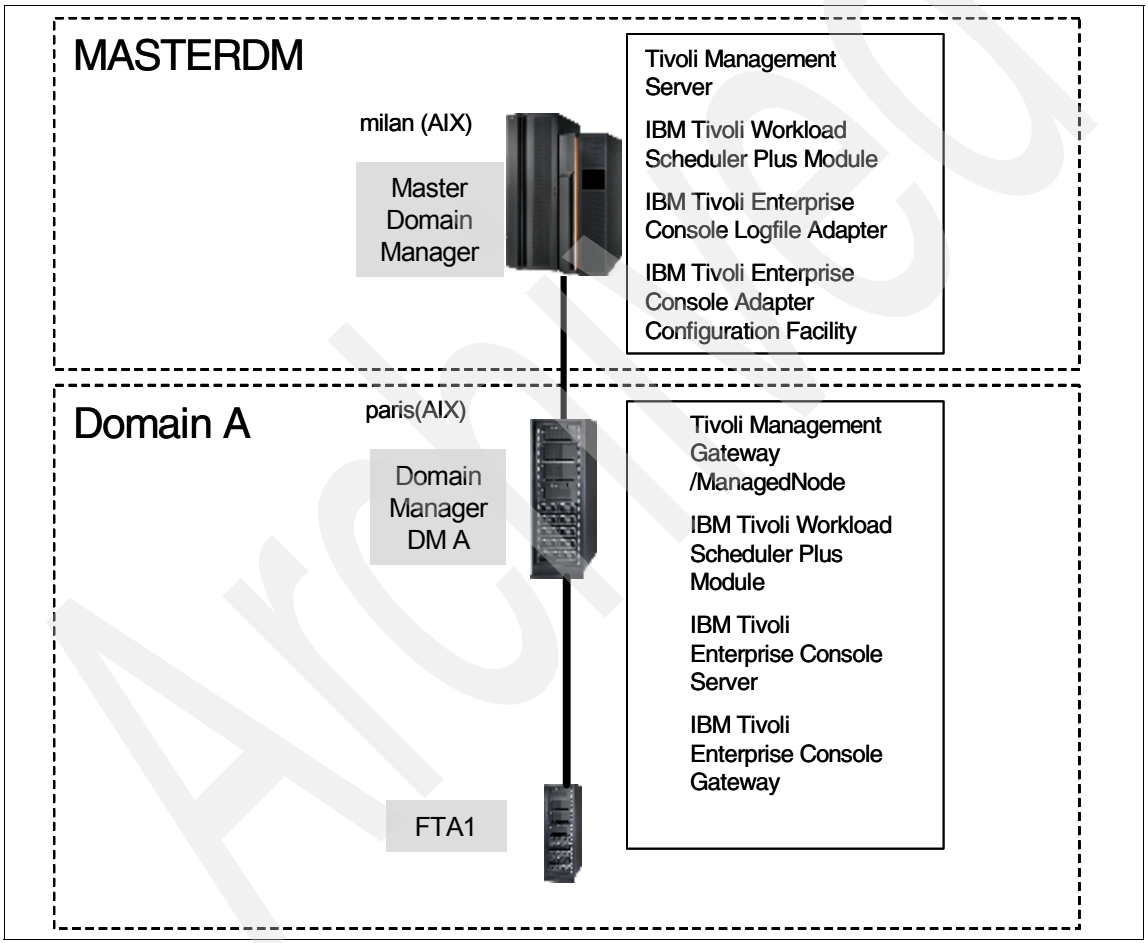

Figure 8-1 Architecture of TWS and Tivoli Enterprise Console in the lab

The server milan runs TWS. The server paris runs Tivoli Enterprise Console.

8.7 Design of the integration

Tivoli Workload Scheduler 8.2 comes with a product called TWS Plus Module that configures the integration between TWS and Tivoli Enterprise Console. This section describes the design of the integration.

Using a mechanism called a TEC logfile adapter, Tivoli Enterprise Console reads entries from the event.log file that is located in the home directory of the *TWSuser* user account on the Master Domain Manager of TWS. These entries describe all the activities that occur in the scheduling network.

The location of the log file event.log is configured through a BmEvents.conf configuration file. This file is the same file that is used to create the integration with NetView. The following line in the file places the event.log file in the home directory of TWS on milan, the MDM in the lab servers that we used for this book.

```
FILE=/opt/tws/m82/event.log
```

While other values can be used, you would have to manually configure the entire integration with Tivoli Enterprise Console and would not be able to take advantage of the TWS Plus Module.

Restriction: Changing where the event.log file is written or its name is not recommended unless you are very familiar with Tivoli Enterprise Console and are prepared to create and customize the integration manually.

The event.log file is truncated by the TWS batchman process during every Jnextday. The size of the file depends upon the activity in the scheduling network. Planning for 20 MB of disk space for this file is more than sufficient for all but the largest and busiest scheduling networks of more than a hundred FTAs with tens of thousands of jobs.

AFter the TEC logfile adapter is installed on the MDM, it runs from the endpoint cache directory. The exact location of this directory depends upon options your Tivoli Framework administrator selected during the installation of the endpoint software and how the endpoint software was installed. For example, multiple instances of the endpoint software can be installed on a single physical machine under certain circumstances. Consult your Tivoli Framework administrator or IBM service provider for details about the endpoint software installation on servers in your scheduling network. In the lab server milan that we use for this book, the cache directory was in the /opt/Tivoli/lcf/dat/1/cache/ directory path.

The TEC logfile adapter reads each line as it is appended to the event.log file, interprets the line, and generates an event to send to the TEC server. A configuration file is used by the TEC logfile adapter to determine where to send

the event. This configuration file is usually controlled from the TMR server. In the lab that we use for this book, the TMR server is milan. When the event is received by the TEC server, it is processed as any other TEC event.

Figure 8-2 shows the detailed steps that are involved in the event generation process flow.

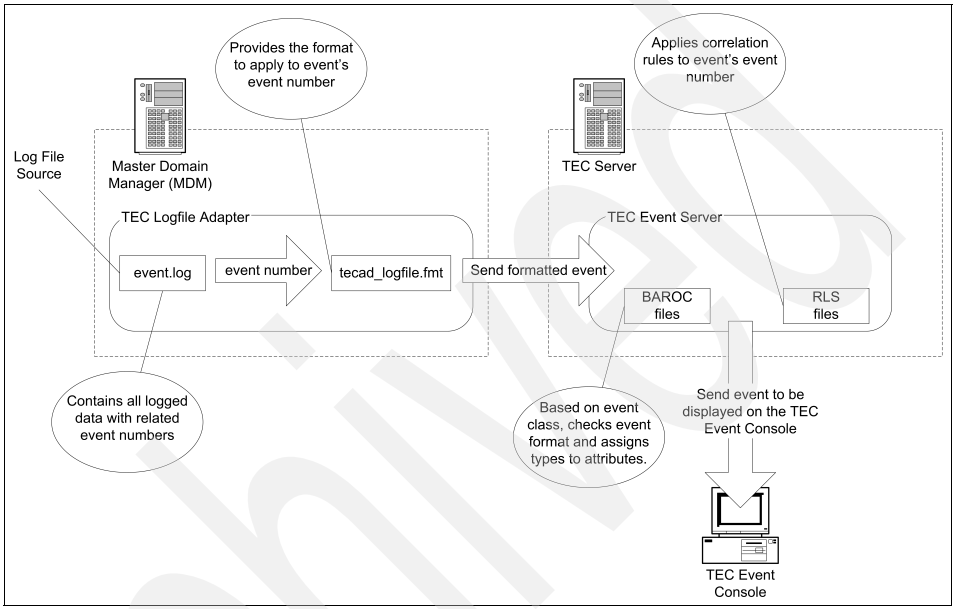

Figure 8-2 Event generation process flow

The steps that are involved in this process are:

1. The event generated at the source is characterized with an event number, which is an identifier of the type of event information.
2. If the event number is one that is specified in the EVENT field of the BmEvents.conf file, it is logged in the file that is specified in FILE field of the BmEvents.conf file (in our lab environment, this was the event.log file).
3. The TEC logfile adapter reads this information inside the log file, formats it using the structure that is stored in the FMT file (tecad_logfile.fmt), and forwards it to the TEC event server using the TEC gateway.
4. On the TEC event server, the structure of the formatted information is checked against the information that is stored in the BAROC files and, if correct, is accepted. Otherwise, a parsing failure is generated.
5. When the event is accepted by the TEC event server, a check on possible predefined correlation rules or automatic responses for that event number is

made using the information that is stored in the RLS files. If defined, the correlation rules or automatic responses are triggered, and the event is sent to the TEC event console to be displayed on the defined Event Console.

An important aspect to consider when configuring the integration with Tivoli Enterprise Console using event adapters is whether to monitor only the Master Domain Manager or to monitor every TWS agent. If you integrate only the Master Domain Manager, all the events coming from the entire scheduling environment are reported because the log file on a master domain manager logs the information from the entire scheduling system. Therefore, all events on the TEC event server and TEC event console look as though they originate from the Master Domain Manager, regardless of the TWS agent on which they were generated. The workstation name, job name, and job stream name are reported still to Tivoli Enterprise Console but as part of the message inside the event.

If instead you install a TEC logfile adapter on every TWS Fault Tolerant Agent (FTA), this results in a duplication of events both from the Master Domain Manager and from each FTA. Creating and using a TEC rule that detects these duplicated events (based on job_name, job_cpu, schedule_name, and schedule_cpu) keeps the event from the log file on the TWS agent, thereby addressing the duplication problem. The same consideration also applies to the backup Master Domain Manager, if one exists.

For information about creating new rules for Tivoli Enterprise Console, refer to *IBM Tivoli Enterprise Console Rule Developer's Guide Version 3.9*, SC32-1234. For information about the description of events and rules that are provided by the TWS Plus Module, refer to *IBM Tivoli Workload Scheduler Plus Module User's Guide Version 8.2*, SC32-1276.

8.8 Implementation of the integration

As shown in Figure 8-1 on page 331, the TMR server that we used in the lab for this book was installed on the TWS Master Domain Manager server milan. A Managed Node and Gateway are installed on the TWS Master Domain Manager server paris. In our environment, we installed the Tivoli Enterprise Console Server, the Tivoli Enterprise Console Gateway, and the Tivoli Enterprise Console client console all on the same server, paris.

For TWS to use the capabilities of Tivoli Enterprise Console, we installed and then configured the TWS Plus Module. This module provides the required support (rules, event formats, message classes, and actions) for the Tivoli Enterprise Console components.

Note: All commands that are run on the primary TMR, Tivoli Enterprise Console, and MDM servers that are shown in this section assume that the Tivoli environment is already sourced.

Source the Tivoli environment using the following command.

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

Your Tivoli Framework administrator or IBM service provider might have configured your shell environment to do this each time you log in. Refer to them for details.

8.8.1 Installation

You can find the TWS Plus Module on Disc 1 of the CD media, or you can download the file C52XZML.tar from your online Passport Advantage® account and uncompress to a directory. Within the main directory of the media, the subdirectory TWSPLUS contains the installation files for the TWS Plus Module.

The overall sequence of product and patch installations that we used to deliver the integration in our lab environment were as follows:

- Tivoli Workload Scheduler Plus Module Support (Link Binaries) 3.2.r
- Tivoli Workload Scheduler Plus Module 8.2
- 8.2.0-TIV-TWSPLUS-FP0005 patch for Tivoli Workload Scheduler Plus Module for Tivoli, 8.2

Install the TWS Plus Module, which is divided into two installable components, in the following order:

1. Tivoli Workload Scheduler Plus Module Support (Link Binaries) 3.2.r
2. Tivoli Workload Scheduler Plus Module 8.2

For instructions on how to install the Tivoli Workload Scheduler Plus Module 8.2, refer to the *IBM Tivoli Workload Scheduler Version 8.2 Plus Module User's Guide Version 8.2*, SC32-1276. You can obtain a copy of the guide at:

http://publib.boulder.ibm.com/tividd/td/WorkloadScheduler8.2.html

In our lab, after changing directories to the TWSPLUS subdirectory, we used the following commands run as root user on milan to install the components of TWS:

```
winstall -c `pwd` -i LINK.IND
winstall -c `pwd` -i TWS.IND \
   TWSUSER=m82 \
   TWSDIR=/opt/tws/m82 \
   JSCONSOLEDIR=/opt/JSC_1_3
```

We recommend installing at least Fix Pack 6 for Tivoli Workload Scheduler 8.2 and using later fix packs if they are appropriate and available. You can download Tivoli Workload Scheduler Fix Pack 6 from:

ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_8.2.0/8.2.0-TIV-TWS-FP0006/

The corresponding patches for Tivoli Workload Scheduler Plus Module 8.2 are in the TWSPLUS subdirectory of this URL. After downloading and expanding the tar archive found in this subdirectory, we ran the following command as root user on milan to install the patch:

```
wpatch -c `pwd` -i TWS.IND \
   TWSUSER=m82 \
   TWSDIR=/opt/tws/m82 \
   JSCONSOLEDIR=/opt/JSC_1_3
```

Note that we installed TWS Plus Module on all Managed Nodes in our lab environment because we only had two Managed Nodes in our lab environment, as shown in Example 8-1 of the **odadmin** command run as root user on milan.

Example 8-1 Command odadmin that is used to identify Managed Nodes in a TMR

```
milan:/#  odadmin odlist
Region          Disp  Flags  Port          IPaddr   Hostname(s)
1187404249         1    ct-    94         9.3.5.57   milan.itsc.austin.ibm.com,milan
                   2    ct-    94         9.3.5.45   paris.itsc.austin.ibm.com
```

Both Managed Nodes are used for the integration: milan as the MDM and paris as the TEC server. TWS Plus Module was installed on both because both servers were part of the scheduling network. In a production deployment, it should be installed on every Managed Node that is part of the scheduling network.

In your TMR environment, however, it is possible that some Managed Nodes are not TWS processors in a scheduling network. If so, be sure to use the appropriate installation command or procedure to limit the installation to only the Managed Nodes that participate in your scheduling network at most. For most sites, installing only on the primary TMR server(s) and TEC server(s) should be sufficient. See your Tivoli Framework administrator or IBM service provider for

assistance on limiting the Managed Nodes on which the TWS Plus Module is installed.

If your scheduling network does not follow a uniform standard for the *TWSuser* user account and *TWSHome* home directory, you must run the commands for the TWS.IND installation file and patch file separately for each set of servers that use a unique combination of *TWSuser* and *TWSHome*. This requirement of separate installations extends to the JSC directory path variable JSCONSOLEDIR as well. If the location of the JSC directory path changes, then you need a separate installation for each unique combination of *TWSuser*, *TWSHome*, and the JSC directory.

Note that the JSC directory variable is only significant if you have installed the JSC on a UNIX processor. If you do not run the JSC through the X Window System from a UNIX processor, then any value for the JSCONSOLEDIR variable is sufficient. Thus, this variable has no meaning on Windows Managed Nodes that are processors in a scheduling network.

For example, suppose there are three servers in a scheduling network that are named server1, server2, and server3. The first two use maestro as the *TWSuser* user account and /usr/local/tivoli/tws as the *TWSHome* home directory. The third server that is named server3 uses m82 as the *TWSuser* user account and /opt/tws as the *TWSHome* home directory. The installation and patch commands for the first two are shown in Example 8-2. These commands are run as root user on any of the three servers that have the installation media available.

Example 8-2 Sample commands for installation on server1 and server2

```
winstall -c `pwd` -i LINK.IND server1 server2
winstall -c `pwd` -i TWS.IND \
   TWSUSER=maestro \
   TWSDIR=/usr/local/tivoli/tws \
   JSCONSOLEDIR=/opt/JSC_1_3 \
   server1 server2
cd /directory/path/to/fix/pack6/TWSPLUS
wpatch -c `pwd` -i TWS.IND \
   TWSUSER=maestro \
   TWSDIR=/usr/local/tivoli/tws \
   JSCONSOLEDIR=/opt/JSC_1_3 \
   server1 server2
```

The installation and patch commands for the third server are shown in Example 8-3 on page 338, which is run as root user on any of the three servers that have the installation media available.

Example 8-3 Sample commands for installation on server3

```
winstall -c `pwd` -i LINK.IND server3
winstall -c `pwd` -i TWS.IND \
   TWSUSER=m82 \
   TWSDIR=/opt/tws \
   JSCONSOLEDIR=/opt/JSC_1_3 \
   server3
cd /directory/path/to/fix/pack6/TWSPLUS
wpatch -c `pwd` -i TWS.IND \
   TWSUSER=m82 \
   TWSDIR=/opt/tws \
   JSCONSOLEDIR=/opt/JSC_1_3 \
   server3
```

Note: The TWS Plus Module installation files are designed to also install additional components of another IBM Tivoli product — Tivoli Distributed Monitoring 3.7 — if it is discovered in the TMR environment. This product is no longer supported, as shown in the following URL.

http://www-306.ibm.com/software/sysmgmt/products/support/eos.html

Even if your TMR environment might still run Tivoli Distributed Monitoring and even if TWS installs components for it while it is installing the components for Tivoli Enterprise Console, we do not recommend that you use the components installed for Tivoli Distributed Monitoring because it is now an unsupported product.

8.8.2 Configuration

After the integration product and patch files are installed, you need to configure TWS Plus Module to identify the Tivoli Enterprise Console server and integrate with the TWS MDM. To complete the integration:

- Configure the tws_plus.sh file
- Configure installation options on FTAs with endpoints
- Distribute TEC logfile adapter to MDM
- Configure the Tivoli Enterprise Console server
- Configure the logfile adapter

Configuring the tws_plus.sh file is not documented in the manual *IBM Tivoli Workload Scheduler Plus Module User's Guide Version 8.2*, SC32-1276 but is required before the remaining configuration steps can work. The next sections show how to perform the configuration, and then explain why the configuration is

necessary. The explanation is helpful for Tivoli Framework administrators who generally complete this step but might want to customize the configuration.

Configuring the tws_plus.sh file

After installation of the TWS Plus Module product, a file is written to /etc/Tivoli/tws_plus.sh on all Managed Nodes that received the installation. In our lab environment, this file was written on both milan and paris. The original contents of this file in the lab that we used for this book is shown in Example 8-4.

Example 8-4 Original contents of the tws_plus.sh file

```
MAESTROUSER=m82
MAESTRODIR="/opt/tws/m82"
JSCONSOLEDIR="/opt/JSC_1_3"
```

Note: This file simply records the original installation variables that were used during the initial installation of the TWS Plus Module product (as well as for its patch).

This configuration step of the TWS Plus Module must be performed on the MDM and optionally performed on any other servers on which the Plus Module was installed. Configure the file by appending code to the end of the file as shown in Example 8-5 of a fully configured tws_plus.sh file (the original lines are in **bold**).

Example 8-5 Configured tws_plus.sh file

```
MAESTROUSER=m82
MAESTRODIR="/opt/tws/m82"
JSCONSOLEDIR="/opt/JSC_1_3"
export MAESTROUSER MAESTRODIR JSCONSOLEDIR
INTERNAL_LCF_INSTANCE=1
if [ -f /etc/Tivoli/lcf/${INTERNAL_LCF_INSTANCE}/lcf_env.sh ] ; then
        . /etc/Tivoli/lcf/${INTERNAL_LCF_INSTANCE}/lcf_env.sh
else
        echo "FATAL ERROR: endpoint environment lcf_env.sh not found,
exiting..."
        exit 1
fi
TEMP_LCF_INSTANCE=`basename $LCF_DATDIR`
echo "${TEMP_LCF_INSTANCE}" | grep [0-9] > /dev/null 2>&1
rc=$?
if [ ! -z "${TEMP_LCF_INSTANCE}" -a ${rc} -eq 0 ] ; then
        INTERNAL_LCF_INSTANCE=${TEMP_LCF_INSTANCE}
fi
if [ -d $LCF_BINDIR/../../../dat/${INTERNAL_LCF_INSTANCE}/cache ] ; then
        LCF_CACHEDIR=${LCF_BINDIR}/../../../dat/${INTERNAL_LCF_INSTANCE}/cache
        export LCF_CACHEDIR
```

```
else
        echo "FATAL ERROR: endpoint cache directory \
$LCF_BINDIR/../../../dat/${INTERNAL_LCF_INSTANCE}/cache not found, exiting..."
        exit 1
fi
```

In our lab environment, the tws_plus.sh file on milan (the MDM and primary TMR server) received the modification. You can download a more comprehensive version of this file that includes comments from the following URL:

ftp://www.redbooks.ibm.com/redbooks/SG246648

This file can be used as shown in Example 8-5 on page 339 by TMR environments that have created only one instance of an endpoint, where that instance is assigned an endpoint data directory number 1. Determine the endpoint data directory number by sourcing in the Lightweight Client Framework (LCF) environment and inspecting the LCF_DATDIR environment variable. How to source the LCF environment depends upon how the endpoint was installed. See your Tivoli Framework administrator or IBM service provider on how to complete this task.

In the lab for this book, we ran commands that are shown in Example 8-6 as root user on milan to source in the LCF environment and determine the endpoint data directory number.

Example 8-6 Finding the LCF data directory number

```
milan:/#  . /etc/Tivoli/lcf/1/lcf_env.sh
milan:/#  echo $LCF_DATDIR
/opt/Tivoli/lcf/dat/1
```

In our environment, the endpoint directory number is 1, which is the last element of the directory path in the LCF_DATDIR environment variable. This value is used in the first line that was appended to the original tws_plus.sh file, as shown in Example 8-7.

Example 8-7 First line of additional code for tws_plus.sh

```
INTERNAL_LCF_INSTANCE=1
```

If the value was for the endpoint directory was 3, for example, then we would make the variable INTERNAL_LCF_INSTANCE equal to 3.

As a best practice, we recommend that all servers on which TWS Plus Module is installed receive a modified tws_plus.sh file for the sake of consistency.

Modify the INTERNAL_LCF_INSTANCE variable in the additional code to accommodate the endpoint data directory number of each server. A consistent tws_plus.sh file that exports the same environment variables on each processor in the scheduling network that is a Managed Node (with the values accommodating each server's endpoint data directory number) ensures that the Plus Module can be easily reconfigured from another processor in the future if necessary.

Customizing the tws_plus.sh configuration

This information is useful for Tivoli Framework administrators who want to understand how tws_plus.sh interacts with the Configure_NON_TME_Logfile_Adapter Framework task that is described in 8.8.5, "Configuring the Tivoli Enterprise Console server" on page 369. For example, you might need to customize the task if there are unusual site-specific enhancements to the Tivoli Enterprise Console server installation.

The tws_plus.sh file is used during initial installation and configuration of the integration with Tivoli Enterprise Console. Various Framework tasks installed by TWS Plus Module also use this file when they are executed to locate the *TWSuser* user account, *TWSHome* directory, or the JSC directory. For the purposes of this book, we are concerned only with the installation and configuration aspects that deliver the integration with Tivoli Enterprise Console.

More specifically, the tws_plus.sh file is referenced by a Framework task in the logfile adapter configuration for TWS task library. After installation of TWS Plus Module, you can verify that the Task Library exists by running the **wlookup** command as root user on the primary TMR server (milan in our lab environment for this book), as shown in Example 8-8.

Example 8-8 Verifying Task Library logfile adapter configuration for TWS task library

```
milan:/#  wlookup -Lar TaskLibrary
     .
     .
     .
Logfile Adapter Configuration for TWS
     .
     .
     .
```

The task that uses tws_plus.sh is called Configure_TME_Logfile_Adapter. *TME* is a Framework term that stands for *Tivoli Managed Environment*. You can verify that the task exists by running the **wlstlib** command as root user on the primary TMR server (milan in our lab environment for this book) as shown in Example 8-9 on page 342.

Example 8-9 Verifying Task Configure_TME_Logfile_Adapter is installed

```
milan:/#  wlstlib 'Logfile Adapter Configuration for TWS'
(task)  Configure_NON_TME_Logfile_Adapter
(task)  Configure_TME_Logfile_Adapter
(job)   Configure_NON_TME_Logfile_Adapter (TWS)
(job)   Configure_TME_Logfile_Adapter (TWS)
```

This book does not show how to implement non-TME adapter-based integration because that approach is for environments where the Framework endpoint software cannot be installed on the MDM. We do not recommend this architecture because non-TME adapters are unable to take advantage of the additional features like encryption that TME adapters use by taking advantage of the Framework services the endpoint software delivers.

If your site requirements are such that you must use non-TME adapters, however, the Configure_NON_TME_Logfile_Adapter Framework task is used for the installation and configuration instead of its TME-based counterpart. It also uses the tws_plus.sh file, in the same way as described in this section. The Configure_TME_Logfile_Adapter Framework task runs a script that was previously stored internally within the object database that is used by Tivoli Management Framework. A copy of this script is installed onto each Managed Node on which TWS Plus Module is installed. The location of this copy can be determined using the **wgettask** command run as root user on the primary TMR server as shown in Example 8-10.

Example 8-10 Determining location of copy of script used by Configure_TME_Logfile_Adapter

```
milan:/#  wgettask 'Configure_TME_Logfile_Adapter' 'Logfile Adapter Configuration for TWS'
Task Name           :   Configure_TME_Logfile_Adapter
User Name           :   $root_user
Group Name          :
Task ACL            :   admin
Supported Platforms
  default <install-dir>/generic_unix/TAS/TASK_LIBRARY/bin/1187404249/Logfile_Adap_jxmywzea
Task Comments
  Task Name           : Logfile Adapter Configuration for TWS/Configure_TME_Logfile_Adapter
  Task Created        : Tue Mar  1 18:37:40 CST 2005
  Task Created By     : root@milan.itsc.austin.ibm.com
  Task Files
      default milan  /usr/local/tivoli/bin/generic_unix/TME/PLUS/TWS/./config_tme_logadapter.sh
  Distribution Mode   : ALI
  Task Comments       :
-----------------------------------------------------------
  Task Modified       : Tue Mar  1 18:37:59 2005
  Task Modified By    : root@milan.itsc.austin.ibm.com
  Task Files
      default milan  /usr/local/tivoli/bin/generic_unix/TME/PLUS/TWS/config_tme_logadapter.sh
  Distribution Mode   : ALI
  Task Comments       :
-----------------------------------------------
```

The copy simply serves as a reference and documentation of what the Framework task Configure_TME_Logfile_Adapter will run from its internally stored copy. Modifying the copy shown here does not modify what the Framework task runs.

The sample output from Example 8-10 on page 342 indicates that the copy of the script is stored in the file config_tme_logadapter.sh. Analysis of this script reveals that it has two crucial dependencies:

- The LCF environment needs to be sourced into the script.
- The environment variable LCF_CACHEDIR needs to point to the cache directory of the endpoint installation that will be used during the distribution of the TME logfile adapter as described in 8.8.4, “Distributing the TEC logfile adapter to MDM” on page 352.

Further analysis reveals that tws_plus.sh is sourced in early during the execution of the script. This is the basis of the changes made to tws_plus.sh as shown in Example 8-5 on page 339 and the reason for making the changes to tws_plus.sh as opposed to another file.

Tivoli Framework administrators who are responsible for environments that heavily customize the installation of the endpoint software can choose to either customize the Configure_TME_Logfile_Adapter Framework task directly or to modify the tws_plus.sh file further. See your Tivoli Framework administrator or IBM service provider for additional advice if you need to make these kinds of changes.

8.8.3 Configuring install options on FTAs with endpoints

Skip this step if you choose an integration of Tivoli Enterprise Console with only the MDM.

If you want to separately integrate Tivoli Enterprise Console with each FTA in your scheduling network, you must complete this configuration step. For each unique combination of the values for *TWSuser*, *TWSHome*, and the JSC directory that are used by one or more FTAs, complete the following tasks:

1. Start and log into the Tivoli Desktop. Your Tivoli Framework administrator can show you how to start a Tivoli Desktop session that is appropriate for your Tivoli environment.
2. Open the TivoliPlus collection object as shown in Figure 8-3 on page 344.

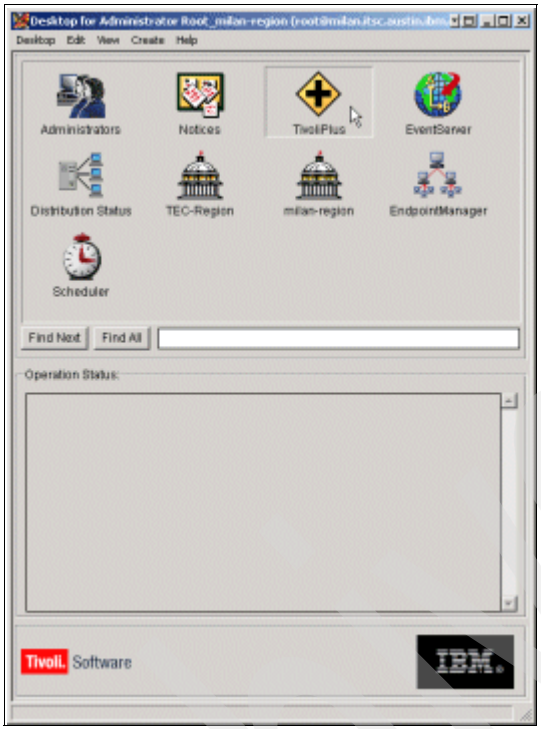

Figure 8-3 Opening the TivoliPlus collection object

The TivoliPlus window displays as shown in Figure 8-4.

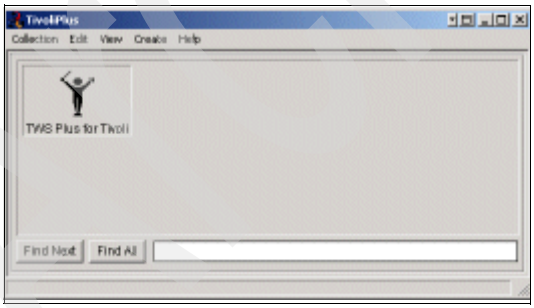

Figure 8-4 TivoliPlus window

3. Double-click the TWS Plus for Tivoli icon. The TWS Plus for Tivoli window displays as shown in Figure 8-5 on page 345.

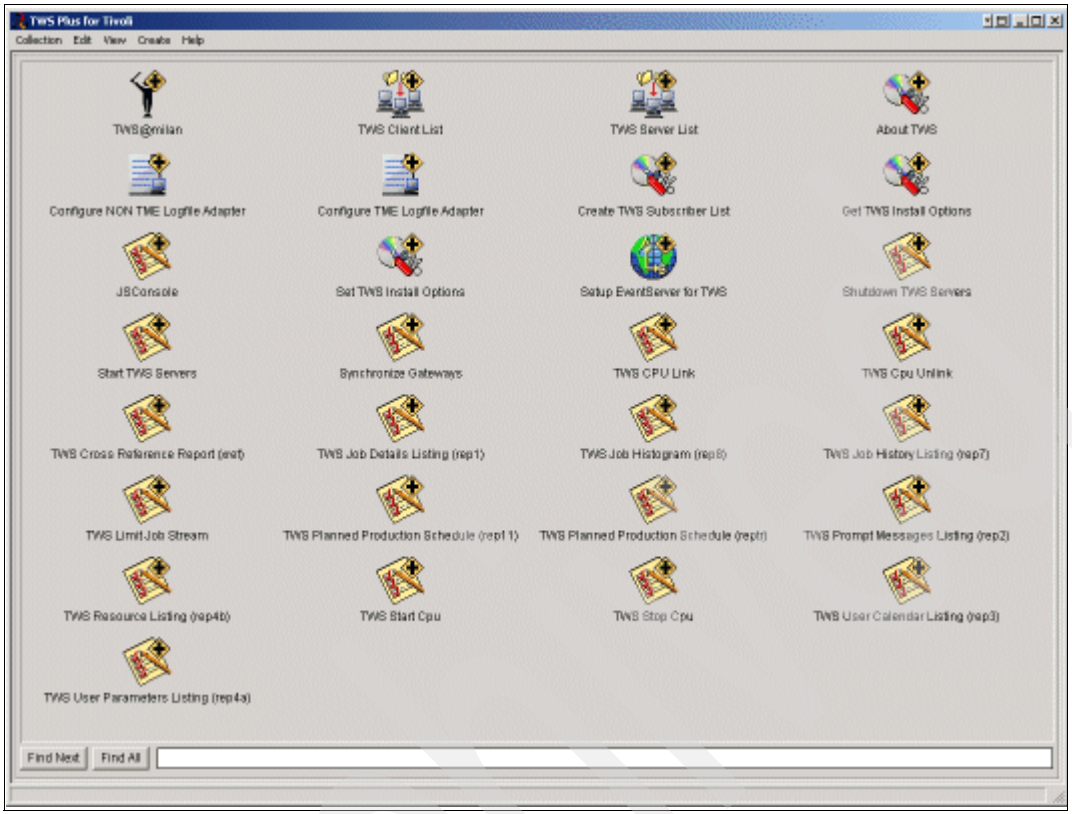

Figure 8-5 TWS Plus for Tivoli window

4. Right-click the Set TWS Install Options icon and select **Run on selected subscribers...** as shown in Figure 8-6 on page 346.

Tip: There is another icon called Get TWS Install Options. Be sure not to mistake it with Set TWS Install Options.

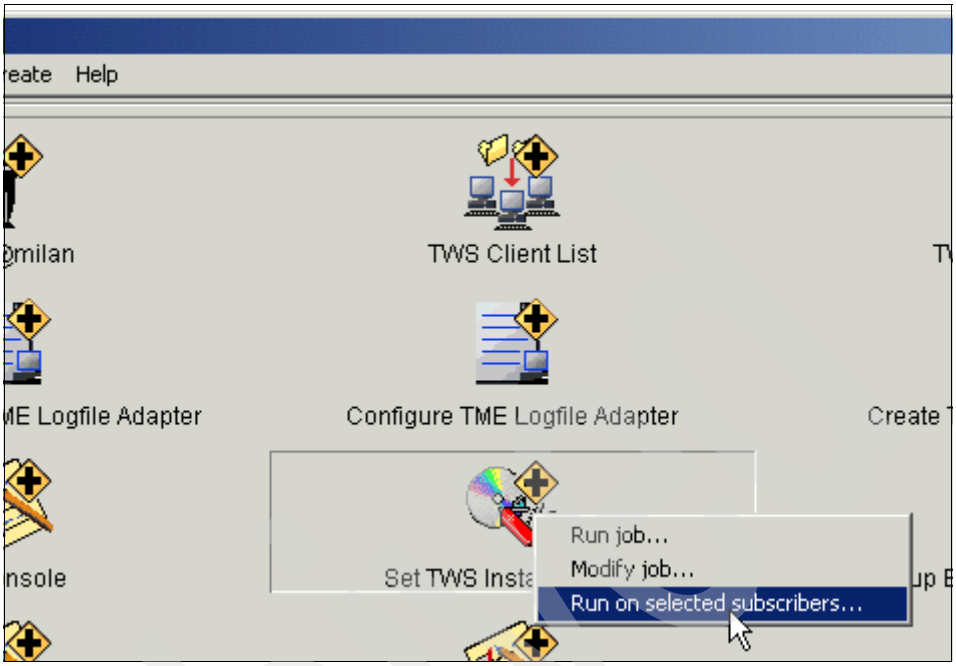

Figure 8-6 Running Set TWS Install Options on user-specified endpoints

The Execute Task window displays as shown in Figure 8-7 on page 347.

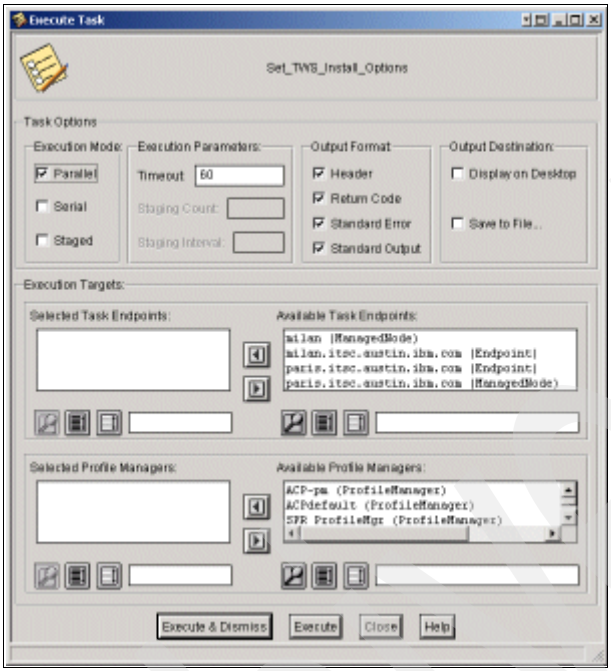

Figure 8-7 Execute Task window.

5. From the Available Task Endpoints list, select one or more FTAs with the same combination of values for *TWSuser*, *TWSHome*, and the JSC directory. Select the left-pointing arrow to move these FTAs into the Selected Task Endpoints list, as shown in Figure 8-8 on page 348.

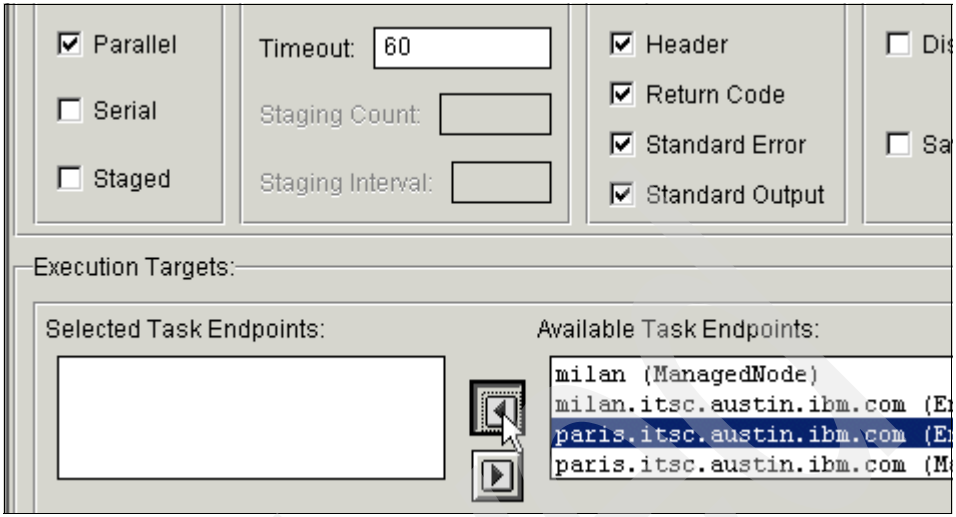

Figure 8-8 Detailed view of selecting an endpoint for the Set_TWS_Install_Options task

In our environment for this book, although paris already had the TWS Plus Module installed, as an example, we chose the endpoint instance of paris, which is denoted by the string `(Endpoint)` beside its IP host name, and moved it into the Selected Task Endpoints list as shown in Figure 8-9 on page 349.

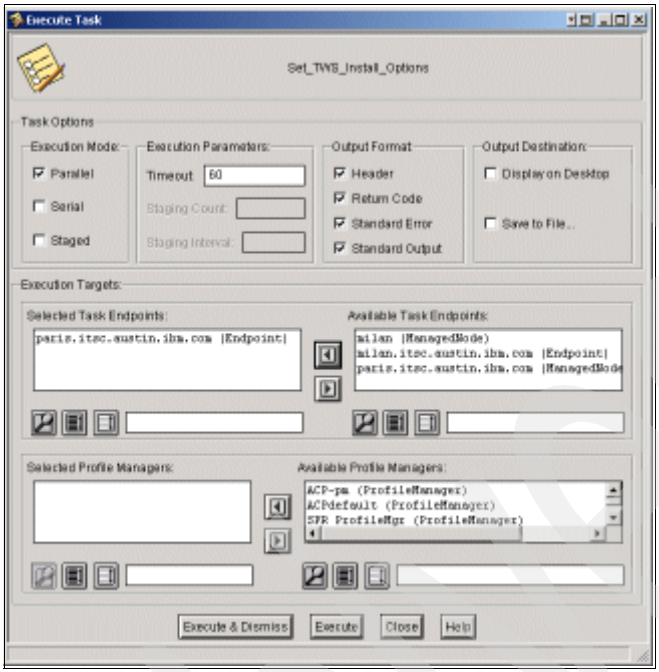

Figure 8-9 FTA selected for configuring TWS Plus Module options

6. Select **Display on Desktop** in the Output Destination group as shown in Figure 8-10.

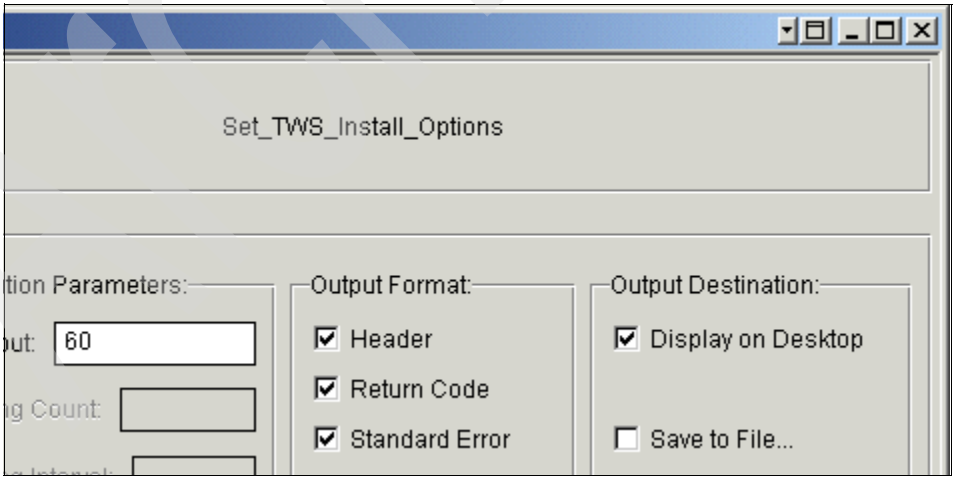

Figure 8-10 Selecting output destination for the task Set_TWS_Install_Options

7. Select **Execute & Dismiss**. The Set_TWS_Install_Options window displays as shown in Figure 8-11.

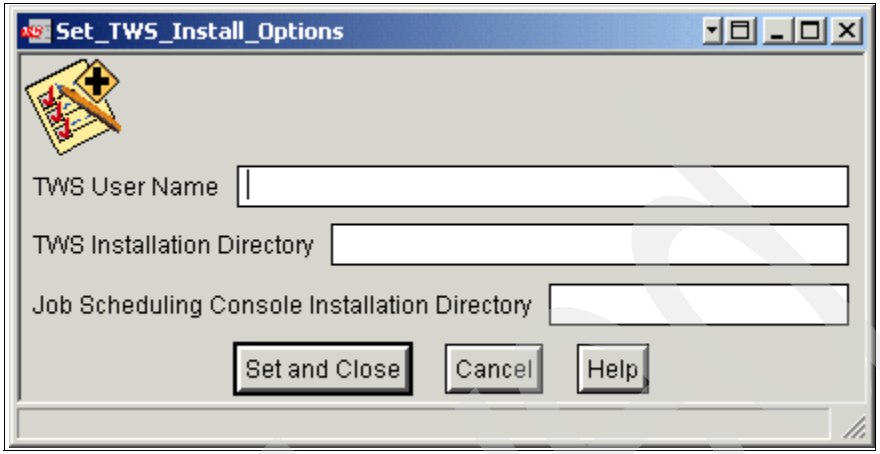

Figure 8-11 Set_TWS_Install_Options window

8. Enter the values for *TWSuser* in the TWS User Name text field, *TWSHome* in the TWS Installation Directory text field, and the JSC directory in the Job Scheduling Console Installation Directory text field. In our environment for this book, we used the values m82, /opt/tws/m82, and /opt/JSC_1_3, respectively, as shown in Figure 8-12.

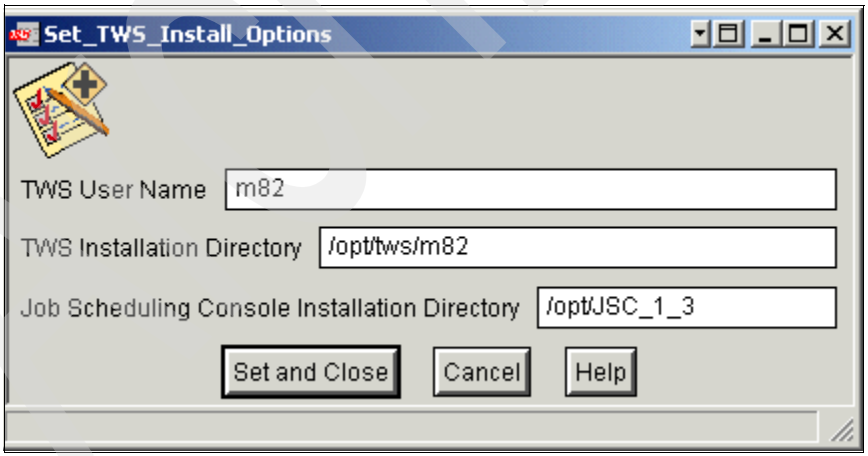

Figure 8-12 Values used in the lab environment in Set_TWS_Install_Options window

9. Select **Set and Close** in the Set_TWS_Install_Options window.

10. The task is executed and progress displays in the Set_TWS_Install_Options Output window, as shown in Figure 8-13.

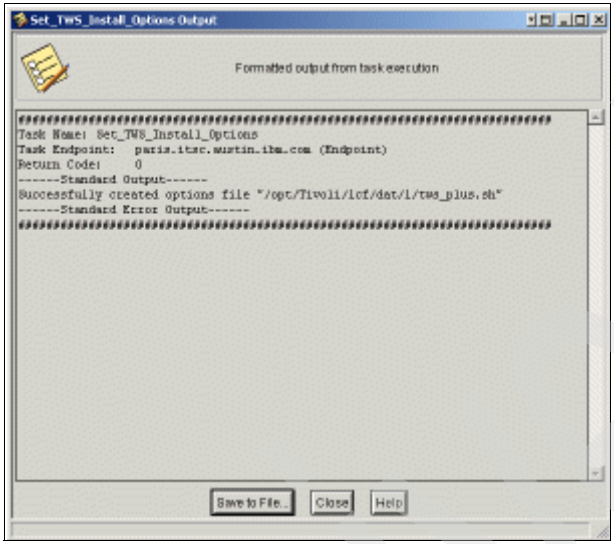

Figure 8-13 Set_TWS_Install_Options Output window

In the environment for this book, the output shows that the tws_plus.sh file is created for the endpoint instance of paris as shown in Example 8-11.

Example 8-11 Output of Set_TWS_Install_Options task for lab environment

```
################################################################################
Task Name:Set_TWS_Install_Options
Task Endpoint:paris.itsc.austin.ibm.com (Endpoint)
Return Code:0
------Standard Output------
Successfully created options file "/opt/Tivoli/lcf/dat/1/tws_plus.sh"
------Standard Error Output------
################################################################################
```

If the task was run on a Managed Node instead, the file would be created in the /etc/Tivoli directory.

11. Select **Close** to close the Set_TWS_Install_Options Output window.

12. Repeat steps 4 on page 345 through 11 for each unique combination of the values for *TWSuser*, *TWSHome*, and the JSC directory that are used by one or more FTAs.

If you have a medium to large scheduling network of 50 or more FTAs, your Tivoli Framework administrator can help you create and use subscriber lists to

simplify the selection of the lists of FTAs on which to run the task. These subscriber lists make the step described in 8.8.4, “Distributing the TEC logfile adapter to MDM” on page 352 more convenient as well. The task Create TWS Subscriber List in the TWS Plus for Tivoli window helps create these subscriber lists.

Tip: Sites that configure multiple instances of FTAs on a single physical server can use multiple endpoint software installations on the same server and create a one-to-one mapping between an endpoint and an FTA instance.

In this advanced scenario, the endpoint installation parameters differentiate among the endpoint instances by installing into different directory locations, communicating through different ports, and taking on different endpoint labels. Multiple endpoint instances can all communicate over the same IP address, but options for binding different instances to different interfaces also exist. See your Tivoli Framework administrator or IBM service provider for assistance with installing and configuring this advanced scenario.

8.8.4 Distributing the TEC logfile adapter to MDM

Before starting this step, ensure that the MDM has the Tivoli Management Framework (TMF) endpoint code installed. This code is also called the Lightweight Client Framework (LCF) code. This task will not succeed unless the endpoint code is already installed. The endpoint code is part of the TMF, which is bundled with Tivoli Workload Scheduler 8.2. See your Tivoli Framework administrator for details on how to install the endpoint software.

These instructions are a detailed expansion of instructions included in *IBM Tivoli Workload Scheduler Plus Module User’s Guide Version 8.2*, SC32-1276 for the benefit of TWS administrators who might not be familiar with Tivoli Enterprise Console.

To distribute the TEC logfile adapter to MDM:

1. Start and log into the Tivoli Desktop. Your Tivoli Framework administrator can show you how to start a Tivoli Desktop session that is appropriate for your Tivoli environment.
2. Open an appropriate Policy Region for creating a new profile manager as directed by your Tivoli Framework administrator. In our lab environment, we opened the Policy Region milan-region as shown in Figure 8-14 on page 353.

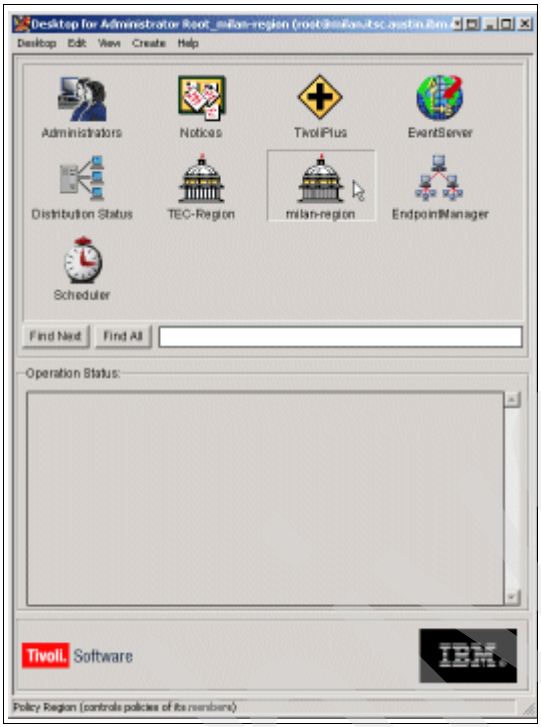

Figure 8-14 Opening the milan-region Policy Region to create a profile manager

The Policy Region window opens to display the managed objects that are within the policy region. In our environment, the window displayed as shown in Figure 8-15.

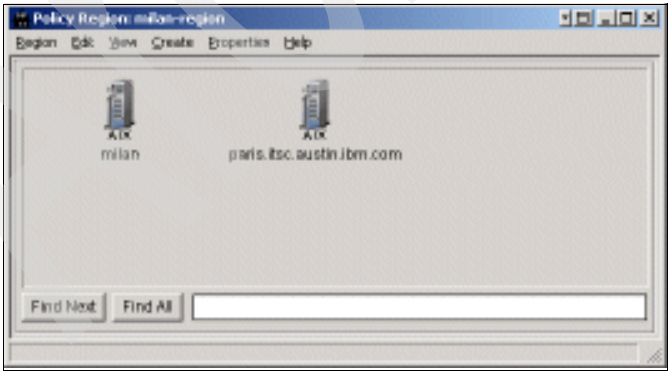

Figure 8-15 Policy Region window milan-region

3. Select **Create** and then **ProfileManager** as shown in Figure 8-16.

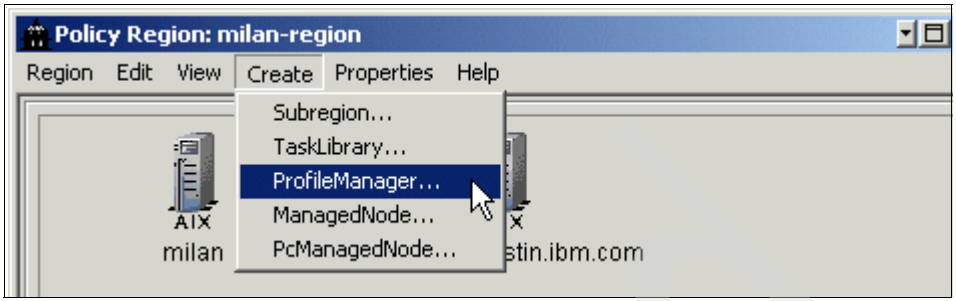

Figure 8-16 Creating a profile manager

The Create Profile Manager window displays as shown in Figure 8-17.

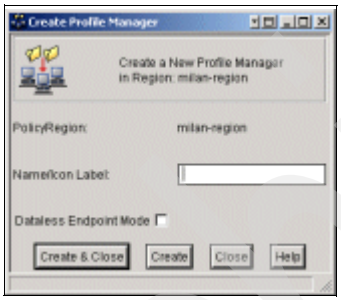

Figure 8-17 Create Profile Manager window

4. Enter a name for the new profile manager and select **Dataless Endpoint Mode**. In our environment for this book, we chose the name tec-tws.pm as shown in Figure 8-18 on page 355.

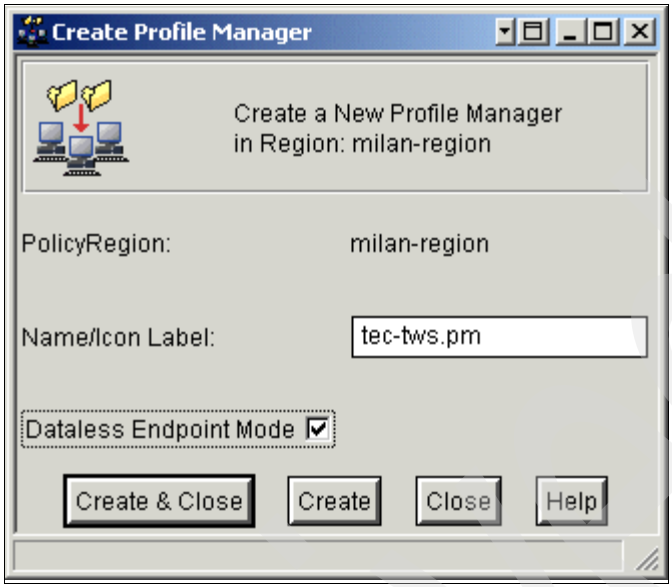

Figure 8-18 Create Profile Manager window

Select **Create & Close**. The Create Profile Manager window closes, and the profile manager is created in the Policy Region window as shown in Figure 8-19.

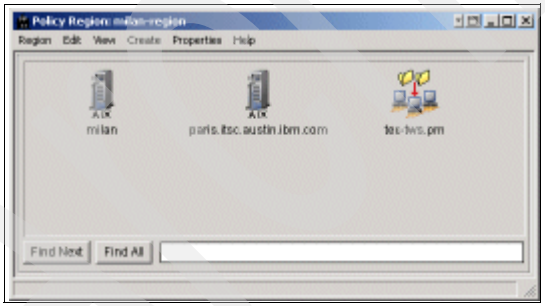

Figure 8-19 Policy region milan-region after creating tec-tws.pm profile manager

5. Select the **Properties** menu, and then select **Managed Resources...** as shown in Figure 8-20 on page 356.

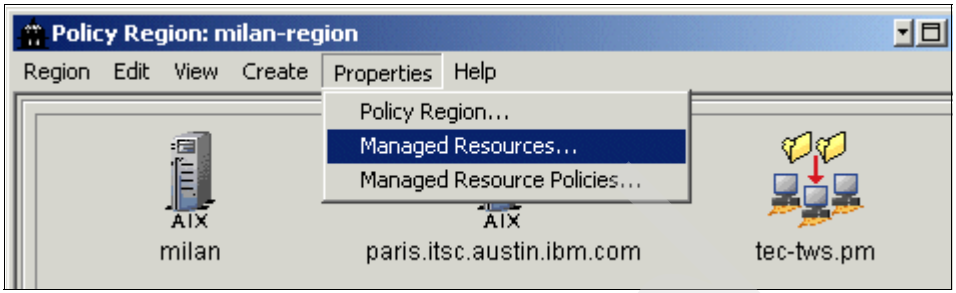

Figure 8-20 Configuring Managed Resources of a Policy Region

The Set Managed Resources window displays as shown in Figure 8-21.

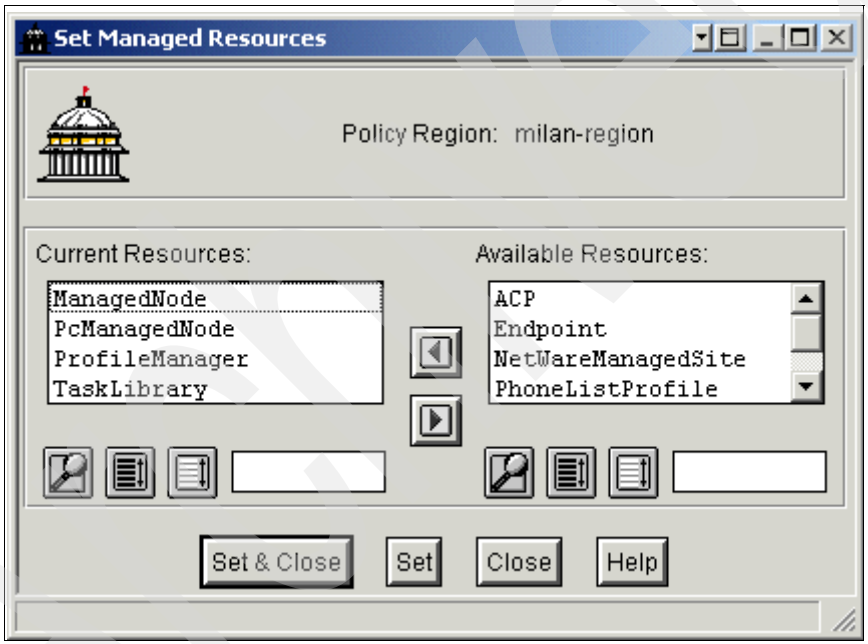

Figure 8-21 Set Managed Resources window

6. Select **ACP** from the Available Resources list on the right-hand side and select the left-pointing arrow button to move it into the Current Resources list on the left-hand side as shown in Figure 8-22 on page 357.

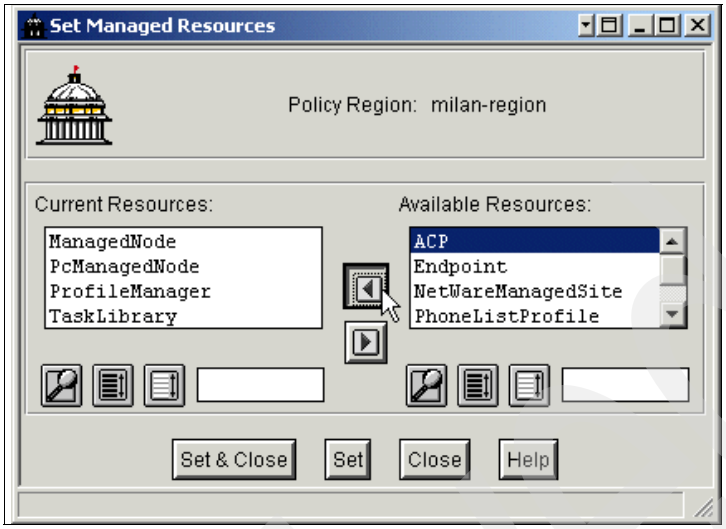

Figure 8-22 Making ACP resource a managed resource for a policy region

The Current Resources list displays the ACP resource as shown in Figure 8-23.

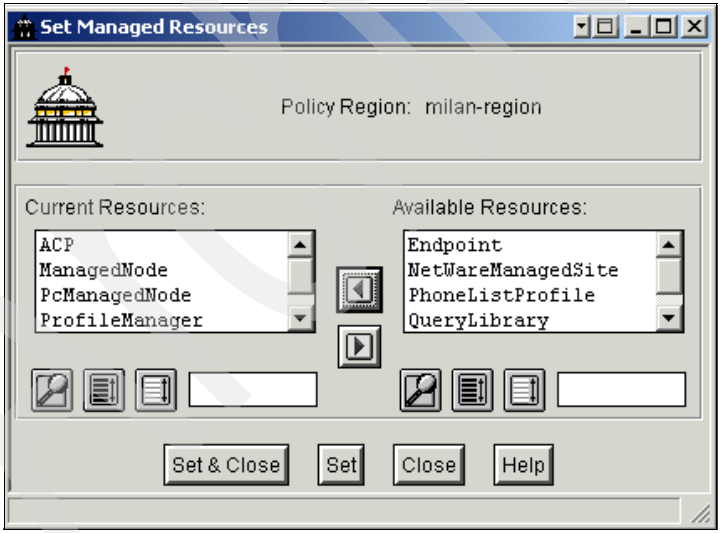

Figure 8-23 ACP resource a managed resource for policy region milan-region

Select **Set & Close**. The Policy Region window displays again.

7. Double-click the icon of the profile manager that you just created. In our environment for this book, we chose the tec-tws.pm icon as shown in Figure 8-24.

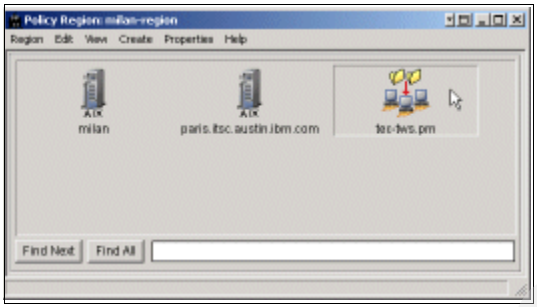

Figure 8-24 Opening the profile manager tec-tws.pm

The Profile Manager window opens as shown in Figure 8-25.

Figure 8-25 Profile Manager window for tec-tws.pm profile manager

8. Add the MDM to the Subscribers of the profile manager. Select **Profile Manager**, and then select **Subscribers...** as shown in Figure 8-26 on page 359.

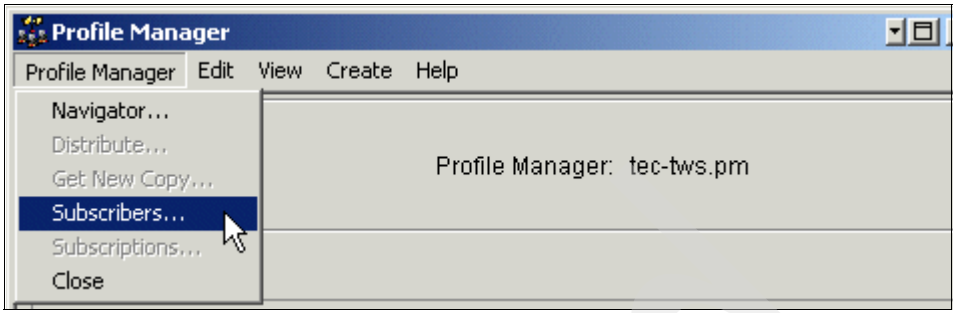

Figure 8-26 Adding a subscriber to the tec-tws.pm profile manager.

The Subscribers window displays as shown in Figure 8-27.

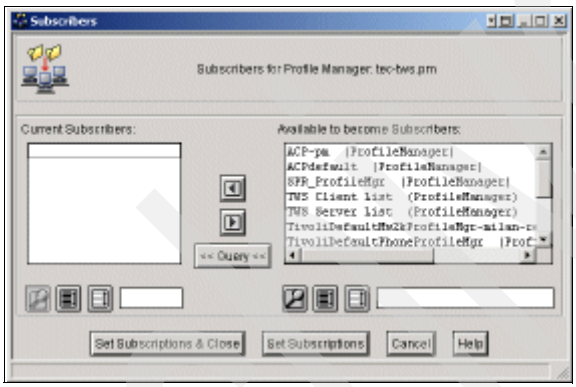

Figure 8-27 Subscribers window

9. Select the endpoint instance of the MDM in the Available to become Subscribers list on the right-hand side of the window and move it to the Current Subscribers list on the left-hand side of the window by pressing the left-pointing arrow. The endpoint instance is indicated by the string `(Endpoint)` beside the IP host name.

   In our environment for this book, we selected milan.itsc.austin.ibm.com (Endpoint) as shown in Figure 8-28 on page 360.

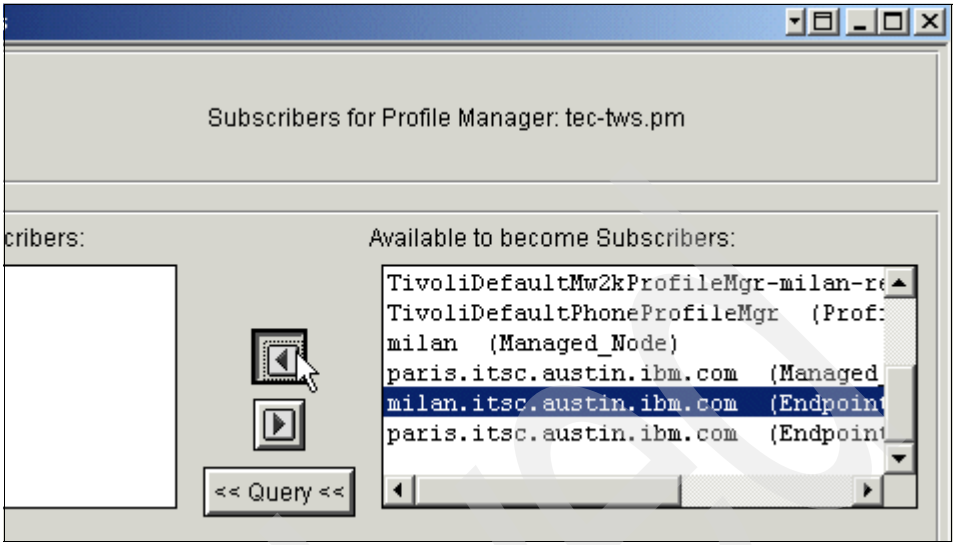

Figure 8-28 Detailed view of selecting endpoint instance of MDM as a subscriber

The endpoint instance of the MDM is moved into the Current Subscribers list as shown in Figure 8-29.

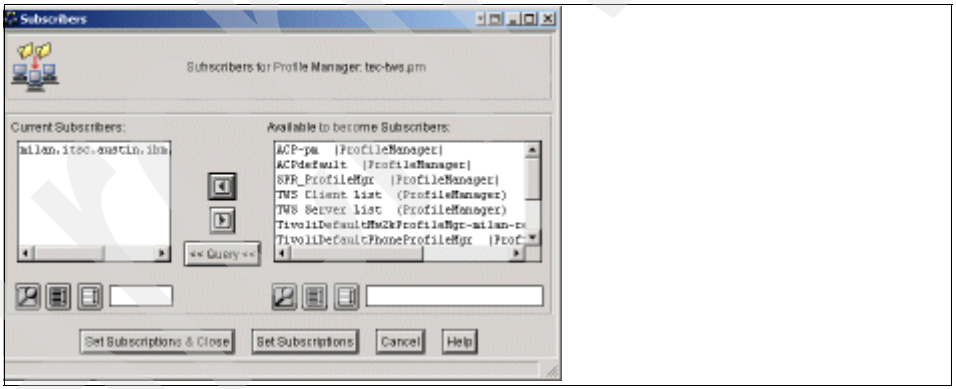

Figure 8-29 Endpoint instance of MDM configured as subscriber in profile manager

Note: If you choose to integrate Tivoli Enterprise Console separately with each FTA in your scheduling network, also add the endpoint instances of each FTA to the Current Subscribers list.

10. Select **Set Subscriptions & Close**. This closes the Subscribers window and returns you to the Profile Manager window. The Subscribers list in the profile

manager displays the endpoint instance of the MDM as a subscriber, as shown in Figure 8-30.

Figure 8-30 Profile manager with endpoint MDM as a subscriber

11. Add an ACP profile to hold the TEC logfile adapter. Select **Create** → **Profile...** as shown in Figure 8-31.

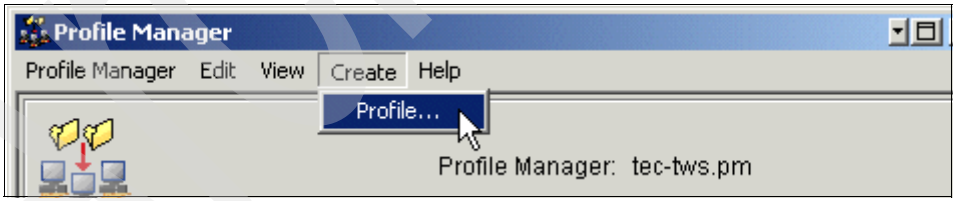

Figure 8-31 Adding an ACP profile to the tec-tws.pm profile manager

The Create Profile window displays as shown in Figure 8-32 on page 362.

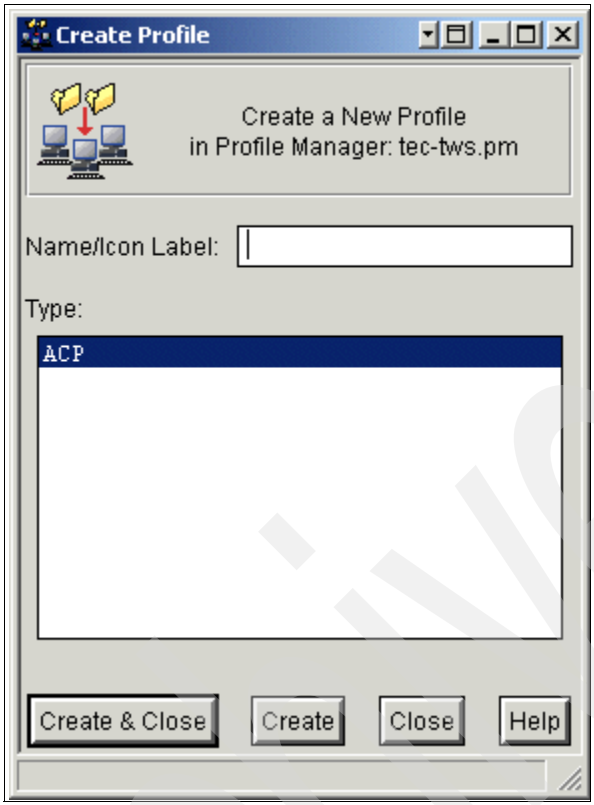

Figure 8-32 Create Profile window

12. Enter a name for the ACP profile and ensure that ACP is selected in the Type list. In our environment for this book, we used the name tec-tws.pf as shown in Figure 8-33 on page 363.

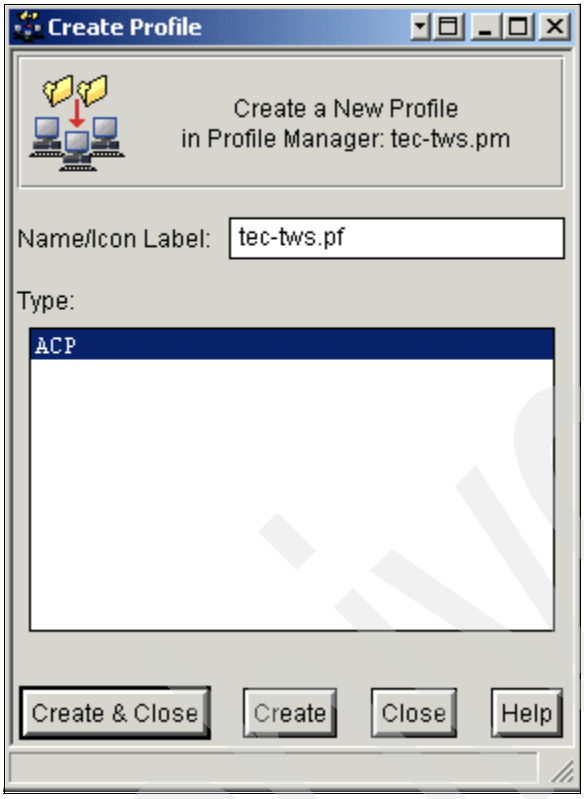

Figure 8-33 Entering new ACP profile name

13. Select **Create & Close**. This returns you to the Profile Manager window, which now displays the new ACP profile as shown in Figure 8-34 on page 364.

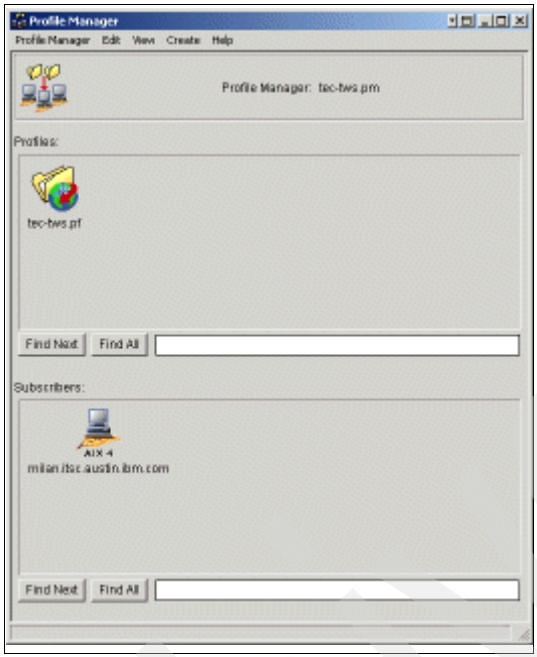

Figure 8-34 New ACP profile tec-tws.pf in Profile Manager window

14. Double-click the new ACP profile icon. The Adapter Configuration Profile window displays as shown in Figure 8-35.

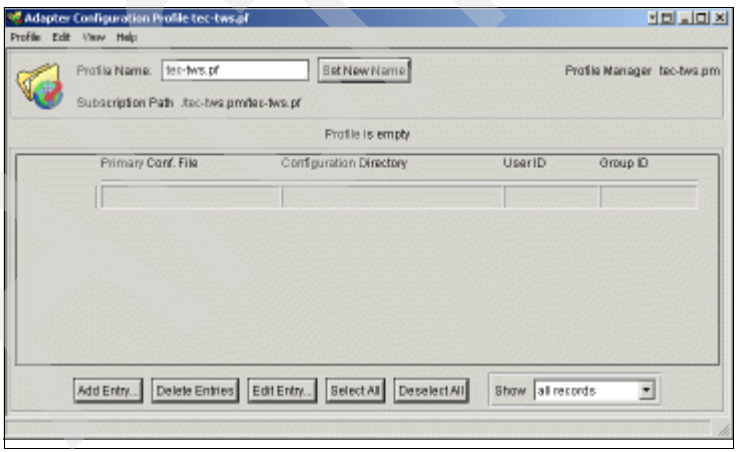

Figure 8-35 Adding a TEC logfile adapter to the Adapter Configuration Profile

15. Select **Add Entry**. The Add Adapter Configuration window displays with a list of TEC logfile adapters as shown in Figure 8-36 on page 365.

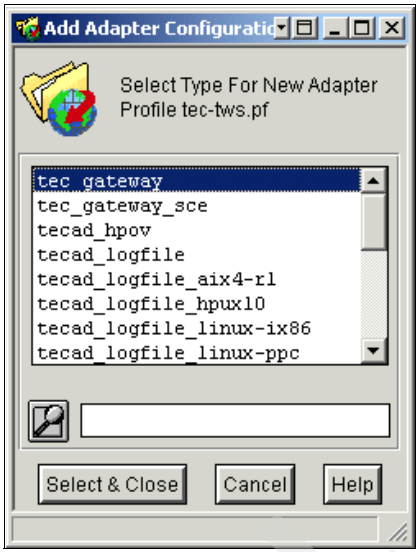

Figure 8-36 Add Adapter Configuration window

16. Select the TEC logfile adapter that corresponds to the platform of the MDM, *not* the primary TMR server or TEC server.

Attention: Do not select tecad_logfile unless you are very familiar with Tivoli Enterprise Console and prepared to configure a generic TEC logfile adapter manually.

In our environment for this book, milan was our MDM, and it was an AIX server. So, we chose the TEC logfile adapter tecad_logfile_aix4-r1 as shown in Figure 8-37 on page 366.

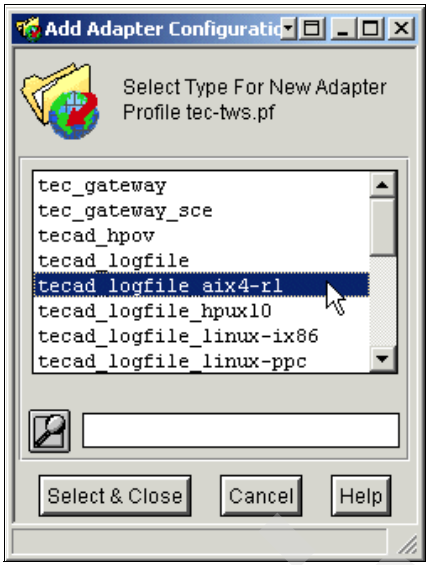

Figure 8-37 Choosing the TEC logfile adapter for an MDM running on AIX

17. Click **Select & Close**. The Edit Adapter window displays as shown in Figure 8-38.

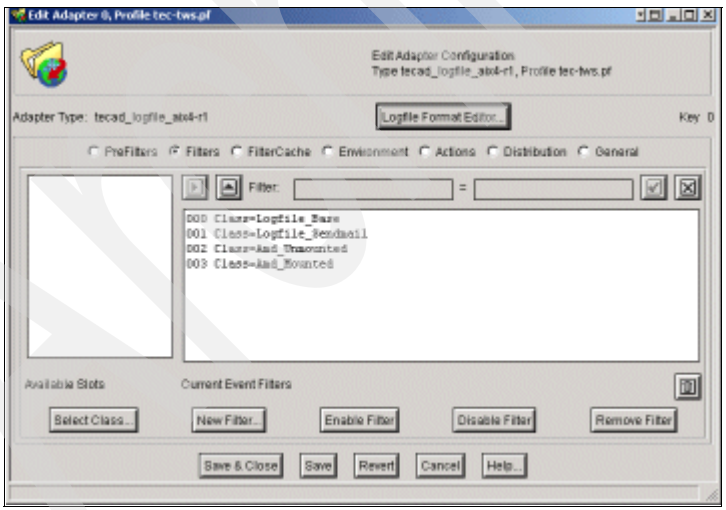

Figure 8-38 Edit Adapter window

18. Select **Save & Close**. This returns you to the Adapter Configuration Profile window with the new TEC logfile adapter displayed in the adapter list as shown in Figure 8-39 on page 367.

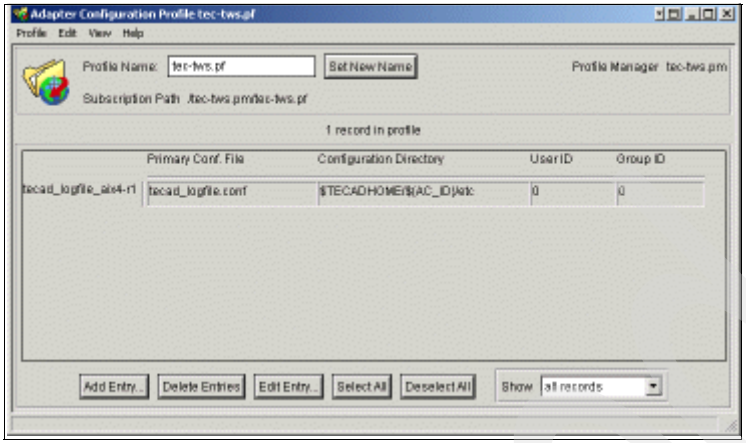

Figure 8-39 Adapter Configuration Profile with new TEC logfile adapter

19.Select **Profile** → **Distribution Defaults...** as shown in Figure 8-40.

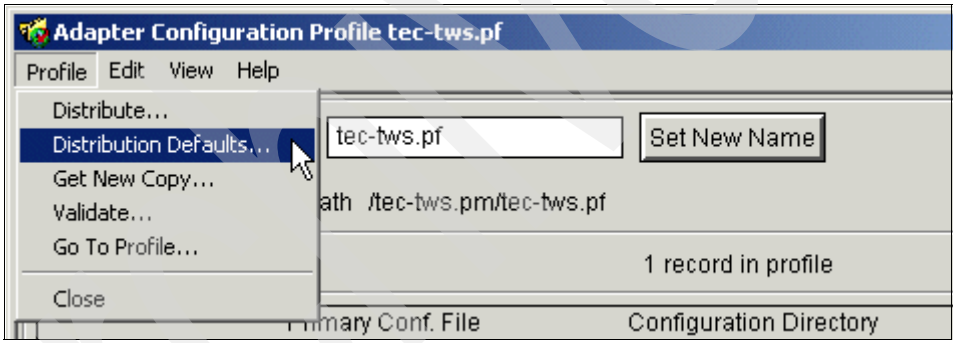

Figure 8-40 Setting distribution defaults of an ACP profile

The Set Distribution Defaults window displays as shown in Figure 8-41.

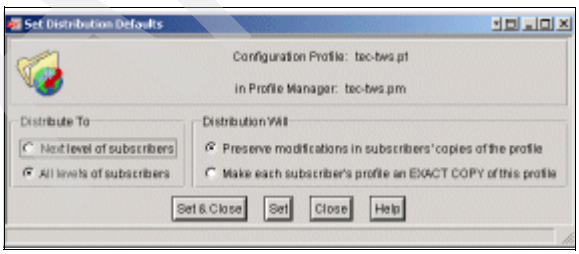

Figure 8-41 Set Distribution Defaults window

20. Ensure that *All levels of subscribers* in the Distribute To group on the left-hand side of the window is selected. Also, ensure that *Make each subscriber's profile an EXACT COPY of this profil*e in the Distribution Will group on the right-hand side of the window is selected, as shown in Figure 8-42.

Figure 8-42 Detailed view of Set Distribution Defaults window

21. Select **Set & Close** to return to the Adapter Configuration Profile window.
22. Select **Profile** → **Close** as shown in Figure 8-43.

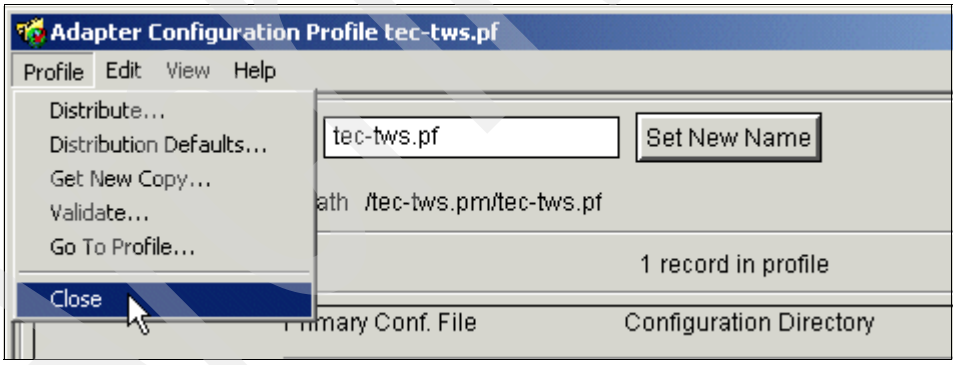

Figure 8-43 Closing the Adapter Configuration Profile window

This returns you to the Profile Manager window.

23. In the Profile Manager window, select **Profile Manager** → **Distribute...** as shown in Figure 8-44 on page 369.

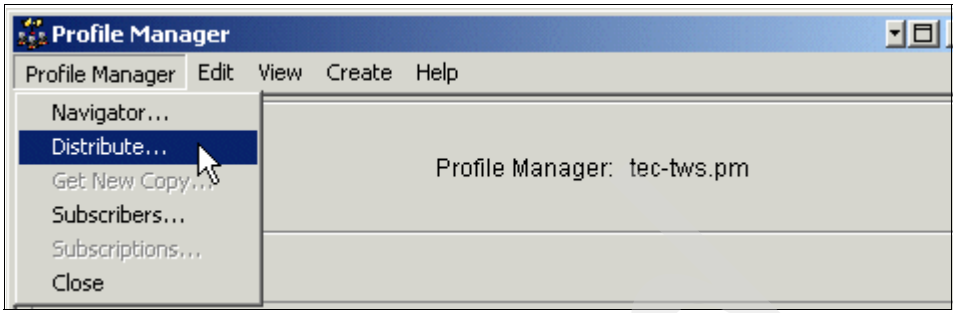

Figure 8-44 Distributing the TEC logfile adapter to the MDM

The Distribute Profile windows displays as shown in Figure 8-45.

Figure 8-45 Distribute Profiles window

24. Select **Distribute Now**. This distributes the TEC logfile adapter to the MDM using ACP.

See your Tivoli Framework administrator or IBM service provider if you need additional assistance, if you have to perform this task inside a different Policy Region, or if you use an existing profile manager or Adapter Configuration Profile.

8.8.5 Configuring the Tivoli Enterprise Console server

The Setup EventServer for TWS task in the Tivoli Management Framework allows you to configure automatically the Tivoli Enterprise Console server to manage events that come in from the scheduling environment. You run the Setup EventServer for TWS task only once, after the TWS Plus Module is installed. Whenever the module is upgraded in the future, you need to run this task again. If you have followed our recommendation to directly install the 8.2.0-TIV-TWSPLUS-FP0005 patch for TWSPlus Module for Tivoli 8.2 from TWS

Fix Pack 6 (or later, if applicable), then you only need to run this task once during your initial installation of the TWS Plus Module.

To run the Setup EventServer for TWS task:

1. Start and log into the Tivoli Desktop. Your Tivoli Framework administrator can show you how to start a Tivoli Desktop session that is appropriate for your Tivoli environment.
2. Open the TivoliPlus collection object as shown in Figure 8-46.

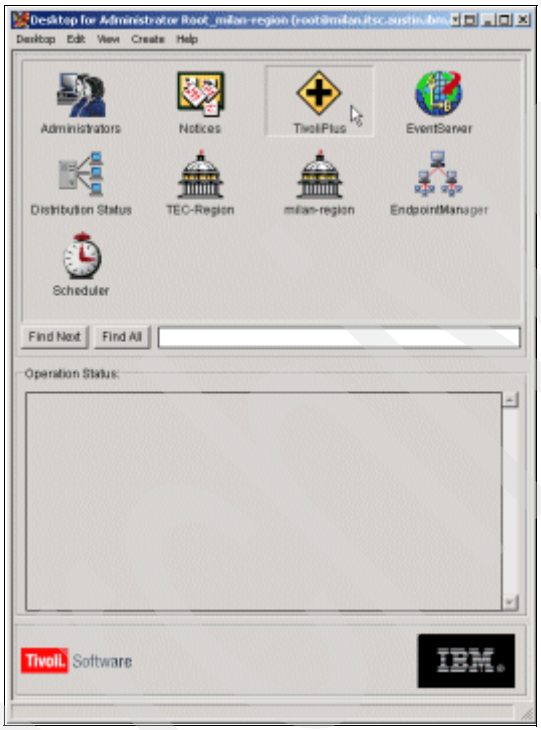

Figure 8-46 Opening the TivoliPlus collection object

The TivoliPlus window displays as shown in Figure 8-47 on page 371.

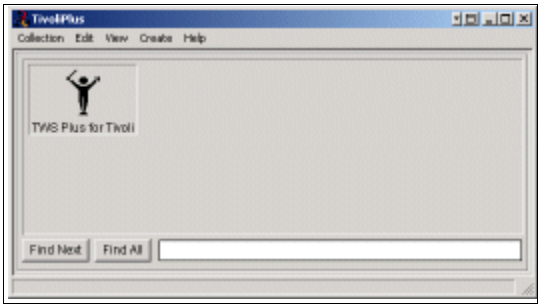

Figure 8-47 TivoliPlus window

3. Double-click the TWS Plus for Tivoli icon. The TWS Plus for Tivoli window displays as shown in Figure 8-48.

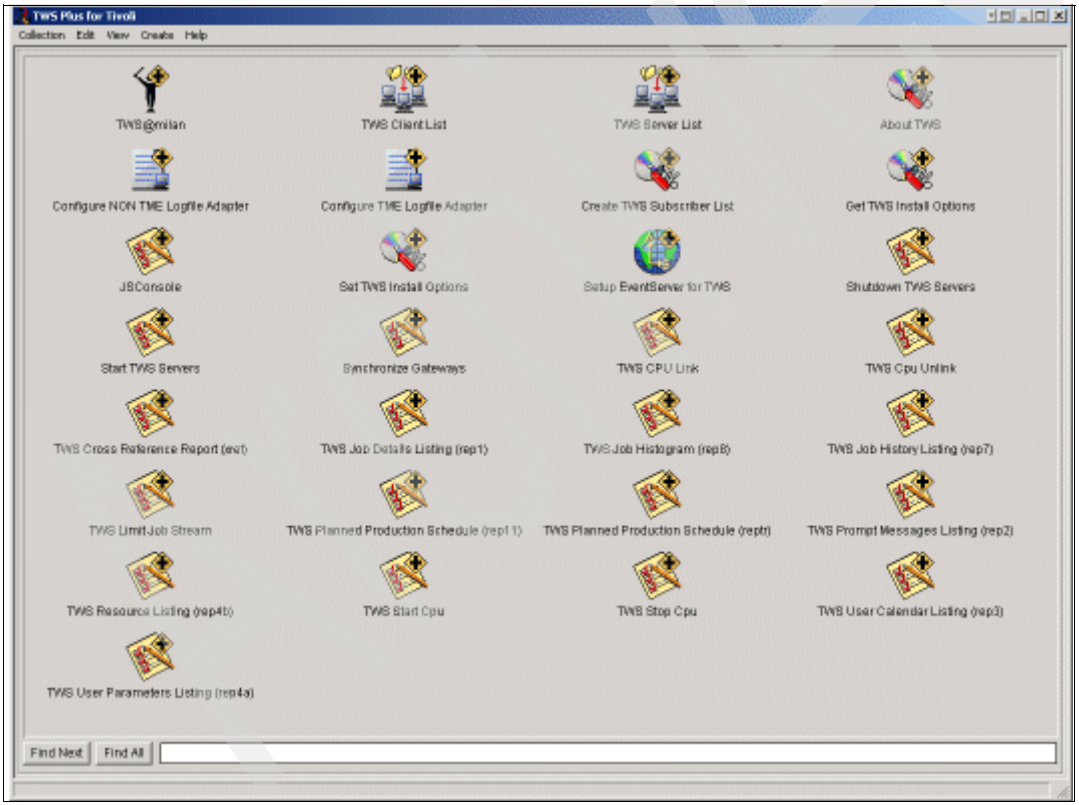

Figure 8-48 TWS Plus for Tivoli window

4. Double-click the Setup EventServer for TWS icon as shown in the Figure 8-49.

Figure 8-49 Starting Setup EventServer for TWS Framework task

The Setup_EventServer_for_TWS window opens as shown in Figure 8-50 on page 373.

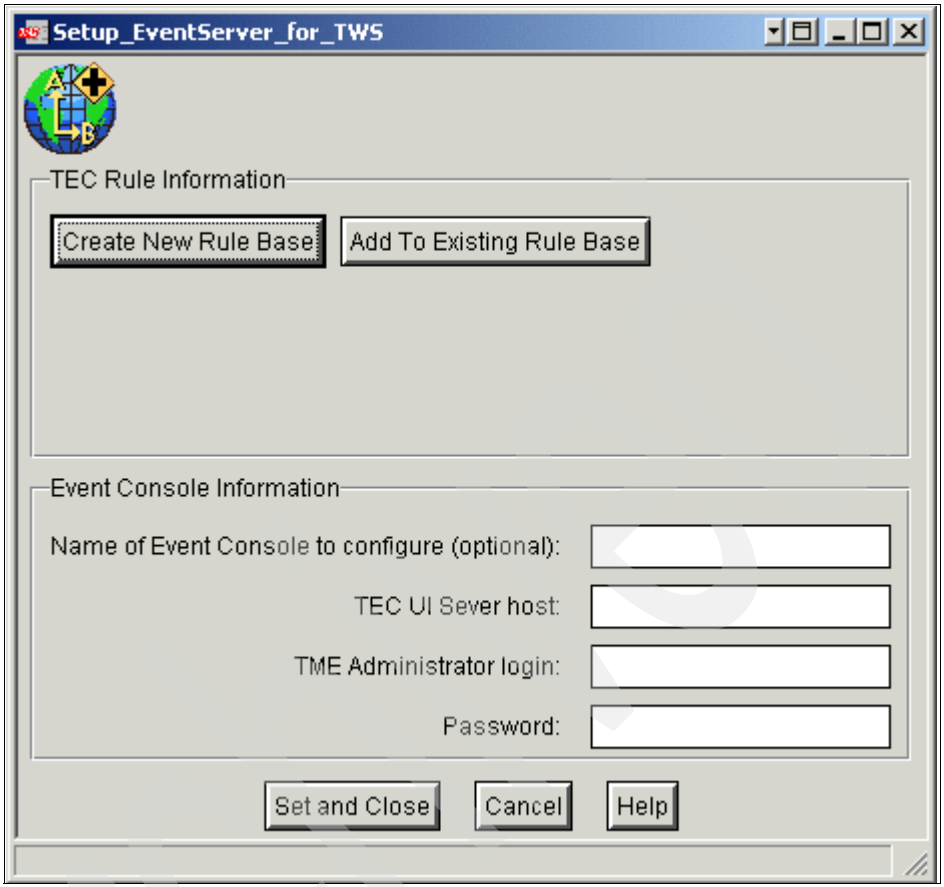

Figure 8-50 Setup_EventServer_for_TWS window

5. Select **Create New Rule Base**. New text fields with default values are created in the TEC Rule Information group as shown in Figure 8-51 on page 374.

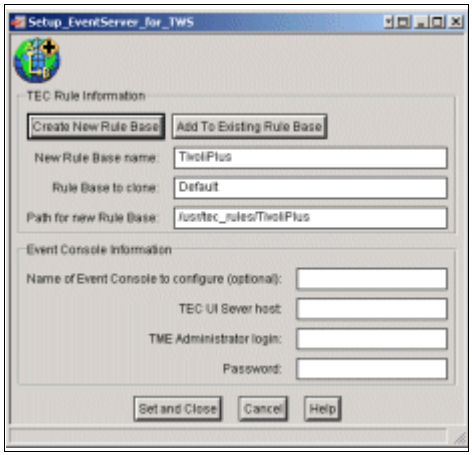

Figure 8-51 New text fields revealed after pressing the Create New Rule Base button

6. Fill in the fields in the TEC Rule Information group as follows:
   - New Rule Base Name

     Specify the name of the new rule base, one that has rule information necessary to manage events from the scheduling environment. Do *not* specify Default in this field because doing so overwrites the default rule base with the new one.
   - Rule Base to Clone

     If you are using the Default rule base, specify Default in this field.
   - Path for new Rule Base

     Specify the directory path to store the rule base on the Tivoli Enterprise Console server.

   In our environment for this book, we chose the values 200503012041 for the New Rule Base Name field, Default for the Rule Base to Clone field, and /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041 for the Path for new Rule Base field, as shown in Figure 8-52 on page 375.

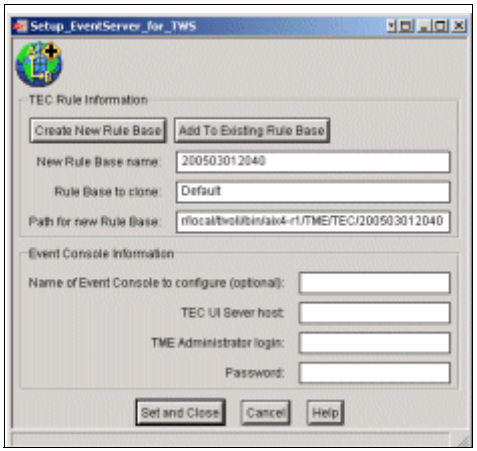

Figure 8-52 TEC Rule Information values used in environment for this book

7. Fill in the fields in the Event Console Information group as follows:
   - Name of Event Console to configure

     Specify the name of the event console where you want to display the events. This field is required to successfully run the Setup EventServer for TWS Framework job *although it is marked optional*.
   - TEC UI Server Host

     Specify the host name of the Tivoli Enterprise Console UI Server
   - TME Administrator login

     Specify the Tivoli Framework administrator user name
   - Password

     Specify the Tivoli Framework administrator password

   If you do not know the name of the event console to configure, see your Tivoli Framework administrator or IBM service provider. The **wconsole** command that is run from the TEC server under an authorized account can also be used to display a list of all event consoles for a specific user account on a host that runs the TEC UI server.

   In our lab environment for this book, we ran the **wconsole** command as root user on paris to determine the available event consoles for the root user account on paris (the TEC server) as shown in Example 8-12 on page 376.

Example 8-12 Getting a list of event consoles for a user account on a TEC UI server

```
paris:/# wconsole -lsconsole -h paris.itsc.austin.ibm.com -u root -p ******
AdministrativeConsole
twsconsole
```

In our lab environment for this book, we chose the values `twsconsole` for the Name of Event Console to configure text field, `paris.itsc.austin.ibm.com` (our TEC and TEC UI server) for the TEC UI Server Host text field, and the root user and password for the TEC Administrator login and Password text fields respectively, as shown in Figure 8-53.

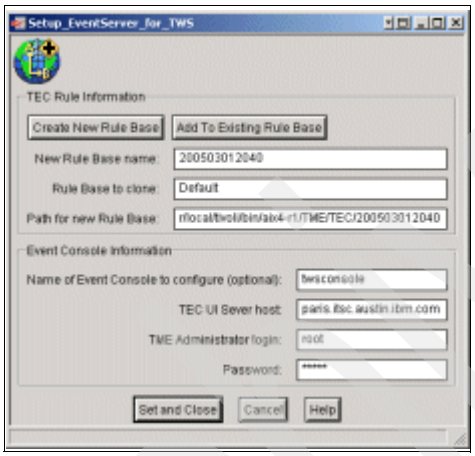

Figure 8-53 Event Console Information values used in environment for this book

8. Select **Set and Close**. The Setup_EventServer_for_TWS (TWS) Output window opens as shown in Figure 8-54 on page 377.

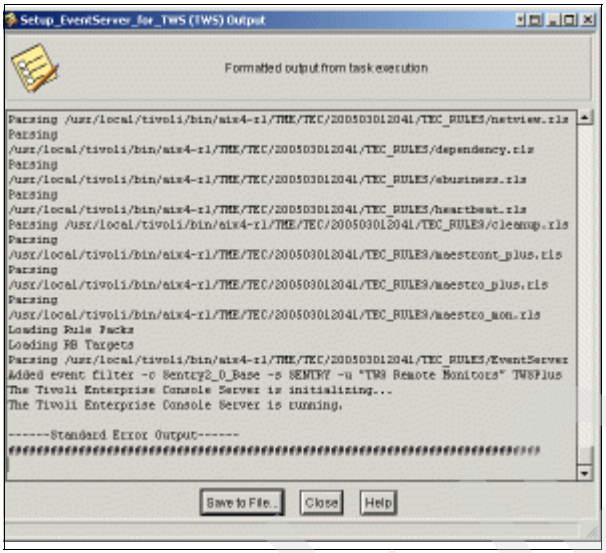

Figure 8-54 Setup_EventServer_for_TWS (TWS) Output window

It can take about 10 to 30 minutes for this task to run, depending upon a site's configuration and architecture. An excerpted sample of the output in the window is shown in Example 8-13.

Example 8-13 Sample output from Setup_EventServer_for_TWS (TWS) Output window

```
##########################################################################
Task Name:Setup_EventServer_for_TWS
Task Endpoint:paris.itsc.austin.ibm.com#milan-region (ManagedNode)
------Standard Output------
*** Executing config_tec.sh (/tmp/taskmDhWqa.BAT)
Creating new rulebase 200503012041
Loading Classes
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/root.baroc
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/tec.baroc
    .
    .
    .
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/TWS_Monitors.baroc
Loading Rule Sets
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maintenance_mode.rls
    .
    .
    .
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/TWS_Monitors.baroc
Loading Rule Sets
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maintenance_mode.rls
    .
    .
```

```
    .
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestront_plus.rls
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestro_plus.rls
Loading Classes
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/root.baroc
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/tec.baroc
    .
    .
    .
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/TWS_Monitors.baroc
Loading Rule Sets
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maintenance_mode.rls
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/netview.rls
    .
    .
    .
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestront_plus.rls
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestro_plus.rls
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestro_mon.rls
Loading Classes
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/root.baroc
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/tec.baroc
    .
    .
    .
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/maestro.baroc
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/maestront.baroc
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/universal.baroc
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_CLASSES/TWS_Monitors.baroc
Loading Rule Sets
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maintenance_mode.rls
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/netview.rls
    .
    .
    .
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestront_plus.rls
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestro_plus.rls
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/maestro_mon.rls
Loading Rule Packs
Loading RB Targets
Parsing /usr/local/tivoli/bin/aix4-r1/TME/TEC/200503012041/TEC_RULES/EventServer
Added event filter -c Sentry2_0_Base -s SENTRY -u "TWS Remote Monitors" TWSPlus
The Tivoli Enterprise Console Server is initializing...
The Tivoli Enterprise Console Server is running.

------Standard Error Output------
##########################################################################
```

9. Select **Close** to dismiss the Setup_EventServer_for_TWS (TWS) Output window.
10. Verify the new rule base exists by running the **wrb** command on the TEC server under an authorized account. In our lab environment for this book, we

ran the **wrb** command on paris under the root user account, as shown in Example 8-14.

Example 8-14 Verifying new rule base exists

```
paris:/# wrb -lsrb
Rule Base Name
--------------
Default
200503012041
```

11. Verify that the new rule base is activated by running the **wrb** command on the TEC server under an authorized account. In our lab environment for this book, we ran the **wrb** command on paris under the root user account, as shown in Example 8-15.

Example 8-15 Verifying new rule base is activated

```
paris:/# wrb -lscurrb
The currently used rule base was loaded from the rule base named
'200503012041'.
```

12. Verify that the new rule base contains event classes for TWS by running the **wrb** command on the TEC server under an authorized account. In our lab environment for this book, we ran the **wrb** command on paris under the root user account, as shown in Example 8-16.

Example 8-16 Verifying new rule base contains event classes for TWS

```
paris:/# wrb -lsrbclass 200503012041 | grep -i tws
TWS_Base
TWS_Reset
TWS_Process
TWS_Process_Reset
TWS_Process_Gone
TWS_Process_Abend
TWS_Job
   .
   .
   .
```

13. Verify that the new rule base contains rule sets for TWS by running the **wrb** command on the TEC server under an authorized account. In our lab environment for this book, we ran the **wrb** command on paris under the root user account, as shown in Example 8-17 on page 380.

Example 8-17 Verifying new rule base contains rule sets for TWS

```
paris:/# wrb -lsrbrule 200503012041 | grep -i maestro
maestront_plus.rls
maestro_plus.rls
maestro_mon.rls
```

Proceed with configuring the TEC logfile adapter that was distributed earlier to the MDM as shown in 8.8.4, “Distributing the TEC logfile adapter to MDM” on page 352 (and if you chose an integration design that integrated Tivoli Enterprise Console with each FTA, distributed earlier to all FTAs as well).

8.8.6 Configuring the TEC logfile adapter

Note: Before you begin configuring the TEC logfile adapter, note that the Adapter Configuration Facility must be installed on the same managed node as the endpoint gateway so that adapters and adapter-specific files can be distributed to the endpoints. This issue is specific to Tivoli Enterprise Console architecture and should be addressed by your Tivoli Framework administrator or IBM service provider.

The TWS Plus Module provides two tasks that allow configuration of the TEC logfile adapter, depending upon how the TEC logfile adapter was distributed. This book shows how to use the Configure TME adapter task for the TME TEC logfile adapter distributed as shown in 8.8.4, “Distributing the TEC logfile adapter to MDM” on page 352.

This book does not show how to implement non-TME adapter based integration, because that approach is for environments where the Framework endpoint software cannot be installed on the MDM. We do not recommend this architecture because non-TME adapters are unable to take advantage of additional features such as encryption that TME adapters use by taking advantage of the Framework services that the endpoint software delivers.

Using an MDM without the endpoint software also significantly reduces the opportunities to integrate other IBM Tivoli products with TWS. The Configure non-TME Adapter task performs the configuration of non-TME TEC logfile adapters. For more information about configuring a non-TME TEC logfile adapter, see *IBM Tivoli Workload Scheduler Plus Module User’s Guide Version 8.2*, SC32-1276.

To run the Configure TME logfile adapter task:

1. Start and log into the Tivoli Desktop. Your Tivoli Framework administrator can show you how to start a Tivoli Desktop session that is appropriate for your Tivoli environment.
2. Open the TivoliPlus collection object as shown in Figure 8-55.

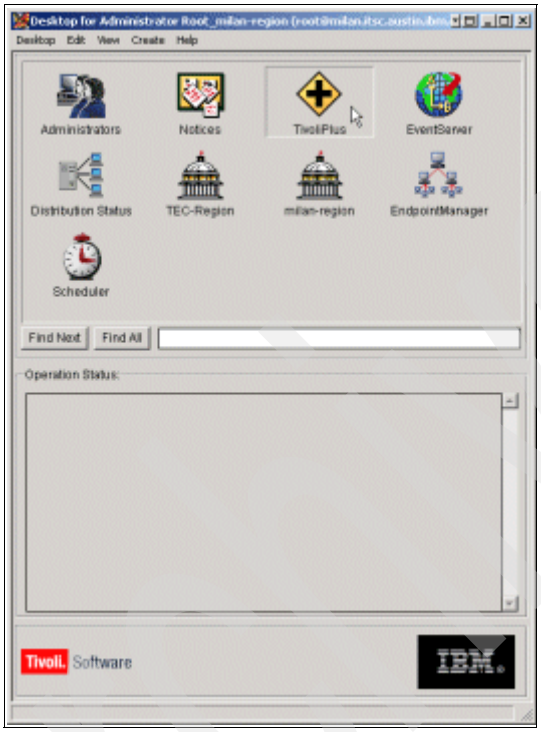

Figure 8-55 Opening the TivoliPlus collection object

The TivoliPlus window displays as shown in Figure 8-56.

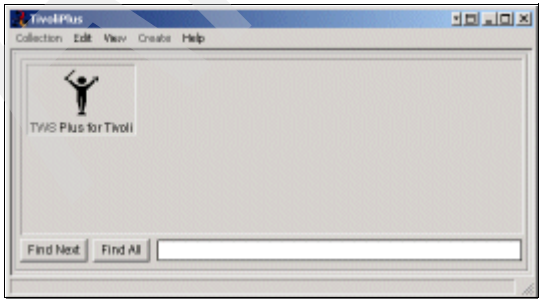

Figure 8-56 TivoliPlus window

3. Double-click the TWS Plus for Tivoli icon. The TWS Plus for Tivoli window displays as shown in Figure 8-58 on page 383.

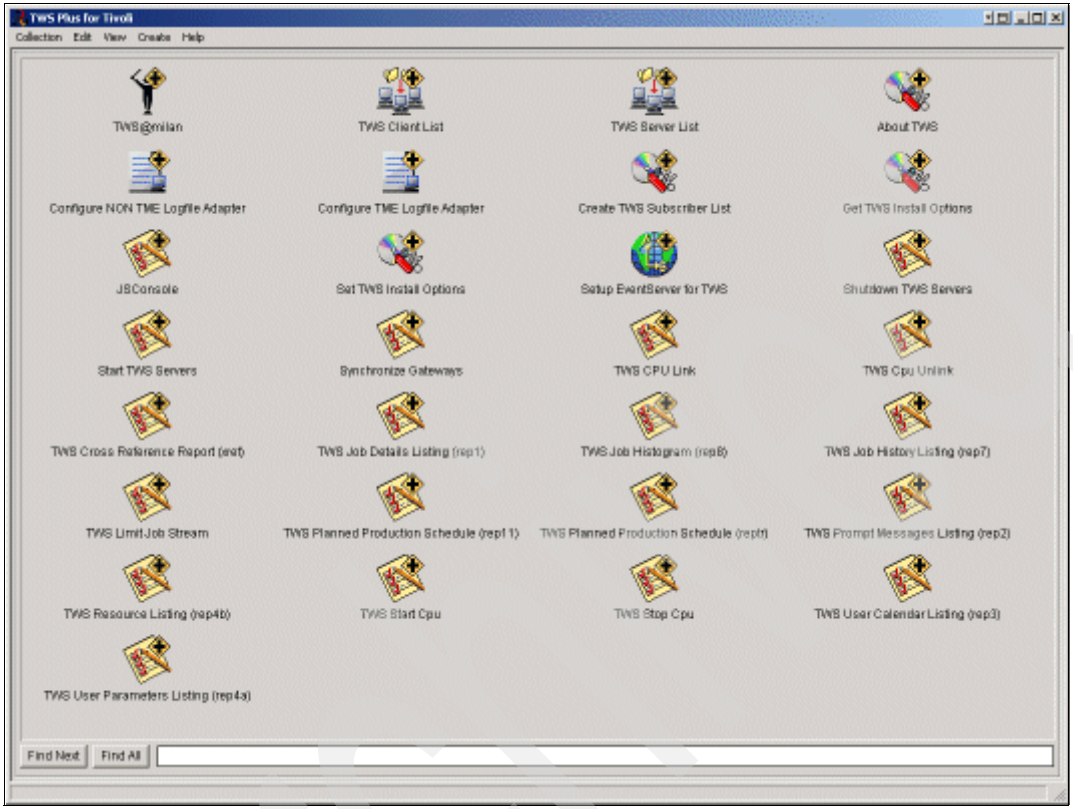

Figure 8-57 TWS Plus for Tivoli window

4. Double-click the Configure TME logfile adapter icon as shown in Figure 8-58 on page 383.

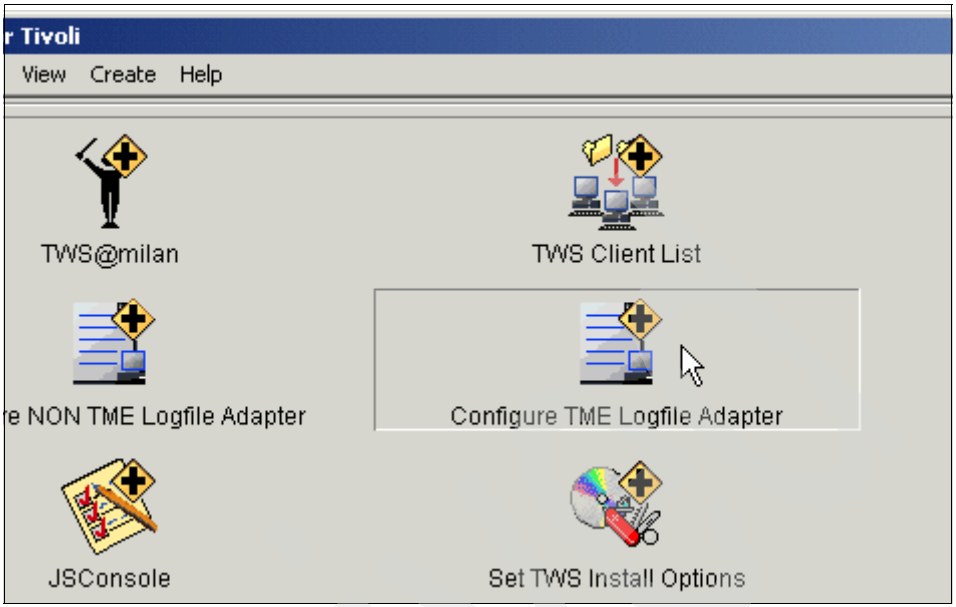

Figure 8-58 Starting Configure TME logfile adapter Framework task

The Configure_TME_Logfile_Adapter (TWS) Output window opens as shown in Figure 8-59.

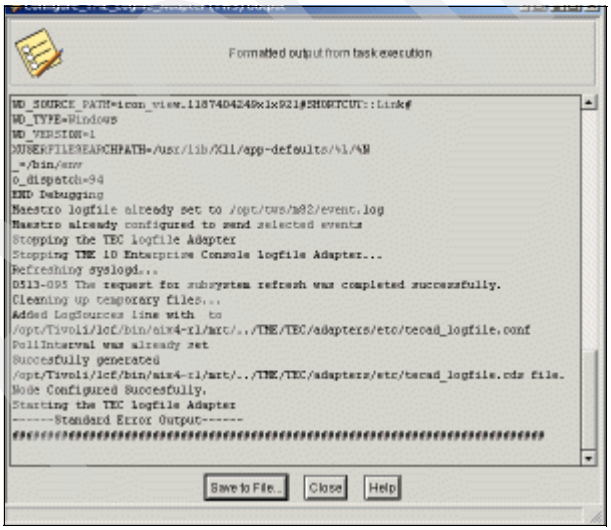

Figure 8-59 Configure_TME_Logfile_Adapter (TWS) Output window

The output of the task is similar to Example 8-18 which was taken from our environment for this book.

Example 8-18 Sample text in Configure_TME_Logfile_Adapter (TWS) Output window

```
################################################################################
Task Name:Configure_TME_Logfile_Adapter
Task Endpoint:milan (ManagedNode)
------Standard Output------
Maestro logfile already set to /opt/tws/m82/event.log
Maestro already configured to send selected events
Stopping the TEC Logfile Adapter
Stopping TME 10 Enterprise Console Logfile Adapter...
Refreshing syslogd...
0513-095 The request for subsystem refresh was completed successfully.
Cleaning up temporary files...
Added LogSources line with  to
/opt/Tivoli/lcf/bin/aix4-r1/mrt/../TME/TEC/adapters/etc/tecad_logfile.conf
PollInterval was already set
Succesfully generated /opt/Tivoli/lcf/bin/aix4-r1/mrt/../TME/TEC/adapters/etc/tecad_logfile.cds
file.
Node Configured Succesfully.
Starting the TEC Logfile Adapter
------Standard Error Output------
################################################################################
```

Tip: If you want to verify the configuration of the TEC logfile adapter afterwards, save the output of the task to a file for later reference by selecting **Save to File**.

5. When the task has finished running, select **Close** to close the Configure_TME_Logfile_Adapter (TWS) Output window.

   If you chose to integrate Tivoli Enterprise Console with just the MDM (the most common design choice), the integration is done and you can proceed with verifying the integration. If you chose to integrate Tivoli Enterprise Console with each FTA in a scheduling network, proceed with the remaining steps.

6. Right-click the Configure TME® logfile adapter icon as shown in Figure 8-60 on page 385.

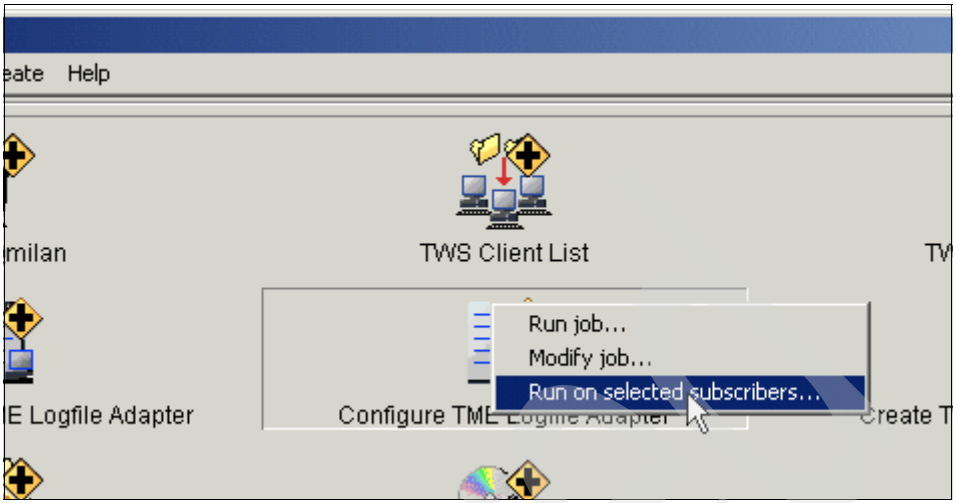

Figure 8-60 Configure TME logfile adapter on additional FTAs

The Execute Task window opens as shown in Figure 8-61.

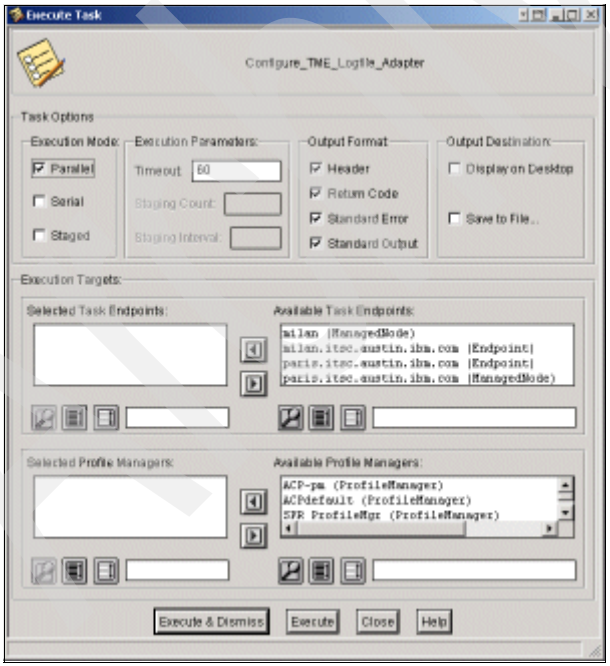

Figure 8-61 Execute Task window for Configure TME logfile adapter Framework task

7. In the Output Destination group, select **Display on Desktop**.

8. Select the endpoint instances of the FTAs from the Available Task Endpoints list and move them into the Selected Task Endpoints list. The Execute Task window will look similar to Figure 8-62.

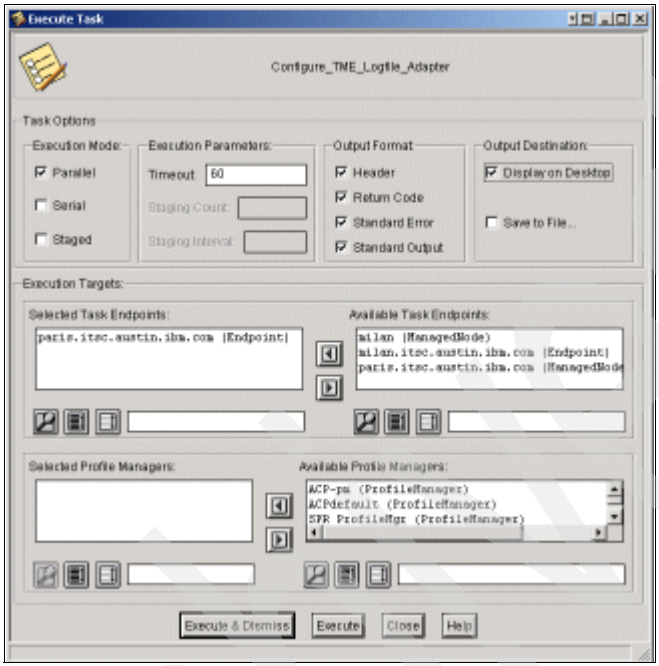

Figure 8-62 Execute Task window after selecting output destination and task endpoints

9. Select **Execute & Dismiss**. The Execute Task window closes, and the Configure_TME_Logfile_Adapter (TWS) Output window opens.

You next need to verify the logfile adapter configuration by either examining the TEC console for events from TWS after running a job or unlinking an FTA or directly inspecting the TEC logfile adapter configuration files.

To inspect the TEC logfile adapter configuration files:

1. Look in the output of the Configure TME logfile adapter Framework task for the location of the CDS file. In our environment for this book, using the output shown in Example 8-18 on page 384, the file is at:

   /opt/Tivoli/lcf/bin/aix4-r1/mrt/../TME/TEC/adapters/etc/tecad_logfile.cds.

2. Search for the string `TWS` in the file. On Windows MDMs, we recommend using the **`findstr`** command. On UNIX MDMs, use the **`grep`** command as shown in Example 8-19 on page 387, which is taken from our environment for this book and is run as root user on the MDM server (milan).

Example 8-19 Verifying TEC logfile adapter configuration for TWS entries.

```
milan:/#  grep TWS \
/opt/Tivoli/lcf/bin/aix4-r1/mrt/../TME/TEC/adapters/etc/tecad_logfile.cds
CLASS TWS_Reset
    msg = PRINTF("TWS has been reset on host %s", $V2);
CLASS TWS_Process_Reset
    msg = PRINTF("TWS process %s has been reset on host %s", $V5, $V2);
CLASS TWS_Process_Gone
    msg = PRINTF("TWS process %s is gone on host %s", $V5, $V2);
CLASS TWS_Process_Abend
    msg = PRINTF("TWS process %s has abended on host %s", $V5, $V2);
CLASS TWS_Job_Abend
CLASS TWS_Job_Abend
    .
    .
    .
```

About 150 lines of output should be returned. If there are substantially more or less lines of output, see your Tivoli Framework administrator or IBM service provider.

The adapter configuration commands set up the file BmEvents.conf in the *TWSHome* home directory. This configuration file determines what information the production processes (batchman and mailman) writes in the *TWSHome*/log_source_file file and how the information is written. By default this file information is written to the event.log.

You can change the name of the log file by modifying:

- The FILE field in the BmEvents.conf file and restart the TWS processes.
- The LogSource field in the tecad_logfile.conf file and restart the TEC logfile adapter.

We do not recommend changing these fields without advice from your Tivoli Framework administrator or IBM service provider. For more information about the fields in the BmEvents.conf file, refer to Appendix A of *IBM Tivoli Workload Scheduler Plus Module User's Guide Version 8.2*, SC32-1276.

The integration between TWS and Tivoli Enterprise Console is complete.

8.9 Using the integration

When the integration is completed, there are many ways to use it. This section discusses examples that use our environment for this book.

The simplest and most direct usage is simply viewing the TWS events in the TEC console. Your Tivoli administrator can show you how to start the TEC console for your environment. Typically, the TEC console is started either from a UNIX server with the **tec_console** command and is run across the X Window System or from a Windows desktop client.

In our environment for this book, we used the Windows desktop client installed from Tivoli Enterprise Console Fix Pack 2. We used the installer in the COMPONENTS/COMPONENTS/NON_TME/CONSOLE/W32-IX86/ subdirectory, expanded from the 3.9.0-TEC-FP02-NON_TME-W32-IX86.tar archive of Fix Pack 2. We logged into the TEC server paris (because it ran the TEC UI server), as shown in Figure 8-63.

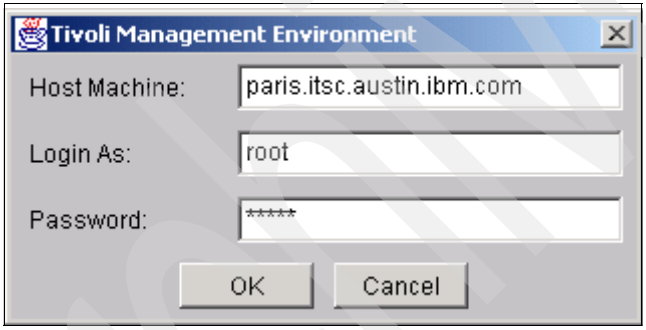

Figure 8-63 Logging into the TEC console

After logging in, our environment for this book was configured to show the event console view as shown in Figure 8-64 on page 389. Note the number of events that is displayed in the small gray box if the mouse pointer hovers over a set of events grouped by severity.

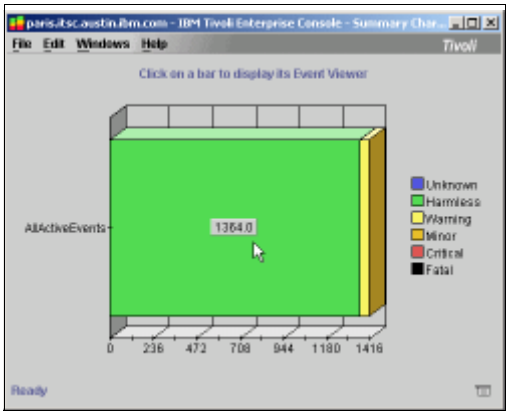

Figure 8-64 Initial Event Viewer display

Clicking any of the bar chart segments displays a detailed event viewer similar to that shown in Figure 8-65. The exact view depends upon how your Tivoli administrator configured the event consoles for Tivoli Event Console.

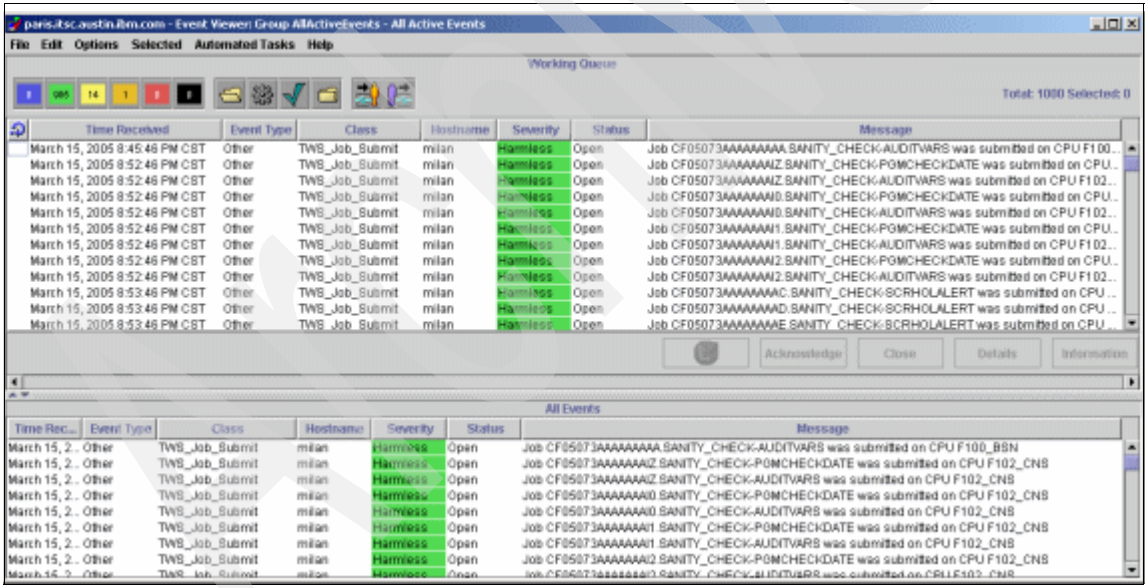

Figure 8-65 Detailed event viewer

Note that the event class column indicates that the events displayed in Figure 8-65 are TWS events. Any of these events can have a number of actions directed upon them such as acknowledge, close, open trouble ticket, start a Framework task, and so forth. For more details on how to use the TEC console,

see *IBM Tivoli Enterprise Console User's Guide Version 3.9*, SC32-1235. Your Tivoli administrator or IBM service provider can also show you how to use the TEC console's event viewer.

A less visible but more powerful application of the integration is to work with a TEC specialist to develop rules for your environment. This would let you define a rule, for example, that senses when an event indicates that disk space has run out and the failure of a job that is known to use that disk space. The rule would then correlate the two events together into the root cause (disk space exhausted). The rule can then acknowledge both events automatically, issue a trouble ticket for the disk space condition, and wait for an event signaling that more disk space has been freed up. When more disk space has been freed up, the rule can close the trouble ticket automatically and close the original two events. Finally, the rule can issue a **`job restart`** command to TWS, assuming that the job's owner has already designed it to be arbitrarily restartable.

Implementing this type of rule turns what is ordinarily a time- and labor-intensive job recovery due to a disk space outage into a completely automated solution that only requires manual intervention to confirm the allocation of additional disk space. This rule is a simple example of a defensive, reactive rule.

You can create rules for virtually any business requirement of the scheduling environment to proactively protect the scheduling environment. For example, suppose an event is received that a mainframe which has several jobs that depend upon file downloads is suddenly unreachable over the network. You can create a rule to compute automatically how much time is left before the first job that depends upon the mainframe is due to launch. Then it can issue an e-mail that alerts administrators that they have that many minutes left to address the network connectivity problem before the production plan is adversely impacted.

See your Tivoli administrator or IBM service provider for assistance in writing rules, or refer to *IBM Tivoli Enterprise Console Rule Developer's Guide Version 3.9*, SC32-1234 for rule programming information and *IBM Tivoli Enterprise Console Rule Set Reference*, SC32-1282 for documentation on pre-built rules that are installed with Tivoli Enterprise Console.

8.10 Possible enhancements

This section shows the power of integrating across other Tivoli products with TWS, using the integration with Tivoli Enterprise Console as a starting point. These are simply ideas that can help you identify other needs in your environment that can be addressed by deploying Tivoli solutions.

When Tivoli Enterprise Console is integrated with TWS, the scheduling network can be proactively monitored as discussed in 8.9, “Using the integration” on page 388. However, the possibility exists of also proactively provisioning new FTAs as well. In large scheduling environments with more than 200 FTAs under federated control of departmental level IT organizations geographically spread around the world, this solution is a significant time saver.

One way to provision FTAs automatically is to deploy Tivoli Intelligent Orchestrator. This method enables provisioning all the way from the “bare metal” level (that is, configuring the server without even starting from a formatted hard disk or installed operating system). Another method uses Tivoli NetView and Tivoli Configuration Manager. We examine the latter method in Figure 8-66 on page 392, because it is significantly easier for most organizations to adopt and deploy.

The IBM Redbook *Implementing Automated Inventory Scanning and Software Distribution After Auto Discovery*, SG24-6626, shows how to use Tivoli NetView and Tivoli Configuration Manager to detect new servers, install endpoint code, and perform inventory scans and required software distributions automatically. The completed solution would have an architecture similar to that shown in Figure 8-66 on page 392.

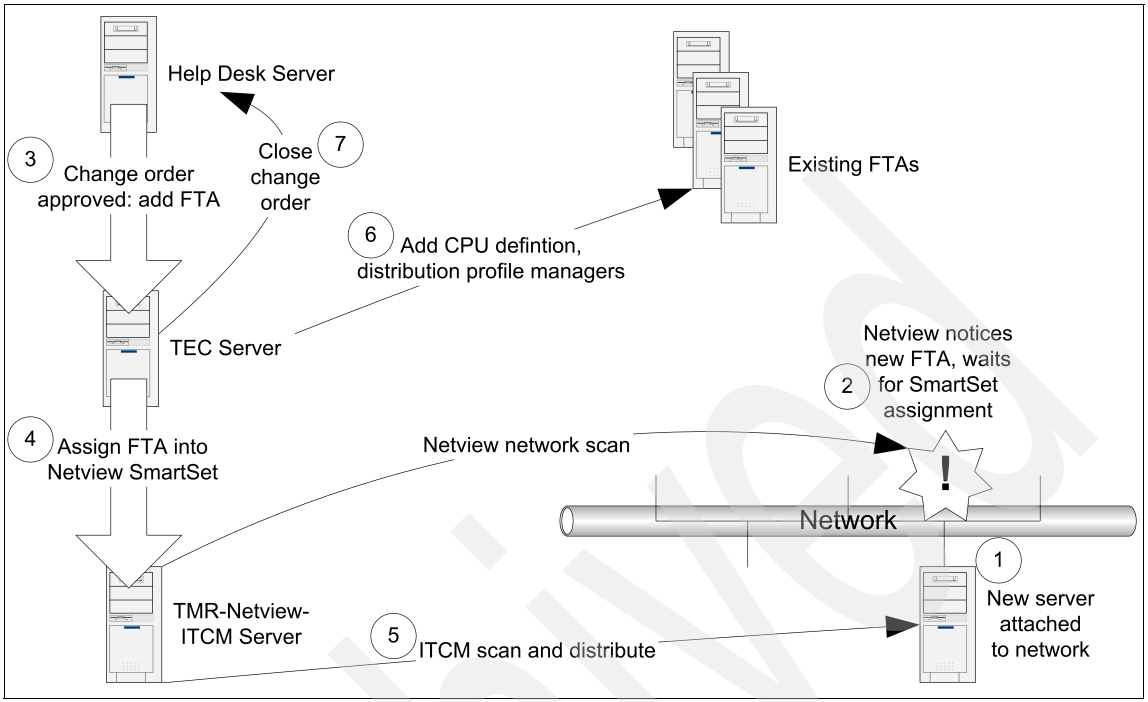

Figure 8-66 Architecture of a possible multi-product enhancement to TEC and TWS integration

Whenever you need to add a new FTA to a scheduling network, you can attach the physical server to the network and then approve a change order ticket. All other tasks are handled automatically as described in the following list:

1. The new server is built and attached to the network.
2. During its periodic network scans, NetView notices the new server and creates an object for the new server. An event is sent to Tivoli Enterprise Console notifying it that a new server has been detected in the network. NetView takes no other immediate action. Instead it waits for the new server to be assigned into a SmartSet.
3. A manager approves the change order ticket in the help desk system to add the new FTA.
4. The help desk server sends an event to the TEC server notifying it to add a new FTA. The TEC server observes that an event has been received indicating that the new FTA has already been attached to the network. If the change order was executed before the new server was attached to the network instead, then the TEC server would merely wait until NetView sends the event indicating the new server is attached to the network. The TEC

server executes a Framework task on the NetView server adding the new server to the NetView SmartSet for all FTAs.

5. NetView directs the TMR server to install the endpoint software on the new server. It then directs Tivoli Configuration Manager to perform an inventory scan of the new server. The results of the scan are used to determine whether or not to install the FTA software on the new server.
6. NetView sends an event to Tivoli Enterprise Console indicating that a new FTA has been created. Tivoli Enterprise Console adds a corresponding new processor definition to the MDM, and adds the endpoint instance of the FTA to appropriate profile managers.

A similar set of actions can completely manage the decommissioning of an FTA as well.

The end result is that whenever a server is added or deleted from the scheduling network, it is installed automatically as an FTA or uninstalled based upon business workflow rules. The major benefit is that FTAs can be commissioned and decommissioned by business analysts without interaction from technical staff except for attaching the server to the network.

If Tivoli Monitoring was added to the solution mix, you can also monitor the FTA automatically for the appropriate scheduler-related resources or uninstall monitoring from the FTA as needed. Provisioning a new server to join the scheduling network then becomes as easy as approving a change order ticket.

The combinations of solutions are as variable as the business issues that need solving. We encourage you to refer the specific business issues that your organization faces in its scheduling network to your Tivoli administrator or IBM service provider to determine how a solution can be created to address it.

9

Integrating Tivoli Monitoring

This chapter describes the integration between Tivoli Workload Scheduler 8.2 and Tivoli Monitoring 5.1.2. This integration creates an integrated scheduling solution that provides monitoring for essential system resources to detect bottlenecks and potential issues and to recover automatically from critical situations.

This chapter discusses the benefits of integration and provides definitions of Tivoli Monitoring terminology. It also provides details about the integration and configuration process as well as examples of possible ways to use the integration.

If you are primarily a Tivoli Workload Scheduler (TWS) administrator who is responsible for implementing Tivoli Management Framework and Tivoli Monitoring, you should thoroughly read the documentation for those products. See “Related publications” on page 691 for a list of documentation and the corresponding Web addresses.

9.1 Benefits of integration

Tivoli Monitoring 5.1.2 is a Tivoli application that applies pre-configured best practices to the automated monitoring of essential system resources. The application detects bottlenecks and other potential problems and provides for the automatic recovery from critical situations, eliminating the need for system administrators to scan manually through extensive performance data. This application was previously called Tivoli Distributed Monitoring (Advanced Edition).

Integration with Tivoli Workload Scheduler (TWS) allows you to monitor system resources that impact TWS, such as disk space. More powerfully, Tivoli Monitoring and its related add-ons, such as Tivoli Monitoring for Databases, can monitor system resources on which jobs depend. For example, if a job extracts rows from a database, Tivoli Monitoring and Tivoli Monitoring for Databases (or just Tivoli Monitoring with custom resource models written in-house by a site) can identify whether the database is available before the job is launched. If the database becomes unavailable at any time, an event is sent to alert operators that the job will fail because the database is unavailable.

9.2 Tivoli Monitoring terminology

Before discussing the entire integration environment and the use of Tivoli Monitoring, we need to cover some basic terminology. Some of these descriptions assume a basic familiarity with the Tivoli Management Framework architecture. Refer to the list of resources on page “Main components of Tivoli Monitoring” on page 398 for detailed documentation of Tivoli Management Framework. The following list displays the main concepts of Tivoli Monitoring relevant to an integration with TWS:

Resource models Tivoli Monitoring uses out-of-the box, predefined *resource models* to specify which resource data are accessed from the system at runtime and how this data is processed. For example, the Process resource model obtains data that is related to processes running on the system. Performance data is collected automatically by the resource model and processed by an appropriate algorithm to determine whether the system is performing to your expectations.

Cycles When a resource model is run at an endpoint, it gathers data at regular intervals, known as *cycles*. The duration of a cycle is the *cycle time*. A resource model with a cycle time of 60 seconds gathers information every 60 seconds. The data collected is a snapshot of the status of the

resources that are specified in the resource model. Each of the supplied resource models has a default cycle time, which you can modify as required.

Thresholds Each resource model defines one or more *thresholds*. A threshold is a named property of the resource with a default value that you can modify in the customization phase. Typically, the value that is specified for a threshold represents a significant reference level of a performance-related entity, which a system administrator might want to know if exceeded or not reached. However, some thresholds are used as reference values to limit the scope of the resource model.

Parameters Some resource models have one or more *parameters*. Each parameter can take the form of a list of strings, a list of numeric values, a Boolean list of predetermined values from which you can make any combination of selections, or a choice list of mutually exclusive alternatives. For example, the Windows Parametric TCP/IP Ports resource model has a parameter where you list the monitored ports and another to choose the port states to monitor.

Indication Each resource model generates an *indication* if certain conditions that are implied by the resource model's thresholds are not satisfied in a given cycle. Each resource model has its own algorithm to determine which combinations of thresholds should generate an indication.

Occurrence An *occurrence* refers to a cycle during which an indication occurs for a given resource model.

Hole A *hole* refers to a cycle during which an indication does not occur for a given resource model. In other words, none of the conditions that are specified for the generation of any indication have been met. This does not mean that none of the thresholds have been exceeded. For example, in the Windows Logical Disk resource model a High Read Bytes per Second indication is not created when the percentage disk time is higher than the High Percent Usage threshold, provided that the Low Disk Space threshold is exceeded.

Event An *event* is used to verify the persistence of a given indication by eliminating unrepresentative peaks and troughs for the indication. For example, a process that generates the ProcessHighCPU indication in one cycle might be behaving perfectly normally and is no threat to

other processes if the high usage is not repeated. However, an indication that persists over several cycles is more likely to be an issue. Thus, an event defines the number of consecutive occurrences of the indication that are significant.

However, given that you have decided that a certain number of consecutive cycles of Process High CPC greater than the chosen threshold value is significant, you might feel that if, during the accumulation of the consecutive occurrences, one or two cycles fall below the threshold, it should not stop the counting of consecutive occurrences. Thus, an event allows you to define how many consecutive holes in the sequence of consecutive occurrences are permitted.

So, an event is an aggregation of a defined number of consecutive occurrences during which there can be a defined number of consecutive holes.

Recovery actions *Recovery actions* can be run automatically for any event. Recovery actions can be of two types: built-in actions and Tivoli Framework Tasks. The recovery actions can take positive steps to remedy the situation or can ensure that information about the event is distributed to the appropriate authorities or entities.

Profiles Tivoli Monitoring is a profile-based application that runs in a Tivoli environment. Different *profiles* can be defined that contain different selections of resource models. You can modify all aspects of existing profiles, including the addition, deletion, and customization of resource models. You can distribute multiple profiles to each endpoint.

These are only some of the key high-level concepts. We recommend that you refer to *IBM Tivoli Monitoring User's Guide Version 5.1.2*, SH19-4569, for a detailed discussion of these and other concepts that are not discussed in this book.

Main components of Tivoli Monitoring

The following list describes the main components of Tivoli Monitoring that are relevant to an integration with TWS:

- Tivoli Monitoring Base

  This product is installed on the Tivoli Management Region (TMR) server and on all gateways (a Managed Node configured as an endpoint gateway) to which are attached endpoints that you want to monitor.

The component comprises a graphical user interface and a command line interface that are available at both server and gateway, and all functions of the product can be controlled from either node. However, the database of available default resource models is maintained at the server, and commands issued to manage resource models from gateways are routed to and performed on the server.

In addition, the component can be configured to operate the heartbeat function for all endpoints directly attached to the system on which it is installed.

- Web Health Console

  The Web Health Console is the Web-based graphical interface for Tivoli Monitoring that runs on any system that can be connected through TCP/IP to the TMR. The Web Health Console allows you to view real time information about a specific problem, and check the status (or health) of a set of endpoints. You can use the Web Health Console to work with real-time data or with historical data that has previously been logged to a local database.

- Endpoint component

  The endpoint component (which requires a Tivoli management agent) performs the resource management through one or more resource models that are distributed to the endpoint with a Tivoli Monitoring profile. The endpoint component is installed automatically when a Tivoli Monitoring profile is distributed to the endpoint for the first time.

9.3 Solution components

The following related IBM products should be installed and configured along with Tivoli Monitoring. These products are essential parts of the integration:

- Tivoli Management Framework 4.1.1

  Tivoli Management Framework is a prerequisite for Tivoli Monitoring. It supplies the foundation necessary for Tivoli Monitoring to operate.

- WebSphere Application Server 4.0

  WebSphere is used by Tivoli Monitoring to present a Web interface.

This chapter takes a stock installation of Tivoli Monitoring and integrates it with TWS. For details about installing and configuring Tivoli Monitoring, see your Tivoli Framework administrator or IBM service provider.

9.4 Tivoli Monitoring configuration

Before we proceeded with the integration, we ensured that the following configuration requirements for Tivoli Monitoring were met in our lab for this book:

- Tivoli Management Framework 4.1.1 with at least Fix Pack 2 installed. The fix pack is available for download from:

  ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.1.1/4.1.1-TMF-FP02/

- Tivoli Monitoring 5.1.2 with at least Fix Pack 3 installed. The fix pack is available for download from:

  ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_5.1.2/5.1.2-ITM-FP03/

- Java Runtime Environment (JRE) from Tivoli Monitoring Tools 5.1.2 installed on all Fault Tolerant Agents under /usr/java131-itm.

- Tivoli Monitoring Web Health Console 5.1.1 (there is no version 5.1.2 for this component) with at least Fix Pack 3 installed. The fix pack is available for download from:

  ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_5.1.2/5.1.2-ITM-FP04/

We recommend using Tivoli Monitoring for all processors in a scheduling network where possible and extending beyond the scheduling network where appropriate. While we only show integration with Fault Tolerant Agents in this book, in a production environment, we recommend that you deploy Tivoli Monitoring on all systems that impact the production schedule, even if they are not part of the scheduling network itself.

For example, if a server is not part of the scheduling network (not a processor), but it uploads a file to an FTA every day, monitoring that server for system outage conditions (like a full disk on the partition where the file is originally created) would enable advance warning that the production plan might be adversely affected. Refer to your Tivoli Framework administrator or IBM service provider for assistance in planning and deploying to servers that are not part of your scheduling network.

9.5 TWS configuration

The Master Domain Manager (MDM) of each scheduling network that requires integration with Tivoli Monitoring must be installed as both a Managed Node and an endpoint in Tivoli Management Framework before performing the integration. See your site's Tivoli Framework administrator or your IBM service provider for details on how to accomplish this prerequisite.

Attention: Do not skip creating endpoints. Even if Managed Nodes already exist on the servers, endpoints are mandatory for the integration to succeed.

There are other benefits available by installing endpoints in Tivoli Management Framework on all Fault Tolerant Agents and Domain Manager in the scheduling network. These benefits are not directly related to the integration with Tivoli Monitoring. But it is helpful to create these endpoints at the same time as the endpoint for Tivoli Management Framework is installed. In our lab for this book we installed and configured Tivoli Workload Scheduler 8.2 at the Fix Pack 6 level. We recommend that you apply the latest available fix pack that is applicable for your site.

9.6 Server architecture

In our lab, the following machines were relevant to the architecture of this solution scenario:

- milan
- paris

These machines ran the application servers for the solution scenario, as shown in Figure 9-1 on page 402. The server milan runs TWS as an MDM. The server paris runs TWS as an FTA.

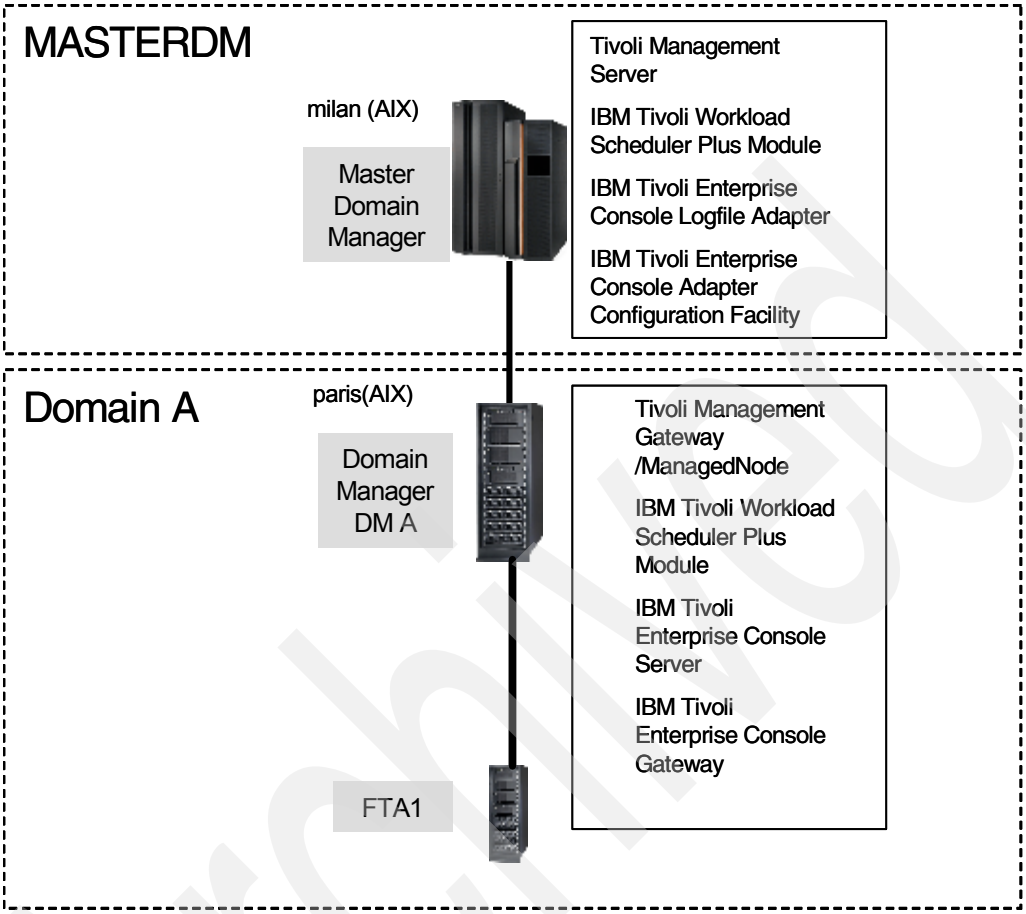

Figure 9-1 Architecture of the TWS and Tivoli Enterprise Console in the lab

9.7 Design

This section describes the design of the integration. It discusses why we recommend not using TWS Plus Module, even though some sites are capable of using it. Tivoli Workload Scheduler 8.2 comes with a product called TWS Plus Module that configures the integration between TWS and Tivoli Distributed Monitoring, the name of an older version of Tivoli Monitoring. The old and new versions are not compatible. Thus, you cannot use TWS Plus Module with Tivoli Monitoring.

We do not recommend using TWS Plus Module with the older version known as Tivoli Distributed Monitoring or Tivoli Distributed Monitoring Advanced Edition. These older versions are either withdrawn from marketing and unavailable for purchase or are withdrawn from support as described at:

http://www-306.ibm.com/software/sysmgmt/products/support/eos.html

Some sites might still be running Tivoli Distributed Monitoring 3.7, which is currently supported on select operating systems. This version can coexist with Tivoli Monitoring 5.1.2. If you are the TWS administrator at such a site and if using Tivoli Monitoring 5.1.2 is not an option, we recommend that you follow the directions in *IBM Tivoli Workload Scheduler Version 8.2 Plus Module User's Guide Version 8.2*, SC32-1276 to install and run TWS Plus Module to integrate TWS with Tivoli Distributed Monitoring.

Other versions of Tivoli Distributed Monitoring (such as version 4.1) cannot run alongside Tivoli Monitoring 5.1.2, and they also do not work with TWS Plus Module. We recommend reading the release notes for TWS Plus Module, Tivoli Monitoring, and Tivoli Distributed Monitoring to determine whether or not your specific site's configuration can use TWS Plus Module with the older versions of Tivoli Distributed Monitoring. For assistance, refer to your Tivoli Framework administrator or IBM service provider.

Instead of installing TWS Plus Module, we refer its design as the basis of designing our own integration that mimics most of the functions that would have been implemented by TWS Plus Module using the older Tivoli Distributed Monitoring product. For details about the monitors that are installed and configured by the TWS Plus Module on designated servers that participate in and impact the scheduling network, see *IBM Tivoli Workload Scheduler Version 8.2 Plus Module User's Guide Version 8.2*, SC32-1276.

Table 9-1 on page 404 provides a mapping from monitors for TWS resources installed by TWS Plus Module under Tivoli Distributed Monitoring, to resource models in Tivoli Monitoring or activities that we need to undertake to deliver the equivalent monitoring under Tivoli Monitoring. For more information about Tivoli Monitoring resource models, refer to *IBM Tivoli Monitoring Resource Model Reference Guide Version 5.1.2*, SH19-4570.

Table 9-1 Mapping between Tivoli Distributed Monitoring and Tivoli Monitoring

Tivoli Distributed Monitoring	Tivoli Monitoring
Host status Availability	Activity: Create custom Resource Model or use Tivoli NetView (and possibly IBM Switch Analyzer)
Application Status (batchman)	Process Resource Model
Application Status (mailman)	Activity: Create custom Resource Model
Application Status (netman)	Process Resource Model
Application Status (jobman)	Activity: Create custom Resource Model
TWS SpaceFree	FileSystem Resource Model
TWS SpaceUsed (stdlist)	Activity: Create custom Resource Model
TWS SpaceUsed (schedlog)	Activity: Create custom Resource Model
Page-Outs	Memory Resource Model
Swap Space Available	Memory Resource Model

The custom resource models included in this book are for UNIX systems. If you want to use the same or equivalent resource models with Microsoft Windows systems, there are several options:

- Download and install onto the Windows systems the open source cygwin Linux-like environment for Microsoft Windows from:

  `http://www.cygwin.com/`

  As long as the Windows system environment variables are correctly configured to run cygwin binaries from within the desired Windows user accounts, the resource models, which only rely upon Bourne shell semantics and some common UNIX utilities, will run correctly.

- Download and install onto the Windows systems the AT&T Research open source uwin version 4.0 (or later) package from:

  `http://www.research.att.com/sw/tools/uwin/`

  As long as the Windows system environment variables are correctly configured to run uwin binaries from within the desired Windows user accounts, the resource models, which only rely upon Bourne shell semantics and some common UNIX utilities, will run correctly.

- Download and install onto the Windows systems the Microsoft Services for UNIX version 3.5 (or later if appropriate) product from:

  `http://www.microsoft.com/windows/sfu/`

As long as the Windows system environment variables are correctly configured to run Microsoft Services for UNIX binaries from within the desired Windows user accounts, the resource models, which only rely upon Bourne shell semantics and some common UNIX utilities, will run correctly.

- Let the resource models that run custom shell scripts use the Bash shell built-in with the Tivoli Management Framework's installation of the Lightweight Client Framework (LCF) software on the Windows system. How to access the Bash shell on a Windows endpoint with the LCF software installed on it can vary depending upon how your Tivoli Framework administrator configured the endpoint installation.

  All of the resource models also rely upon some common UNIX utilities, which also need to be available through the same LCF environment that supplies access to the Bash shell. These utilities can be made available through a variety of ways via the LCF software; typically this is accomplished through the LCF bundle mechanism and using the **wdepset** command. Refer to your Tivoli Framework administrator or your IBM service provider for details on how to access the Bash shell installed on the Windows endpoints at your site.

- Re-write the resource model scripts to use an equivalent function script that runs on a scripting environment that already exists on the Windows systems. For example, these scripts can be Visual Basic for Applications or JScript scripts under the Microsoft Windows Scripting Host service, Perl, Python, or other popular scripting languages. You can find more information about the Microsoft Windows Scripting Host service at:

  `http://msdn.microsoft.com/scripting/`

  You can find more information about Perl at:

  `http://www.perl.com/`

  You can find more information about Python at:

  `http://www.python.org/`

  Refer to your IBM service provider for alternative scripting languages that can be used under Microsoft Windows.

- Re-write the resource model to run an equivalent function program running under Windows instead of a script. These programs can be written in the usual programming languages available under Microsoft Windows.

Any of the first three options are typically recommended for most sites and configurations. The fourth option might not be feasible for some sites. For example, if the location of the Bash shell cannot be guaranteed by the team that manages Tivoli Framework for your site, then you cannot use this option. For additional assistance implementing these alternatives, refer to your IBM service provider.

Cross-platform resource models that have no dependencies on Bourne shell scripts are also possible. A simple example is shown in Chapter 11 of *IBM Tivoli Monitoring Version 5.1.1: Creating Resource Models and Providers*, SG24-6900.

Environments that have already deployed Tivoli Distributed Monitoring to manage their scheduling networks can use the **dmae_sentryanalyser_enh.sh** command to help convert existing Tivoli Distributed Monitoring 3.7 monitors into resource models. You can download this utility from:

http://www-1.ibm.com/support/docview.wss?uid=swg24004683

This utility does not convert all Tivoli Distributed Monitoring monitors. So, manual development of equivalent Tivoli Monitoring resource models is required to convert some monitors. The custom resource models that we show in this book could not be created using this utility by converting existing monitors from Tivoli Distributed Monitoring.

The following sections examine each monitor that is provided by TWS Plus Module for Tivoli Distributed Monitoring and how we designed its equivalent resource model for Tivoli Monitoring.

Important: The custom resource models that we show in this book should not be deployed into production environments without close examination by your Tivoli Monitoring specialist. In many cases, the resource model has to be significantly enhanced to withstand the rigors of production grade environments and to meet more stringent requirements common to production systems. For example, to simplify the presentation in this book, we do not show how to modify the decision tree script of the custom resource models that are discussed. Therefore, the custom resource models presented in this book are incapable of handling multiple parameters. If you need assistance with adapting the custom resource models that we show in this book to your environment, see your IBM service provider.

9.7.1 About host status availability

We recommend using Tivoli NetView (and Tivoli Switch Analyzer if necessary) to implement the equivalent functionality of host status availability instead of creating a custom resource model. While we show how to create the custom resource model that is equivalent to the monitors that are created by TWS Plus Module using the old Tivoli Distributed Monitoring product, both the original monitor and its replacement resource model lack the root cause network failure analysis that Tivoli NetView can provide.

Host availability in the context of the integration with Tivoli Monitoring is defined as a host responds to a ping request from a central host that sends out ping

requests to all systems on which we want to detect host availability. Note, however, that under certain circumstances, a host's compute services can become unavailable yet still respond to ping requests.Therefore, it is important to use host availability as just one measurement out of many for a machine's actual availability status.

9.7.2 Host availability using NetView

Detecting host availability using a monitor or resource model depends upon running the monitor or resource model from one machine, the source machine. This source machine pings the machines on which we are interested in detecting a host down event, the target machine. In Figure 9-2, the source machine is the ITM server, and the target machines are the FTAs.

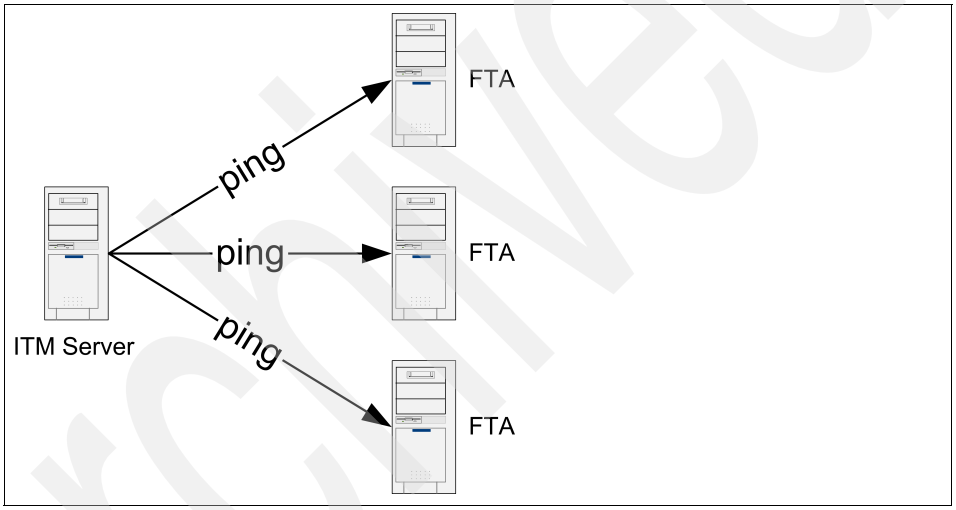

Figure 9-2 Architecture of resource model-based host availability detection

This architecture is sufficient in production environments if there really was a direct connection between the source and targets. For example, if the source server has a quad-port Ethernet host based adapter, and each target machine is plugged directly into a port on the adapter, then Figure 9-2 is an accurate representation of the path between the source and target machines.

However, this path is rarely the case. More often, the network architecture in most sites uses one or more routers between the source and target machines, as shown in Figure 9-3 on page 408.

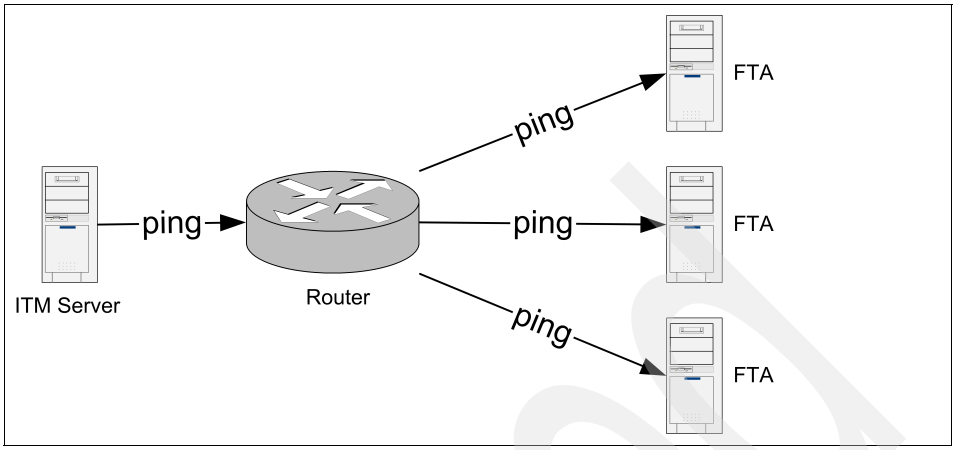

Figure 9-3 Network path of resource model-based host availability detection

In these environments, the network path between the source and target machines goes through a network router. If the router becomes unavailable for any reason, the source machine fails to reach any of the target machines to which it is connected via the router and assumes that all target machines are unavailable.

Many sites use network switches instead of routers today. The architecture is very similar to the router-based architecture. With both architectures, there exists another possible failure mode. An individual port on the router or switch could fail, and the router and target machine could actually still be available. Yet, the target machine would appear to be unavailable, as shown in Figure 9-4.

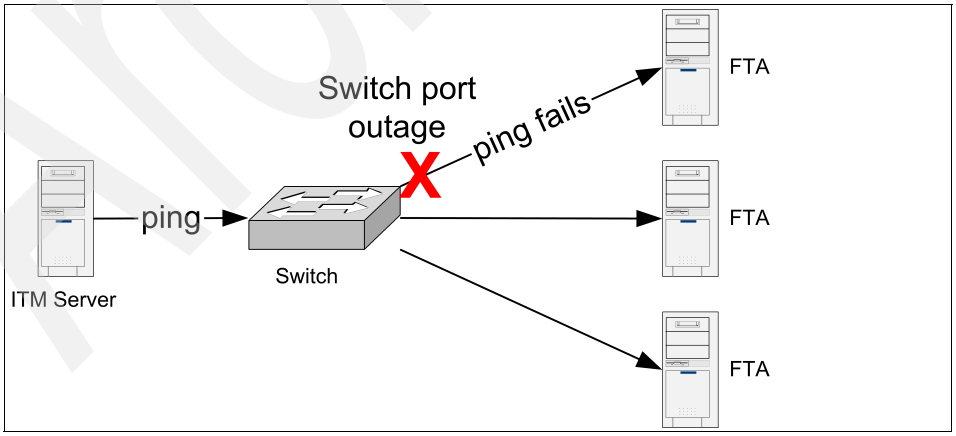

Figure 9-4 Single port failure on switch and impact on resource model measurement

These examples are only the simplest cases. In many production environments, there can be many routers and switches between the source and target machines. Developing a resource model that accounts for all the possible combinations of failures would be very difficult.

The simplest method of addressing these issues is to deploy a network monitoring solution such as Tivoli NetView. For environments that use network switches, you should also deploy Tivoli Switch Analyzer to detect Layer 2 faults on the switch. If a switch fails in a network that is managed by Tivoli NetView and Tivoli Switch Analyzer, the switch is correctly identified as the cause of failure. All network devices behind the switch, such as the FTAs in Figure 9-5, are marked by Tivoli NetView as only possible failures, pending the switch's return to service.

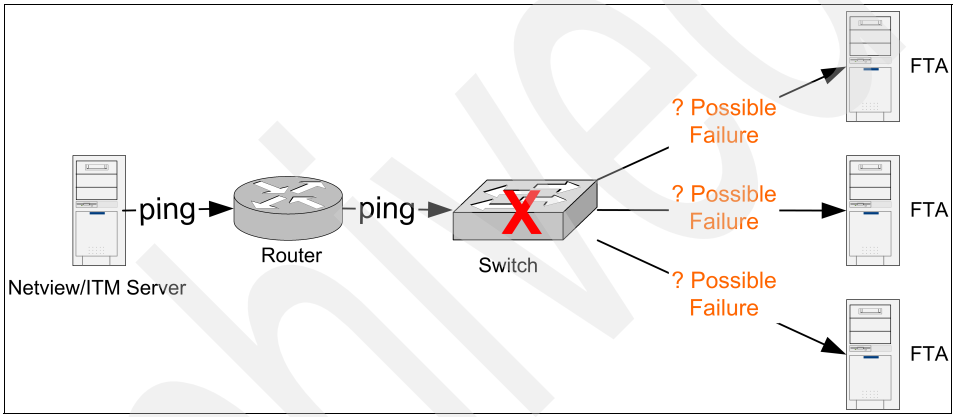

Figure 9-5 Host availability detection using Tivoli NetView and Tivoli Switch Analyzer

If you decide to use Tivoli NetView (and Tivoli Switch Analyzer if you use network switches) to monitor host availability to complement Tivoli Monitoring, you can integrate the detection of host availability using a variety of methods.

The simplest detection method is appropriate for relatively small scheduling networks (roughly 50 processors or less). In this environment, you define a directory to contain a set of zero-length files to indicate the availability of each processor. Then, you can configure a NetView rule such that when NetView positively identifies that a processor (as opposed to a network switch or router in the intermediate network path) is not available, it deletes the file. When the processor is available, the rule uses the **`touch`** command on the file. You can configure a Tivoli Monitoring File resource model that detects the FileNotPresent event for each file.

When a processor is not available, a Tivoli Monitoring event is generated for its corresponding file on the central source machine and the event appears in the Tivoli Monitoring Web Health Console. This method is quick to deploy, but it

requires users of the Tivoli Monitoring Web Health Console to understand that the missing file alert means that a server is down. You can also create a Tivoli Monitoring event that actually indicates host availability from an SNMP trap that is delivered by NetView. It is possible to convert an SNMP trap that is delivered by NetView into a Tivoli Monitoring event by creating a custom resource model that receives SNMP traps and forwards the desired traps to Tivoli Monitoring Web Health Console as Tivoli Monitoring events. See your IBM service provider if you need assistance with the NetView and Tivoli Monitoring integration.

A more sophisticated approach is needed for larger and more complex environments (if you have more than 50 servers or if the majority of the servers to monitor are at branch offices so that there are lots of routers and switches in the environment). In these environments, it is often desirable to receive alerts on intermediate, possible failure conditions, instead of only simple up or down notifications. For these environments, we recommend integrating with Tivoli Enterprise Console. See your IBM service provider for assistance in evaluating how to design automated management of these more complex environments.

9.7.3 Host availability using a resource model

Tivoli Monitoring does not provide a resource model to detect host availability out of the box. This section shows how to design a custom resource model to match the host availability monitor's functionality. Using information from *IBM Tivoli Workload Scheduler Version 8.2 Plus Module User's Guide Version 8.2*, SC32-1276, we determine the initial requirements for the resource model:

- Every 15 minutes, a determination of host availability must be made by a ping test.
- If the host becomes unavailable, send a critical event.
- If the host becomes available (that is, changes from an unavailable state to an available state), send a harmless event.
- Send events to Tivoli Enterprise Console if it is installed in the environment.

For the actual resource model design, we improved upon these initial requirements by increasing the frequency that a host availability determination is made to once every three minutes. We accomplished this by specifying a ping test once a minute and by requiring three failures in a row before a host availability event is generated. This requirement delivers a more consistent result for the ping tests. Transient network anomalies are less likely to be misinterpreted as unavailable hosts.

Further increasing the robustness of the design, the ping test itself consists of three pings against the target host. We ran the script in Example 9-1 on page 411 for each ping test.

Note: You should modify this script appropriately to run it on a Windows system using one of the techniques discussed in 9.7, “Design” on page 402.

This resource model adds functionality over its corresponding Tivoli Distributed Monitoring monitor by sending events to both Tivoli Enterprise Console (which the original monitor also performed) and Tivoli Monitoring Web Health Console (which the original monitor did not perform).

Example 9-1 Script hostavail.sh run for each ping test of HostAvail resource model

```
#!/bin/sh

HOST=$1

case "`uname`" in
    AIX)
        /etc/ping "${HOST}" 56 3 > /dev/null 2>&1 ; echo $?
    ;;
    HP-UX)
        /etc/ping "${HOST}" 56 3 > /dev/null 2>&1 ; echo $?
    ;;
    SunOS)
        /usr/etc/ping -s "${HOST}" 56 3 > /dev/null 2>&1 ; echo $?
    ;;
    Linux)
        /bin/ping -c 3 -s 56 "${HOST}" > /dev/null 2>&1 ; echo $?
    ;;
    *)
    echo "1"
    echo "ERROR:  Unknown platform"
    exit 1
esac

exit 0
```

Table 9-2 describes the resource model design that fulfills these requirements.

Table 9-2 Description for HostAvail resource model

Description	
Name	HostAvail
Resource model distribution	Distribute this resource model to UNIX or Linux systems.
Purpose	Monitor ping availability of IP devices, from the perspective of systems that subscribe to this resource model. Events are generated when the devices transition from available to unavailable state or vice versa.

Table 9-3 shows the key characteristics of this resource model.

Table 9-3 Key characteristics of HostAvail resource model

Resource model at a glance	
Built-in actions	No
Category	TWS Integration
Clearing events	Yes
Default cycle time	60 seconds
Internal name	HostAvail
Parameters	Yes
Thresholds	No

Indications and events

Table 9-4 lists the event that can be generated by the HostAvail resource model, the name of the indication from which the event is generated, and the default severity of the event. Table 9-5 provides the default settings for this indication.

Table 9-4 Indications and events of HostAvail resource model

Event	Indication	Severity
PingFail	Ping Test Failed	Warning

Table 9-5 Default settings for PingFail indication

Setting	Default
Clearing Event	Yes
Holes	0
Occurrences	3
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Critical

The PingFail indication is sent when the result of running the ping test script is equal to one (1), which means failure. The indication contains the ScriptResult attribute, which is the return result of the ping test script.

Parameters

Table 9-6 lists the parameter that you can set for the HostAvail resource model.

Note: To provide a simplified presentation in this book, this custom resource model only supports one instance of a parameter. You need to add a loop to the decision tree script to support multiple parameters. For details, refer to *IBM Tivoli Monitoring Version 5.1.1: Creating Resource Models and Providers*, SG24-6900.

Table 9-6 Parameter for HostAvail resource model

Parameter	Description	Default
IP	IP address or host name of network device to ping test.	localhost

Logging

Table 9-7 shows the resource, context, and properties for which data can be logged.

Table 9-7 Resources available for logging in HostAvail resource model

Resource	Context	Properties
PingResult	HostAvail	ScriptResult The return result from each run of the ping test script.

The HostAvail resource model does not use a Meta-Object Facility (MOF) file.

9.7.4 Application status (batchman)

Tivoli Monitoring has the Process resource model that you can use to determine whether the batchman process is running. Using the information *IBM Tivoli Workload Scheduler Version 8.2 Plus Module User's Guide Version 8.2*, SC32-1276, we determined the following requirements for the resource model:

- Once every minute, check for presence of batchman process.
- If the process does not exist, send a critical event.
- Send events to Tivoli Enterprise Console if it is installed in the environment.

During configuration of the resource models into profiles, a separate profile should be created for this resource model, so that it can be placed on a schedule that matches the production day. This profile ensures that the resource model stops running when batchman is stopped during Jnextday and avoids false alerts.

This resource model adds functionality over its corresponding Tivoli Distributed Monitoring monitor by sending events to both Tivoli Enterprise Console (which the original monitor also performed) and Tivoli Monitoring Web Health Console (which the original monitor did not perform).

Description

Table 9-8 describes the resource model design that fulfills these requirements.

Table 9-8 Description for Process resource model for batchman process

Description	
Name	Process
Resource model distribution	Distribute this resource model to UNIX or Linux systems.
Purpose	Monitor process availability of batchman process. Events are generated when the process is unavailable.

Table 9-9 shows the key characteristics of this resource model.

Table 9-9 Key characteristics of Process resource model for batchman process

Resource model at a glance	
Built-in actions	No
Category	UNIX - Linux
Clearing events	Yes
Default cycle time	30 seconds
Internal name	DMXProcess
Parameters	Yes
Thresholds	Yes

Indication and event

Table 9-10 lists the events that can be generated by the Process resource model for the batchman process. The table includes the name of the indication from which each event is generated, the default severity of the event. The sections that follow provide details on each indication.

Table 9-10 Indications and events of Process resource model for batchman process

Event	Indication	Severity
HighZombieProcesses	High Number of Zombie Processes	Warning
ProcessHighCPU	Process Consuming High CPU	Minor
ProcessKilledOrNotExisting	Process Killed or Nonexistent	Critical
ProcessStopped	Process Stopped	Critical

High Number of Zombie Processes

This indication is sent when the number of zombie processes is high. A process is a zombie when it has terminated, but its results have not been gathered by the parent process. An excessive number of zombie processes can indicate a failing network daemon (such as rsh) or application.

The indication contains the following attributes.

name "Total". The name attribute is a key attribute.

numZombie The number of zombie processes.

This indication is dependent on the threshold as shown in Table 9-11.

Table 9-11 Threshold of High Number of Zombie Processes indication

Threshold	Description	Default
Maximum Number of Zombie Processes (HighZombieProcess)	The number of zombie processes must exceed this threshold.	20

Table 9-12 shows the default settings for this indication.

Table 9-12 Default settings for High Number of Zombie Processes indication

Setting	Default
Clearing Event	Yes
Holes	0
Occurrences	3
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Warning

Process Consuming High CPU

This indication is sent when a process is using an excessive amount of processor time.

The indication contains the following attributes.

IDProcess The ID of the process monitored. IDProcess is a key attribute.

PrcProcessorTime The percentage of processor time used by the process.

Process The process name. Process is a key attribute.

state The status of the process

This indication is dependent on the threshold as shown in Table 9-13.

Table 9-13 Threshold of Process Consuming High CPU indication

Threshold	Description	Default
Percentage of processor Used (HighCPUUsed)	The percentage of processor time used by the process must exceed this threshold.	60

Table 9-14 shows the default settings for this indication.

Table 9-14 Default settings for Process Consuming High CPU indication

Setting	Default
Clearing Event	Yes
Holes	1
Occurrences	10
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Minor

Process Killed or Nonexistent

This indication is sent when a monitored process is not found in the system. The process may have been killed or may never have been started. Specify the list of monitored processes through the Processes parameter.

The indication contains the following attribute:

name The process name. The name attribute is a key attribute.

Table 9-15 shows the default settings for this indication.

Table 9-15 Default settings for Process Killed or Nonexistent indication

Setting	Default
Clearing Event	Yes
Holes	0
Occurrences	1
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes

Process Stopped

This indication is sent when a monitored process is in the stopped state. The process is present in the system and can be monitored using the **ps** command. Specify the list of monitored processes through the Processes parameter.

The indication contains the following attributes.

name The process name. The name attribute is a key attribute.

ParentProcessID The process identifier (PID) of the parent process.

ProcessID The process identifier (PID). ProcessID is a key attribute.

ProcessStatus The status of the process.

Table 9-16 shows the default settings for this indication.

Table 9-16 Default settings for Process Consuming High CPU indication

Setting	Default
Clearing Event	Yes
Holes	0
Occurrences	1
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Critical

Thresholds

Table 9-17 lists the thresholds that you can set for the Process resource model for the batchman process. For each threshold, the table shows the name, a short description, and the default value.

Table 9-17 Thresholds for Process resource model for batchman process

Threshold	Description	Default
Maximum Number of Zombie Processes (HighZombieProcess)	This threshold is the maximum number of zombie processes.	20

Parameters

Table 9-18 lists the parameters that can be set for the Process resource model for the batchman process.

Table 9-18 Parameters for HostAvail resource model.

Parameter	Description	Default
Processes	This parameter is the set of filters used in the search for stopped or non existent processes. The filters may vary in length and are composed of key=value pairs. See *BM Tivoli Monitoring Resource Model Reference Guide Version 5.1.2* for details.	basename=batchman

The syntax of the filter parameter for the Processes parameter is as follows:

```
property1=value1&property2=value2&...
```

The following properties are valid filters for the Processes parameter.

arguments The arguments of the process.

basename The base name of the executable file, equivalent to that generated by the **`ps -e`** command.

effectiveGroup The effective group name of the process.

effectiveUser The effective user name of the process.

group The group name of the process.

name The fully qualified path name of the executable file, equivalent to that generated by the **`ps -ef`** command.

status The current status of the process.

terminal The controlling terminal of the process.

user The user name of the process.

Logging

Table 9-19 shows the resource, context, and properties for which data can be logged.

Table 9-19 Resources available for logging in Process resource model

Resource	Context	Properties
Process	Processor Usage	**PercentProcessUsage** The percentage of processor time used by a process. **PID** The process identifier. **Process** The process name.

Description of the MOF file properties

The Process resource model uses two MOF files:

- DMXProcess.mof
- DMXSecurity.mof

See DMXProcess.mof in *IBM Tivoli Monitoring Resource Model Reference Guide Version 5.1.2*, SH19-4570 for a description of the DMXProcess.mof CIM classes and properties that are associated with this file. See DMXSecurity.mof in *IBM Tivoli Monitoring Resource Model Reference Guide Version 5.1.2*, SH19-4570 for a description of the DMXSecurity.mof CIM classes and properties that are associated with this file.

9.7.5 Application status (netman)

The design of the resource model for application status of the netman process is the same as for the batchman process except for the configuration of the resource models into profiles. The Process resource model for the batchman process has to be configured for a schedule that matches the production day. No such requirement exists for the Process resource model for the netman process. Instead, it should run at all times, unless TWS has to be shut down completely for activities such as applying software patches.

Planned maintenance activities that shut down the netman process should follow your environment's planned outage procedures for appropriately turning off either the Tivoli Monitoring engine or just the affected resource models on the affected systems and for turning back on the engine or affected resource models.

9.7.6 Application status (mailman)

The design of the resource model for application status of the mailman process is the same as for the batchman process, as described in 9.7.4, “Application status (batchman)” on page 414.

9.7.7 Application status (jobman)

The design of the resource model for application status of the jobman process is the same as for the batchman process, as described in 9.7.4, “Application status (batchman)” on page 414.

9.7.8 TWS SpaceFree

Tivoli Monitoring has the File System resource model that you can use to determine the amount of free disk space in the TWS directory (*TWSHome*). Using the information from *IBM Tivoli Workload Scheduler Version 8.2 Plus Module User’s Guide Version 8.2*, SC32-1276, we determined the following requirements for the resource model:

- Once every 15 minutes check amount of free disk space in *TWSHome*.
- If the amount of free disk space is less than 2 MB, send a critical event.
- Send events to Tivoli Enterprise Console if it is installed in the environment.

The File System resource model does not provide multiple levels of notification corresponding to different severity levels. Thus, we cannot duplicate the original Tivoli Distributed Monitoring monitor that sent a severe event if the free disk space fell below 4 MB, a warning event if it fell below 6 MB, and a harmless event if free disk space was greater than 6 MB. If multiple levels of notification are desired, there are two possible solutions.

The first solution is to create separate profiles for each triggering level, and populate a single File System resource model in each profile (only one instance of a type of resource model can exist in a resource model). The drawback to this method is the multiple profiles will each send events for thresholds configured in their resource model, leading to duplicate events that have to be suppressed. The suppression can be performed with a product like Tivoli Enterprise Console but not with the Web Health Console.

We recommend using the second solution. Develop a custom resource model that accepts a parameter string that is comprised of fields delimited by a reserved character. The first field is for the file system path (which cannot contain the reserved character), and additional fields supply trigger levels for fatal, critical, minor, warning and harmless notification levels. The custom resource model would then use multiple thresholds corresponding to each notification

level, and the script return code would indicate which threshold to has been triggered, and thus which event to send.

More sophisticated versions of this custom resource model could use a modified VisitTree() function of the resource model's decision tree to directly query the script result without comparing it with a threshold, then directly send the desired event. This would eliminate the need to define multiple thresholds within the resource model (which are not necessary because the parameter contains the thresholds to use).

This resource model adds functionality over its corresponding Tivoli Distributed Monitoring monitor by sending events to both Tivoli Enterprise Console (which the original monitor also performed) and Tivoli Monitoring Web Health Console (which the original monitor did not perform).

Description

Table 9-20 describes the resource model design that fulfills these requirements.

Table 9-20 Description for File System resource model for TWS

Description	
Name	File system
Resource model distribution	Distribute this resource model to UNIX or Linux systems.
Purpose	Provides information about file system usage. Events are generated when available space in the file system is low, the percentage of available space in the file system is low, the file system is fragmented, or the percentage of i-nodes in the file system is low.

Table 9-21 shows the key characteristics of this resource model.

Table 9-21 Key characteristics of File System resource model for TWS

Resource model at a glance	
Built-in actions	No
Category	UNIX - Linux
Clearing events	Yes
Default cycle time	120 seconds
Internal name	DMXFileSystem
Parameters	Yes
Thresholds	Yes

Indication and event

Table 9-22 lists the events that can be generated by the File System resource model, the name of the indication from which each event is generated, and the default severity of the event. The sections that follow provide details about the indication.

Table 9-22 Indications and events of File System resource model for TWS

Event	**Indication**	**Severity**
FragmentedFileSystem	Fragmented File System	Minor
LowKAvail	Low Space Available	Critical
LowPercInodesAvail	Low Percentage of Available i-nodes	Warning
LowPercSpcAvail	Low Percent Space Available	Critical

Fragmented File System

This indication is sent when the file system is fragmented. The percentage of i-nodes in use is high and the percentage of used file system space is low. To optimize the use of the available space, the percentage of i-nodes used should not be too high in comparison to the space used by the file system. The indication contains the following attributes:

mountname The mount point on which the file system is mounted. The mountname attribute is a key attribute.

percinodeused The percentage of used i-nodes in the file system.

perckbused The percentage of space used by the file system.

This indication is dependent on the thresholds as shown in Table 9-23.

Table 9-23 Threshold of Fragmented File System indication

Threshold	Description	Default
Percentage of File System Space Used (PrcUsedKspace)	The percentage of used space in the file system must not exceed this threshold.	85
Percentage of I-nodes Used (PrcUsedInodes)	The percentage of i-nodes in use in the file system must exceed this threshold.	80

Table 9-24 shows the default settings for this indication.

Table 9-24 Default settings for High Number of Zombie Processes indication

Setting	Default
Clearing Event	Yes
Holes	1
Occurrences	4
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Minor

Low Space Available

This indication is sent when the available space in the specified file system is low. The indication contains the following attributes.

kb_avail The number of kilobytes available on the specified file system.

mountname The directory on which the file system is mounted. The mountname attribute is a key attribute.

This indication is dependent on the threshold as shown in Table 9-25.

Table 9-25 Threshold of Low Space Available indication.

Threshold	Description	Default
Available Space (AvailableSpace)	The available space on the file system, in Kilobytes, must not exceed this threshold.	7000

Table 9-26 shows the default settings for this indication.

Table 9-26 Default settings for Process Consuming High CPU indication

Setting	Default
Clearing Event	Yes
Holes	1
Occurrences	10
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Minor

Low Percentage of Available I-nodes

This indication is sent when the percentage of space available in the specified file system is low. The indication contains the following attributes:

mountname The mount point on which the file system is mounted. The mountname attribute is a key attribute.

PercSpcAvail The percent space available.

Table 9-27 shows the default settings for this indication.

Table 9-27 Default settings for Low Percentage of Available I-nodes indication

Setting	Default
Clearing Event	Yes
Holes	1
Occurrences	4
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes

Low Percent Space Available

This indication is sent when the percentage of available i-nodes is below the threshold.

For VxFS file systems, version 2.0 or greater, i-nodes are allocated dynamically from a pool of free blocks, and the number of i-nodes appears unlimited. However, if you receive this indication, the internal i-node list file might be larger than the maximum file size limit of 2 GB. You must then configure the VxFS file system for large file support.

Confirm the following to ensure that your VxFS file system configuration enables large file support:

- The VxFS file system is disk layout, version 3.0 or greater.
- Your operating system version where the VxFS file system resides provides large file support.
- Large file support is enabled on your VxFS file system.
- If large file support is not enabled, set the largefiles flag on the file system using the **`fsadm`** command.

The indication contains the following attributes.

mountname The mount point on which the file system is mounted. The mountname attribute is a key attribute.

percavailinodes The percentage of available i-nodes

This indication is dependent on the threshold as shown in Table 9-28.

Table 9-28 Threshold associated with Low Percent Space Available indication

Threshold	Description	Default
Percentage of Available I-nodes (PrcAvailInodes)	The percentage of available i-nodes must not exceed this threshold.	20

Table 9-29 shows the default settings for this indication.

Table 9-29 Default settings for Process Consuming High CPU indication

Setting	Default
Clearing Event	Yes
Holes	1
Occurrences	4
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Warning

Thresholds

Table 9-30 lists the thresholds that can be set for the File System resource model. For each threshold, the table shows the name, a short description, and the default value.

Table 9-30 Thresholds for File System resource model for TWS

Threshold	Description	Default
Available Space (AvailableSpace)	This threshold is the minimum amount of available space, in kilobytes, in a file system.	7000
Percentage of Available I-nodes (PrcAvailInodes)	This threshold is the minimum percentage of available i-nodes in a file system.	20
Percentage of Available Space (PrcAvailKspace)	This threshold is the minimum percentage of available space in a file system.	15
Percentage of Used Space (PrcUsedKspace)	This threshold is the maximum percentage of used space in a file system.	85
Percentage of Used I-nodes (PrcUsedInodes)	This threshold is the maximum percentage of used i-nodes in a file system.	80

Parameters

Table 9-31 lists the parameters that can be set for the File System resource model.

Table 9-31 Parameters for File System resource model for TWS

Parameter	Description	Default
Monitored File Systems (FileSystemsToMonitor DiffThres)	This parameter is the list of file systems monitored, along with their threshold values. If no values are specified with this parameter, the default behavior of the resource model is to monitor all file systems of the supported types (for example, UFS or VxFS) using the global threshold values. This parameter allows you to override the threshold values for specific file systems. See *IBM Tivoli Monitoring Resource Model Reference Guide Version 5.1.2*, SH19-4570 for details.	None
Ignored File Systems (IgnoredFileSystems)	This parameter is the list of file systems that are not monitored. If this parameter is empty, then all file systems are monitored. The file system name may contain one or more wildcard characters, which is an asterisk ('*'). The wildcard '*' matches one or more characters in the file system name.	None

The syntax of the Monitored File Systems parameter is:

```
FS_name | a | b | c | d | e
```

In this command:

- *FS_name* is the file system name
- *a* is the percentage of i-nodes used
- *b* is the percentage of file system used space
- *c* is the available space, in kilobytes
- *d* is the percentage of available i-nodes
- *e* is the percentage of file system available space

The file system name can contain one or more wildcard characters, such as an asterisk (*). The wildcard matches one or more characters in the file system name.

Logging

Table 9-32 shows the resource, context, and properties for which data can be logged.

Table 9-32 Resources available for logging in File System resource model for TWS

Resource	Context	Properties
File System	File System Availability	**mountpoint** The directory on which the file system is mounted. **percUsed** The percentage of file system space used **percInodesUsed** The percentage of i-nodes used **percAvail** The percentage of file system space available

Note: For VxFS file systems, the i-node properties that are reported are based on the number of i-nodes that are currently allocated in your configuration. For VxFS version 2.0 or greater, i-nodes are allocated dynamically from a pool of free blocks. The i-node properties change as additional i-nodes are added.

Description of the MOF file properties

The File System resource model uses one MOF file: DMXFileSystem.mof.

See DMXFileSystem.mof in *IBM Tivoli Monitoring Version 5.1.1 Creating Resource Models and Providers*, SG24-6900 for a description of the DMXFileSystem.mof CIM classes and properties that are associated with this file.

9.7.9 TWS SpaceUsed (stdlist)

Tivoli Monitoring does not provide a resource model to detect space used underneath a directory out of the box. This section shows how to design a custom resource model to match the space used monitor's functionality. Using the information from *IBM Tivoli Workload Scheduler Version 8.2 Plus Module*

User's Guide Version 8.2, SC32-1276, we determined the following initial requirements for the resource model:

- Every 15 minutes, a determination of space used underneath a directory must be made by a custom script.
- If the directory contains more than 50 000 1024-byte blocks, send a critical event.
- If the directory contains less than 40 000 1024-byte blocks, send a harmless event.
- Send events to Tivoli Enterprise Console if it is installed in the environment.

For the actual resource model design, we improve upon these initial requirements by increasing the frequency that the space used determination is made to once every three minutes. We accomplish this by specifying that the custom script run once a minute and by requiring three failures in a row before a space used critical event is generated. This requirement delivers a more consistent result for the space used tests. Temporary files are less likely to be misinterpreted as filling up the directory.

We ran the script in Example 9-2 to test the space used under a directory.

Note: You should modify this script appropriately to run it on a Windows system using one of the techniques that are discussed in 9.7, "Design" on page 402.

This resource model adds functionality over its corresponding Tivoli Distributed Monitoring monitor by sending events to both Tivoli Enterprise Console (which the original monitor also performed) and Tivoli Monitoring Web Health Console (which the original monitor did not perform).

Example 9-2 Script dirspace.sh run for each ping test of SpaceUsed resource model

```
#!/bin/sh

TARGET_DIR=$1

if [ ! -d "${TARGET_DIR}" ] ; then
    echo 1.1
    exit 0
fi

du -sk "${TARGET_DIR}" | awk '{ print $1 }'
```

Table 9-33 describes the resource model design that fulfills these requirements.

Table 9-33 Description for SpaceUsed resource model

Description	
Name	SpaceUsed
Resource model distribution	Distribute this resource model to UNIX or Linux systems.
Purpose	Monitor space used in 1024-byte blocks of user-specified directory. Events are generated when the exceeds or falls below a given number of blocks.

Table 9-34 shows the key characteristics of this resource model.

Table 9-34 Key characteristics of SpaceUsed resource model

Resource model at a glance	
Built-in actions	No
Category	TWS Integration
Clearing events	Yes
Default cycle time	60 seconds
Internal name	SpaceUsed
Parameters	Yes
Thresholds	No

Indications and events

Table 9-35 lists the events that can be generated by the SpaceUsed resource model, the name of the indication from which each event is generated, and the default severity of the event. The sections that follow describe the indications.

Table 9-35 Indications and events of SpaceUsed resource model

Event	**Indication**	**Severity**
SpaceHigh	Directory Space High	Critical
SpaceLow	Directory Space Low	Harmless

Directory Space High

This indication is sent when the standard output of running the space used test script is an unformatted integer greater than or equal to 50 000, meaning that the given directory uses an abnormally high number of 1024-byte blocks. The indication contains the ScriptResult attribute, which is the return result of the space used test script.

Table 9-36 shows the default settings for this indication.

Table 9-36 Default settings for Directory Space High indication

Setting	Default
Clearing Event	Yes
Holes	0
Occurrences	3
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Critical

Directory Space Low

This indication is sent when the standard output of running the space used test script is an unformatted integer less than or equal to 40 000, meaning that the given directory uses a low number of 1024-byte blocks that might mean that too many files have been rotated out of the directory. The indication contains the ScriptResult attribute, which is the return result of the space used test script.

Table 9-37 shows the default settings for this indication.

Table 9-37 Default settings for Directory Space Low indication

Setting	Default
Clearing Event	Yes
Holes	0
Occurrences	3
Send indications to Tivoli Business Systems Manager	No
Send indications to Tivoli Enterprise Console	Yes
Severity	Harmless

Parameters

Table 9-38 lists the parameter that you can set for the HostAvail resource model.

Table 9-38 Parameter for HostAvail resource model

Parameter	Description	Default
DirPath	Full directory path of directory to test.	/tmp

Logging

Table 9-39 shows the resource, context, and properties for which data can be logged.

Table 9-39 Resources available for logging in HostAvail resource model

Resource	Context	Properties
SpaceUsedDir	SpaceUsed	**ScriptResult** The standard output from each run of the space used test script.

The SpaceUsed resource model does not use an MOF file.

9.7.10 TWS SpaceUsed (schedlog)

The design of the resource model for the space that is used in the schedlog directory is the same as for the stdlist directory, as described in 9.7.9, “TWS SpaceUsed (stdlist)” on page 430, except that during the configuration phase of the resource model, you use a different parameter.

9.8 Implementation

As shown in Figure 9-1 on page 402, the Tivoli Management Region (TMR) Server in the lab that we used for this book is installed on the TWS Master Domain Manager (MDM) server milan. A Managed Node and Gateway are installed on the TWS Domain Manager server paris. In our environment, we installed the Tivoli Monitoring base component and the Tivoli Monitoring Web Health Console server component on the same server (milan) because it was the TMR server.

Note: All commands that we ran on the primary TMR, ITM, and MDM servers shown in this section assumed that the Tivoli environment was already sourced.

You source the Tivoli environment using the following command.

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

Your Tivoli Framework administrator or IBM service provider might have configured your shell environment to source the Tivoli environment each time you log in.

9.8.1 Installing the integration

You install the integration between Tivoli Monitoring and TWS by ensuring that Tivoli Monitoring is configured on all systems which impact the production schedule that need to be monitored. The majority of the integration between Tivoli Monitoring and TWS occurs during the configuration phase. See your Tivoli Framework administrator or IBM service provider for details on how to verify the configuration of the base installation of Tivoli Monitoring.

In the lab for this book, we determined that using the JRE that is supplied by IBM yields the best results. This JRE was copied from the Tools disc of the Tivoli Monitoring media. In our lab, we installed the IBM-supplied JRE files under the directory /usr/java131-itm. It can also be installed using the **wdmdistrib** command. For details, refer to *IBM Tivoli Monitoring User's Guide Version 5.1.2*, SH19-4569.

Multiple JREs can coexist without issue on the same system. Example 9-3 shows how we linked the JRE for Tivoli Monitoring. The linking procedure is not necessary in some Tivoli Monitoring installation scenarios.

Note: This procedure should have been performed for you if your Tivoli Framework administrator installed Tivoli Monitoring on the systems that you want to monitor. We present this example so that you can verify with your Tivoli Framework administrator that this JRE has been linked on the servers that are impacted by the production plan.

Example 9-3 Linking to the IBM-supplied JRE for Tivoli Monitoring

```
milan:/tivoli/media#  wruntask -t DMLinkJre -l 'IBM Tivoli Monitoring Tasks' \
-E -h paris.itsc.austin.ibm.com -m 1800 -a "/usr/java131-itm/jre"
################################################################################
Task Name:      DMLinkJre
Task Endpoint:  paris.itsc.austin.ibm.com (ManagedNode)
Return Code:    0
------Standard Output------
**************************************************************
java version "1.3.1"
Java(TM) 2 Runtime Environment, Standard Edition (build 1.3.1)
```

```
Classic VM (build 1.3.1, J2RE 1.3.1 IBM AIX build ca131-20030630a (JIT enabled:
jitc))
***************************************************************
JRE is correctly installed.
JRE is a suitable version.
The JRE has been successfully linked.
------Standard Error Output------
###############################################################################
```

The specific build string `ca131-20030630a` that is highlighted in red text in the example indicates the exact JRE version that is bundled with Tivoli Monitoring. Refer to the resources listed in “Related publications” on page 691 if you are installing Tivoli Monitoring for the first time.

9.8.2 About the configuration

The majority of the integration between Tivoli Monitoring and TWS is performed through the configuration of profile managers, profiles, subscribers, and resource models. These are all objects within Tivoli Management Framework. The remainder of the sections discuss the configuration tasks.

Tip: All of the configuration tasks included in this book have command line interface equivalents. If you need to quickly configure many resource models or need a way to duplicate the configuration activities (useful for disaster recovery procedures, for example), we recommend that you use the command line interface equivalents. See your Tivoli Framework administrator or IBM service provider for details on how to use the command line for Tivoli Monitoring.

Figure 9-6 displays the hierarchical relationship between the Framework objects that are configured for the integration.

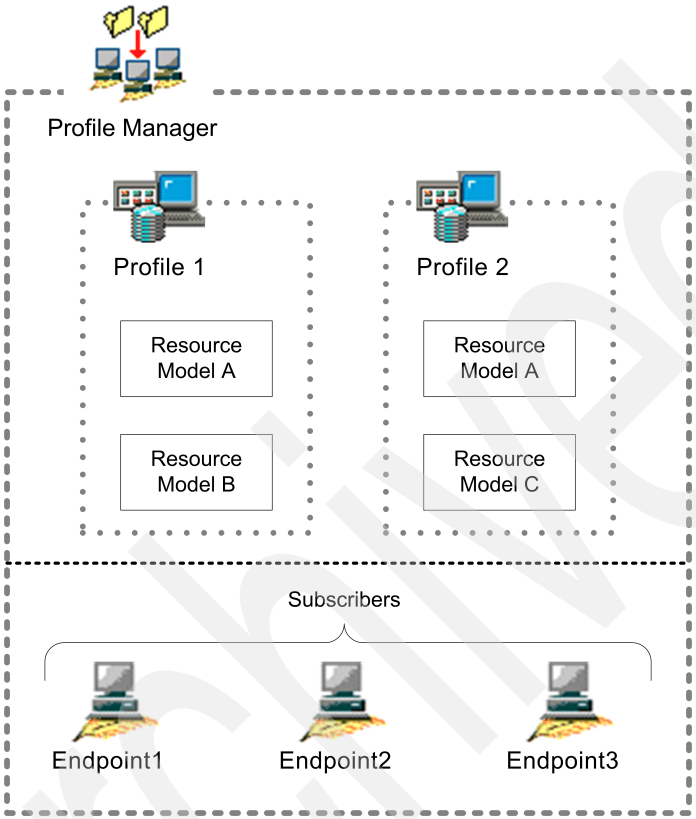

Figure 9-6 Hierarchical relationship between Framework objects used in integration

At least one profile manager is needed for the integration. In some production environments, more than one profile manager can be configured. See your Tivoli Framework administrator or IBM service provider for further advice on profile manager architecture. This book shows all configuration activities within the context of a single profile manager.

One or more profiles and subscribers can be added to a profile manager. Furthermore, one or more resource models can be added to each profile. However, only one resource model can be used in each profile.

For example, in Profile 1 there is Resource Model A and Resource Model B. Tivoli Monitoring's design does not allow another Resource Model A to be added to Profile 1. This restriction impacts the design that is shown in this book because

the simplified custom resource models do not support multiple parameters. The parameter for these custom resource models specifies configurable data such as the IP host name for the Host Status Availability resource model. In the absence of support for multiple parameters, individual profiles have to be created for each resource model, each with a different parameter.

Subscribers in the context of this integration are typically all the processors in the scheduling network that have the LCF software installed on them. In many production environments, typically any server that can impact the production plan, regardless of whether or not it directly participates in the scheduling network as a Fault Tolerant Agent or Extended Agent, is included as a subscriber. In a production environment, the custom resource models would be further developed to support event correlation among other features, enabling more sophisticated configurations within a profile.

There are many options for architecting the organization of subscribers. For example, subscribers can be organized hierarchically, or even generated on demand. In this book we only show using a dataless profile manager with endpoint subscribers to implement the integration. Refer to the Tivoli Management Framework and Tivoli Monitoring manuals for details on other architecture options for subscribers, or see your Tivoli Framework administrator or IBM service provider for assistance.

After profiles are created and populated with the desired resource models, the profiles are distributed to the subscribers in the profile manager. There are many distribution options, but to simplify the presentation in this book we show how to manually distribute selected profiles to individual subscribers. Detailed descriptions of the available distribution options are in the Tivoli Management Framework and Tivoli Monitoring manuals.

The distribution of profiles to subscribers installs the resource models in the profiles onto the subscribers. All executables necessary to start running the resource model are automatically downloaded, configured and started on the subscriber as part of the distribution process. This lets administrators make very quick changes to the resource models on subscribers without having to engage in any distribution issues.

The following sections show the configuration activities necessary to complete the integration between Tivoli Monitoring and TWS.

9.8.3 Creating a profile manager

You create a profile manager to hold the profiles, resource models, and subscribers that we show in this book. To create a profile manager, you need a user account that is authorized to access Tivoli Framework with the appropriate level of access. To log into the Tivoli Management Framework and create a profile manager:

1. Start the Tivoli Desktop GUI client on your system. If you are using a Windows system, we recommend that you install the Tivoli Desktop client from Tivoli Management Framework version 4.1.1 Fix Pack 2. You can downloaded this fix pack from:

   `ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.1.1/4.1.1-TMF-FP02/4.1.1-TMF-FP02.tar`

   After you extract this tar archive and extract the tar archives within the tar file, the installer for the Tivoli Desktop is in the 411TMF19/DESKTOP/NT_95 subdirectory. See your Tivoli Framework administrator or IBM service provider for additional assistance installing the Tivoli Desktop.

   If you are using UNIX, if the Tivoli environment is sourced. Running the **`tivoli`** command starts the Tivoli Desktop.

   The login window for Tivoli Desktop displays when you start it, as shown in Figure 9-7.

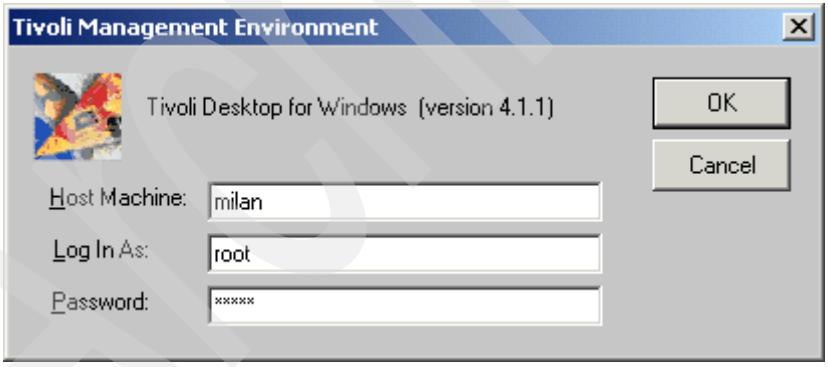

Figure 9-7 Tivoli Desktop login screen

2. Log into the primary TMR server with an authorized account. The splash screen displays while Tivoli Desktop logs into the server.

3. If the login is successful, the main window displays as shown in Figure 9-8.

Figure 9-8 Main window of Tivoli Desktop

4. Double-click the policy region that should contain the profile manager. Your Tivoli Framework administrator can provide guidance on the appropriate policy region. In the lab for this book, we used the milan-region policy region as shown in Figure 9-9 on page 441.

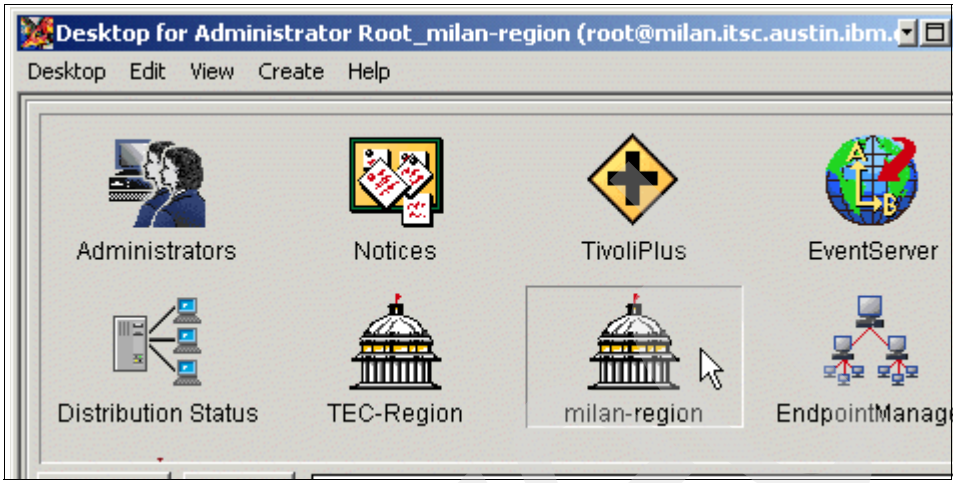

Figure 9-9 Opening the milan-region policy region

5. The Policy Region window displays, as shown in Figure 9-10.

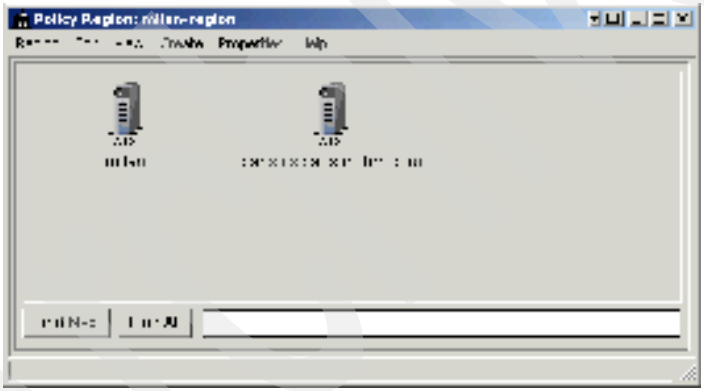

Figure 9-10 Policy Region window

In the lab for this book, this policy region already contained the Managed Node objects that represent milan and paris. Depending on your environment, there might be no objects or many more objects.

6. Add profile managers as a managed resource for the policy region. Select **Properties** → **Managed Resources...** as shown in Figure 9-11 on page 442.

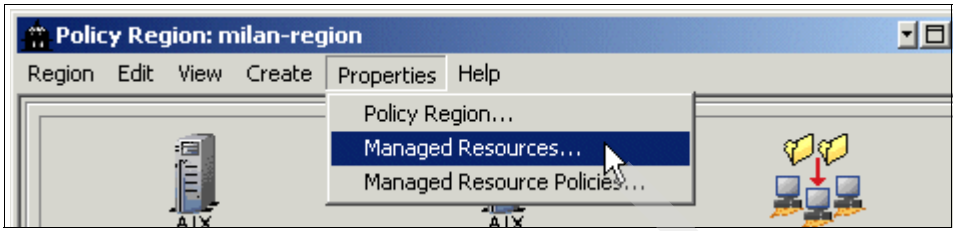

Figure 9-11 Adding Managed Resources

7. The Set Managed Resources window displays as shown in Figure 9-12.

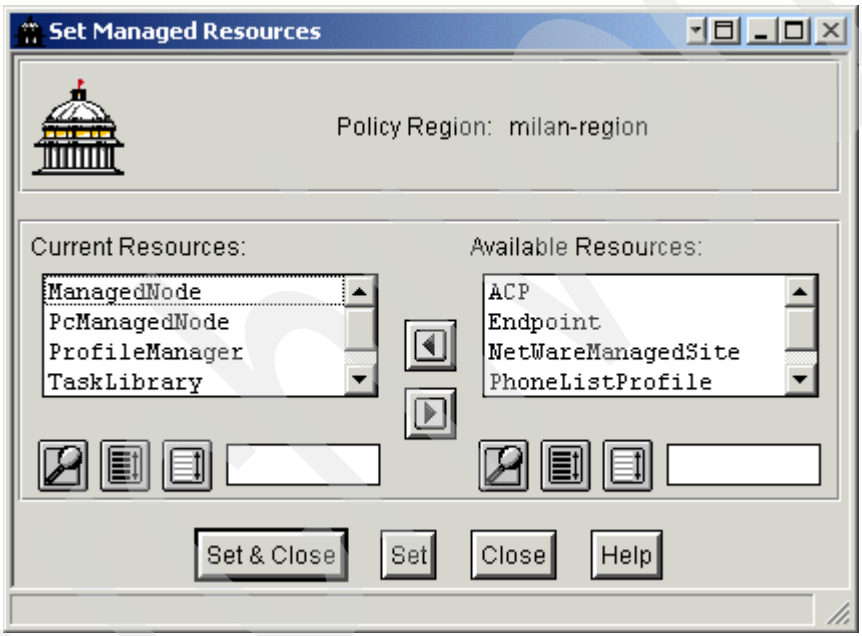

Figure 9-12 Set Managed Resources window

Verify that ProfileManager is displayed in the Current Resources list on the left-hand side of the window as shown. If it is not, follow the next step as a guide to move ProfileManager from the Available Resources list on the right-hand side of the window into the Current Resources list on the left-hand side of the window.

8. Add resource model profiles as a managed resource for this policy region. Scroll down in the Available Resources list on the right-hand of the window until you see the Tmw2kProfile managed resource in the list.

   Highlight the resource, then click the top arrow button that points to the left as shown in Figure 9-13 on page 443.

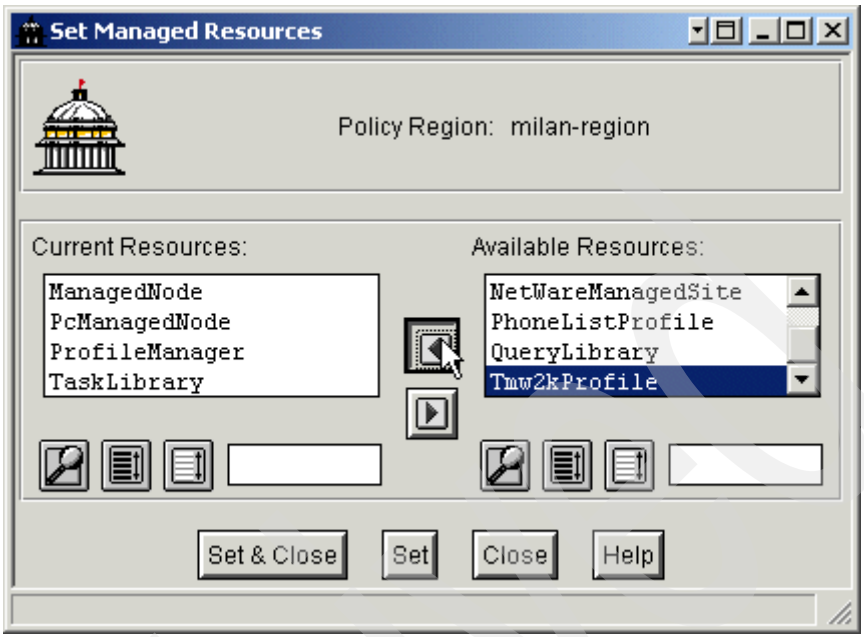

Figure 9-13 Adding resource model profiles as a managed resource

The resource model profile managed resource is moved into the Current Resource list on the left-hand side of the window as shown in Figure 9-14 on page 444.

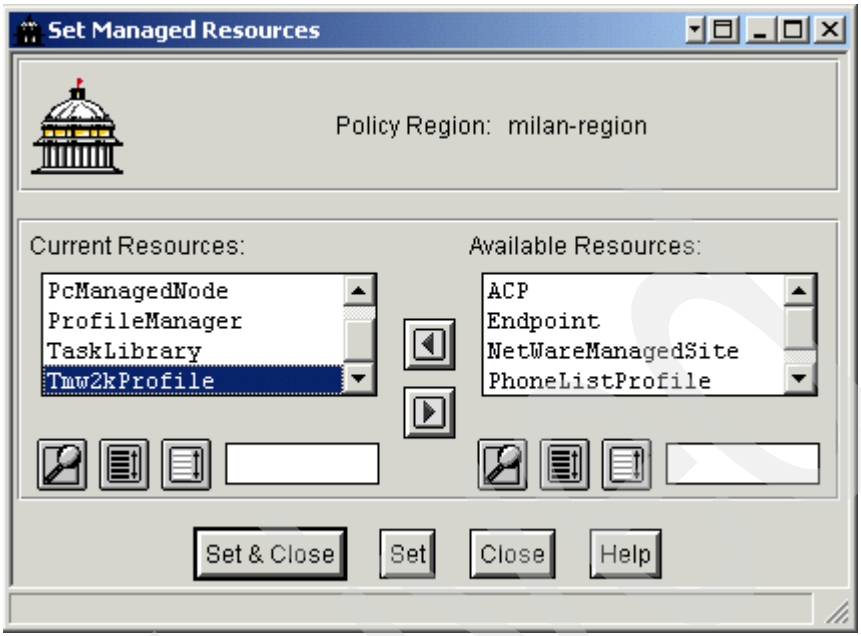

Figure 9-14 Resource model profile as a managed resource of the policy region

If it is not already a managed resource, the ProfileManager managed resource must be in the Current Resources list for the policy region. The Endpoint managed resource should also be added.

9. Select **Set & Close**.

10. Create the profile manager. In the Policy Region window, select **Create** →, **ProfileManager…** as shown in Figure 9-15.

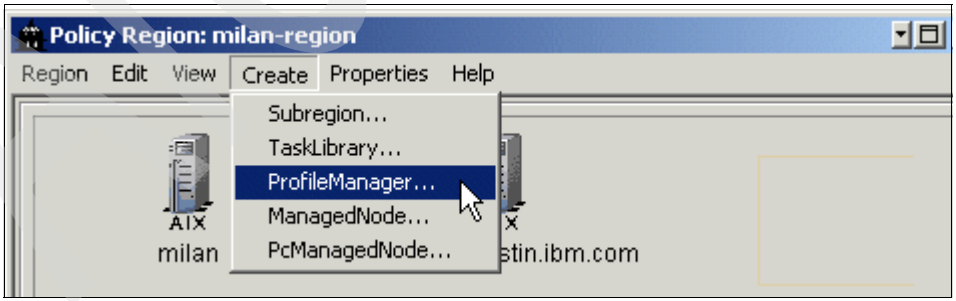

Figure 9-15 Creating a profile manager

11. The Create Profile Manager window displays as shown in Figure 9-16.

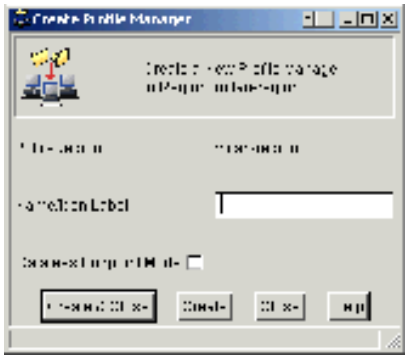

Figure 9-16 Create Profile Manager window

12. Enter a name for the profile manager in the Name/Icon Label text field, and select Dataless Endpoint Mode. In the lab for this book, we entered `itw-tws.pm` as shown in Figure 9-17.

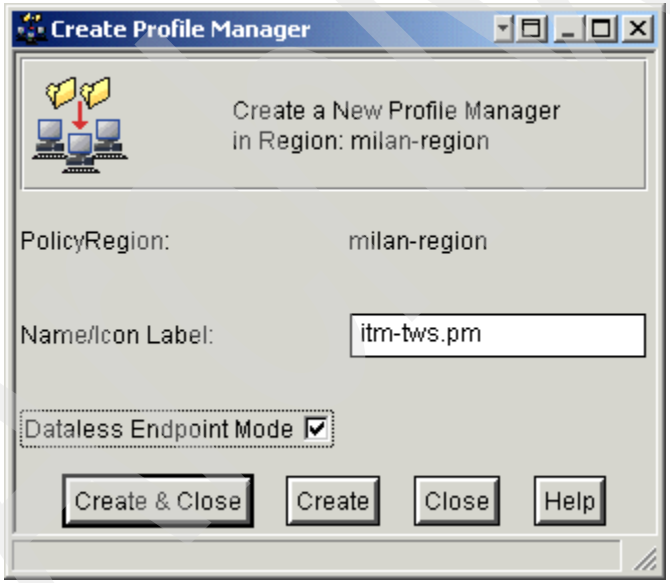

Figure 9-17 Naming the new profile manager

13. Select **Create & Close**. The profile manager is created, and its icon placed in the Policy Region window as shown in Figure 9-18 on page 446.

Figure 9-18 New profile manager in policy region

14. Double-click the new profile manager as shown in Figure 9-19.

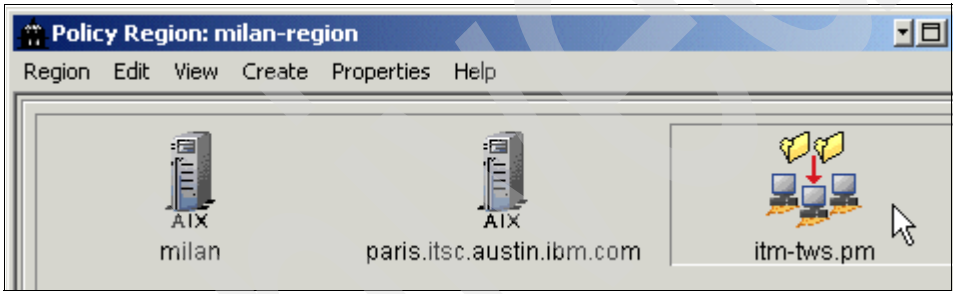

Figure 9-19 Opening the new profile manager

The Profile Manager window opens as shown in Figure 9-20 on page 447.

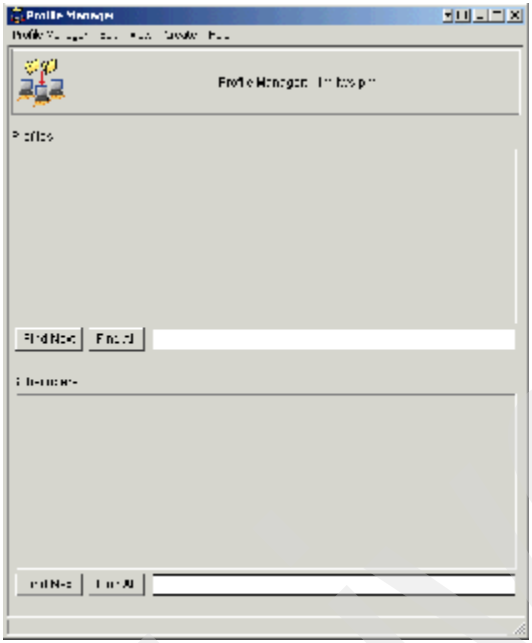

Figure 9-20 New profile manager window

9.8.4 Adding subscribers

After you create a profile manager, you can populate it with profiles and subscribers. Start with an open profile manager window. To add subscribers:

1. Select the **Profile Manager** → **Subscribers…** as shown in Figure 9-21.

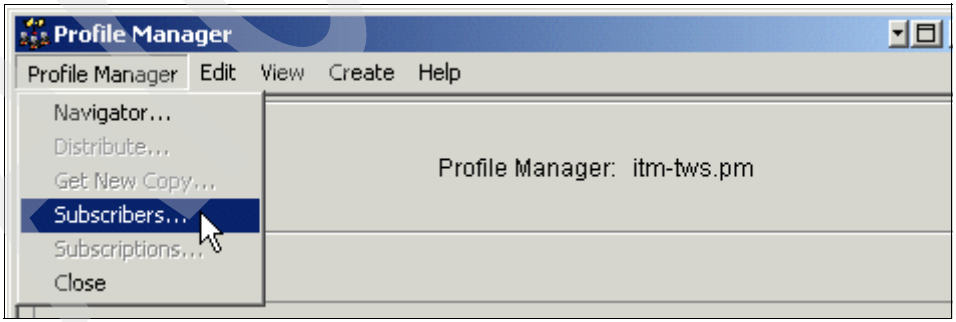

Figure 9-21 Adding subscribers to a profile manager

2. The Subscribers window opens as shown in Figure 9-22.

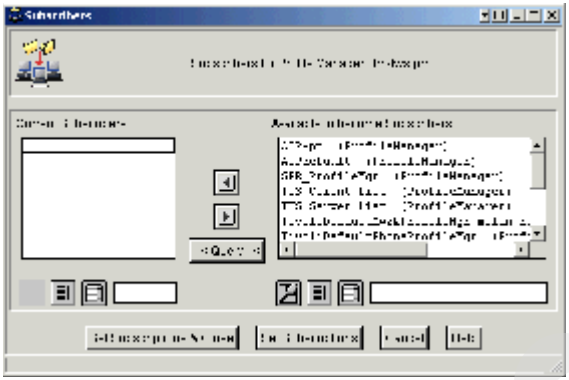

Figure 9-22 Subscribers window

3. Select the endpoint instances of the systems that need to be managed by Tivoli Monitoring from the Available to become Subscribers list on the right-hand side of the window (shift-click or control-click as necessary), then click the top button with an arrow pointing to the left. Integration with TWS typically means selecting all Fault Tolerant Agents and Extended Agents that have the LCF software installed on them.

   Endpoint instances are denoted by the (`Endpoint`) string at the end of their name. In the lab for this book, we selected the endpoint instances of milan and paris and pressed the top arrow button to move them into the Current Subscribers list as shown in Figure 9-23 on page 449.

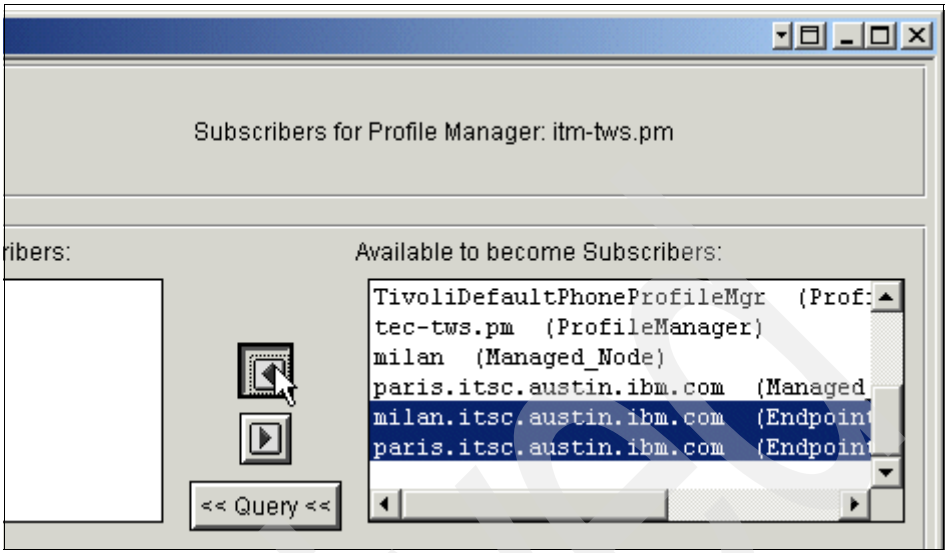

Figure 9-23 Selecting the endpoint instances to be subscribers

The selected endpoints are moved into the Current Subscribers list on the left-hand side of the window as shown in Figure 9-24.

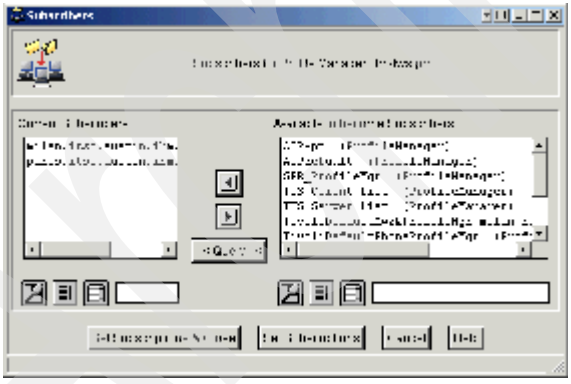

Figure 9-24 Endpoint instances selected to be subscribers

4. Click **Set Subscriptions & Close**. The endpoint objects are added to the Subscribers area of the profile manager as shown in Figure 9-25 on page 450.

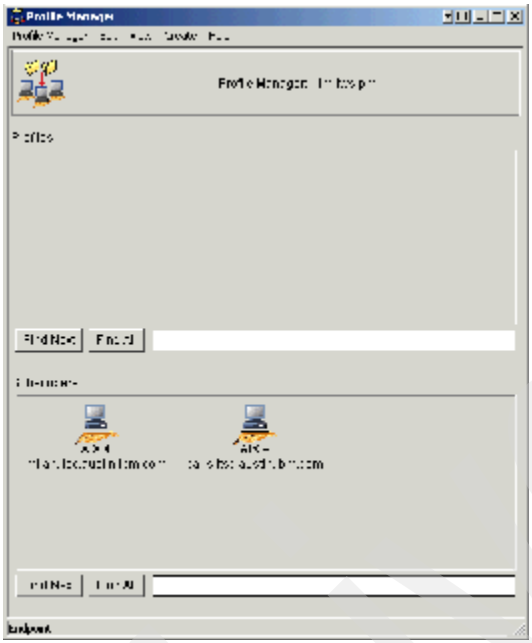

Figure 9-25 Endpoints added as subscribers to a profile manager

9.8.5 Adding profiles

After adding to the profile manager all the subscribers that will be monitored by the resource models, you can add the profiles to hold the resource models. We simplify the presentation in this book to focus more on the integration aspects by showing how to add one profile. We then describe in each resource model configuration section whether the resource model as designed needs a dedicated profile.

You start from the Profile Manager window. To add a profile to the profile manager:

1. Select **Create** → **Profile…** as shown in Figure 9-26.

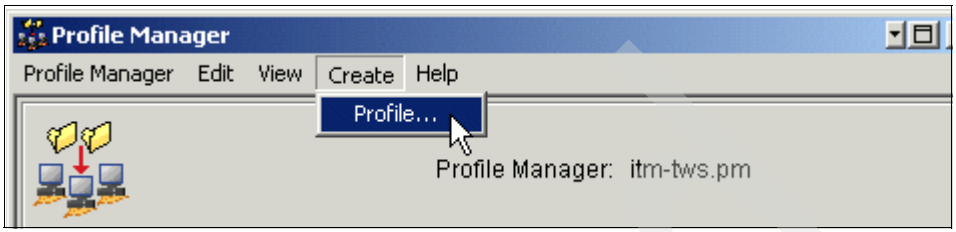

Figure 9-26 Creating a profile to hold resource models.

The Create Profile window opens as shown in Figure 9-27.

Figure 9-27 Create Profile window.

2. Enter a name for the profile in the Name/Icon Label text field, and ensure that the Tmw2kProfile profile type is selected in the Type list. In the lab for this book, we entered the name `itm-tws.pf` for our profile as shown in Figure 9-28 on page 452.

Figure 9-28 Naming the new profile.

3. Select **Create & Close**. The new profile object is placed into the Profiles section of the profile manager window as shown in Figure 9-29.

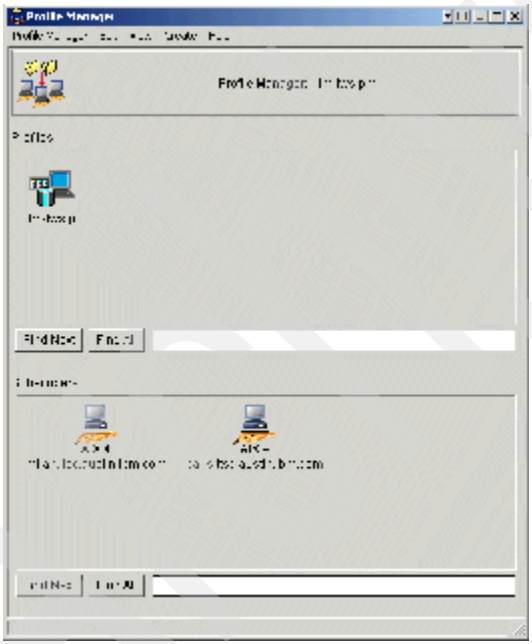

Figure 9-29 New profile in profile manager

4. Double-click the new profile as shown in Figure 9-30.

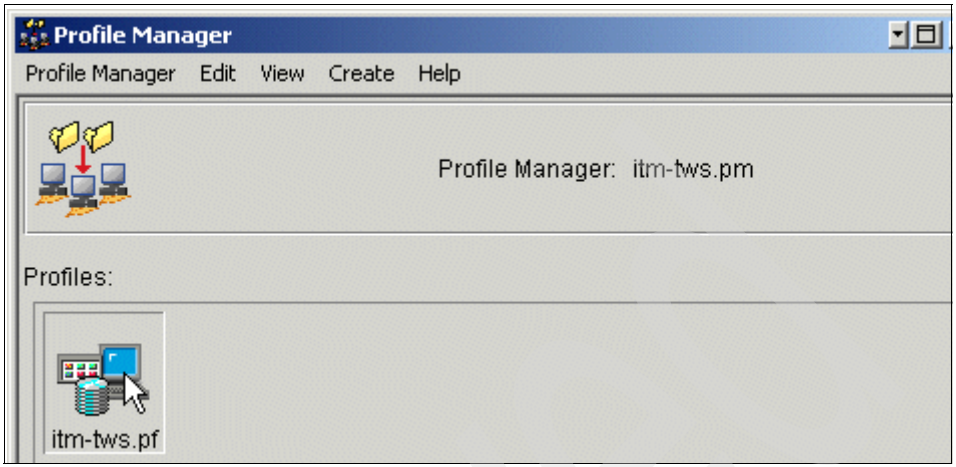

Figure 9-30 Opening the new profile.

5. The Tivoli Monitoring Profile window opens as shown in Figure 9-31 on page 454.

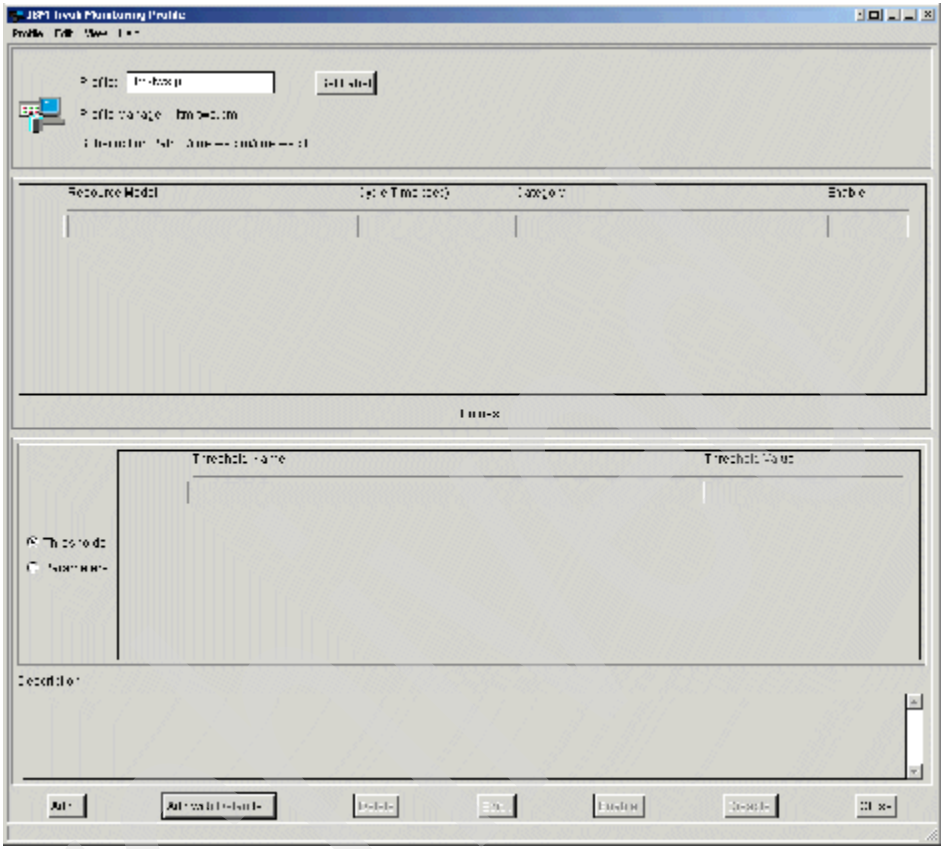

Figure 9-31 Tivoli Monitoring Profile window

6. We recommend creating any additional Tivoli Monitoring profiles that you will need at this time. Return to the Profile Manager window and repeat steps 1 on page 451 through 3 on page 452 for each additional Tivoli Monitoring profile.

   In the lab for this book, we created the following Tivoli Monitoring profiles:

   – itm-tws.pf (for following applications of resource models: netman application status, space free *TWSHome*, space used stdlist, space used schedlog)

   – hostavail.pf

   – appavail-tws.pf (for following applications of resource models: batchman application status, jobman application status, mailman application status)

The profile itm-tws.pf is distributed to all Fault Tolerant Agents and is used for three resource models:

– The standard resource model Process (what TWS Plus Module called Application Status, and the term we used during the design of the integration), with netman passed in as a parameter to the resource model.
– The standard resource model File System (what TWS Plus Module called Space Free, and the term we used during the design of the integration), with *TWSHome* passed in as the parameter to the resource model.
– The custom resource model Space Used, (what TWS Plus Module called Space Used, and the term we used during the design of the integration), with stdlist and schedlog passed in as parameters to the resource model.

In the lab for this book, itm-tws.pf is distributed to milan and paris.

The profile hostavail.pf is distributed only to the MDM server. Because we wanted this profile's resource model distributed to a different set of subscribers than the other two profiles (just one subscriber instead of all subscribers), the resource model must be held in a separate profile. It is used for only one resource model, the custom resource model HostAvail (Host status Availability).

The profile appavail-tws.pf is distributed to all Fault Tolerant Agents, and is used for a Process resource model. Because this resource model must operate on a separate schedule than the Process resource model used for netman, it must be held in a separate profile. The parameters for the batchman, mailman, and jobman daemons are passed to this resource model.

Figure 9-32 on page 456 shows how the profile manager looked in the lab for this book after we created all the recommended profiles.

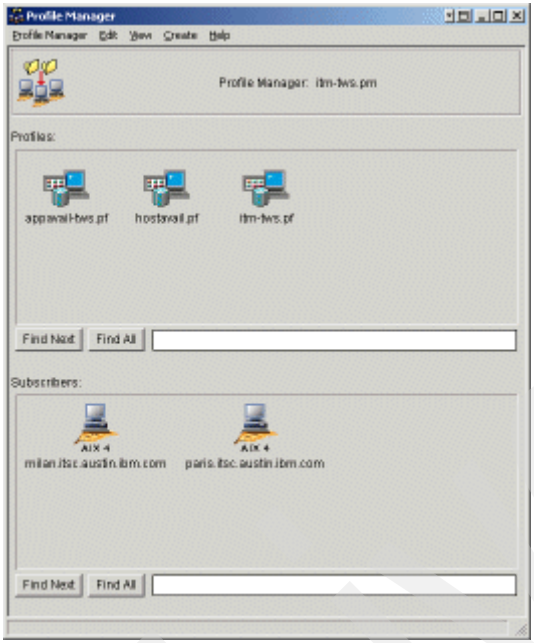

Figure 9-32 All recommend profiles created in itm-tws.pm profile manager

9.8.6 Creating HostAvail custom resource model

This section shows how to use the Resource Model Builder to create a simplified custom resource model to implement the Host status Availability design that is discussed in 9.7.3, “Host availability using a resource model” on page 410.

Note: This section is for the benefit of TWS administrators who want to follow a sample implementation of a simplified custom resource model. If you want to simply install the resource model, go to 9.8.5, “Adding profiles” on page 450.

Note that the custom resource models that are shown in this book implement only a simplified subset of features available to resource models. We do not recommend these custom resource models actually be used in production environments without review by experienced Tivoli Monitoring specialists to assess the fitness for a specific site’s requirements. For example, there is no code to handle non-existent hosts specified to the resource model, which would need to be added for production environments.

You should not use these instructions as a comprehensive tutorial. For detailed instructions on how to use the Resource Model Builder, see *IBM Tivoli Resource Model Builder User's Guide Version 1.1.3*, SC32-1391.

To create the custom resource model for Host status Availability:

1. Download and install Resource Model Builder 1.1.3 from:

   http://www-306.ibm.com/software/tivoli/resource-center/availability/code-monitor-resource.jsp

2. If you intend to import Tivoli Distributed Monitoring 3.7 monitors into resource models, you must install the win32gnu.dll file. Either temporarily install Tivoli Management Framework on your Windows PC as a primary TMR server and extract win32gnu.dll, or install the GNU software for NT DLL.

   ftp://microlib.cc.utexas.edu/microlib/nt/gnu/win32gnu.dll

   See the README file for details.

3. Start Resource Model Builder. The main window displays as shown in Figure 9-33.

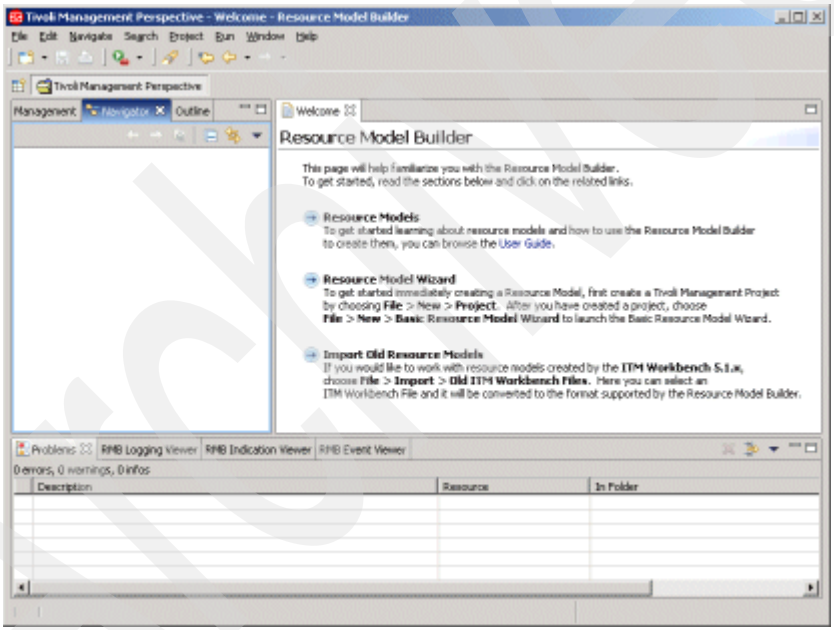

Figure 9-33 Main window of Resource Model Builder.

4. Select **File** → **New** → **Project...** as shown in Figure 9-34.

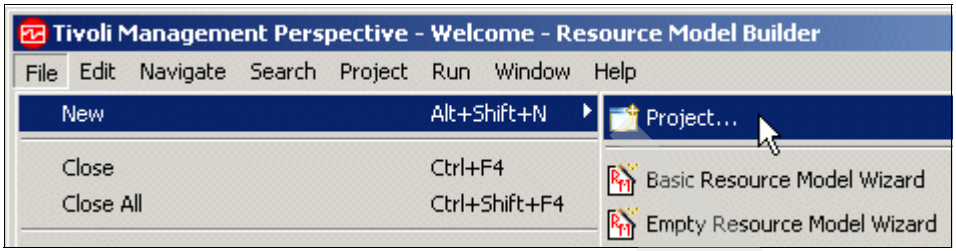

Figure 9-34 Adding a new project

A window with a Tivoli Management Project pre-selected displays as shown in Figure 9-35.

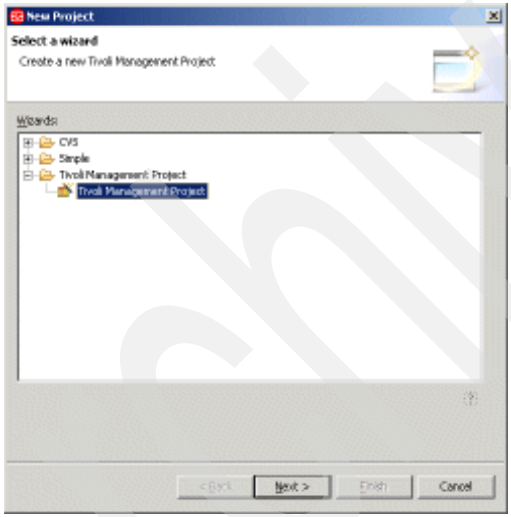

Figure 9-35 New Project window

5. Select **Next**.

6. In the window that displays, enter `HostAvail` in the Project name text field as shown in Figure 9-36.

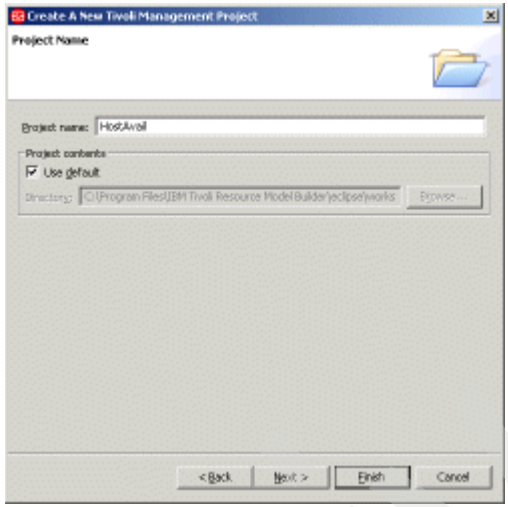

Figure 9-36 Create A New Tivoli Management Project: Project Name window

7. Select **Next**. The window displays as shown in Figure 9-37.

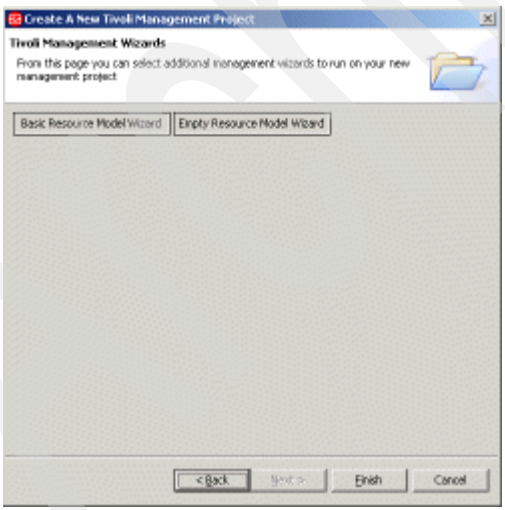

Figure 9-37 Create A New Tivoli Management Project: Tivoli Management Wizards

8. Select **Basic Resource Model Wizard**. A window displays as shown in Figure 9-38 on page 460.

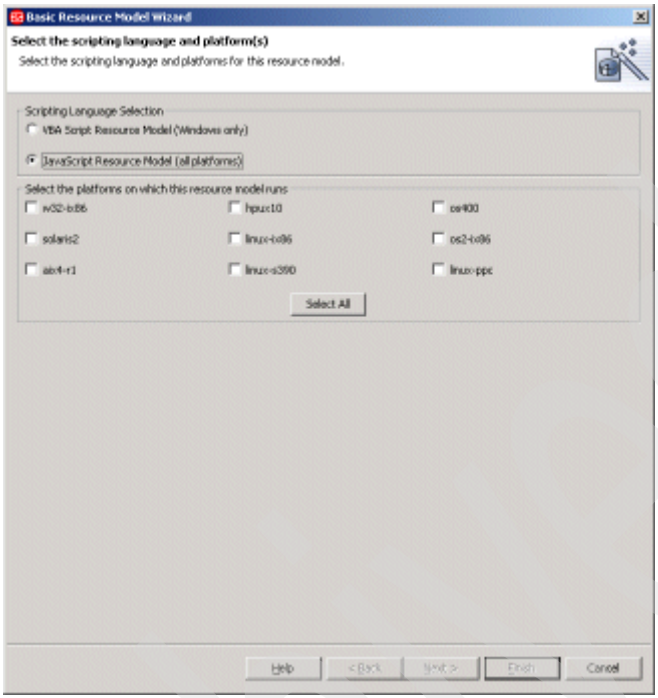

Figure 9-38 Basic Resource Model Wizard: Select the scripting language and platform(s)

9. Select all UNIX and Linux platforms as shown in Figure 9-39 on page 461.

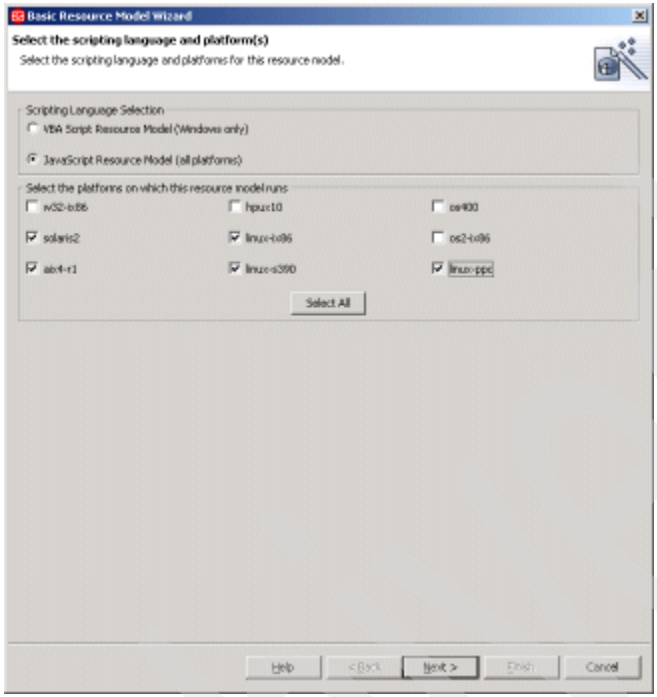

Figure 9-39 Selecting platforms on which the resource model runs

10. Select **Next**. A window displays as shown in Figure 9-40 on page 462.

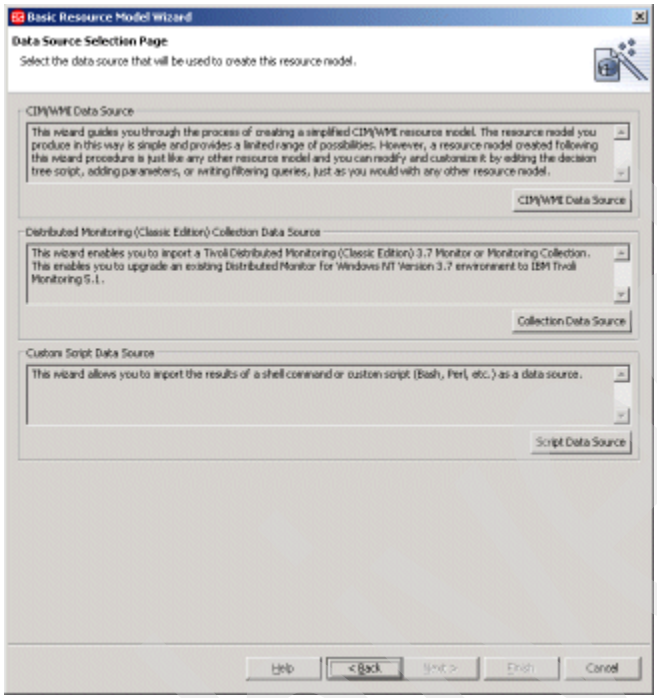

Figure 9-40 Basic Resource Model Wizard: Data Source Selection Page window.

11. Select **Script Data Source**. In the Script to Import text field, enter the path to the hostavail.sh file. You can download this file from the IBM Redbook Web site. For download instructions, see Appendix D, “Additional material” on page 687.

    Modify the Shell Command text field accordingly, as shown in Figure 9-41 on page 463.

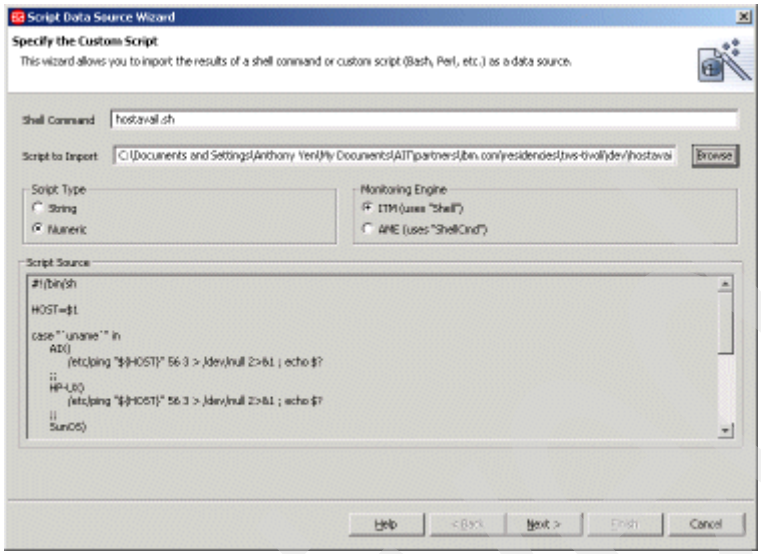

Figure 9-41 Specifying the hostavail.sh script to the resource model

12. Select **Next**. The Script Data Source Wizard: Specify the Event Triggering Conditions window displays. Select **ScriptResult** in the Selected Properties list on the left-hand side of the window, then click the top button that points to the right as shown in Figure 9-42.

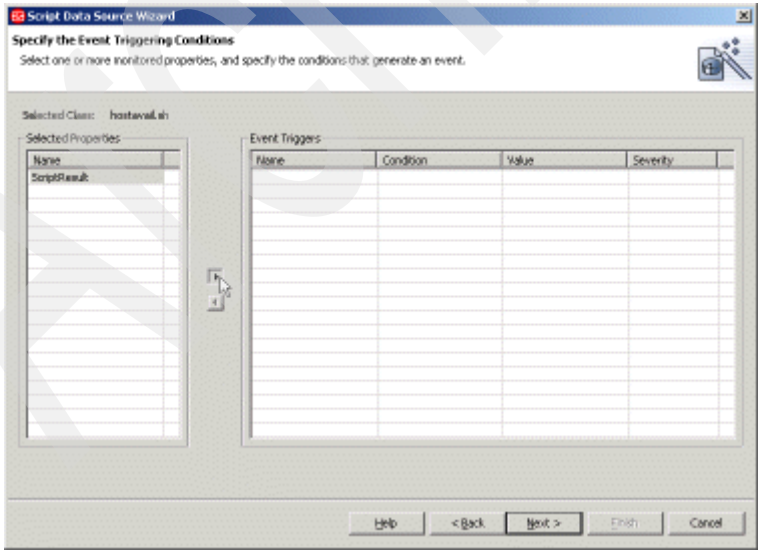

Figure 9-42 Adding ScriptResult as an Event Trigger

13. The Event Triggering Condition window displays. Select the equal sign from the ScriptResult menu, enter **1** in the ScriptResult text field, enter 3 in the Number of occurrences of this condition before the event is generated text field, and disable Send this event to TBSM, as shown in Figure 9-43.

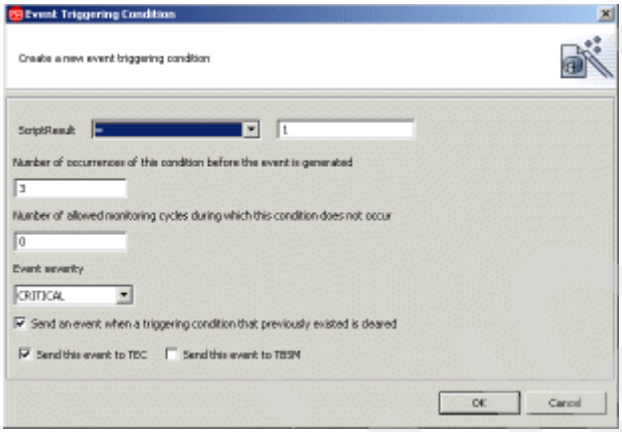

Figure 9-43 Event Triggering Condition window

14. Select **OK**. You are returned to the Script Data Source Wizard: Specify the Event Triggering Conditions window.
15. Select **Next**. The Script Data Source Wizard: Select the Properties to Log window displays. Select **ScriptResult** from the Available Properties list on the left-hand side of the window, and click the top button that points to the right. The ScriptResult property is moved to the Logged Properties list on the right-hand side of the window as shown in Figure 9-44 on page 465.

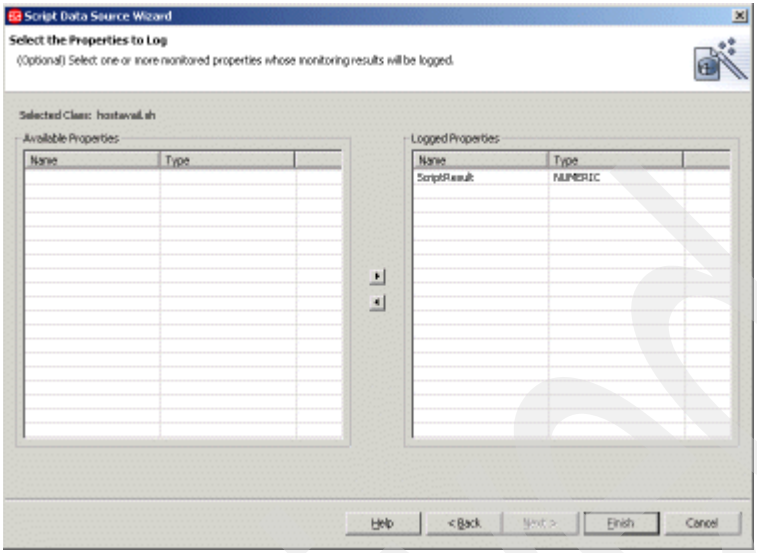

Figure 9-44 Script Data Source Wizard: Select the Properties to Log window

16. Select **Finish**. You are returned to the Basic Resource Model Wizard: Data Source Selection Page window, as shown in Figure 9-45 on page 466.

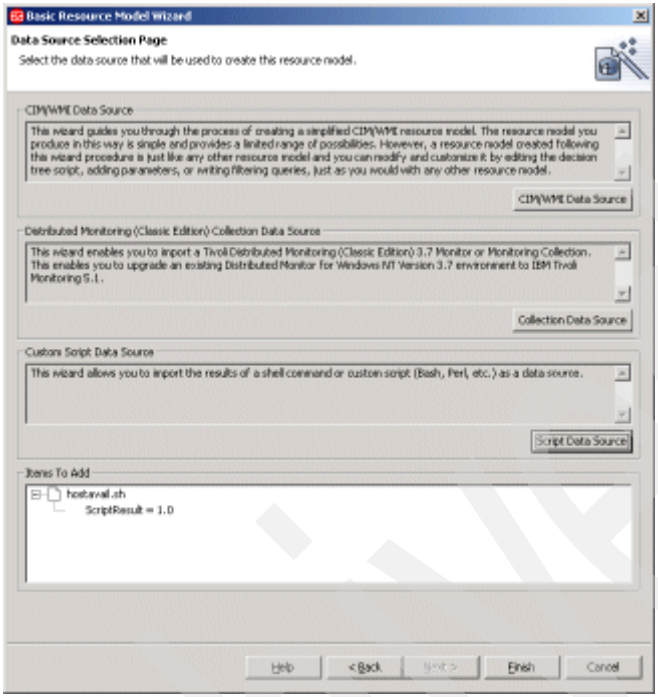

Figure 9-45 Basic Resource Model Wizard: Data Source Selection Page window

17. Select **Finish**. The Finalizing Creation: Summary Page window displays as shown in Figure 9-46 on page 467.

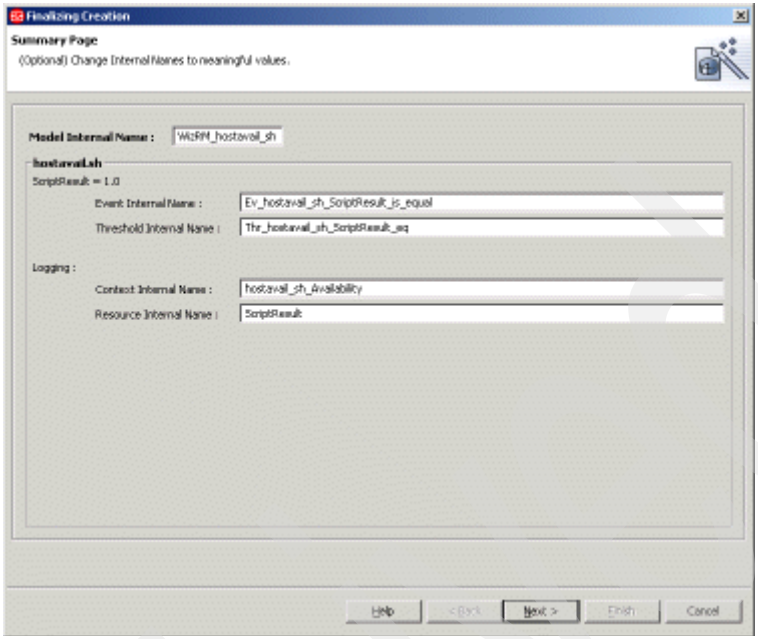

Figure 9-46 Finalizing Creation: Summary Page window

18. Enter `HostAvail` in the Model Internal Name text field and `PingFail` in the Event Internal Name text field, as shown in Figure 9-47 on page 468.

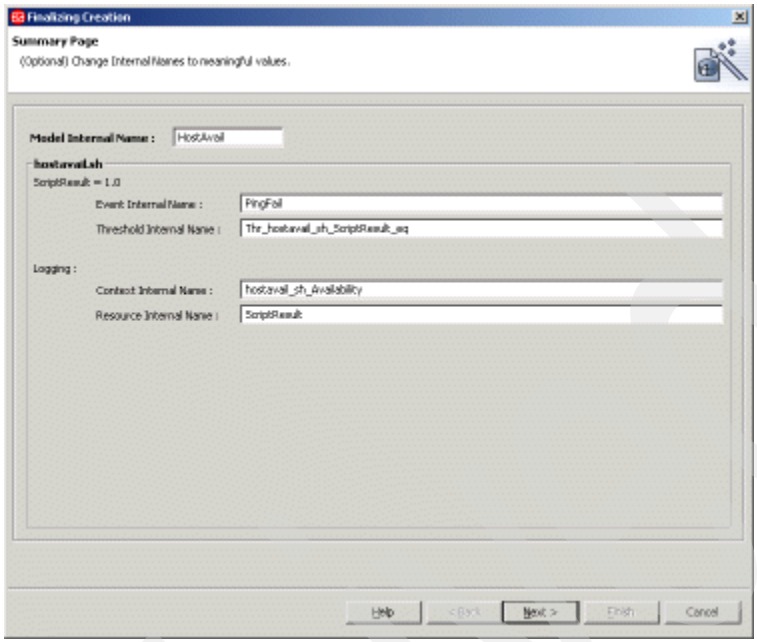

Figure 9-47 Specifying internal names

19. Select **Next**. The Finalizing Creation: Enter the Cycle Time window displays. Enter 120 in the text field, as shown in Figure 9-48 on page 469.

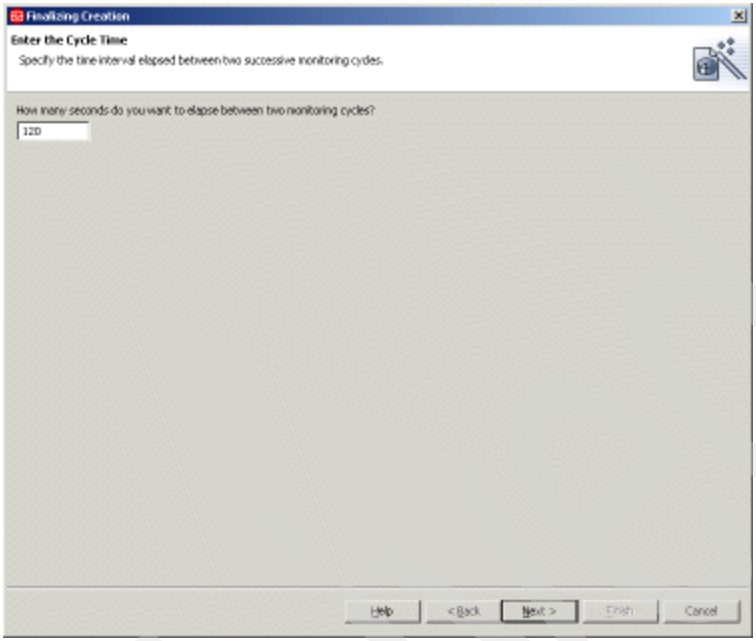

Figure 9-48 Finalizing Creation: Enter the Cycle Time window

20. Select **Next**. The Finalizing Creation: Specify the File Name window displays as shown in Figure 9-49 on page 470.

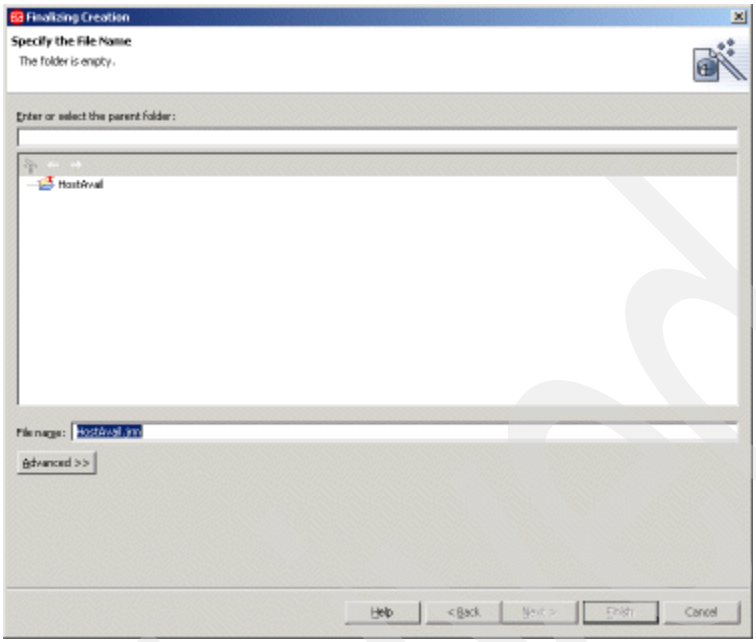

Figure 9-49 Finalizing Creation: Specify the File Name window

21. Select the HostAvail project folder. The Finish button is then enabled, as shown in Figure 9-50 on page 471.

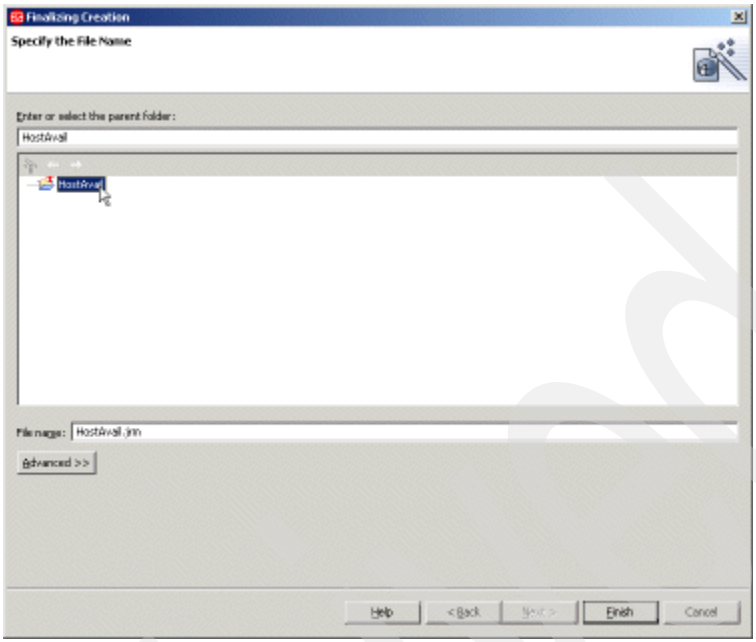

Figure 9-50 Selecting the project folder

22. Select **Finish**. You are returned to the Create A New Tivoli Management Project: Tivoli Management Wizards window.
23. Select **Finish**. You are returned to the main window.
24. Enter `Host status Availability` in the Descriptive name text field, `TWSIntegration` in the Category internal name text field, and `TWS Integration` in the Category descriptive name text field, as shown in Figure 9-51 on page 472.

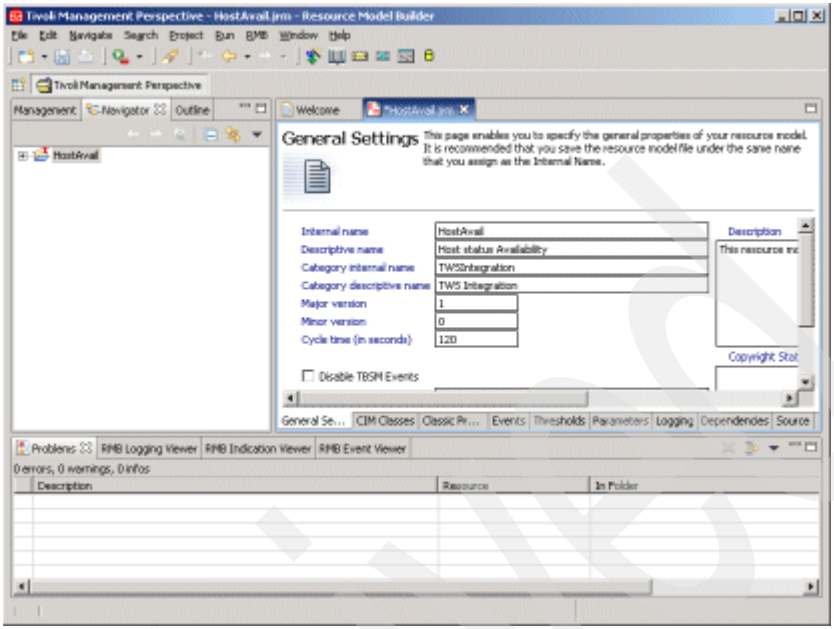

Figure 9-51 Configuring category names

25. Select the Parameters tab. The Parameters Editor pane displays as shown in Figure 9-52 on page 473.

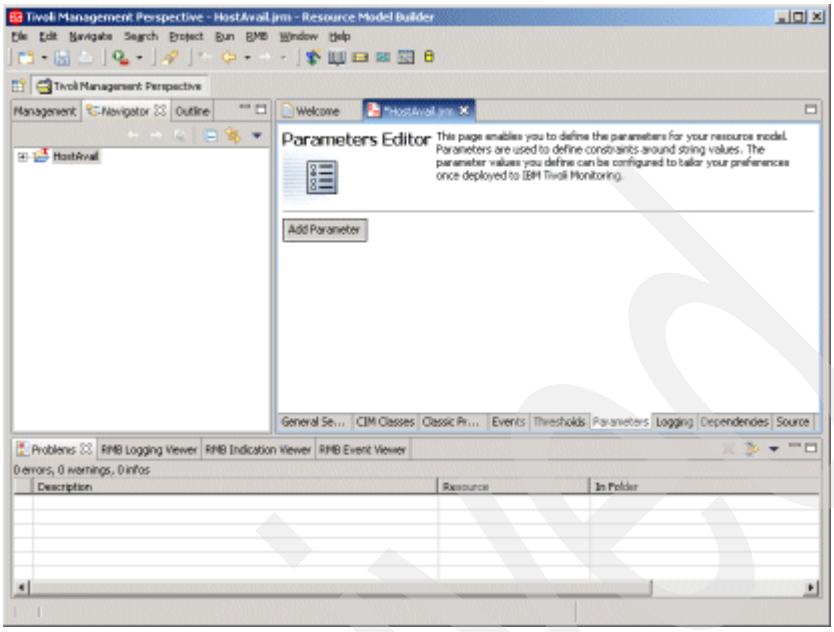

Figure 9-52 Parameters Editor pane

26. Select **Add Parameter**. A parameter is added to the pane.

27. Enter `IPList` in the Internal Name text field and select **Tab**. The Update Visit Tree References window displays as shown in Figure 9-53.

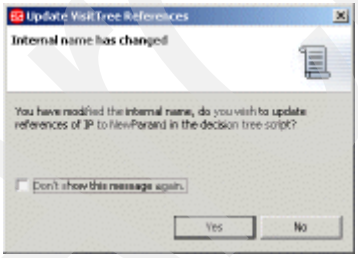

Figure 9-53 Update Visit Tree References window

28. Select **Yes**. You are returned to the Parameters Editor pane, which now shows the IPList parameter (Figure 9-54 on page 474).

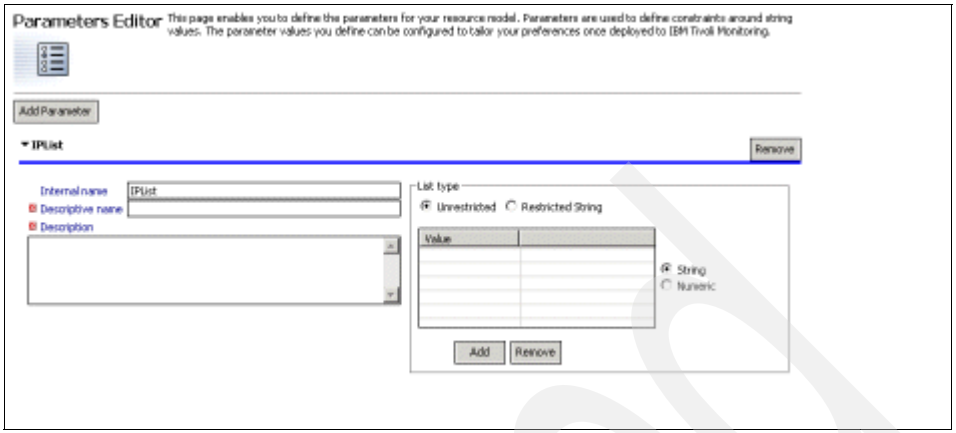

Figure 9-54 Parameters Editor pane with new IPList parameter configuration

29. Enter `Host to Be Checked` in Descriptive name text field and `List of IP addresses or IP hostnames to check for host status availability.` in the Description text field, as shown in Figure 9-55.

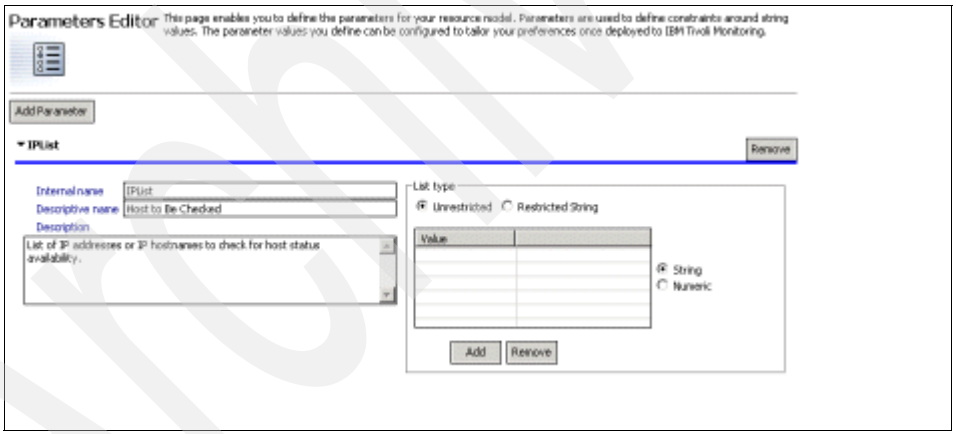

Figure 9-55 Parameters Editor pane with new parameter description

30. Select **Add**. The Add New Parameter Value window displays.

31. Enter `localhost` in the Value text field as shown in Figure 9-56 on page 475.

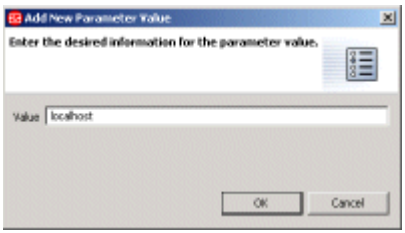

Figure 9-56 Add New Parameter Value window

32. Select **OK**. You are returned to the main window with the Parameters pane, which is updated with the new parameter value as shown in Figure 9-57.

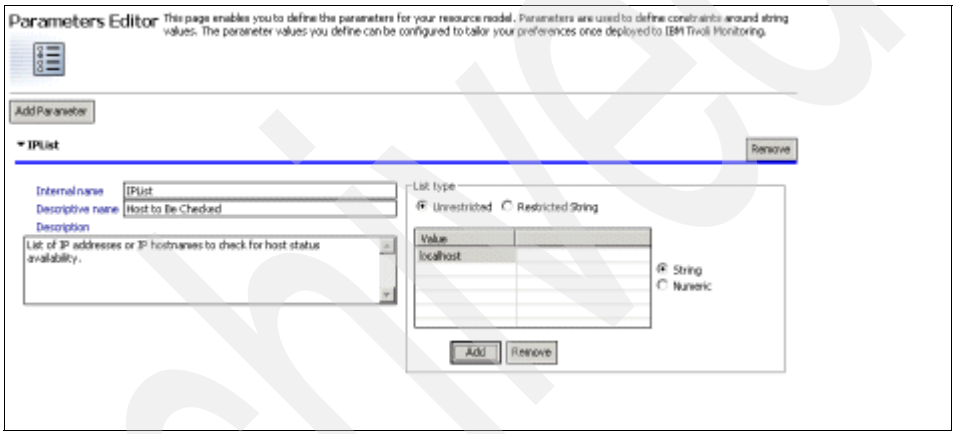

Figure 9-57 New parameter value added

33. Select the Events tab. The Events Editor pane displays as shown in Figure 9-58 on page 476.

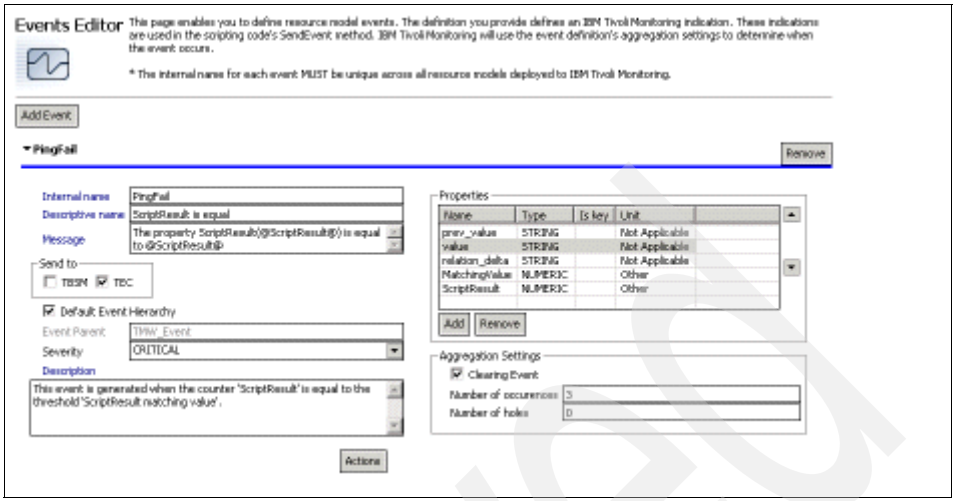

Figure 9-58 Events Editor pane.

Note: This book shows only a subset of the modifications that are needed for a production-grade resource model and focuses on the integration with TWS. For example, you could change the descriptive name and message of the PingFail event to text that is entirely specific to the host status availability function of the resource model.

34. In the Properties area of the Events Editor pane, select **Add**. The Add New Property window displays as shown in Figure 9-59.

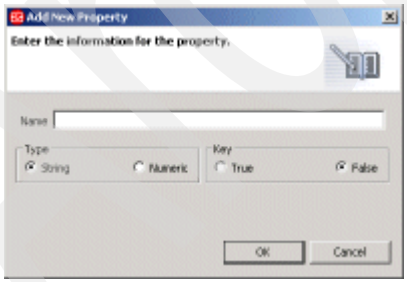

Figure 9-59 Add New Property window

35. Enter `host` in the Name text field, and select **True** in the Key area, as shown in Figure 9-60 on page 477.

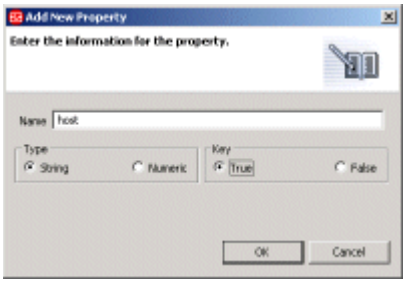

Figure 9-60 Adding a new property

36. Select **OK**. You are returned to the Events Editor pane with the new property listed in the Properties list as shown in Figure 9-61.

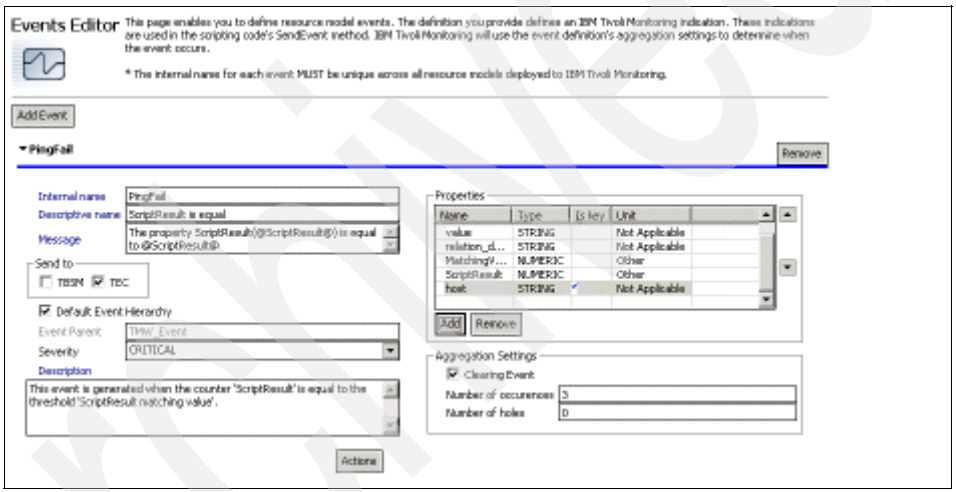

Figure 9-61 New property added to Events Editor pane

This new property is used in the decision tree script to let users identify a more precise cause of a Tivoli Monitoring event.

37. In the Message text field, append the following text to the text already in the field:

```
, because @host@ could not be pinged.
```

Figure 9-62 on page 478 shows the appended text highlighted in the Message text field.

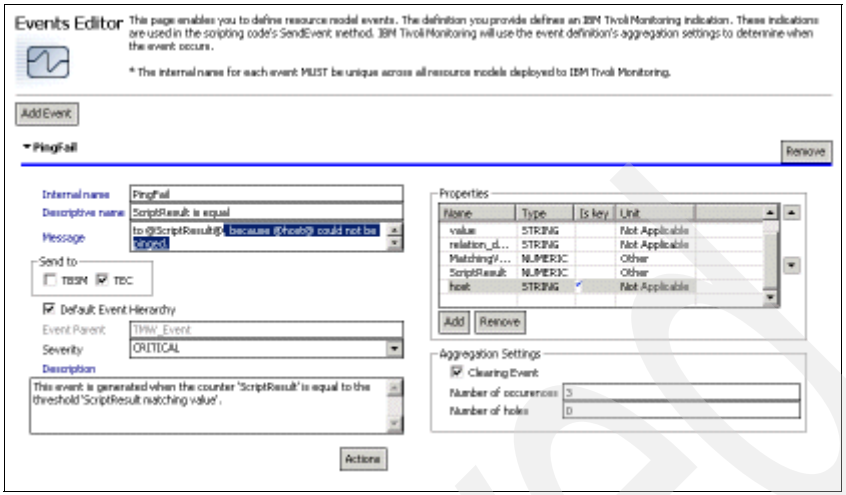

Figure 9-62 Adding event-specific text to the Message text field

38. Select the Source tab. The Javascript source pane with the decision tree script displays as shown in Figure 9-63.

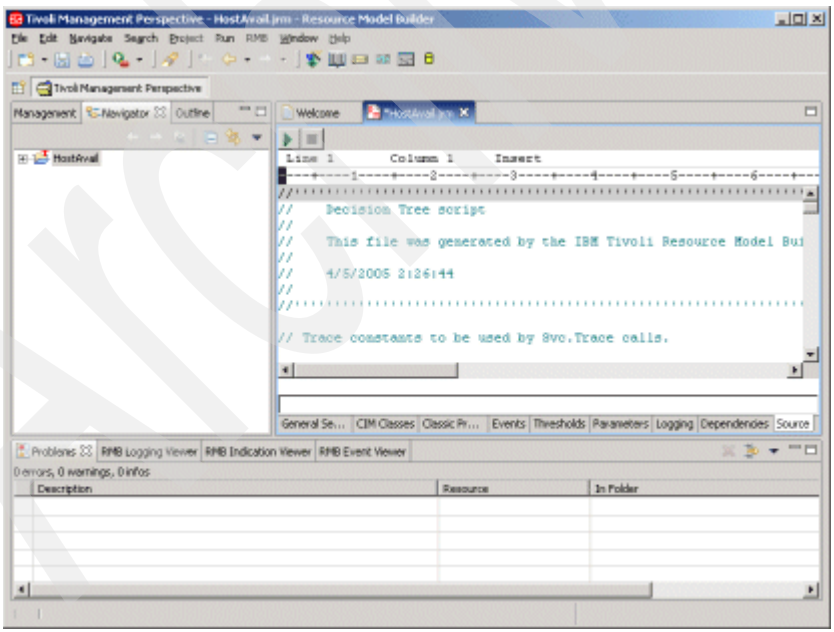

Figure 9-63 Source pane

39. Locate the VisitTree() function in the text editor pane, and replace it with the text that is included in Example 9-4.

Example 9-4 Modified VisitTree() function to use for host status availability resource model.

```
function VisitTree(Svc)
{
   Svc.Trace(TRACE_FINEST, TRACE_SOURCE + "Start VisitTree");

   //vars for data source: hostavail.sh
   var curScriptResult;
   var hashKey;
   var found;
   var k = 0;

   var OldScriptResult;

   var hPropTable;
   var ParamCount;
   var ParamIdx;
   var Different;

        ParamCount = Svc.GetStrParameterCount("IPList");
   hPropTable = Svc.CreateMap();

   for (k = 0; k < ParamCount; k++)
   {
      strHostName = Svc.GetStrParameter("IPList", k);
       //Implementation of the triggers for the script hostavail.sh
          Svc.RemoveMapAll(hPropTable);
       hashKey = "hostavail_sh_ScriptResult";
       curScriptResult = Svc.Shell("hostavail.sh "+strHostName);
       Svc.SetMapNumElement(hPropTable, "ScriptResult", curScriptResult);
       found = Svc.ExistsMapElement(hTableScriptResult, hashKey);
       if (found)
       OldScriptResult = Svc.GetMapNumValue(hTableScriptResult, hashKey);
       else
       OldScriptResult = 0;

       Svc.SetMapNumElement(hPropTable, "OldScriptResult", OldScriptResult);
          Svc.SetMapStrElement(hPropTable, "prev_value", OldScriptResult);
       Svc.SetMapStrElement(hPropTable, "value", curScriptResult);
       Svc.SetMapStrElement(hPropTable, "relation_delta", OldScriptResult - curScriptResult);

       if (curScriptResult == Svc.GetThreshold("Thr_hostavail_sh_ScriptResult_eq")) {
          Svc.SetMapNumElement(hPropTable, "MatchingValue",
Svc.GetThreshold("Thr_hostavail_sh_ScriptResult_eq"));
             Svc.SendEventEx ("PingFail", hPropTable);
       }
```

```
        Svc.SetMapStrElement(hPropTable,"key", "@");
           Svc.LogInstEx ("hostavail_sh_Availability","ScriptResult", hPropTable);

        Svc.SetMapNumElement(hTableScriptResult, hashKey, curScriptResult);
        }

   flagNotFirstRun = true;

   Svc.DestroyMap(hPropTable);

   return (0);

}
```

If you want to explore decision trees in more depth, refer to *IBM Tivoli Resource Model Builder User's Guide Version 1.1.3*, SC32-1391. You can also download this function from the IBM Redbook Web site. Refer to Appendix D, “Additional material” on page 687 for download instructions.

40. Select **File** → **Save** to save the project. Until the HostAvail.jrm resource model file is saved, its tab has an asterisk (*) before the file name as shown in Figure 9-64. Verify that the tab for the HostAvail.jrm resource model file does not have an asterisk in it before proceeding to the next step.

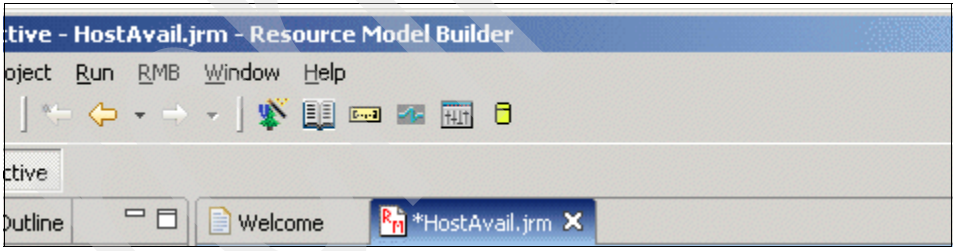

Figure 9-64 Resource model file tab showing unsaved status

41. Select **File** → **Export**. The Export: Select window displays as shown in Figure 9-65 on page 481.

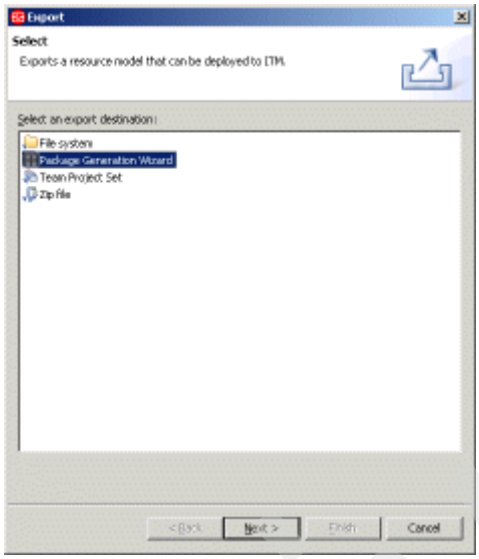

Figure 9-65 Export: Select window

42. Verify that the Package Generation Wizard export destination is selected, and then select **Next**. A window displays as shown in Figure 9-66.

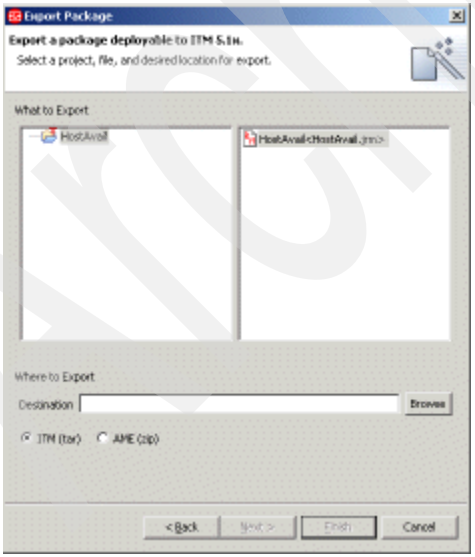

Figure 9-66 Export Package: Export a package deployable to ITM 5.1.x window

43. Enter a fully qualified path to a file in the Destination text field, ensure that ITM (tar) radio is selected, ensure that the HostAvail project folder in the list on the left-hand side of the window is selected, and ensure the HostAvail.jrm resource model is selected in the list on the right-hand side of the window.

    In the lab for this book, we created the file hostavail.tar as shown in Figure 9-67.

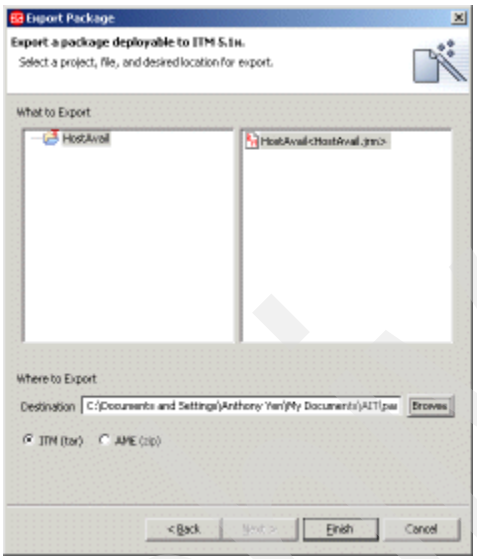

Figure 9-67 Specifying export file

44. Select **Finish**. The export file is created, and you are returned to the main window.

9.8.7 Adding HostAvail custom resource model

After the custom resource model is created and exported into a tar archive by the Resource Model Builder, you need to add it to Tivoli Monitoring. Adding resource models can only be performed as an authorized user on the primary TMR server. In the lab for this book, we added resource models after logging into milan as root user.

To add a custom resource model:

1. Transfer the tar archive created to the primary TMR server.
2. As an authorized user, use the **wdmrm** command to add the resource model.

   In the lab for this book, we used the **wdmrm** command as shown in Example 9-5.

Example 9-5 Adding a resource model to Tivoli Monitoring

```
milan:/#  wdmrm -add hostavail.tar

IBM Tivoli Monitoring - Adding new resource model

Parsing configuration file HostAvail.conf ...
Configuration file successfully parsed.
Checking for event redefinition...
Starting resource HostAvail registration ...
The resource HostAvail has been successfully stored.
Registration completed.

Copying HostAvail.cat msgfile ...
Copying HostAvail.cat zipfile ...

Installation completed.
```

3. Verify that the resource model is added by listing all resource models with the **wdmrm** command and looking for the HostAvail resource model. In the lab for this book, we ran the **wdmrm** command under the root user on milan as shown in Example 9-6.

Example 9-6 Verifying addition of a custom resource model.

```
milan:/# wdmrm -list

IBM Tivoli Monitoring - Listing resource models


Resource -> ASPDiskMirroringStatus400
   NLS name        : ASP Disk Mirroring Status
   product_id      : none
   major_version   : 2
   minor_version   : 5
   platform        : os400
   message catalog : ASPDiskMirroringStatus400
   zip file        : ASPDiskMirroringStatus400.zip
    .
    .
    .
Resource -> HostAvail
```

```
NLS name        : Host status Availability
product_id      : none
major_version   : 1
minor_version   : 0
platform        : solaris2\linux-ix86\aix4-r1\linux-s390\linux-ppc
message catalog : HostAvail
zip file        : HostAvail.zip
 .
 .
 .
```

9.8.8 Creating a SpaceUsed custom resource model

The procedure to create a custom resource model for the SpaceUsed resource model is similar to the procedure that is shown in 9.8.6, “Creating HostAvail custom resource model” on page 456.

Note: If you want to duplicate the effort of creating this custom resource model, first read *IBM Tivoli Resource Model Builder User's Guide Version 1.1.3*, SC32-1391 and *IBM Tivoli Monitoring Version 5.1.1 Creating Resource Models and Providers*, SG24-6900.

You can download this resource model from the IBM Redbook Web site. Refer to Appendix D, “Additional material” on page 687 for download instructions.

9.8.9 Adding a SpaceUsed custom resource model

The procedure to add the SpaceUsed resource model is similar to the procedure that is shown in 9.8.7, “Adding HostAvail custom resource model” on page 482, except that you use the spaceused.tar file with the **wdmrm** command instead of using the hostavail.tar file. In the lab for this book, we transferred the spaceused.tar file to milan, logged into milan as root user, and ran the **wdmrm** command as shown in Example 9-7.

Example 9-7 Adding the Space Used custom resource model.

```
milan:/#  wdmrm -add spaceused.tar

IBM Tivoli Monitoring - Adding new resource model

Parsing configuration file SpaceUsed.conf ...
Configuration file successfully parsed.
Checking for event redefinition...
Starting resource SpaceUsed registration ...
The resource SpaceUsed has been successfully stored.
```

```
Registration completed.

Copying SpaceUsed.cat msgfile ...
Copying SpaceUsed.cat zipfile ...

Installation completed.
```

9.8.10 Configuring the Host Status Availability resource model

When you have created the custom resource model and added them to Tivoli Monitoring, you need to add the custom resource model to the itm-itws.pf profile and then configure it. To configure the host status availability resource model from the Tivoli Monitoring Profile window:

1. In the hostavail.pf profile, select **Add**. The Add Resource Models to Profile window displays as shown in Figure 9-68.

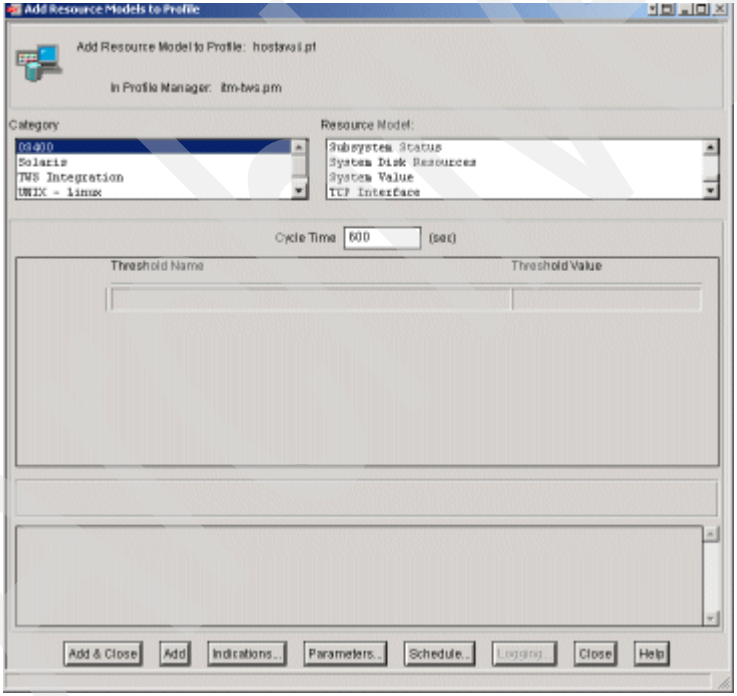

Figure 9-68 Add Resource Models to Profile window.

2. Select **TWS Integration** in the Category list on the left-hand side of the window, and ensure that **Host status Availability** is selected in the Resource Model list on the right-hand side of the window as shown in Figure 9-69.

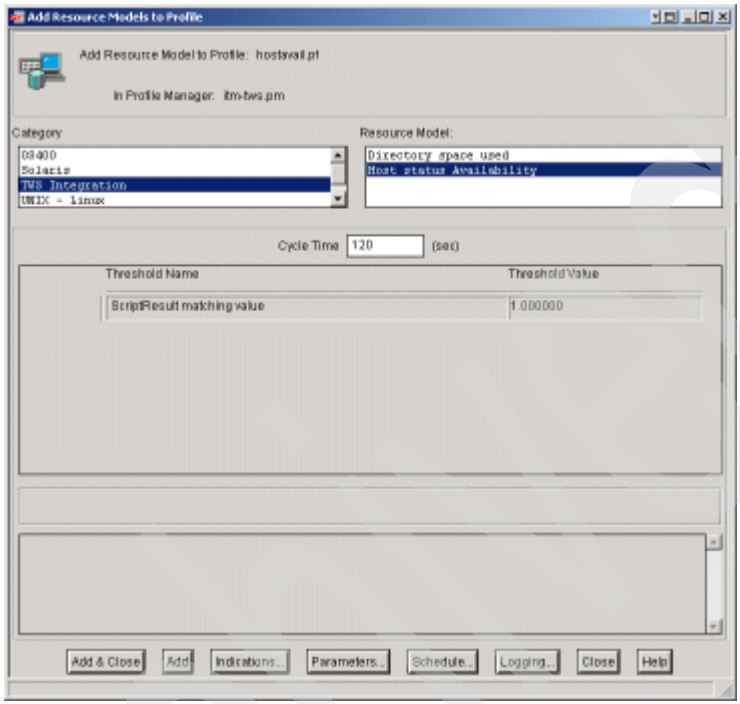

Figure 9-69 Selecting the Host status Availability resource model

3. Select **Add & Close**. You are returned to the Tivoli Monitoring Profile window for the hostavail.pf profile, with the Host status Availability resource model added to the Resource Model list, as shown in Figure 9-70 on page 487.

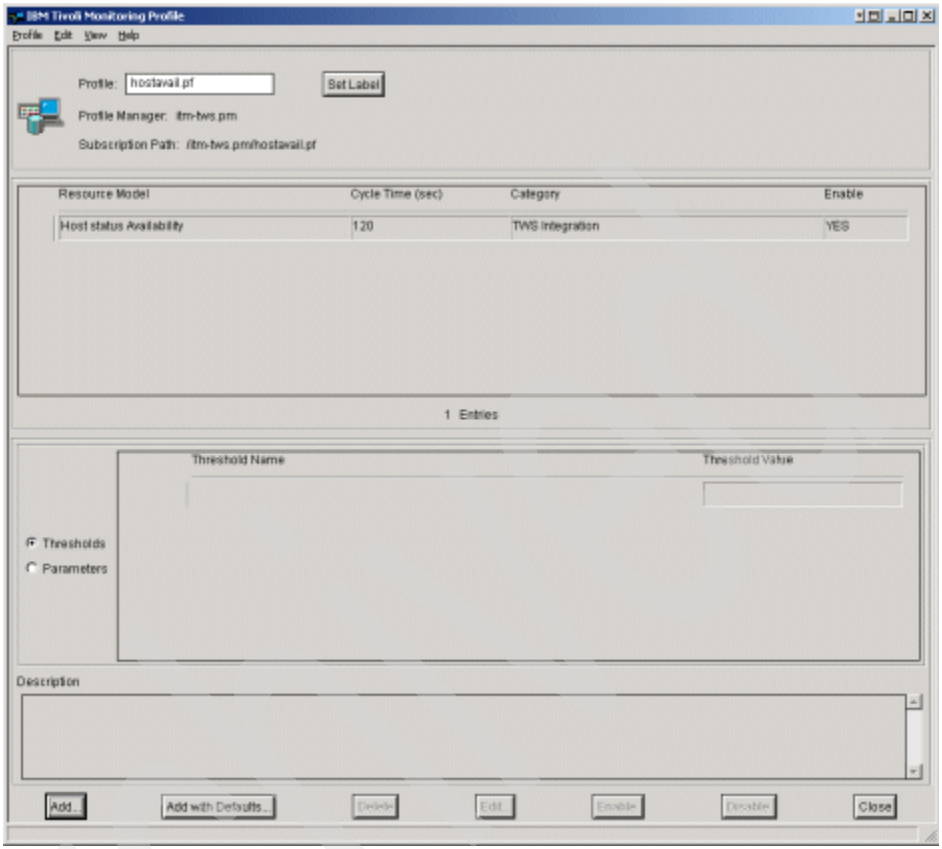

Figure 9-70 Host status Availability resource model added to ITM profile

4. Double-click the **Host status Availability** resource model. The Edit Resource Model window displays as shown in Figure 9-71 on page 488.

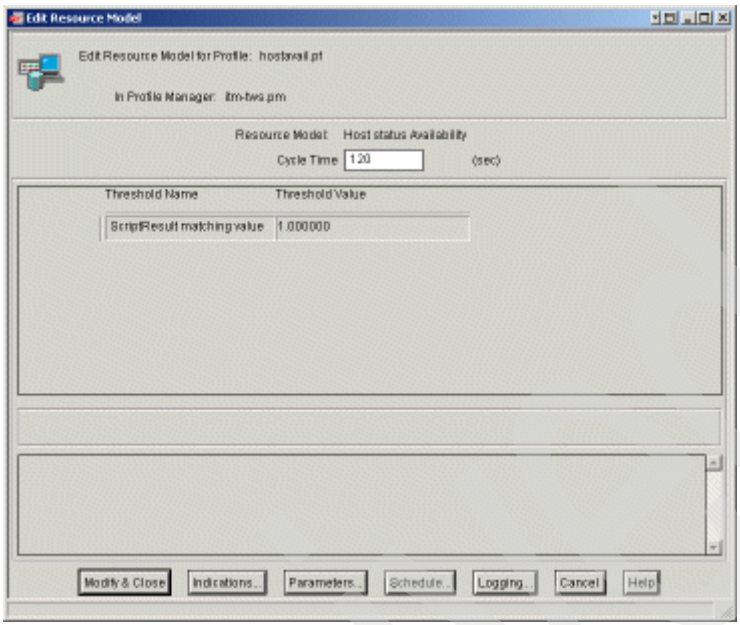

Figure 9-71 Edit Resource Model window

5. Click **Parameters**. The Parameters window displays as shown in Figure 9-72.

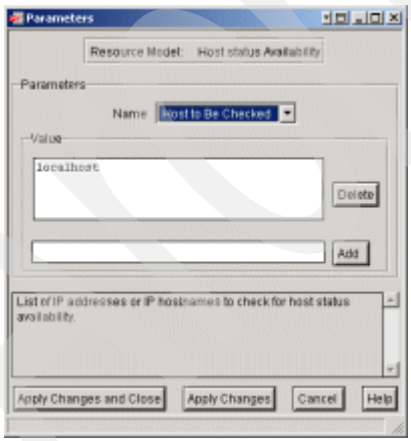

Figure 9-72 Parameters window

6. The localhost entry is for testing purposes. It can be left in the parameter list or removed. To remove it, highlight the entry in the Value list and select **Delete**.

In the lab for this book, we chose to leave the localhost entry in the parameter list.

7. Enter the IP host name or address of the servers that need to be pinged in the Value text field, and then select **Add**.

   In the lab for this book, we added paris as shown in Figure 9-73.

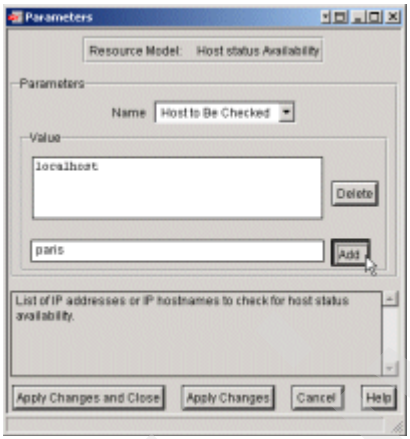

Figure 9-73 Adding an FTA to the list of hosts to be checked

The host is added to the Value list. Add any other hosts that are in the scheduling network in the same manner. The Value list displays the added hosts as shown in Figure 9-74.

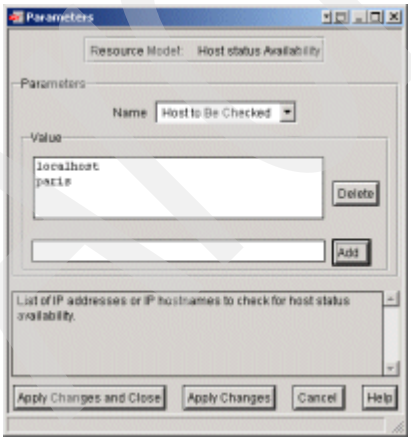

Figure 9-74 List of FTAs added to the list of hosts to be checked

There is practical limit of about 20 to 30 servers that use this resource model. Beyond that limit, the serialized checking can impact the cycle time, among

other adverse effects. The solution is to either re-implement the resource model to check all hosts in the parameter list asynchronously or to use Tivoli NetView.

8. Select **Apply Changes and Close**. You are returned to the Edit Resource Model window.
9. Select **Modify & Close**. You are returned to the Tivoli Monitoring Profile window for the itm-itws.pf profile.
10. Select **Profile** → **Distribute Defaults...** as shown in Figure 9-75.

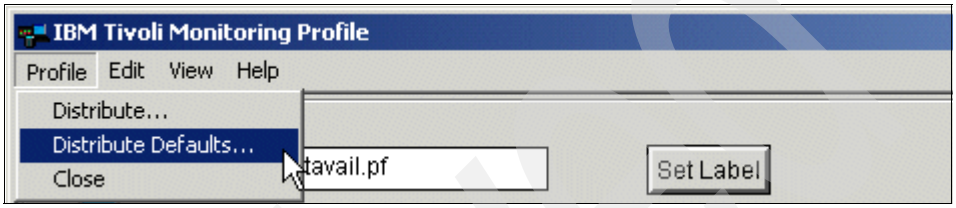

Figure 9-75 Setting distribution defaults of a Tivoli Monitoring profile

The Set Distribution Defaults window displays as shown in Figure 9-76.

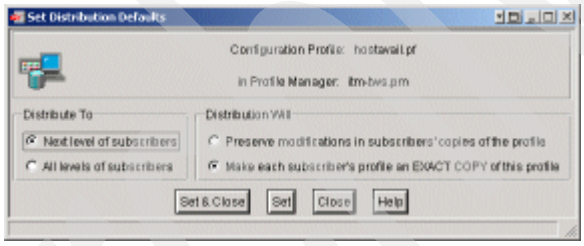

Figure 9-76 Set Distribution Defaults window

11. In the **Distribute To** group, select All levels of subscribers. In the Distribution Will group, select Make each subscriber's profile an EXACT COPY of this profile, as shown in Figure 9-77.

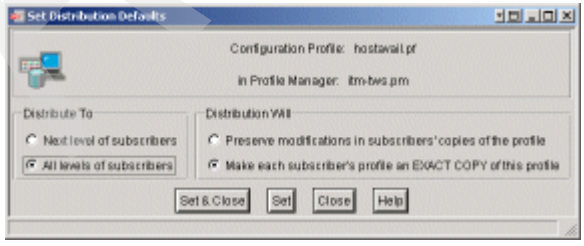

Figure 9-77 Setting distribution defaults for hostavail.pf profile

12. Select **Set & Close**. You are returned to the Tivoli Monitoring Profile window for the itm-itws.pf profile.
13. Select **Close**. You are returned to the Profile Manager window for the itm-tws.pm profile manager, as shown in Figure 9-78.

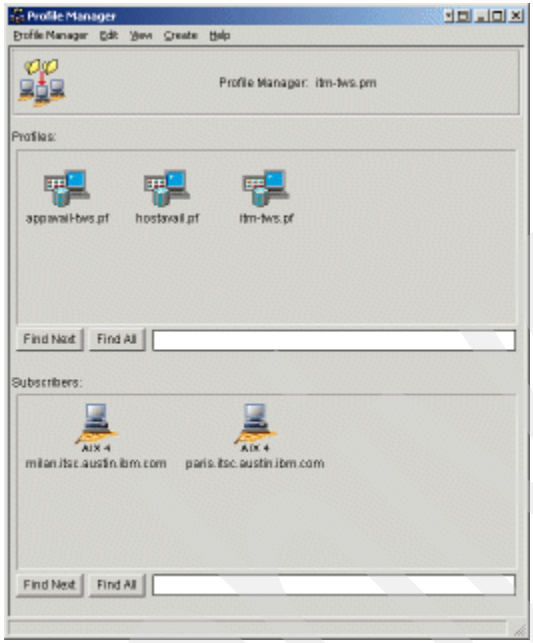

Figure 9-78 Profile Manager window for the itm-tws.pm profile manager

9.8.11 Configuring Application Status (batchman) resource model

From the itm-tws.pm profile manager window, use the following steps to configure application status checking for the batchman process.

Note: This procedure configures the application status monitoring for the batchman, mailman, and jobman processes. Each process is passed as a parameter to a single resource model.

1. Double-click the **appavail-tws.pf profile**, as shown in Figure 9-79.

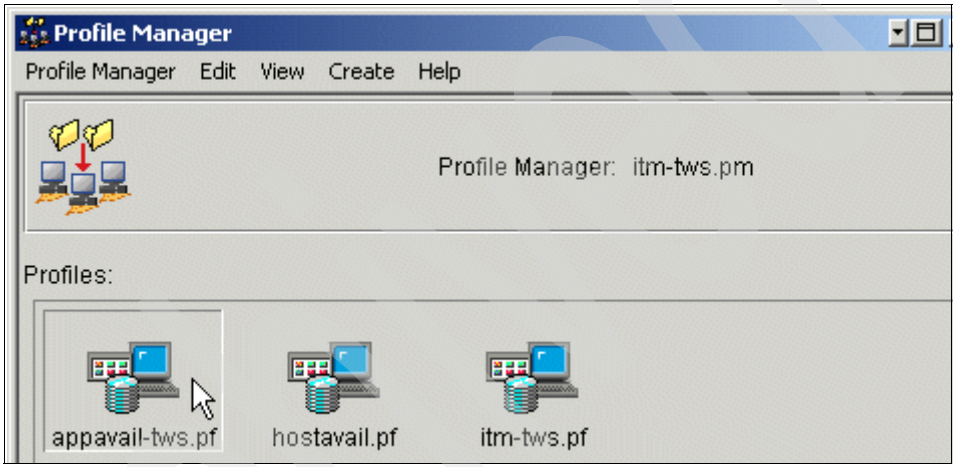

Figure 9-79 Opening the appavail-tws.pf profile

The Tivoli Monitoring Profile window for the appavail-tws.pf profile opens as shown in Figure 9-80 on page 493.

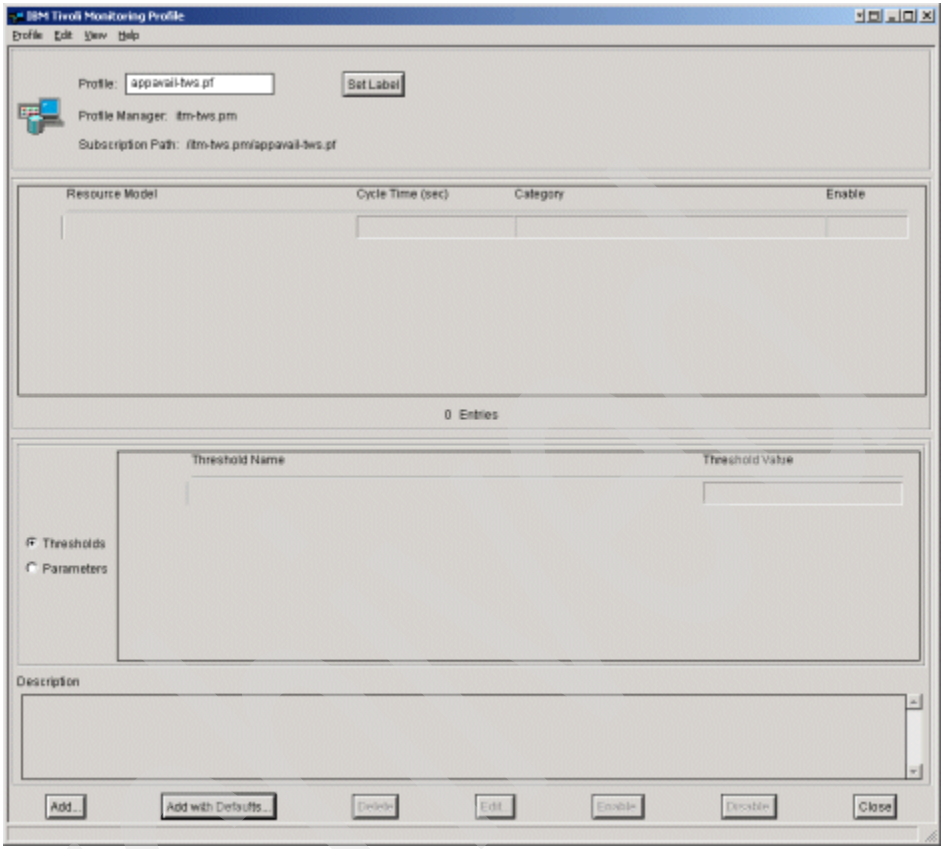

Figure 9-80 Tivoli Monitoring Profile window for the appavail-tws.pf profile

2. Select **Add**. The Add Resource Models to Profile window opens as shown in Figure 9-81 on page 494.

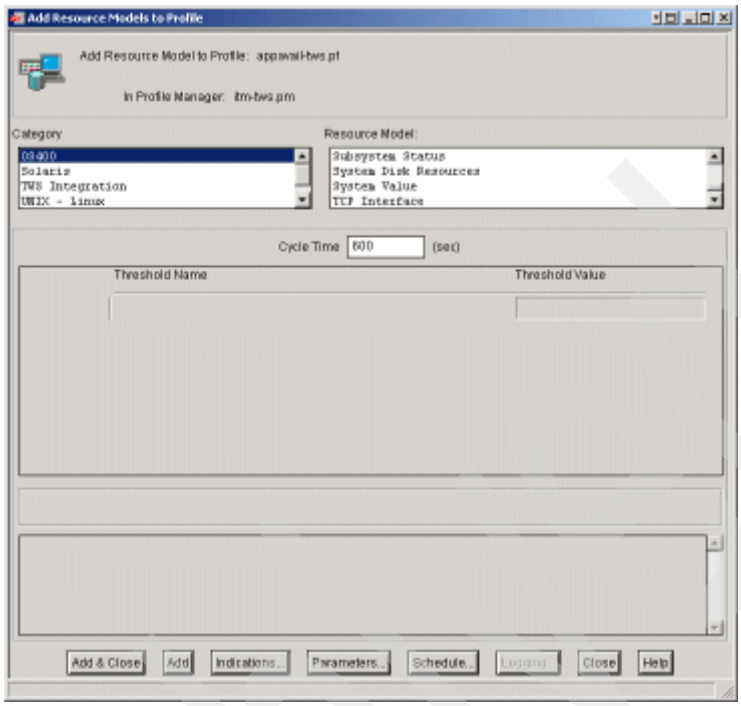

Figure 9-81 Add Resource Models to Profile window

3. Select **UNIX – Linux** in the Category list on the left-hand side of the window, and ensure that Process is selected in the Resource Model list on the right-hand side of the window as shown in Figure 9-82 on page 495.

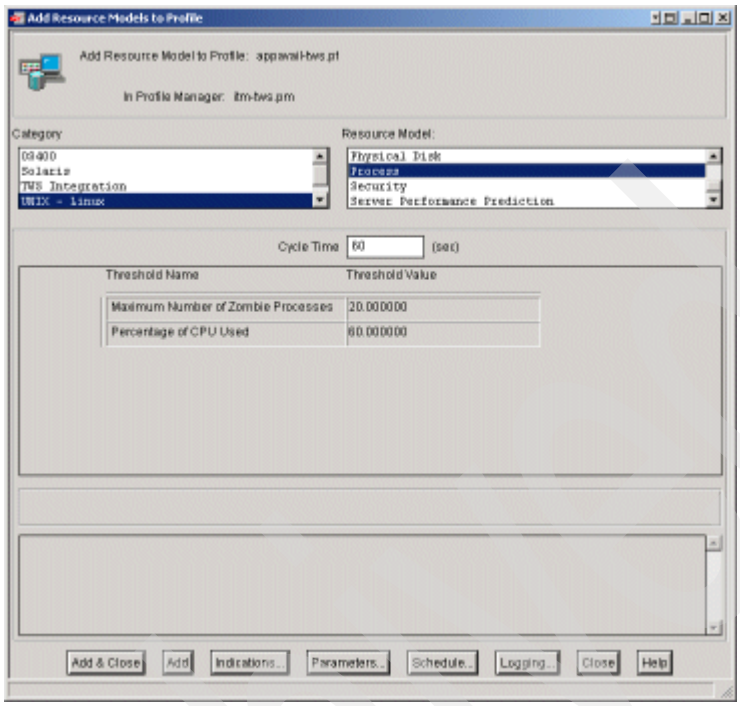

Figure 9-82 Selecting Process resource model

4. Select **Parameters**. The Parameters window opens as shown in Figure 9-83.

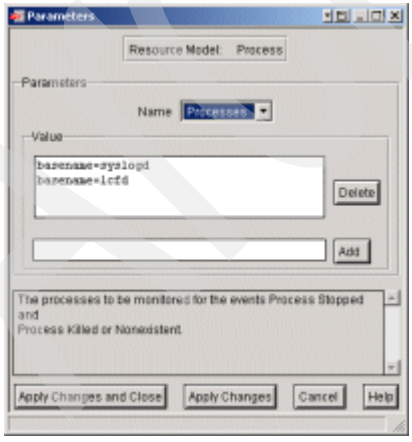

Figure 9-83 Parameters window

5. Shift select both entries (**syslogd** and **lcfd**) in the Value list, and then select **Delete** as shown in Figure 9-84.

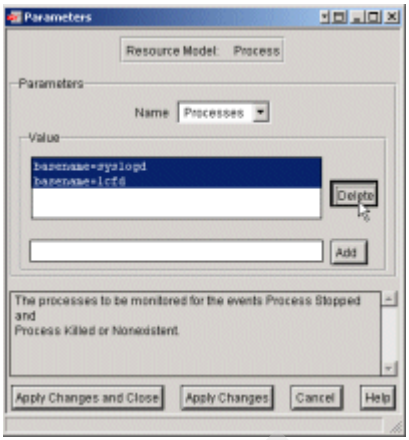

Figure 9-84 Deleting default parameters from Process resource model

6. Enter the text `basename=batchman` in the Value text field, and then select **Add**. The value is added to the Value list as shown in Figure 9-85.

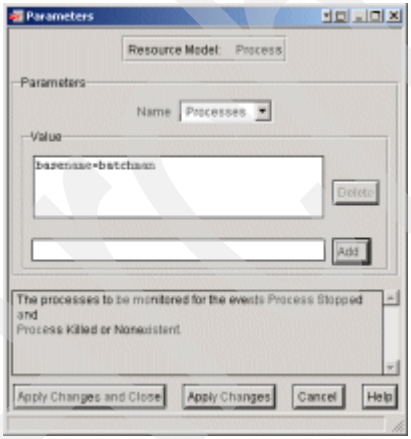

Figure 9-85 Adding batchman to parameter list

7. Add the parameter values `basename=mailman` and `basename=jobman` using the same procedure. The Value list displays the new values as shown in Figure 9-86 on page 497.

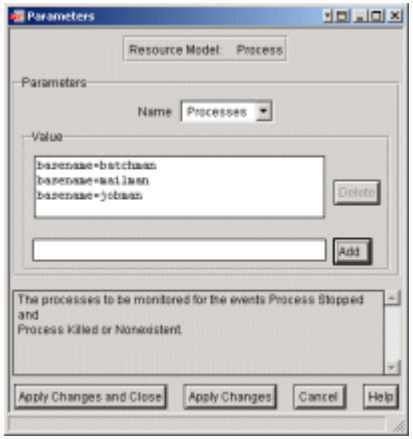

Figure 9-86 Parameters for batchman, mailman, and jobman processes

8. Select **Apply Changes and Close**. You are returned to the Add Resource Models to Profile window.
9. You do not want the monitoring to be in effect during Jnextday, so you need to configure the resource model to stop running during Jnextday. Select **Schedule**. The Scheduling window opens as shown in Figure 9-87.

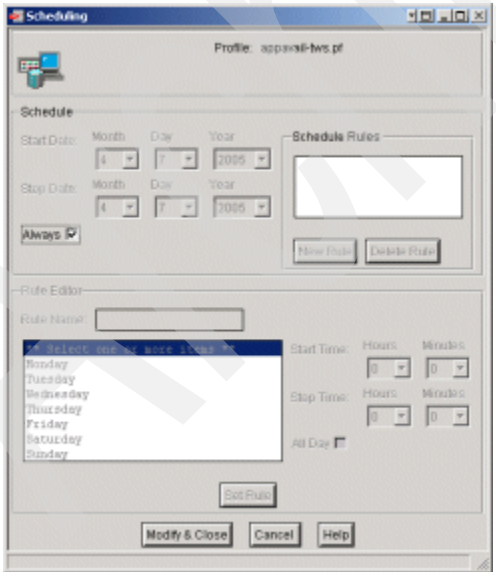

Figure 9-87 Scheduling window

10. In the Schedule group, clear Always, and then set the Stop Date to the last day of 2037, as shown in Figure 9-88.

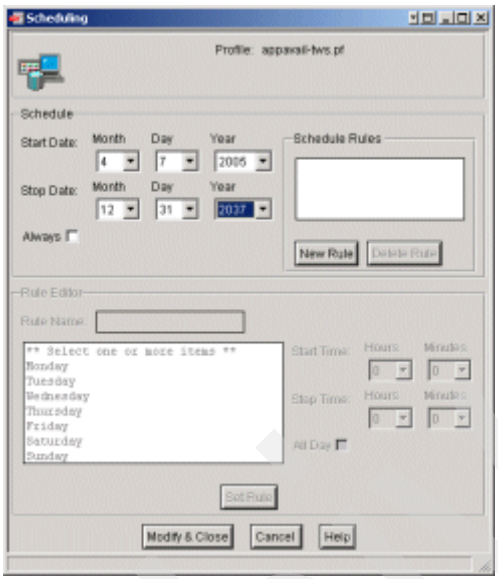

Figure 9-88 Setting schedule stop date

11. Select **New Rule** to enable the Rule Editor section as shown in Figure 9-89 on page 499.

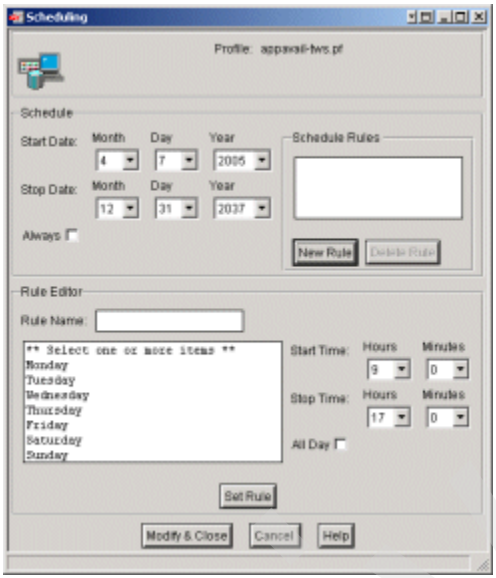

Figure 9-89 Enabling the Rule Editor

12. In the lab for this book, we left Start of Day at the default 0600h. Because Tivoli Monitoring's scheduling operates on just the normal "wall clock" 24 hour period, we have to define two rules, one for the time period before Jnextday and one for the time period after Jnextday.

    Enter `PreJnextday` in the Rule Name text field, shift-select all days of the week in the list of days, set the Start Time to 0 Hours and 0 Minutes, and set the Stop Time to 5 Hours and 55 Minutes, as shown in Figure 9-90 on page 500.

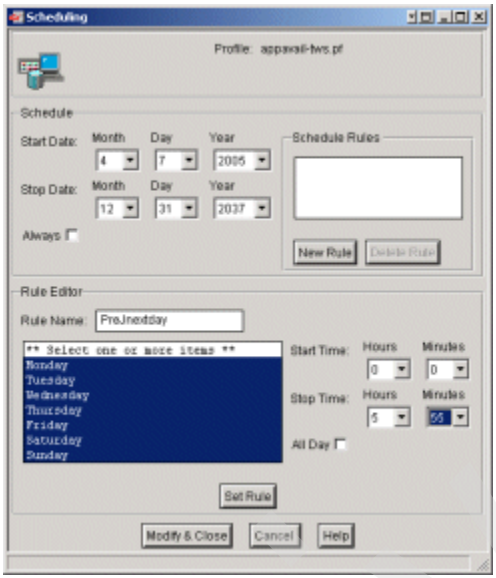

Figure 9-90 Configuring the PreJnextday rule

13. Select **Set Rule**. The rule is entered into the Schedule Rules list and the Rule Editor is disabled as shown in Figure 9-91.

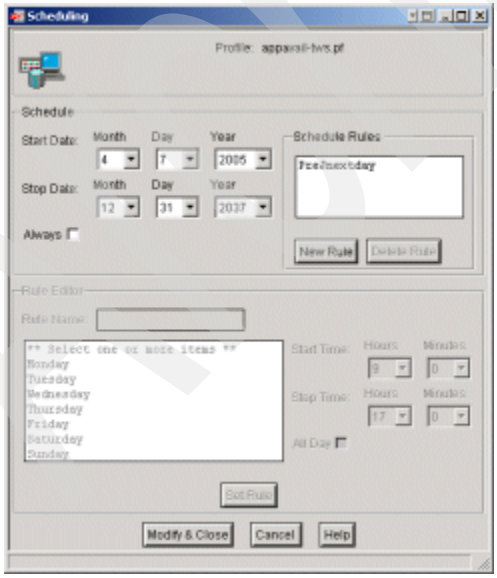

Figure 9-91 PreJnextday rule added to schedule

14. Select **New Rule**, enter `PostJnextday` in the Rule Name text field, select all days of the week in the list of days, set the Start Time to 6 hours and 10 minutes, and set the Stop Time to 24 hours (the minutes pop-up menu dims when you select 24 hours to indicate the midnight following the Start Time), as shown in Figure 9-92.

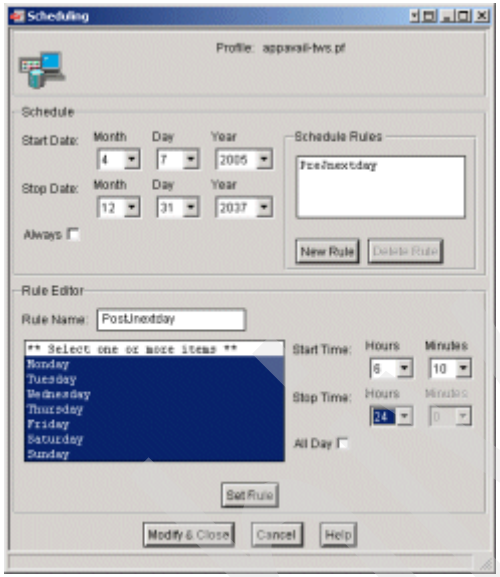

Figure 9-92 Configuring the PostJnextday scheduling rule

15. Select **Set Rule**. The Scheduling window displays both rules in the Schedule Rules list as shown in Figure 9-93 on page 502.

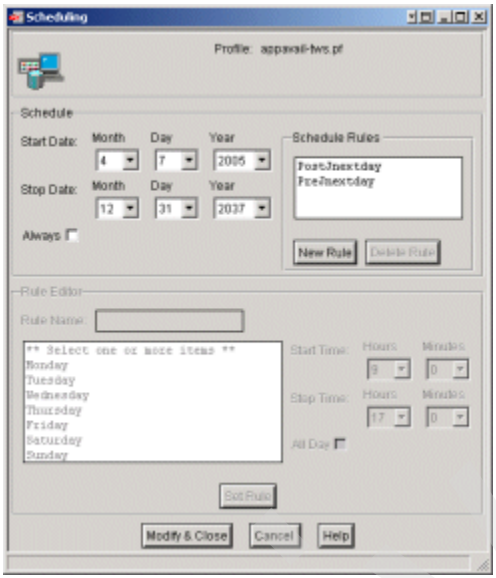

Figure 9-93 Configured scheduling rules for Tivoli Monitoring resource model

16. Select **Modify & Close**. You are returned to the Add Resource Models to Profile window for the appavail-tws.pf profile.

17. Select **Add & Close**. You are returned to the Tivoli Monitoring Profile window for the appavail-tws.pf profile, with the Process resource model added to the Resource Model list, as shown in Figure 9-94 on page 503.

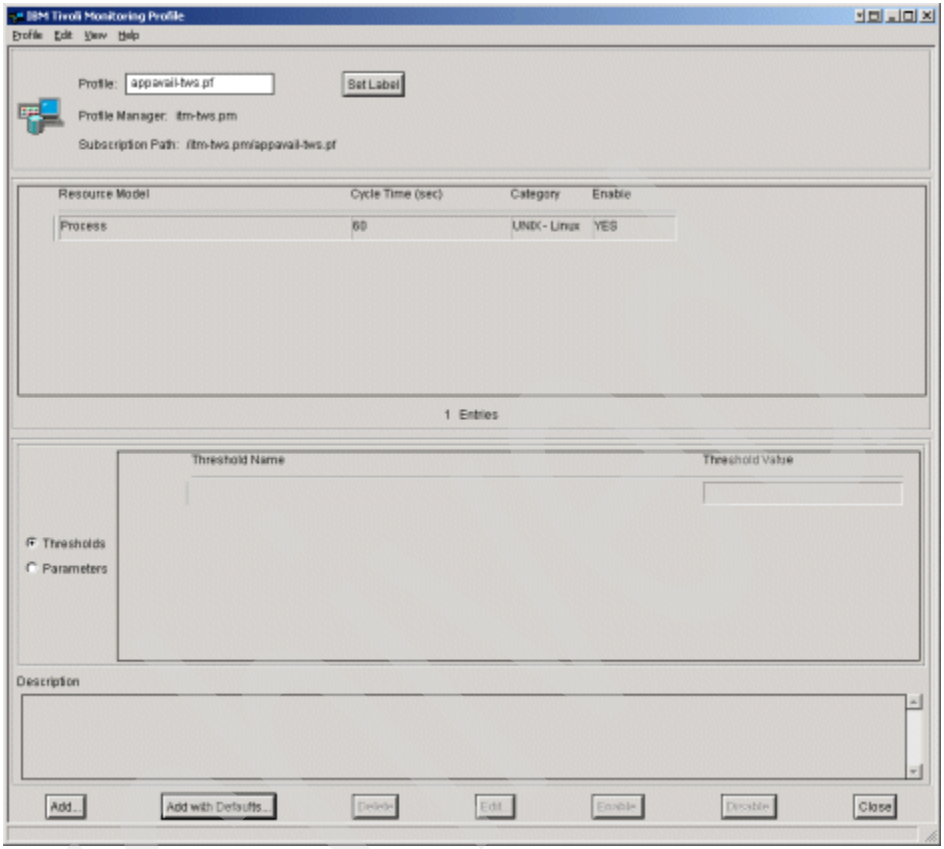

Figure 9-94 Process resource model added to appavail-tws.pf profile

18.Select **Profile** → **Distribute Defaults...** as shown in Figure 9-95.

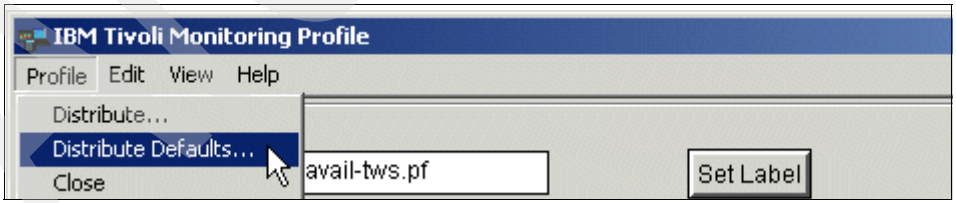

Figure 9-95 Setting distribution defaults for appavail-tws.pf profile

The Set Distribution Defaults window is displayed as shown in Figure 9-96 on page 504.

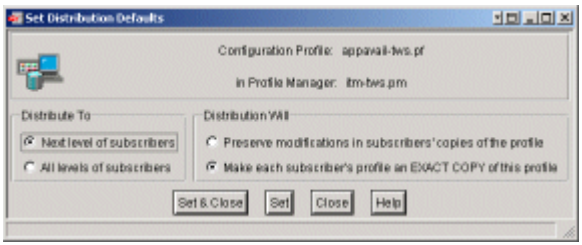

Figure 9-96 Set Distribution Defaults window for appavail-tws.pf profile

19. In the Distribute To group, select **All levels of subscribers**. In the Distribution Will group, select **Make each subscriber's profile an EXACT COPY of this profile**, as shown in Figure 9-97.

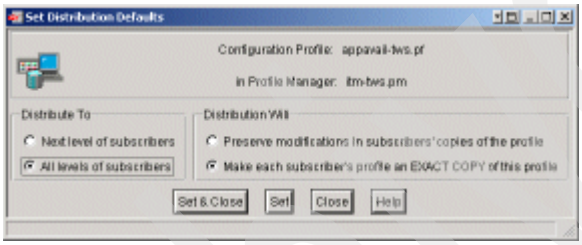

Figure 9-97 Setting distribution defaults for appavail-tws.pf profile

20. Select **Set & Close**. You are returned to the Tivoli Monitoring Profile window for the appavail-tws.pf profile.
21. Select **Close**. The Tivoli Monitoring Profile window for the appavail-tws.pf profile is closed, and you are returned to the Profile Manager window for the itm-tws.pm profile manager, as shown in Figure 9-98 on page 505.

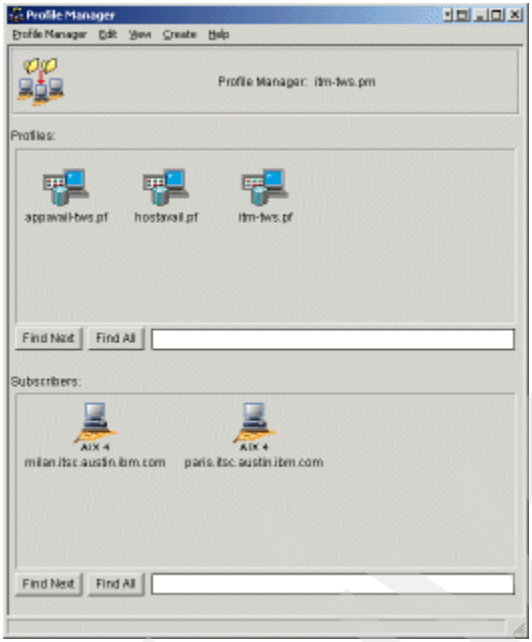

Figure 9-98 Configured appavail-tws.pf profile in itm-tws.pm profile manager

9.8.12 Configuring Application Status (netman) Resource Model

Configuration of the Process resource model for application status monitoring of the netman process is similar to the configuration that is shown for application status monitoring of batchman, mailman and jobman in 9.8.11, “Configuring Application Status (batchman) resource model” on page 492, for the appavail-tws.pf profile. The only difference from the appavail-tws.pf profile is that we added the Process resource model to the tm-tws.pf profile, and because the netman process runs at all times except during special periods such as scheduled maintenance of servers, no rules for a schedule for the resource model were created.

To configure the application status resource model for the netman process:

1. From the itm-tws.pm profile manager, double-click the itm-tws.pf Tivoli Monitoring profile as shown in Figure 9-99.

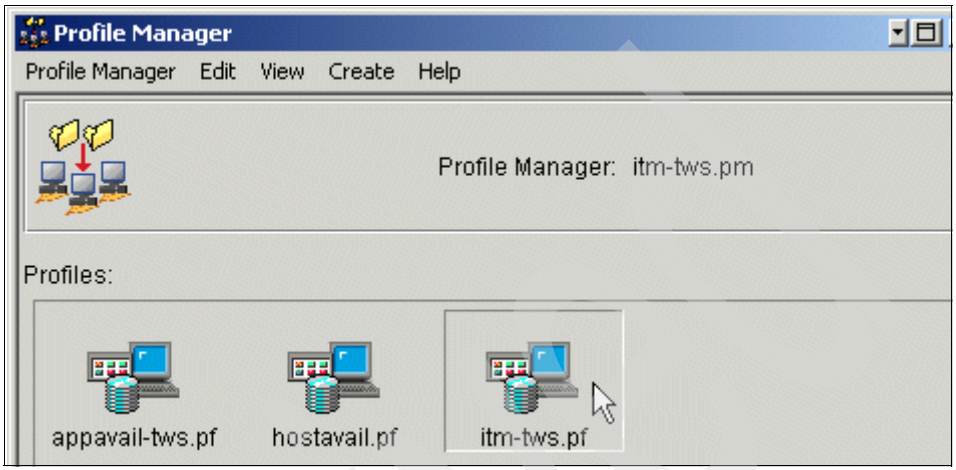

Figure 9-99 Opening the itm-tws.pf Tivoli Monitoring profile

The Tivoli Monitoring Profile window for the itm-tws.pf profile opens as shown in Figure 9-100 on page 507.

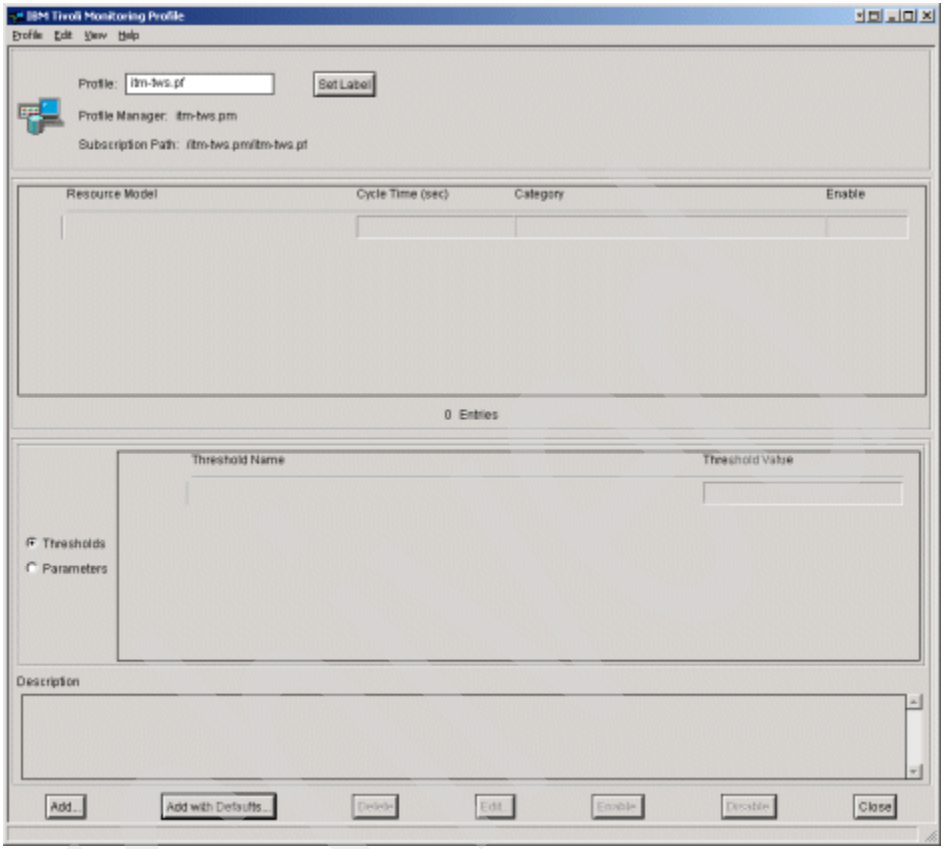

Figure 9-100 Tivoli Monitoring Profile window for itm-tws.pf profile

2. Select **Add**. The Add Resource Models to Profile window opens as shown in Figure 9-101 on page 508.

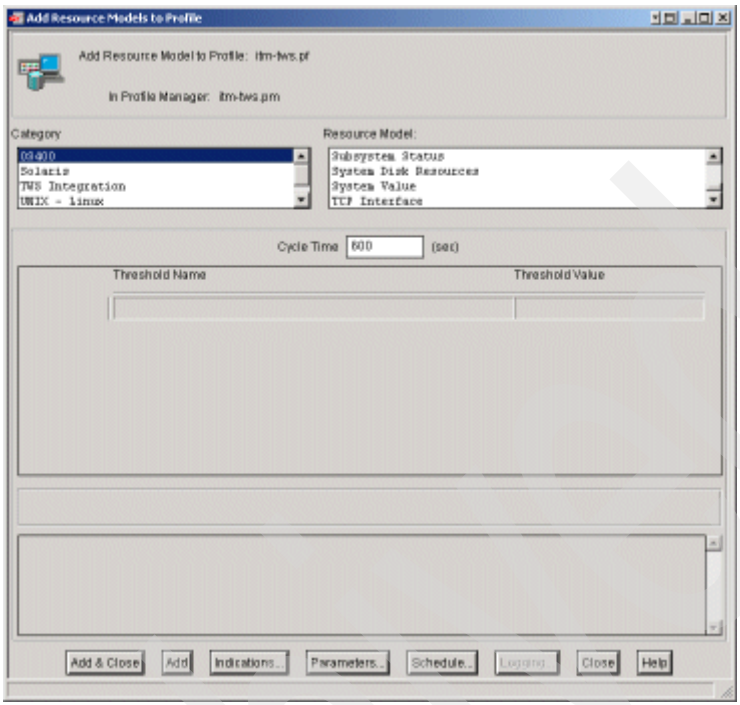

Figure 9-101 Add Resource Models to Profile window

3. Select **UNIX – Linux** in the Category list on the left-hand side of the window, and ensure Process is selected in the Resource Model list on the right-hand side of the window as shown in Figure 9-102 on page 509.

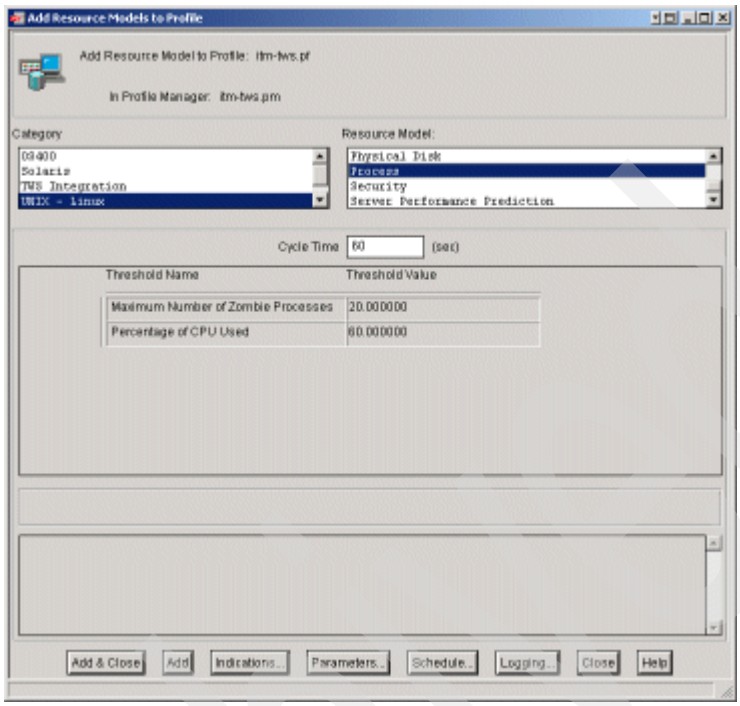

Figure 9-102 Selecting the Process resource model for the netman process

4. Select **Parameters**. The Parameters window opens as shown in Figure 9-103.

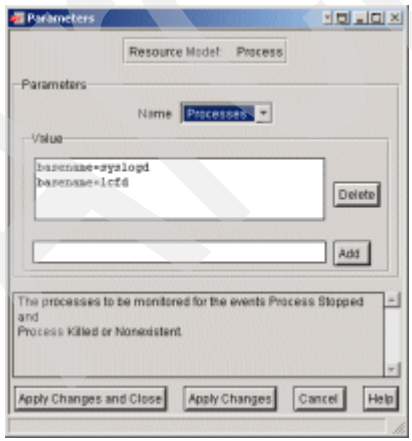

Figure 9-103 Parameters window for application status monitoring of netman process

5. Shift select both entries (**syslogd** and **lcfd**) in the Value list, and then select **Delete** as shown in Figure 9-104.

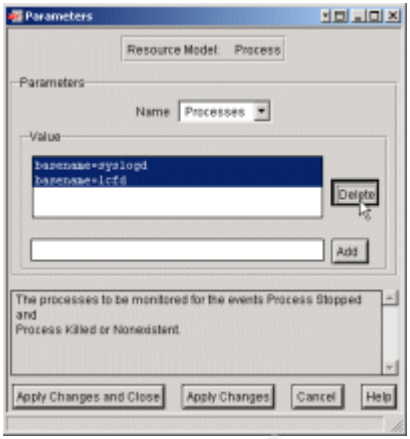

Figure 9-104 Deleting default parameters from Process resource model

6. Enter the text `basename=netman` in the Value text field, and then select **Add**. The value is added to the Value list as shown in Figure 9-105.

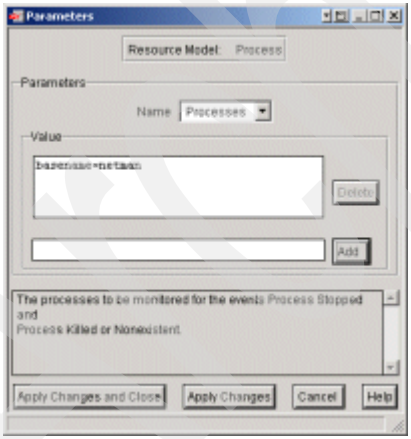

Figure 9-105 Adding netman to parameter list

7. Select **Apply Changes and Close**. You are returned to the Add Resource Models to Profile window.

8. Select **Add & Close**. You are returned to the Tivoli Monitoring Profile window for the itm-tws.pf profile, with the Process resource model added to the Resource Model list, as shown in Figure 9-106.

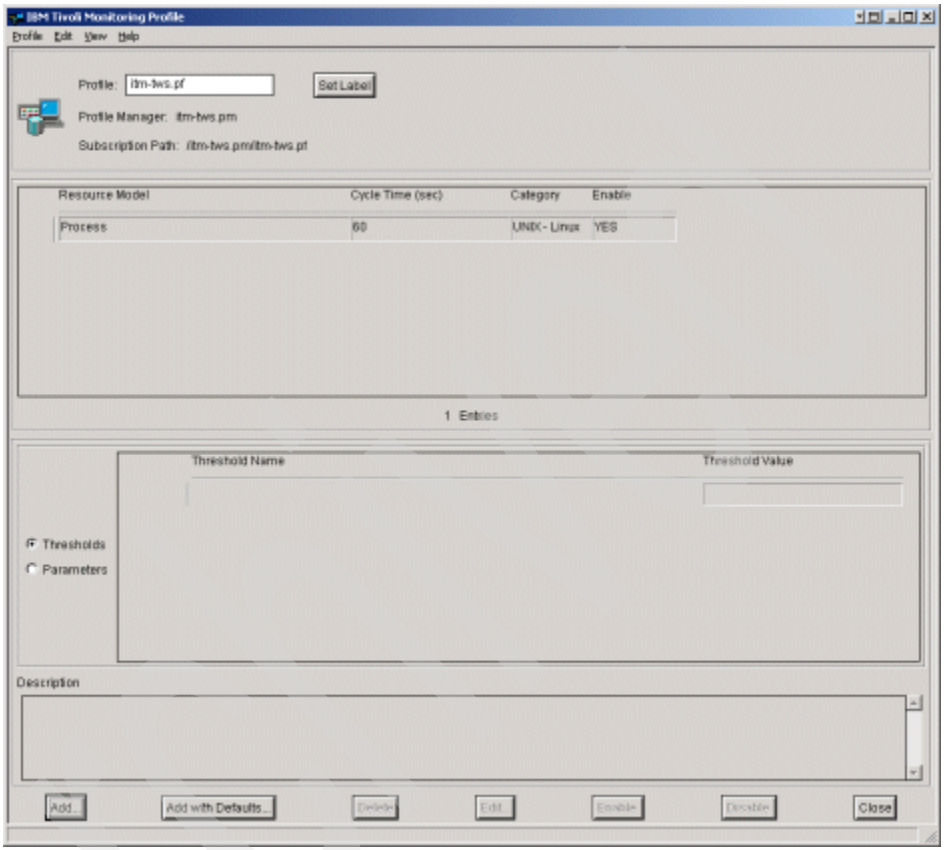

Figure 9-106 Process resource model for netman process added to itm-tws.pf profile

9. Select **Profile** → **Distribute Defaults…** as shown in Figure 9-107.

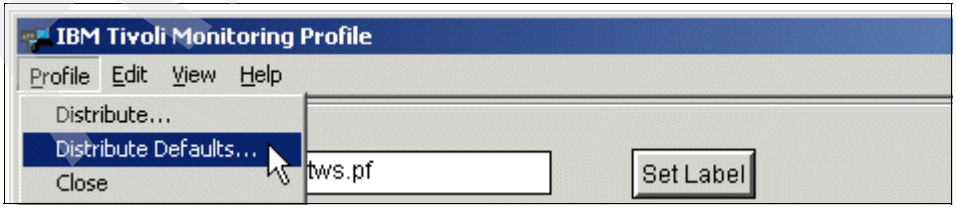

Figure 9-107 Setting distribution defaults for itm-tws.pf Tivoli Monitoring profile

The Set Distribution Defaults window displays as shown in Figure 9-108.

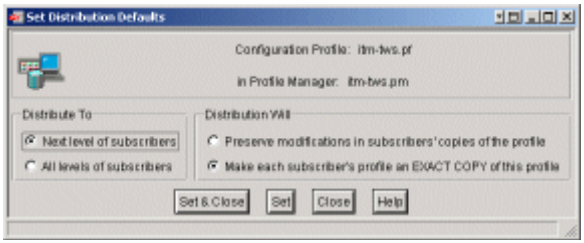

Figure 9-108 Set Distribution Defaults window for itm-tws.pf profile

10. In the Distribute To group, select **All levels of subscribers**. In the Distribution Will group, select **Make each subscriber's profile an EXACT COPY of this profile**, as shown in Figure 9-109.

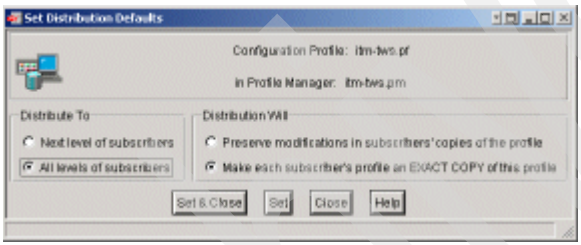

Figure 9-109 Setting distribution defaults for itm-tws.pf profile

11. Select the **Set & Close**. You are returned to the Tivoli Monitoring Profile window for the itm-tws.pf profile.
12. Select **Close**. You are returned to the Profile Manager window for the itm-tws.pm profile manager, as shown in Figure 9-110 on page 513.

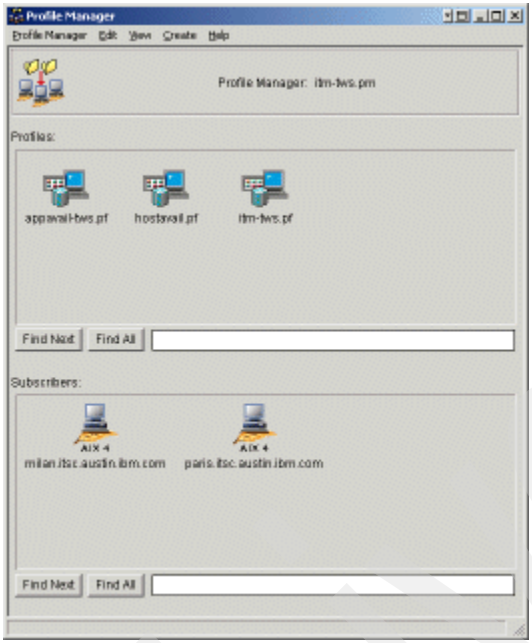

Figure 9-110 Configured itm-tws.pf profile in itm-tws.pm profile manager

9.8.13 Configuring Application Status (mailman) resource model

This configuration task is part of the configuration for the Process resource model for the batchman process, shown in 9.8.11, “Configuring Application Status (batchman) resource model” on page 492. The mailman process is passed as a parameter to the Process resource model in the appavail-tws.pf Tivoli Monitoring profile.

9.8.14 Configuring Application Status (jobman) resource model

This configuration task is part of the configuration for the Process resource model for the batchman process, shown in 9.8.11, “Configuring Application Status (batchman) resource model” on page 492. The jobman process is passed as a parameter to the Process resource model in the appavail-tws.pf Tivoli Monitoring profile.

9.8.15 Configuring Space Free resource model

Configuring the File System resource model to implement Space Free monitoring of the file system that *TWSHome* is located upon takes place in the itm-tws.pf Tivoli Monitoring profile.

To configure the File System resource model for monitoring the location of available space for the file system TWSHome:

1. From the itm-tws.pm profile manager, double-click the itm-tws.pf Tivoli Monitoring profile as shown in Figure 9-111.

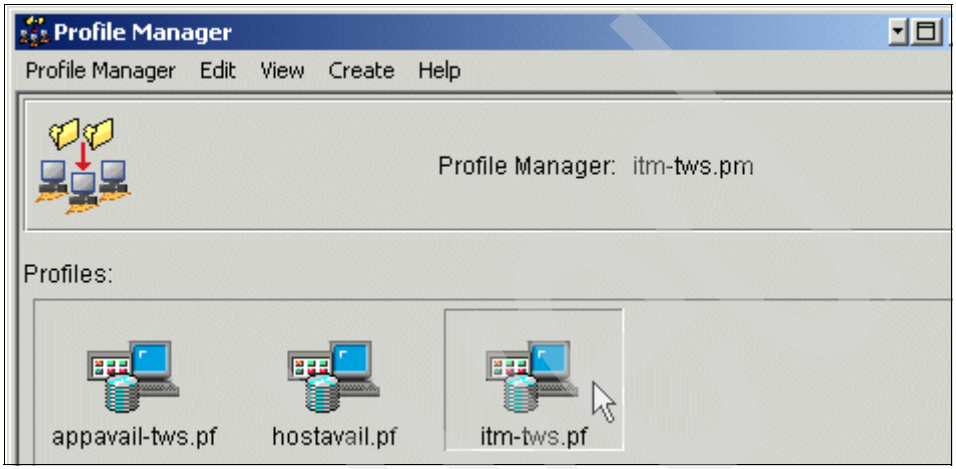

Figure 9-111 Opening the itm-tws.pf Tivoli Monitoring profile

The Tivoli Monitoring Profile window for the itm-tws.pf profile opens.

If you are following the same sequence of steps as shown in this book, a Process resource model (for the netman process) should be configured in this profile as shown in Figure 9-112 on page 515.

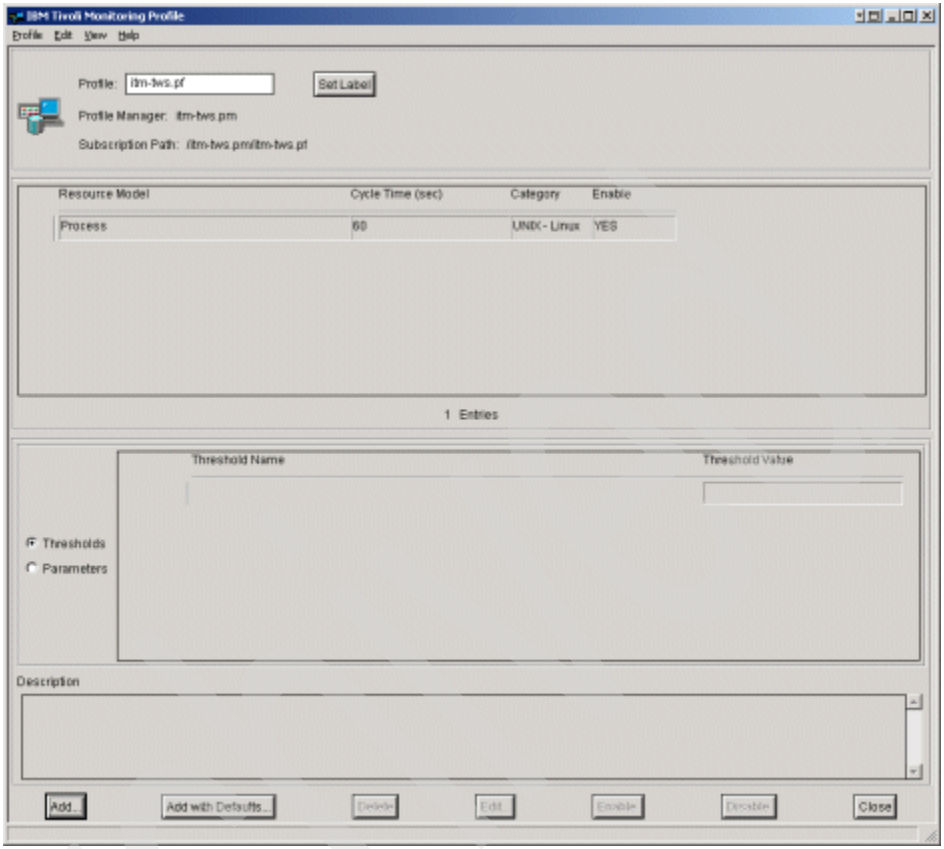

Figure 9-112 Tivoli Monitoring Profile window for itm-tws.pf profile

2. Select **Add**. The Add Resource Models to Profile window opens as shown in Figure 9-113 on page 516.

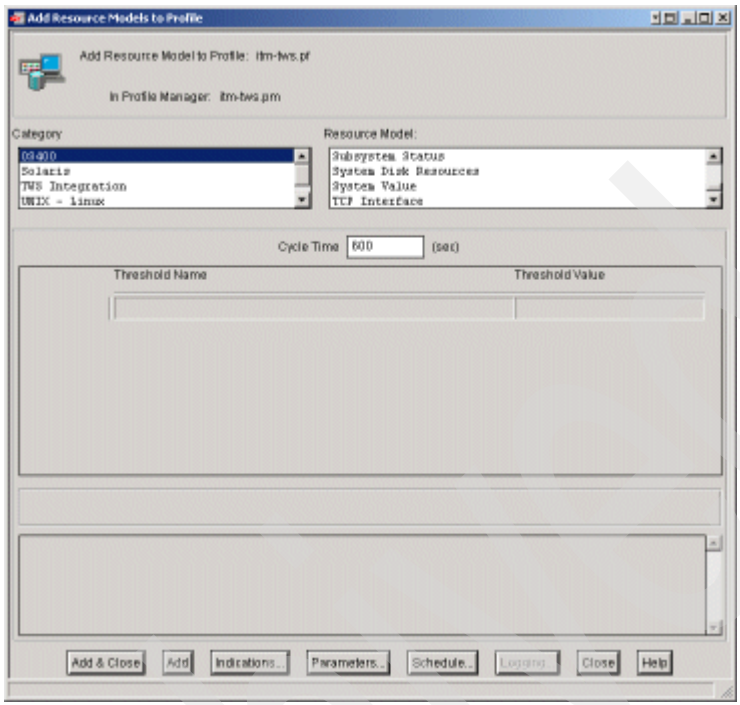

Figure 9-113 Add Resource Models to Profile window

3. Select **UNIX – Linux** in the Category list on the left-hand side of the window, and ensure File System is selected in the Resource Model list on the right-hand side of the window as shown in Figure 9-114 on page 517.

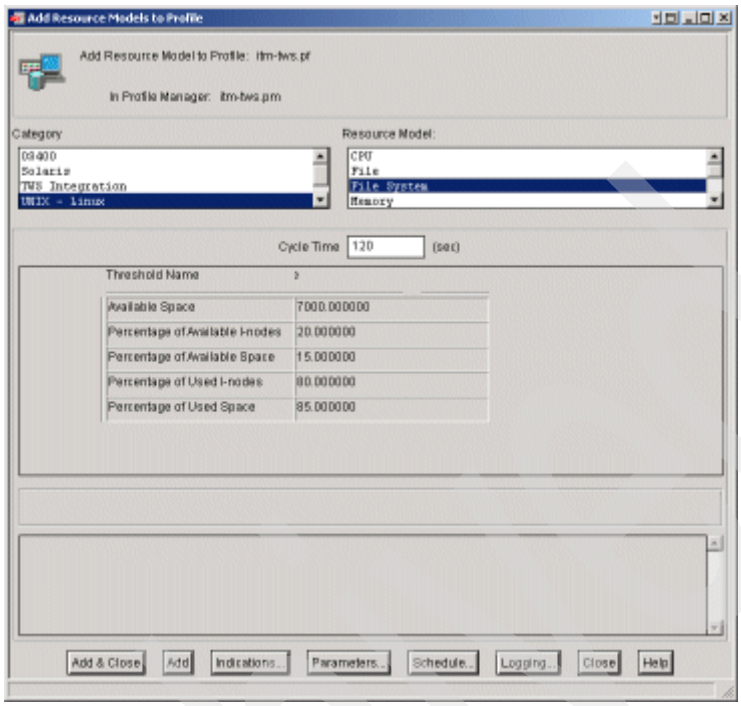

Figure 9-114 Selecting the File System resource model

4. Select **Parameters**. The Parameters window opens as shown in Figure 9-115.

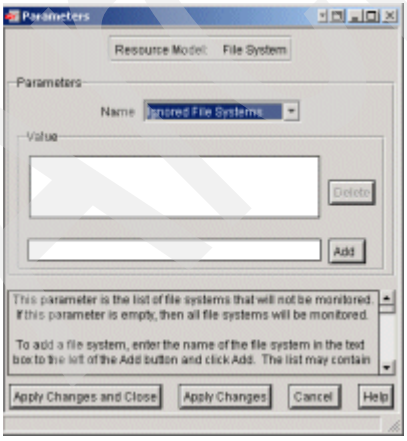

Figure 9-115 Parameters window for File System resource model

5. Select **Monitored File Systems** from the Name pop-up menu as shown in Figure 9-116.

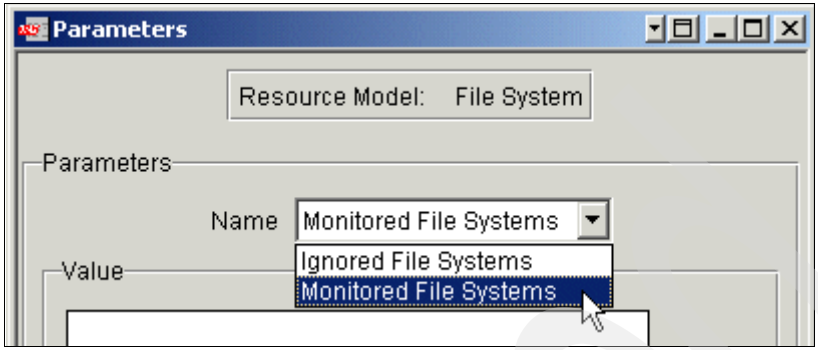

Figure 9-116 Selecting Monitored File Systems parameter

6. Enter the path to *TWSHome* in the Value text field, the additional arguments for the file system, and then select **Add**.

   In the lab for this book, we entered the path and argument string as shown in Figure 9-117.

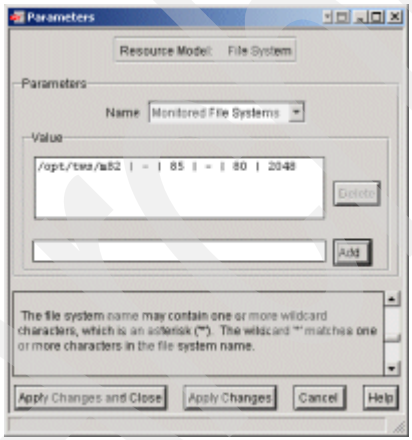

Figure 9-117 Path and monitoring threshold argument string added for TWSHome

7. Select **Apply Changes and Close**. You are returned to the Add Resource Models to Profile window.

8. Select **Add & Close**. You are returned to the Tivoli Monitoring Profile window for the itm-tws.pf profile, with the File System resource model added to the Resource Model list, as shown in Figure 9-118.

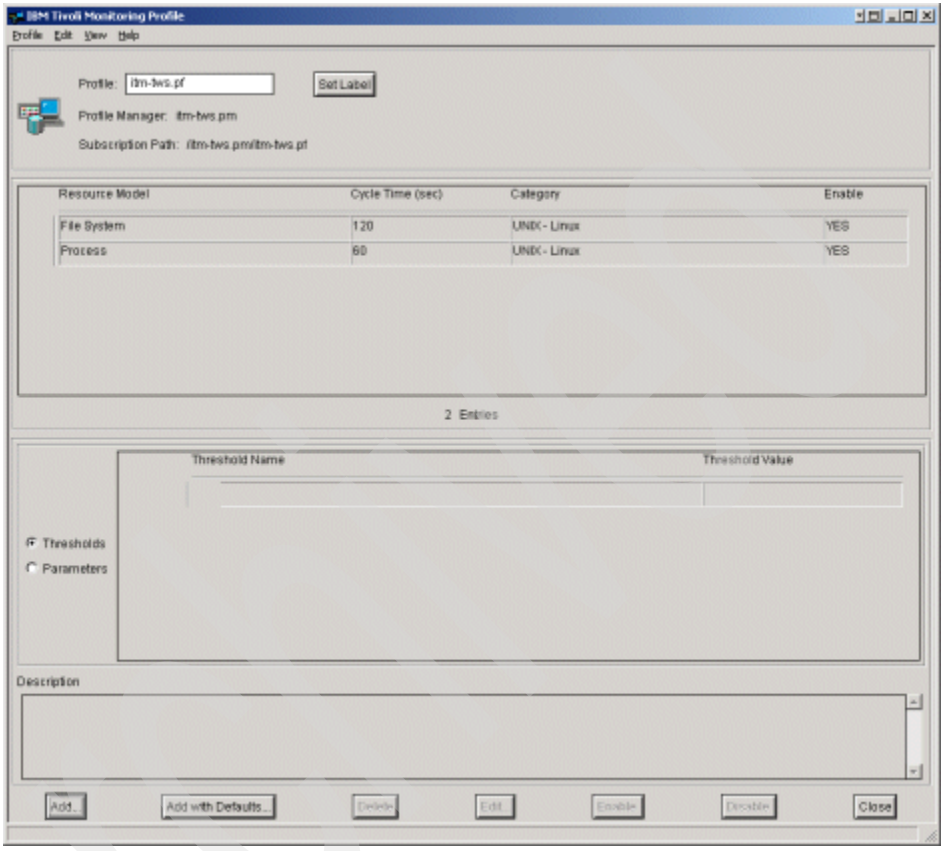

Figure 9-118 File System resource model added to itm-tws.pf ITM profile

If you have skipped directly into this section and have not set the distribution defaults yet, complete steps 9 on page 511 through 11 on page 512 in 9.8.12, “Configuring Application Status (netman) Resource Model” on page 505.

9. Select **Close**. You are returned to the Profile Manager window for the itm-tws.pm profile manager, as shown in Figure 9-119 on page 520.

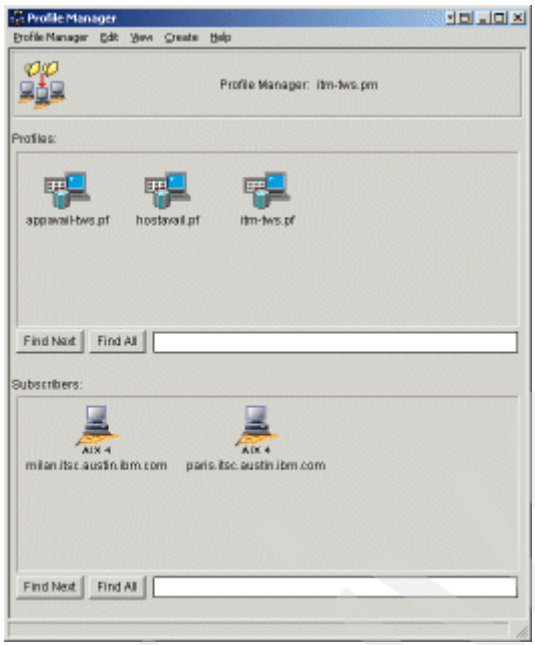

Figure 9-119 Configured itm-tws.pf profile in itm-tws.pm profile manager

9.8.16 Configuring Space Used (stdlist) resource model

The Space Used custom resource model is configured in the itm-tws.pf Tivoli Monitoring profile. This resource model must already be installed before the configuration can be performed. Installation instructions are in 9.8.9, “Adding a SpaceUsed custom resource model” on page 484.

Note: This procedure configures the space used monitoring for both the stdlist and schedlog directories. Each fully qualified directory path is passed as a parameter to a single resource model.

To configure the Space Used custom resource model in the itm-tws.pf Tivoli Monitoring profile:

1. From the itm-tws.pm profile manager, double-click the itm-tws.pf Tivoli Monitoring profile as shown in Figure 9-120.

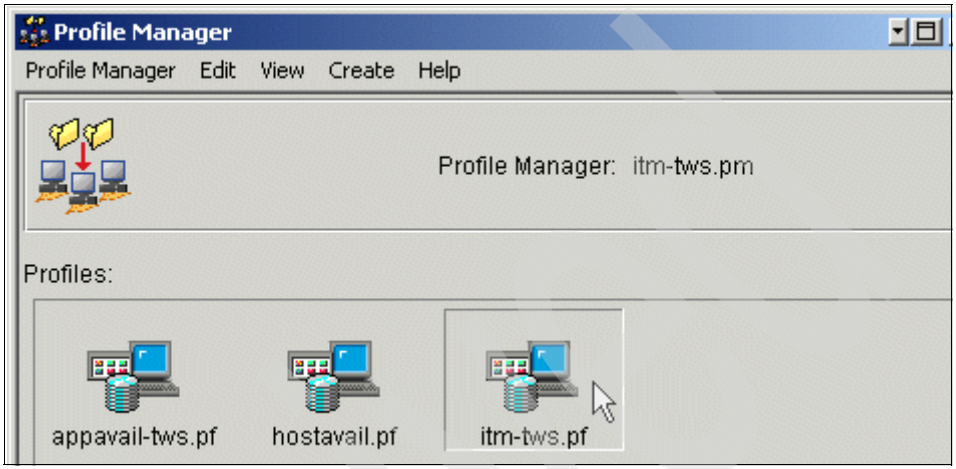

Figure 9-120 Opening the itm-tws.pf Tivoli Monitoring profile

The Tivoli Monitoring Profile window for the itm-tws.pf profile opens.

If you are following the same sequence of steps as shown in this book, a Process resource model (for the netman process) and a File System resource model (for the file system that contains *TWSHome*) should be configured in this profile as shown in Figure 9-121 on page 522.

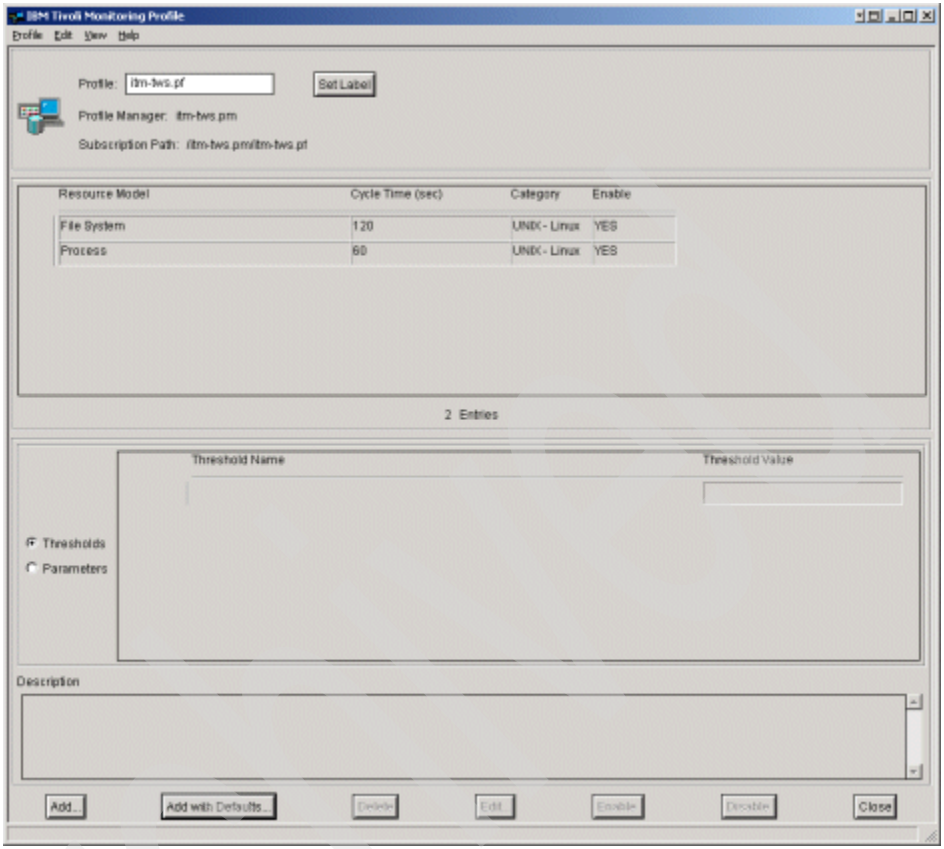

Figure 9-121 Tivoli Monitoring Profile window for itm-tws.pf profile

2. Select **Add**. The Add Resource Models to Profile window opens as shown in Figure 9-122 on page 523.

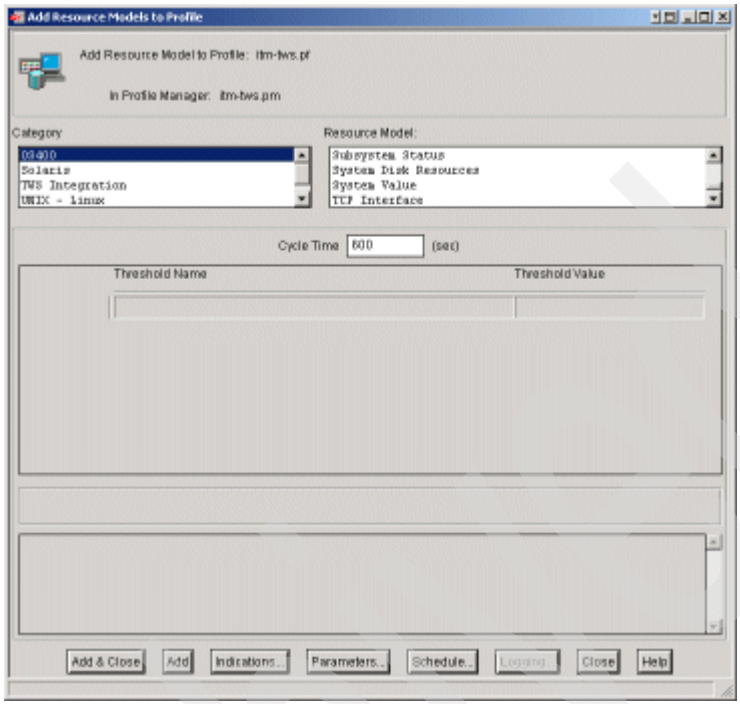

Figure 9-122 Add Resource Models to Profile window

3. Select **TWS Integration** in the Category list on the left-hand side of the window, and ensure Space Used is selected in the Resource Model list on the right-hand side of the window as shown in Figure 9-123 on page 524.

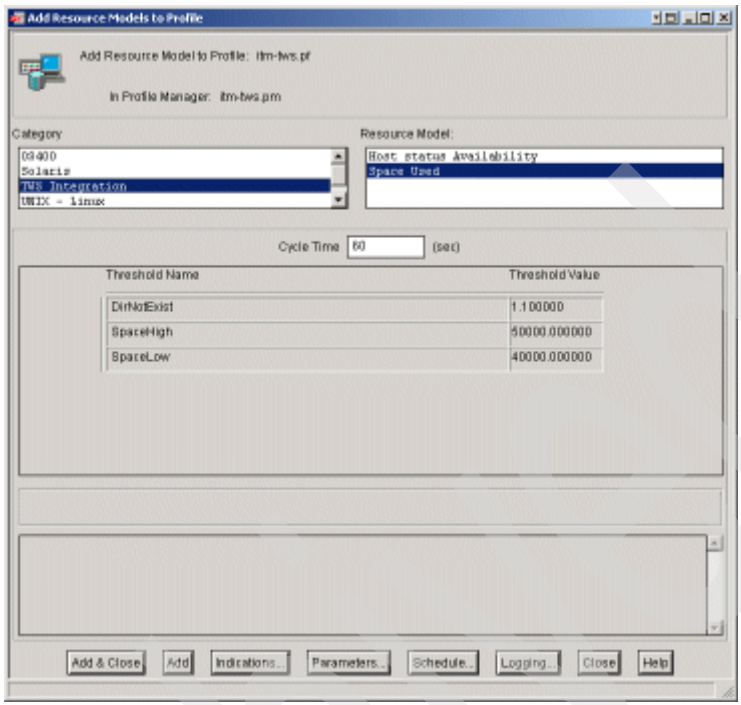

Figure 9-123 Selecting Space Used custom resource model

4. Select **Parameters**. The Parameters window opens as shown in Figure 9-124.

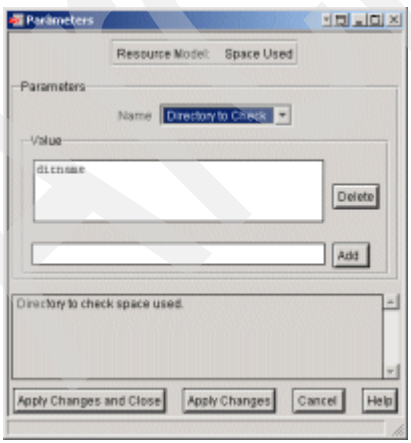

Figure 9-124 Parameters window

5. Select **dirname** in the Value list, and then select **Delete** as shown in Figure 9-125.

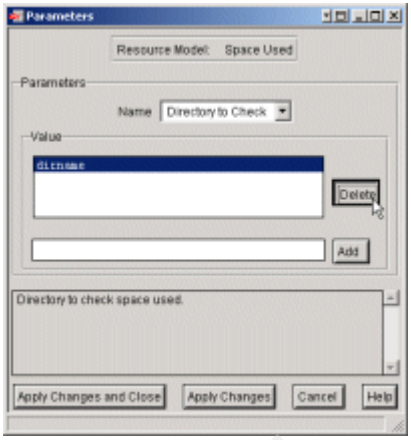

Figure 9-125 Deleting the default directory from the Value list

6. Enter the fully qualified path to the stdlist directory in the Value text field, and then select **Add**.

   In the lab for this book, we entered `/opt/tws/m82/stdlist` as shown in Figure 9-126.

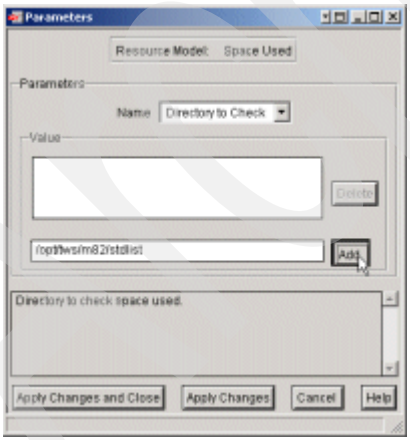

Figure 9-126 Adding the stdlist directory to the parameter value list

The directory path is entered into the Value list as shown in Figure 9-127 on page 526.

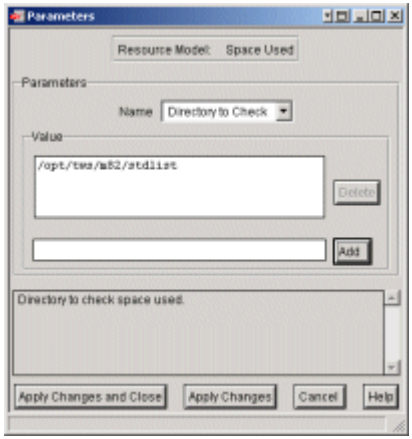

Figure 9-127 Directory for stdlist added to parameter list

7. Repeat step 6 on page 525 for the fully qualified directory path to schedlog.

   In the lab for this book, we entered the path `/opt/tws/m82/schedlog` as shown in Figure 9-128.

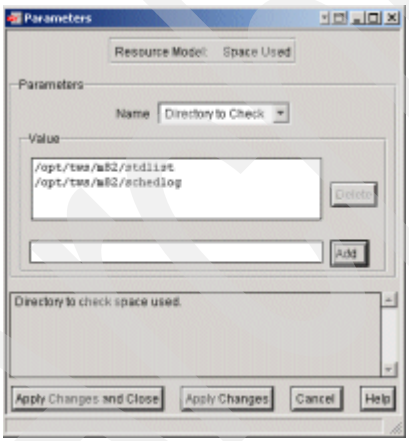

Figure 9-128 Directory for schedlog added to parameter list

8. Select **Apply Changes and Close**. You are returned to the Add Resource Models to Profile window.

9. Select **Add & Close**. You are returned to the Tivoli Monitoring Profile window for the itm-tws.pf profile, with the Space Used resource model added to the Resource Model list, as shown in Figure 9-129.

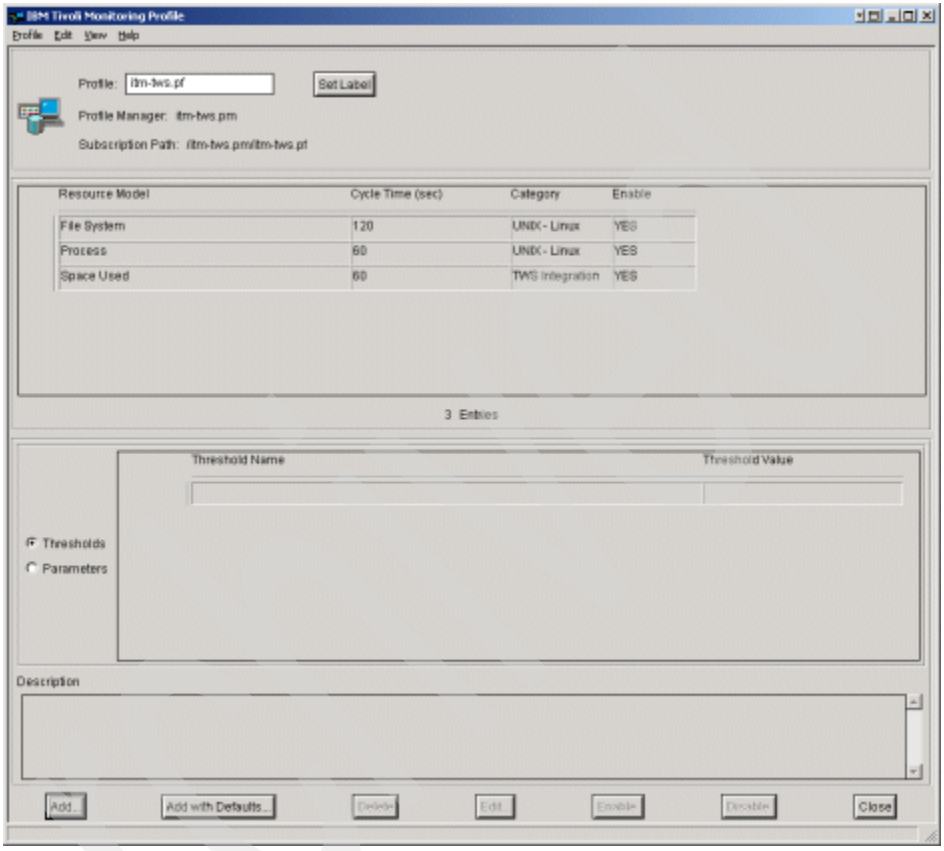

Figure 9-129 Space Used custom resource model added to itm-tws.pf ITM profile.

10. If you have skipped directly into this section and have not set the distribution defaults, complete steps 9 on page 511 through 11 on page 512 in 9.8.12, “Configuring Application Status (netman) Resource Model” on page 505.
11. Select **Close**. You are returned to the Profile Manager window for the itm-tws.pm profile manager, as shown in Figure 9-130 on page 528.

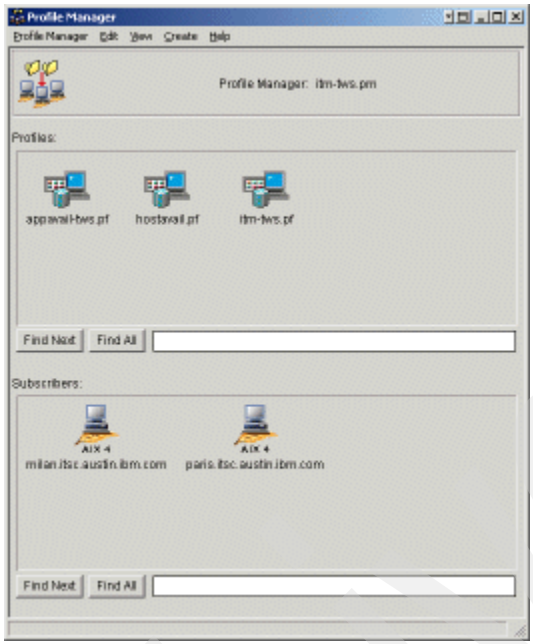

Figure 9-130 Configured itm-tws.pf profile in itm-tws.pm profile manager

9.8.17 Configuring Space Used (schedlog) resource model

This configuration task is part of the configuration for the Space Used custom resource model for the stdlist directory, shown in 9.8.16, “Configuring Space Used (stdlist) resource model” on page 520. The schedlog directory is passed as a parameter to the Space Used custom resource model in the itm-tws.pf Tivoli Monitoring profile.

9.8.18 About configuring an action

This book shows how to achieve a basic level of integration between Tivoli Monitoring and TWS where deployed resource models can detect and send alerts on the system conditions that you are monitoring. However, you can configure much more sophisticated resource models, starting with adding actions.

You can either add actions directly into a resource model or you can configure them as a Tivoli Management Framework task. Either way, you configure the triggering in the same manner. If actions are configured for a resource model, you can select the conditions under which they are triggered by configuring the indications of the resource model.

To view the indications of a resource model and access the actions that can be selected:

1. Open a Tivoli Monitoring profile as shown in Figure 9-131.

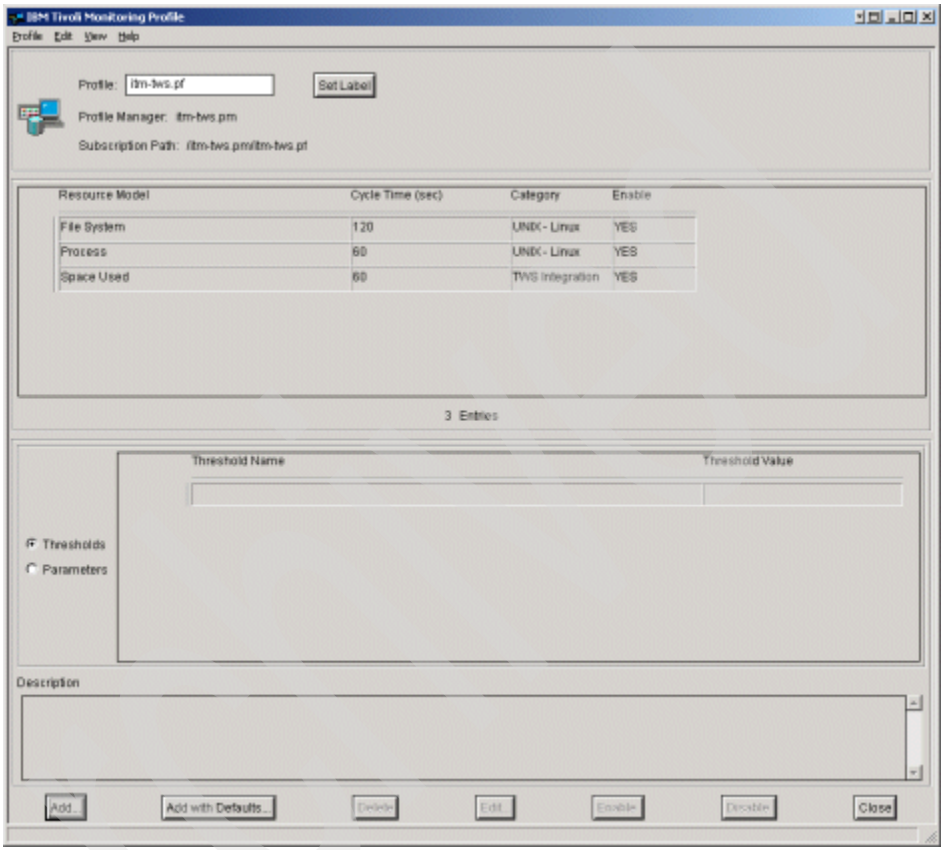

Figure 9-131 An open Tivoli Monitoring profile

2. Double-click a resource model in the Resource Model list (for example, the Space Used resource model as shown in Figure 9-132 on page 530).

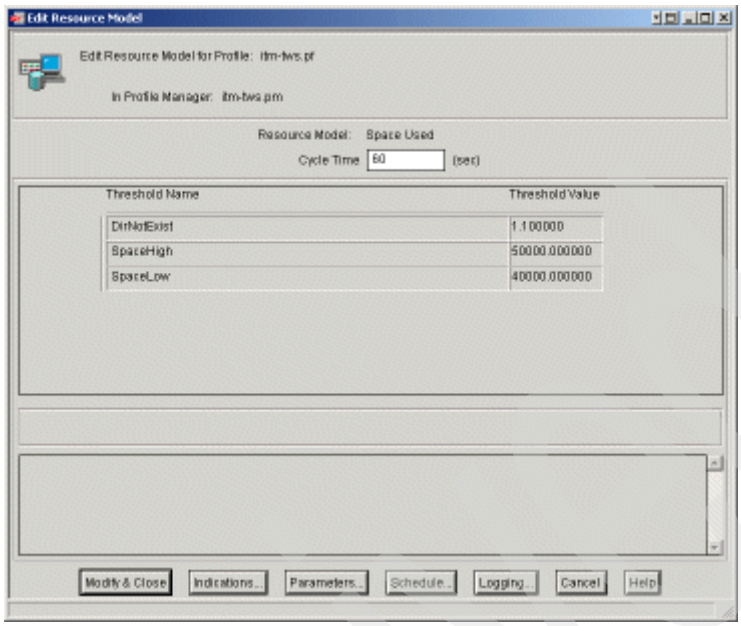

Figure 9-132 Opened resource model

3. Access the indications by selecting **Indications** to open the Indications and Actions window as shown in Figure 9-133 on page 531.

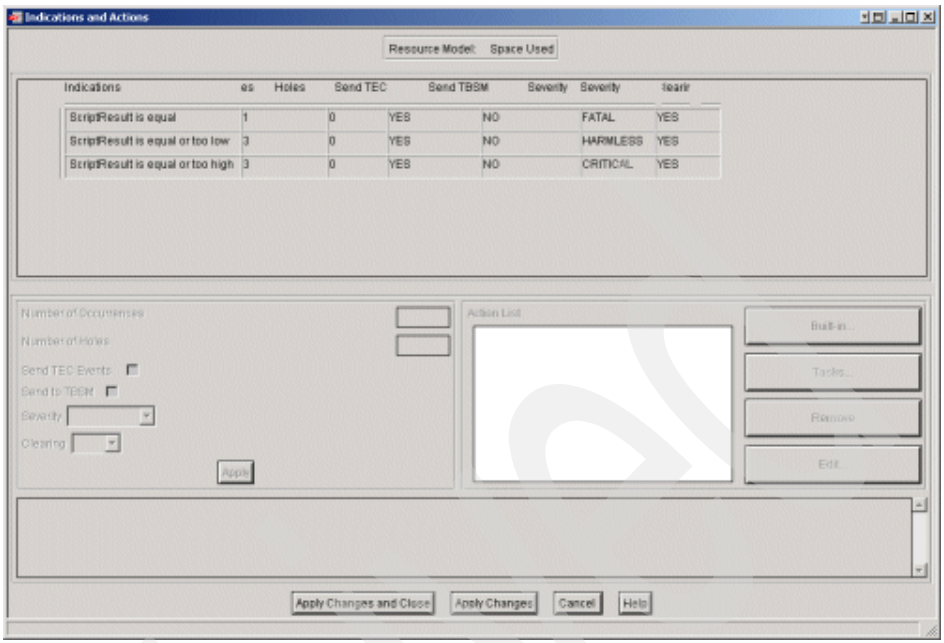

Figure 9-133 Indications and Actions window.

4. Select an indication from the top list. The bottom half of the window is undimmed as shown in Figure 9-134 on page 532 where the ScriptResult is equal indication is selected.

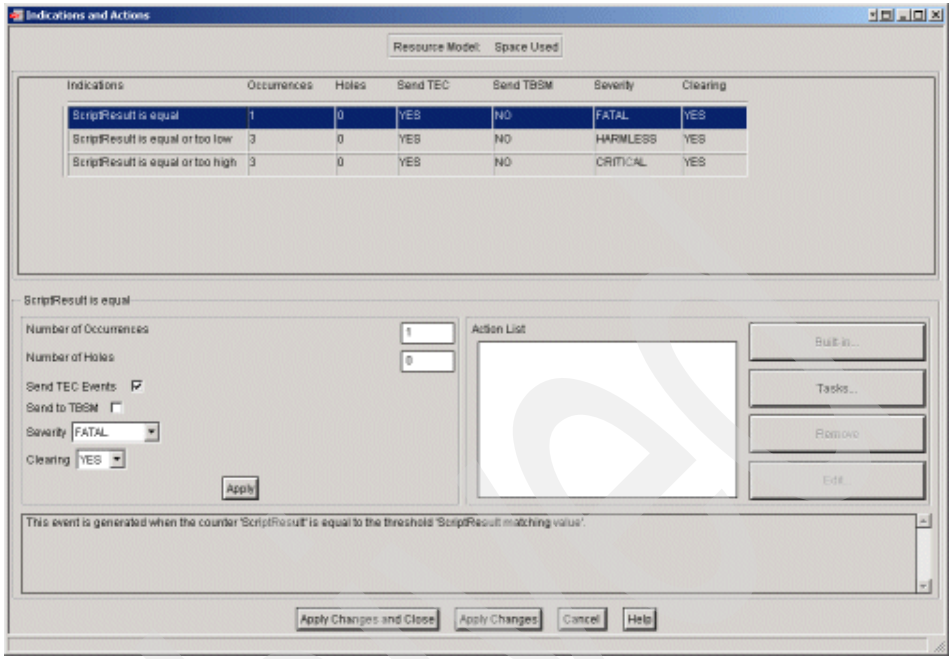

Figure 9-134 Indication selected

If any actions are available in the resource model, they are presented in the Actions List.

For further details on how to develop actions, see your Tivoli Framework administrator or IBM service provider.

9.8.19 Distributing profiles to subscribers

After you have configured the Tivoli Monitoring profiles, you need to distribute them to the subscribers that should run the resource models in the profiles. In the lab for this book, the hostavail.pf profile was only distributed to milan because we wanted the host availability checking to be run only from the MDM. The other profiles, appavail-tws.pf and itm-tws.pf, were distributed to both milan and paris because we wanted the resource models in those profiles to run on all Fault Tolerant Agents in the scheduling network.

To distribute the profiles that were created in this book to the subscribers:

1. Open the itm-tws.pm profile manager as shown in Figure 9-135.

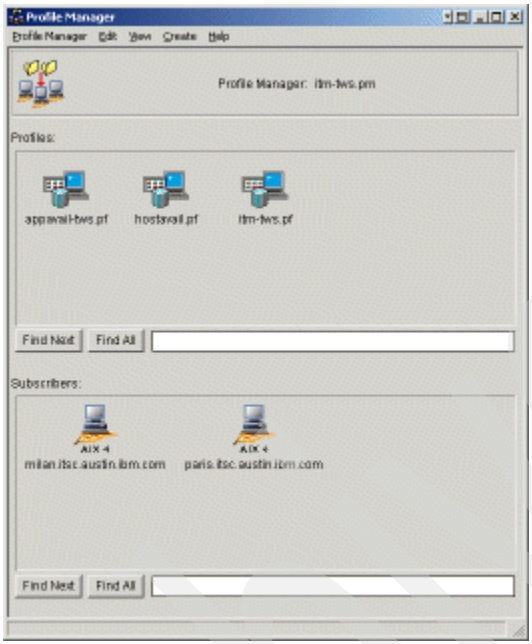

Figure 9-135 Profile manager itm-tws.pm

2. Select the hostavail.pf Tivoli Monitoring profile as shown in Figure 9-136 on page 534.

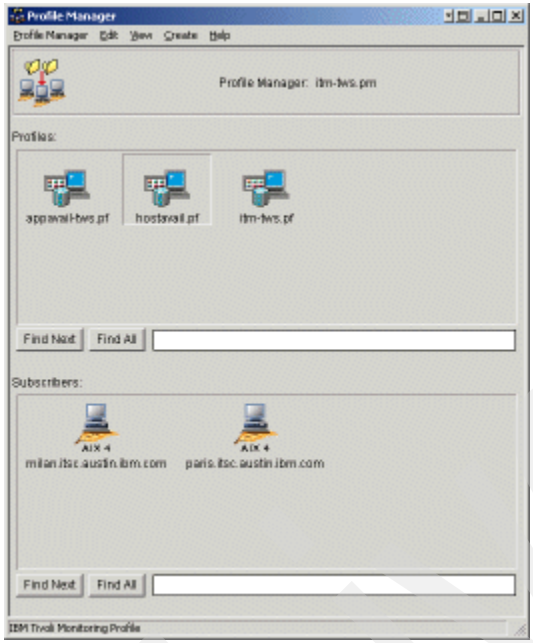

Figure 9-136 Selecting hostavail.pf Tivoli Monitoring profile

3. Select the Master Domain Manager in the Subscribers list. In the lab for this book, we selected milan as shown in Figure 9-137 on page 535.

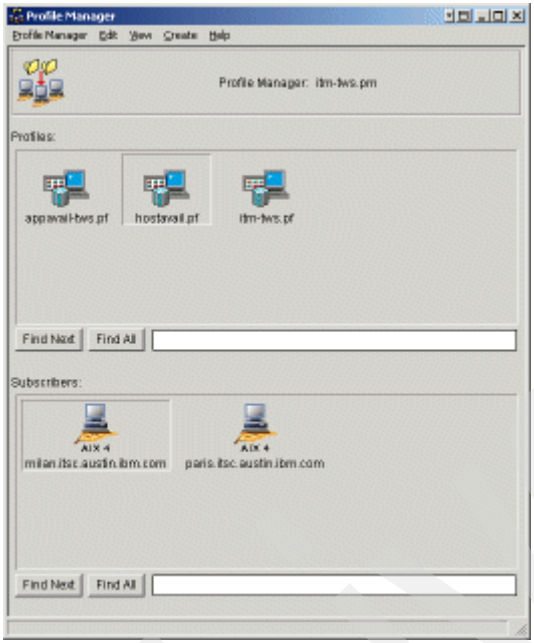

Figure 9-137 Selecting Master Domain Manager in Subscriber list

4. Select **Profile Manager** →, **Distribute...** as shown in Figure 9-138.

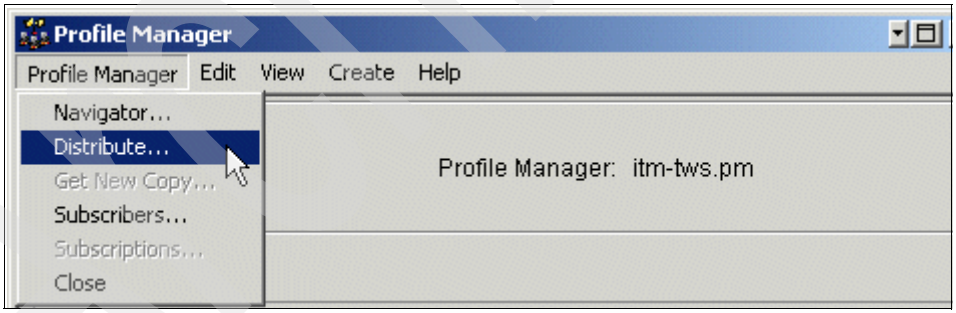

Figure 9-138 Starting distribution of selected profile to selected subscriber

The Distribute Profiles window opens as shown in Figure 9-139 on page 536.

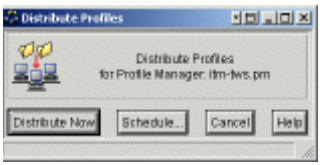

Figure 9-139 Distribute Profiles window

5. Select **Distribute Now**. You are returned to the Profile Manager window for the itm-tws.pm profile manager, and the hostavail.pf profile is distributed to the Master Domain Manager.
6. Verify the distribution by logging into the primary TMR server under a user account that is authorized in Tivoli Management Framework and using the **wdmlseng** command. You might have to wait two to three minutes for the distribution to complete, depending upon the complexity of your environment's topology. In environments where the primary TMR server and the MDM server are the same, you should not have to wait longer than about 30 seconds.

   In the lab for this book, we logged into milan as root user and ran the **wdmlseng** command as shown in Example 9-9 on page 540.

Example 9-8 Verifying distribution of hostavail.pf Tivoli Monitoring profile

```
milan:/#  wdmlseng -e milan.itsc.austin.ibm.com -verbose

Forwarding the request to the endpoint:
milan.itsc.austin.ibm.com  1187404249.4.522+#TMF_Endpoint::Endpoint#

The following profiles are running:

hostavail.pf#milan-region
        HostAvail: Running
                PingFail 100 %
```

7. Select **Edit** → **Profiles** → **Deselect All** as shown in Figure 9-140.

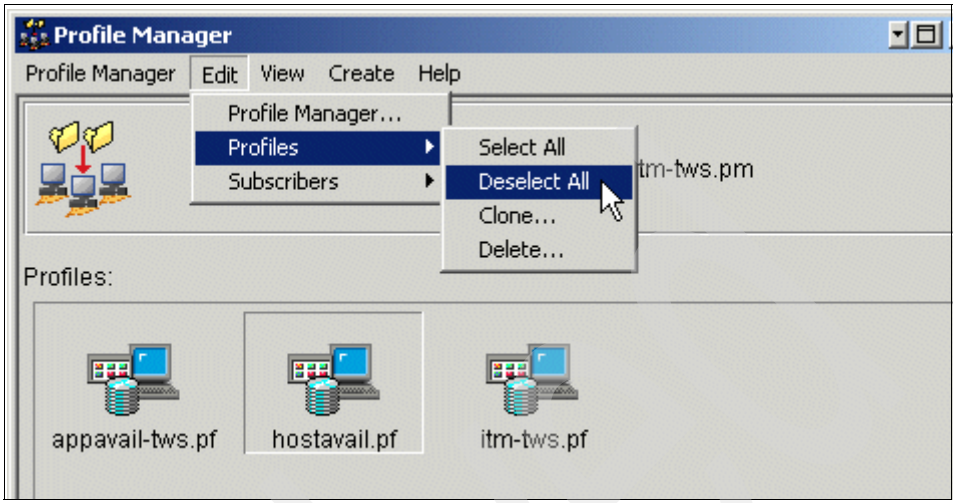

Figure 9-140 Deselecting selected profiles

You can also simply select another profile to deselect the currently selected profile.

8. Control-click the appavail-tws.pf and itm-tws.pf Tivoli Monitoring profiles as shown in Figure 9-141 on page 538.

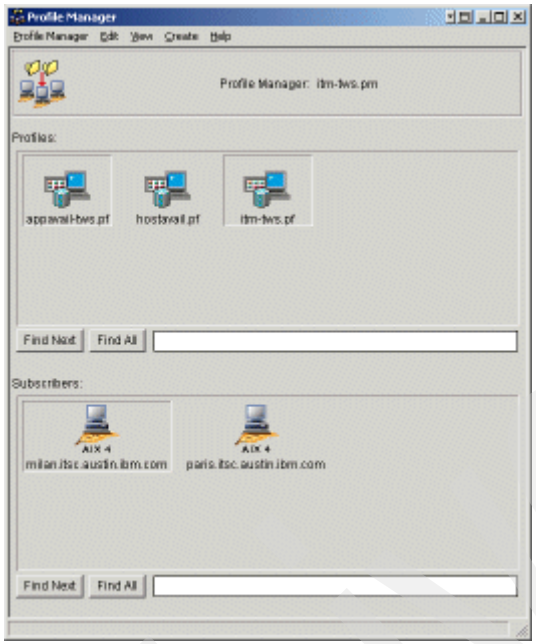

Figure 9-141 Selecting appavail-tws.pf and itm-tws.pf Tivoli Monitoring profiles

9. Select **Edit** → **Subscribers** → **Select All** as shown in Figure 9-142.

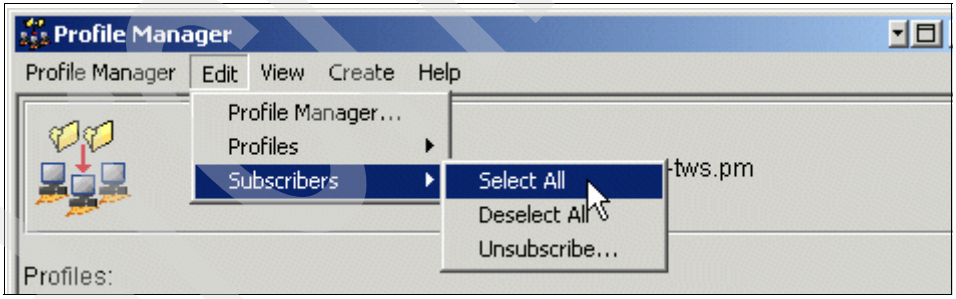

Figure 9-142 Selecting all subscribers.

10. Select **Profile Manager** → **Distribute...** as shown in Figure 9-143 on page 539.

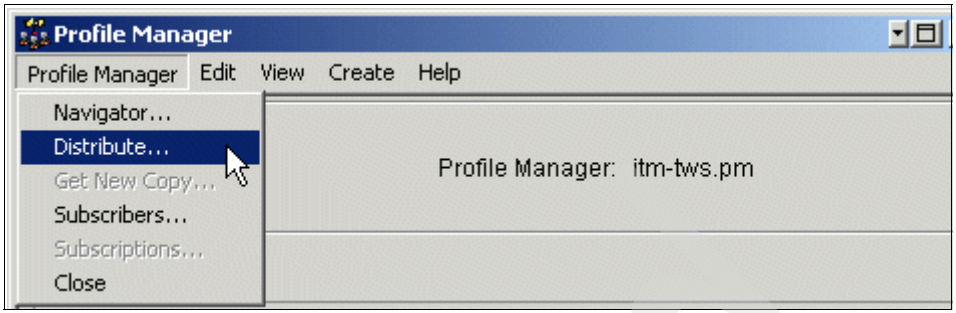

Figure 9-143 Starting distribution of selected profile to selected subscriber

The Distribute Profiles window opens as shown in Figure 9-144.

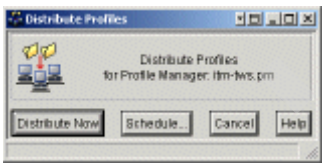

Figure 9-144 Distribute Profiles window

11. Select **Distribute Now**. You are returned to the Profile Manager window for the itm-tws.pm profile manager, and the appavail-tws.pf and itm-tws.pf profiles are distributed to the Master Domain Manager.
12. Verify the distribution by logging into the primary TMR server under a user account that is authorized in Tivoli Management Framework, and using the **`wdmlseng`** command. You might have to wait two to three minutes for the distribution to complete, depending upon the complexity of your environment's topology. In environments where the primary TMR server and the Fault Tolerant Agent servers (or the servers you want to monitor) are not separated by a WAN link and number fewer than 50, you should not have to wait longer than about a minute.

    In the lab for this book, we logged into milan as root user and ran the **`wdmlseng`** command as shown in Example 9-9 on page 540.

Example 9-9 Verifying distribution of appavail-tws.pf and itm-tws.pf profiles

```
milan:/#  wdmlseng -e milan.itsc.austin.ibm.com -verbose

Forwarding the request to the endpoint:
milan.itsc.austin.ibm.com  1187404249.4.522+#TMF_Endpoint::Endpoint#

The following profiles are running:

hostavail.pf#milan-region
        HostAvail: Running
                PingFail 100 %
```

The configuration of the integration between Tivoli Monitoring and TWS is complete.

Tip: For large scheduling networks (100 or more processors), we recommend that you use subscription lists. Especially large scheduling networks (300 or more processors) that have constant additions and deletions every month can benefit from using Queries against manually maintained lists of processors in a relational database, or Queries backed by inventory scans from Tivoli Configuration Manager. See your Tivoli Framework administrator or IBM service provider for assistance in using these features to manage scaled-up scheduling networks.

In complex environments with 10 or more distributions per day, we recommend a full implementation of the MDist 2 feature in Tivoli Management Framework. This feature lets you monitor status and control distributions at a very detailed level using the MDist 2 Distribution Status Console as shown in Figure 9-145 on page 541.

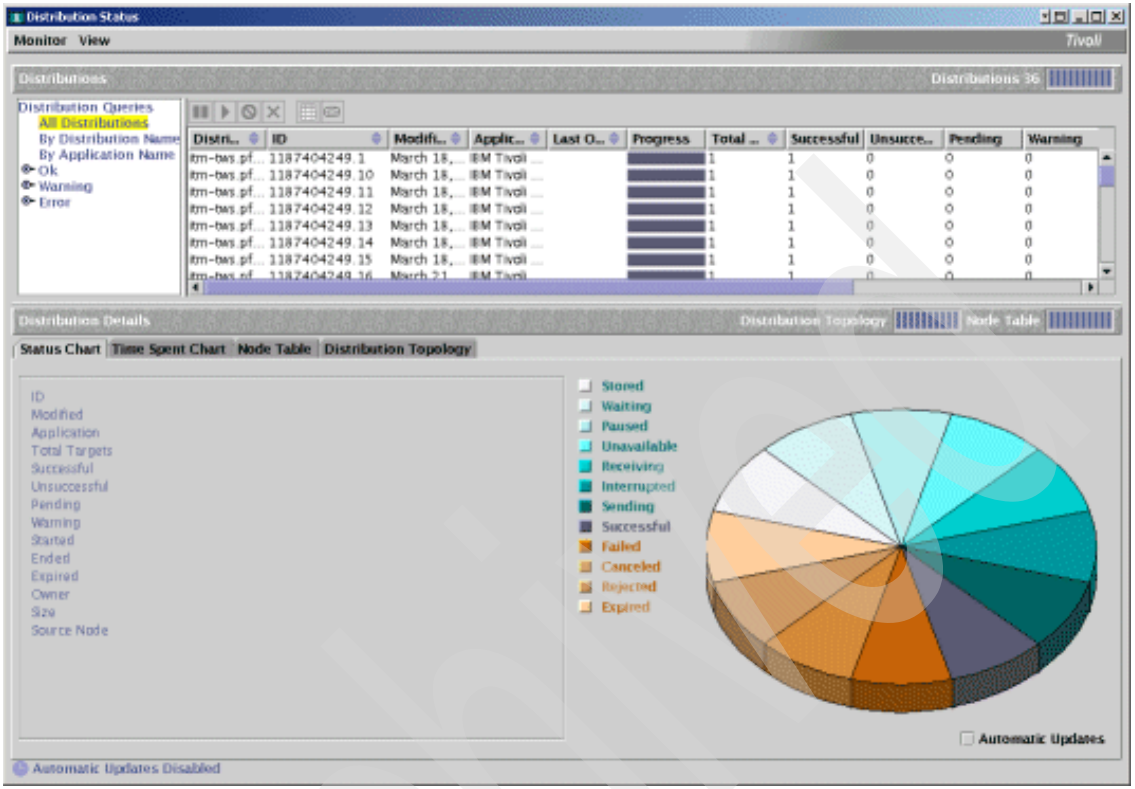

Figure 9-145 Distribution Status Console for monitoring and controlling MDist 2 distributions

Note that this GUI allows very fine-grained monitoring of the state of the distributions. It also enables tracking the location within a network of a distribution, as shown in Figure 9-147 on page 543.

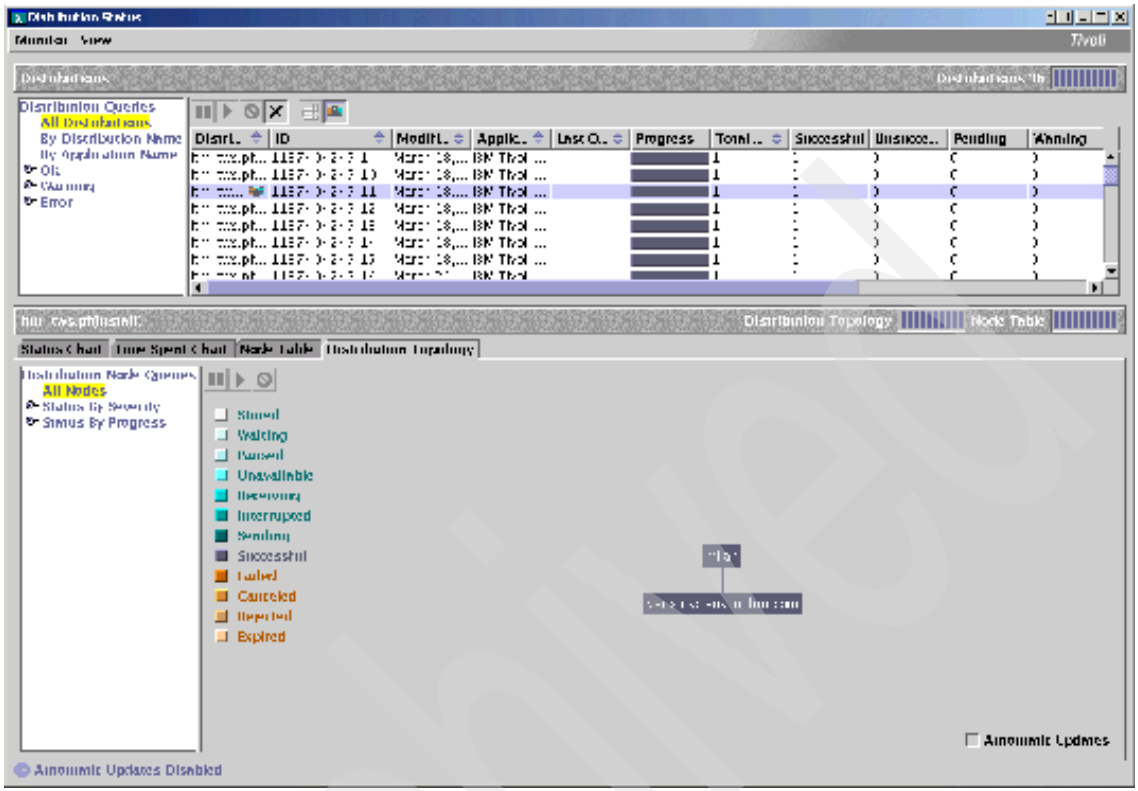

Figure 9-146 Distribution Status Console display of distribution topology

Full implementation requires a RDBMS that is supported by Tivoli Management Framework. See your Tivoli Framework administrator or IBM service provider for additional details.

9.9 Using the integration

A common way to use the integration between Tivoli Monitoring and TWS is to have operators watching the Tivoli Monitoring Web Health Console, or integrate Tivoli Enterprise Console and have operators watch the GUI event console supplied by Tivoli Enterprise Console. In this section we show how to start the Tivoli Monitoring Web Health Console, and inspect the resource models through the command line interface.

In order to use Tivoli Monitoring Web Health Console, you must have it installed and configured. See *IBM Tivoli Monitoring User's Guide Version 5.1.2*, SH19-4569 for details on installing and configuring the Web Health Console.

To start using the Tivoli Monitoring Web Health Console:

1. Open a Web browser on your computer, and go to the Web page for Tivoli Monitoring Web Health Console. The exact URL depends upon how the Web Health Console was configured your environment. The typical URL is that is based off of the primary TMR server is:

   `http://tmr.example.com:9080/dmwhc/`

   In the lab for this book, we used the open source Mozilla browser that pointed to:

   `http://milan.itsc.austin.ibm.com:9080/dmwhc/`

   A logon page displays.

2. Enter a user account with appropriate privileges to Web Health Console via the Tivoli Management Framework in the User text field, the password of the account in the Password text field, and name of the server that Web Health Console runs upon in the Host text field.

   In the lab for this book, we used the root user account for milan as shown in Figure 9-147.

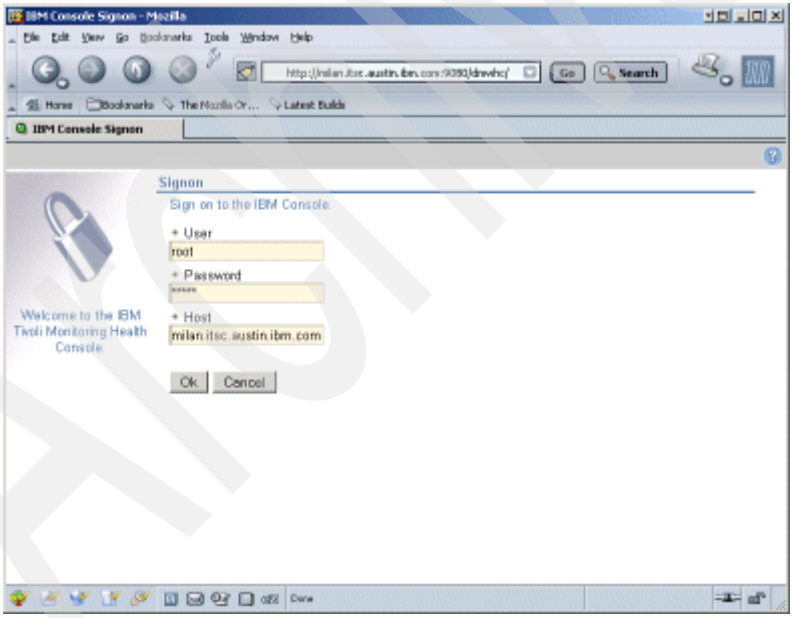

Figure 9-147 Logging into Web Health Console

3. Select **OK**. If this is your very first time logging into Web Health Console, it can take up to a minute to log in. The first time you log into Web Health Console, it displays the Preferences page with the Endpoints frame.

Web Health Console can be configured to show all or only selected endpoints in the environment. This configuration does not affect which endpoints run Tivoli Monitoring. It is just a configuration of the display. The Preferences page, as shown in Figure 9-148, lets users select which endpoints are displayed in Web Health Console.

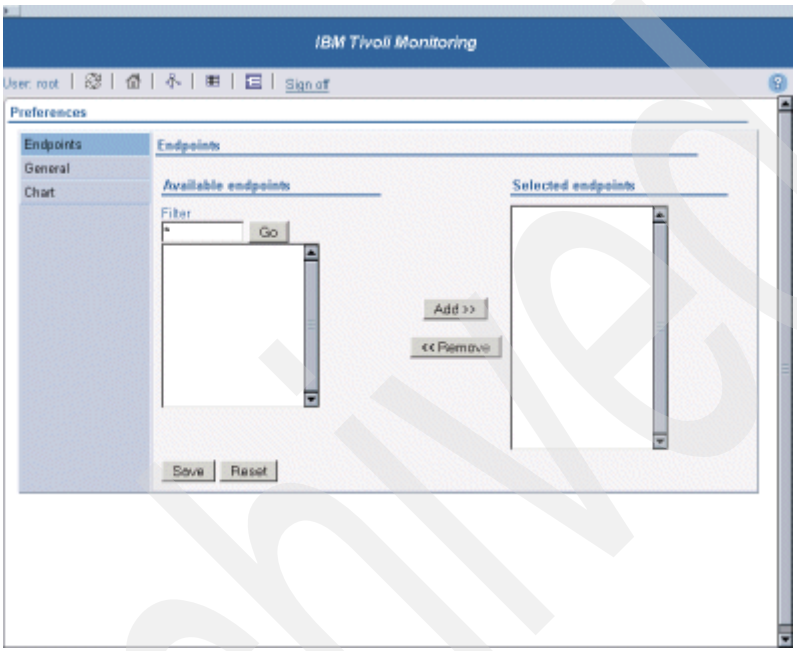

Figure 9-148 Preferences page, Endpoints frame

4. Leave the asterisk (*) in the Filter text field, and select **Go**. The Available Endpoints list is populated with all known endpoints in the environment. If you are in a large environment, the filter helps you quickly locate the endpoints you are interested in including in the Web Health Console view. You might want filter on just the endpoints that belong to the scheduling network, for example.

   In the lab for this book, we retrieved both milan and paris into the Available Endpoints list as shown in Figure 9-149 on page 545.

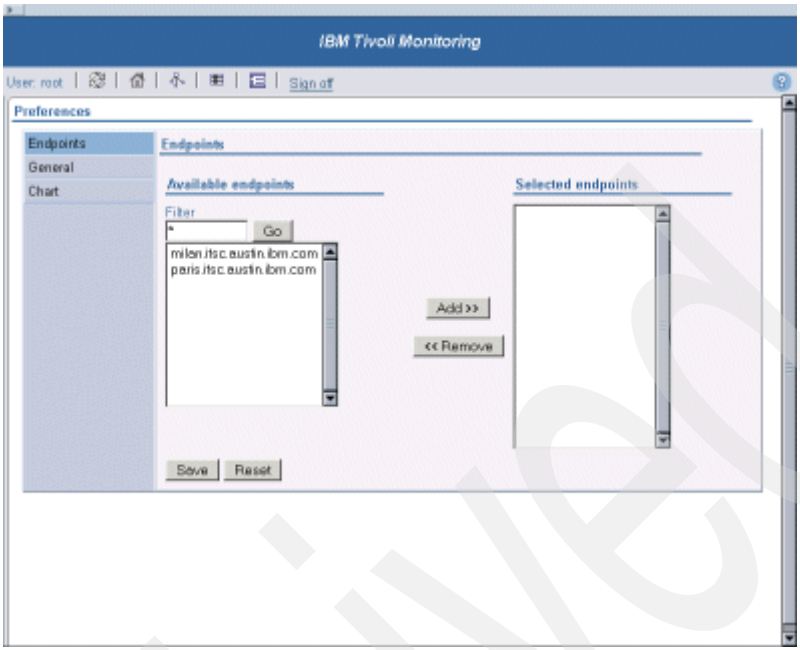

Figure 9-149 Populated Available Endpoints list

5. Select all endpoints that you want to view in the Web Health Console and select **Add >>**. The selected endpoints are copied to the Selected Endpoints list.

   In the lab for this book, we selected both milan and paris as shown in Figure 9-150 on page 546.

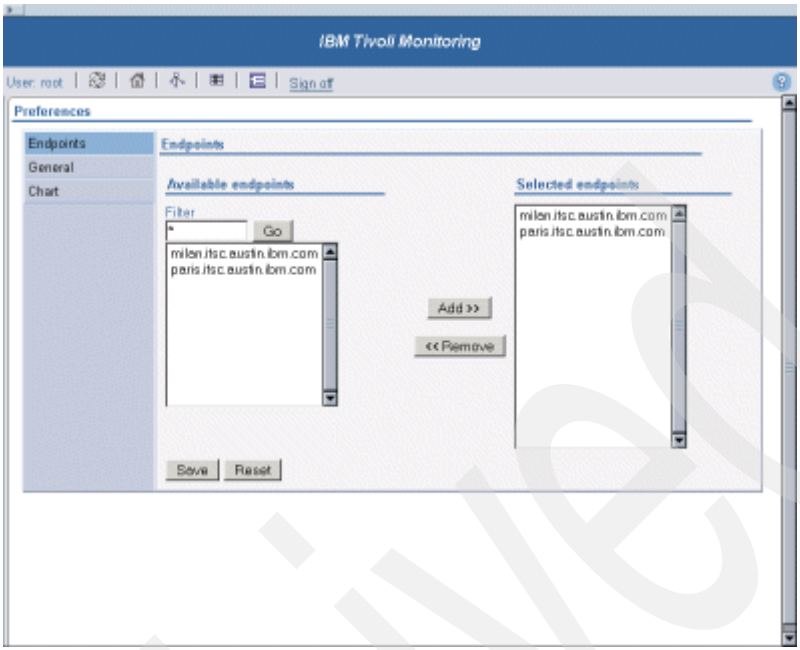

Figure 9-150 Populated Selected Endpoints list

6. Select **Save**. The page refreshes as soon as the selected endpoints are selected.
7. Two basic views are available from within Web Health Console: endpoint or resource model. Select the Endpoint View icon to view a status of the environment by all selected endpoints as shown in Figure 9-151.

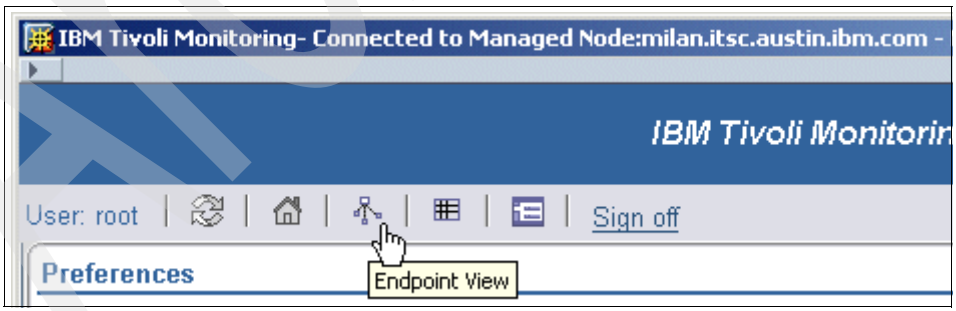

Figure 9-151 Selecting Endpoint view.

Select the Resource Model view to view a status of the environment by all resource models distributed to all selected endpoints as shown in Figure 9-152 on page 547.

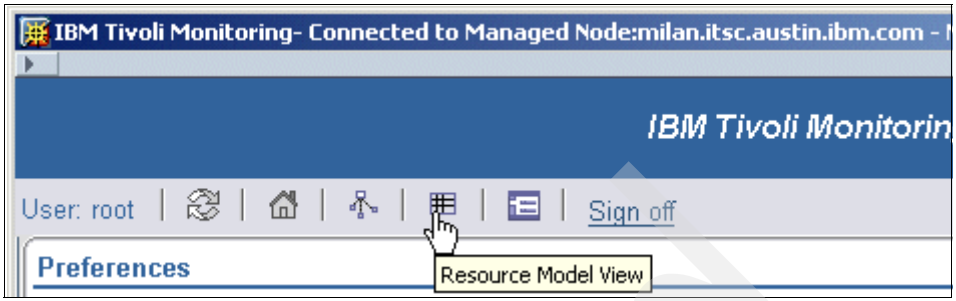

Figure 9-152 Selecting Resource Model view

8. Using the integration at the basic level of implementation that we show in this book consists primarily of viewing the events in the Web Health Console. The instructions in this section show how to initially access the Web Health Console. For has detailed instructions on how to use the Web Health Console see, *IBM Tivoli Monitoring User's Guide Version 5.1.2*, SH19-4569.

The integration can also be accessed through the command line interface. Using the **wdmlseng** command, you can extract an overall view of the resource model status on one or more endpoints through the command line interface. Example 9-10 shows how we extracted the overall resource model status for paris in the lab for this book.

Example 9-10 Viewing overall resource model status

```
milan:/#  wdmlseng -e paris.itsc.austin.ibm.com -verbose

Forwarding the request to the endpoint:
paris.itsc.austin.ibm.com  1187404249.8.522+#TMF_Endpoint::Endpoint#

The following profiles are running:

itm-tws.pf#milan-region
        DMXFileSystem: Running
                LowPercInodesAvail 100 %
                LowPercSpcAvail 100 %
                FragmentedFileSystem 100 %
                LowKAvail 100 %
        SpaceUsed: Running
                SpaceLow 100 %
                SpaceHigh 100 %
                DirNotExist 100 %
        DMXProcess: Running
                HighZombieProcesses 100 %
                ProcessHighCPU 100 %
                ProcessKilledOrNotExisting 100 %
```

```
                ProcessStopped 100 %
appavail-tws.pf#milan-region
        DMXProcess: Running
                HighZombieProcesses 100 %
                ProcessHighCPU 100 %
                ProcessKilledOrNotExisting 100 %
                ProcessStopped 100 %
```

When an indication of a resource model reaches zero percent (0%) in this output, the resource model has determined that a problem exists.

It is also possible to retrieve detailed information about a single resource model on an endpoint, so that the parameter associated with an event or failing resource model is displayed. Example 9-11 shows how we used the **wdmlseng** command in the lab for this book to show the Process resource model at a detailed level when the resource model showed the ProcessKilledOrNotExisting indication has reached zero percent.

Example 9-11 Viewing detailed resource model status

```
milan:/#  wdmlseng -e paris.itsc.austin.ibm.com \
-p 'appavail-tws.pf#milan-region' -r DMXProcess -verbose

Forwarding the request to the endpoint:
paris.itsc.austin.ibm.com  1187404249.8.522+#TMF_Endpoint::Endpoint#

The following profiles are running:

appavail-tws.pf#milan-region
        DMXProcess: Running
                HighZombieProcesses 100 %
                ProcessHighCPU 100 %
                ProcessKilledOrNotExisting 0 %
                        Keys = (name=basename=jobman;) 0 %

                ProcessStopped 100 %
```

In this example, the resource model is reporting that the jobman process no longer exists. Details on how to use the command line interface are in *IBM Tivoli Monitoring User's Guide Version 5.1.2*, SH19-4569.

Some environments have a notification server, or the staff have pagers that can be sent pages via e-mail. In these environments, actions can be configured for the application status resource models to automatically e-mail or notify staff when any of the TWS processes are stopped. We briefly discuss actions in 9.8.18, "About configuring an action" on page 528.

Best practice calls for small (up to 50 processors) scheduling networks for automatically and periodically refreshing the resource model distributions to ensure all processors have a current and up to date set of resource models. Use the **wdmdistrib** command to distribute profiles, and the periodic runs can be scheduled from TWS. See *IBM Tivoli Monitoring User's Guide Version 5.1.2*, SH19-4569 for details on the **wdmdistrib** command.

Larger (50 or more processors) scheduling networks may not want to refresh the profiles on all systems, and may opt to instead report on all configured resource models. The **wdmlseng** command can help in creating reports to address this requirement.

This book does not discuss enabling and configuring the heartbeat function of Tivoli Monitoring because it is not directly related to the integration with TWS. However, in an actual production environment this function would be used to ensure the basic health of endpoints and their connection to their gateways.

These are only some of the basic uses of the integration. More advanced usage involves feeding the data collected by Tivoli Monitoring into additional integrations with help desk ticketing systems for automatically creating tickets when system conditions first start to deteriorate, data warehouses for correlating overall server hardware performance and job processing performance, and so on.

9.10 Possible enhancements

This section shows the power of integrating across other Tivoli products with TWS, using the integration with Tivoli Monitoring as a starting point. These are ideas that can help you identify other needs in your environment that can be addressed by deploying Tivoli solutions.

The base Tivoli Monitoring product has many complementary monitoring products in the Tivoli Monitoring family. Each product incorporates resource models implementing best practice monitoring for specific application servers, and includes the following products:

- Tivoli Monitoring for Applications

  For mySAP.com and Siebel e-business applications.

- Tivoli Monitoring for Business Integration

  For WebSphere MQ, WebSphere MQ Integrator, WebSphere MQ Workflow, and WebSphere Interchange Server.

- Tivoli Monitoring for Databases

  For IBM DB2, IBM Informix®, Oracle, Microsoft SQL Server and Sybase database servers.

- Tivoli Monitoring for Messaging and Collaboration for Lotus® Domino® Servers

  For IBM Lotus Notes® Domino servers.

- Tivoli Monitoring for Messaging and Collaboration for Microsoft Exchange Servers

  For Microsoft Exchange servers.

- Tivoli Monitoring for Microsoft .NET

  For the Microsoft .NET environment's components: Biz Talk Server, Commerce Server, Content Management Server, Host Integration Server, Internet Security and Acceleration Server, Sharepoint Portal Server, and UDDI Services.

- Tivoli Monitoring for Web Infrastructure

  For Apache HTTP Server, Microsoft Internet Information Server, iPlanet Web Server, WebSphere Application Server, and BEA WebLogic Server.

These products represent the easiest way to rapidly install and configure availability management of specific application servers. The extensive line of Tivoli OMEGAMON products can also be integrated into Tivoli Monitoring as well. This provides the enterprise with a broad suite of options from which to select ready-made monitoring solutions.

Comprehensive proactive monitoring of a TWS scheduling network would encompass not only the system level monitoring of computing resources such as processor, disk, file systems, and so forth., but also application level monitoring like databases. This helps ensure that potential problems that might impact the production plan are identified as soon as possible, ideally long before a job launches with dependencies on the application level resources.

Integrating comprehensive monitoring across an enterprise's technology "stack" leads to a lot of data returned by all the monitoring points. Also, even with the local event correlation facility offered by Tivoli Monitoring, there are still many scenarios where the correlation has to cross system boundaries. Tivoli Monitoring's Web Health Console delivers a "micro" view of the managed resources, but in larger or more complex environments a higher level management solution is necessary for effective systems management.

Tivoli Enterprise Console and Tivoli Business Systems Manager provide these higher level views. Tivoli Enterprise Console enables sophisticated dynamic analysis, correlation and root cause analysis of events across all systems. This

allows enterprises to reduce cascades of events to one or a few root cause events.

Tivoli Business Systems Manager enables management of the dependencies between business components and their underlying infrastructure. It organizes events into line of business views, letting enterprises plan, define, and control their business systems (as contrasted with a technical view of the systems).

10

Integrating Tivoli Business Systems Manager

Tivoli Business Systems Manager provides intelligent management software to help businesses increase operational agility by aligning IT operations to business priorities. Tivoli Business Systems Manager (TBSM) is an object-oriented application that provides monitoring and event management of resources, applications and subsystems within the enterprise. The executive dashboard provided with TBSM visualizes the health of the most critical business services and any associated service level agreements.

Following the installation of TBSM, we installed the Intelligent Monitor for Tivoli Workload Scheduler Distributed.

This chapter discusses the prerequisites for the installation as well as the steps of the installation. It also provides an overview of the process, details on the configuration, and information about starting and operating the adapter. Finally, this chapter includes troubleshooting and testing tips.

10.1 Prerequisites

The following software is required for Intelligent Monitor for Tivoli Workload Scheduler (TWS) Distributed:

- Tivoli Business Systems Manager 3.1 (plus Interim Fix 3.1.0.0-TIV-BSM-IF0006)

  or

  Tivoli Business Systems Manager 2.1.1 (plus LAFix 2.1.1-BSM-0012-E05)

> **Note:** Interim Fix 3.1.0.0-TIV-BSM-IF0006 is available. It is available as anonymous ftp (password is your e-mail address). You can get this fix, using either a browser (per the example) or an ftp agent, at:
>
> ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_3.1.0.0/3.1.0.0-TIV-BSM-IF0006

- Tivoli Workload Scheduler Distributed 8.2 with Fix Pack 5 or higher
- The Microsoft JDBC driver for the platform on which this intelligent monitor is being installed (AIX, Linux, Solaris, or Windows). The drivers are available for free from the Microsoft Web site:

  http://www.microsoft.com/downloads/details.aspx?FamilyID=9f1874b6-f8e1-4bd6-947c-0fc5bf05bf71&DislayLang=en

  > **Note:** Tivoli Business Systems Manager uses the Microsoft JDBC driver on both the Common Listener server *and* on the system with the TWS adapter.

The following platforms are supported by the Intelligent Monitor for Tivoli Workload Scheduler:

- AIX 4.3.3, 5.1, 5.2
- Solaris 8, 9
- RedHat Linux 7.2, 7.3
- Windows 2000 Server, Advanced Server, with SP3

For all supported operating system versions, consult *Tivoli Workload Scheduler V8.2 Release Notes (Maintenance Release April 2004)*, SC32-1258.

10.2 Installing Intelligent Monitor for TWS

The following installation kits are available under the install package file, TWSAdapter_Install_code.tar:

- IMfTWS-aix4-r1
- IMfTWS-solaris2.8
- IMfTWS-linux
- IMfTWS-win32.exe

Note: A JDBC driver must be installed before installing the package. The IMfTWS package requires the name of the JDBC install directory when you install the package.

Untar or unzip the kit into a temporary directory on your IBM Tivoli Workload Scheduler server machine. Refer to the installation instructions provided in the package.

- For Linux installations, issue the command:

  ```
  ./IMfTWS-linux {-console}
  ```

- For AIX installations, issue the command:

  ```
  ./IMfTWS-aix-r1 {-console}
  ```

- For Solaris installations, issue the command:

  ```
  ./IMfTWS-solaris2.8 {-console}
  ```

- For Windows installations, issue the command:

  ```
  IMfTWS-win32.exe
  ```

Follow the instructions that are provided by the installer and reply as appropriate.

To uninstall on UNIX systems, execute the following command from the _uninst directory found under the IMfTWS home directory (for example, IMfTWS/_uninst):

```
./uninstaller.bin {–console}
```

To uninstall on Windows, go to the Windows Add/Remove Programs. The display name for removing the adapter is “Tivoli Intelligent Monitor for Tivoli Workload Scheduler”.

The **-console** argument is used when installing or uninstalling from a remote telnet-type session where the GUI installer cannot be displayed.

10.3 Process overview

Figure 10-1 illustrates the high-level design of the process for integrating TBSM with TWS Distributed.

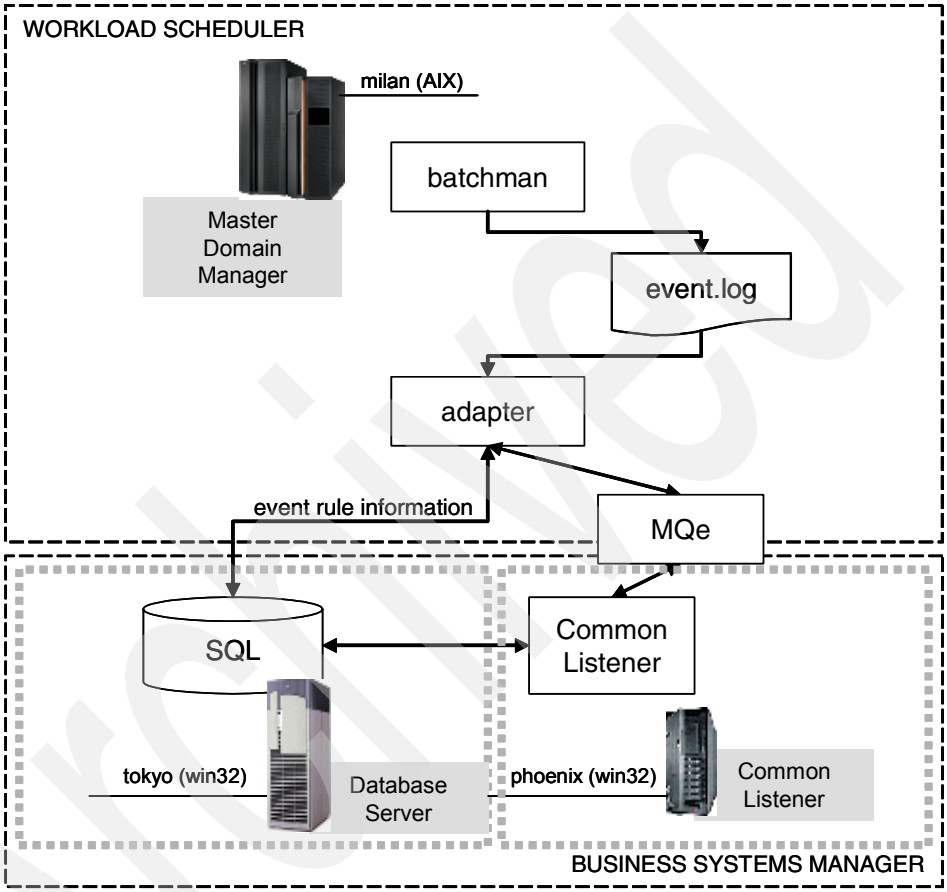

Figure 10-1 High-level design

The IBM Tivoli Workload Scheduler event.log is dependent on the proper setup of the configuration file BmEvents.conf. Refer to “Customizing the batchman events configuration file” on page 664 for complete update instructions. At a minimum, an event.log file must be defined and event messages must be configured to log to this file.

Tip: If the BmEvents.conf file is not found in the *TWSHome* directory, it may be copied from the *TWSHome*/OV directory to the *TWSHome* directory. A restart of batchman is required to pick up changes made to the BmEvents.conf file.

The TBSM adapter daemon maintains a connection with the TBSM Common Listener. This adapter daemon is a Java application responsible for packaging the data and sending it to the remote TBSM Common Listener.

The TBSM event rule information is stored on the TBSM SQL server in the Adapter database. The event rule information consists of three tables in the database that are named Adapter..EventRecognizer, Adapter..EventParser, and Adapter..ObjectAttributes.

Note: The syntax for notating tables in Microsoft SQL server is *Database..Tablename*. We use this convention when referring to specific tables in the Tivoli Business Systems Manager database.

10.3.1 Adapter internals

Figure 10-2 on page 558 depicts the steps that are needed to implement a TBSM adapter according to the cycle time. The steps are:

1. Event Collection
2. Event Tokenizer
3. Event Recognition
4. Event Parsing
5. Event Formatting and Forwarding

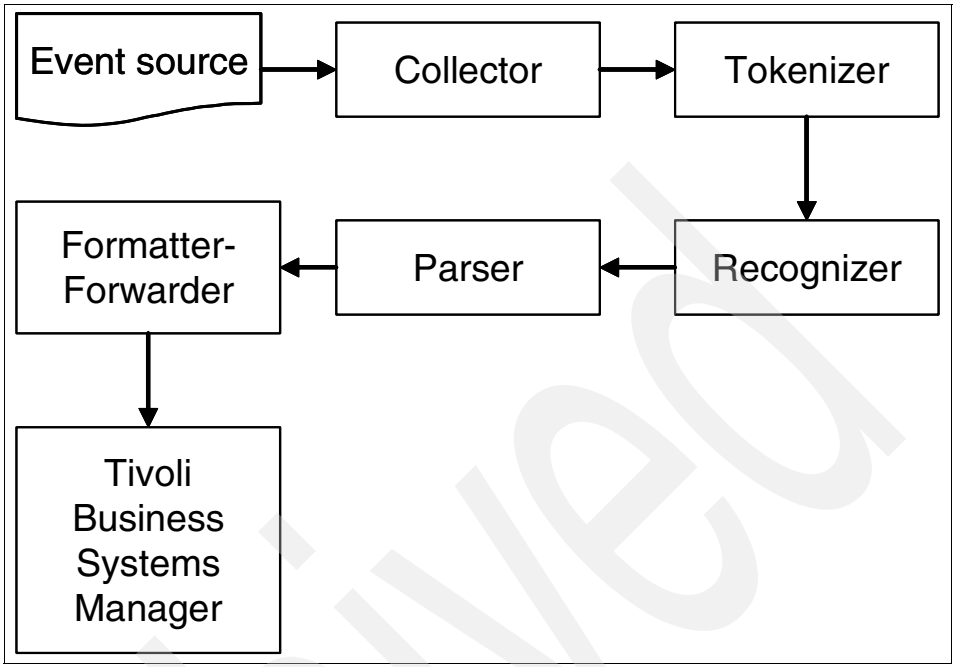

Figure 10-2 Adapter cycle event processing flow

Note: The cycle time is configured in the DriverPlugIns.conf file, on the second line. By default, it reads:

```
CycleTime=120
```

This setting means that the cycle is initiated every 120 seconds. Take care when changing this parameter.

The DriverPlugIn executable calls the event collector to collect the set of events that have been generated since the last cycle and the event collector returns them to the DriverPlugIn executable. DriverPlugIn then hands each of these events to the event tokenizer.

Because not all events are simple space-delimited message strings, the event tokenizer takes the events from the DriverPlugIn and normalizes the events of different types, making the event source name the first item of an array. For example,Tivoli Workload Scheduler messages are passed as TWS. This array is passed to the event recognizer.

The event recognizer determines the message ID for a given message. The event recognizer contains a recognition algorithm which computes an ID for a

given message based on external rules to determine the message's format and interpretation. This is used by the recognizer to access a database table to gather the recognition rules for the given source.

The Event Recognition Rule Table (an example is shown Table 10-1) shows the possible message IDs and the criteria a message must meet to be given the related message ID for each message source. In other words, if an event satisfies the criteria, it is associated with the related message ID. If the message does not satisfy any of the listed criteria, the recognizer returns UNKNOWN as a message ID. UNKNOWN is a reserved word and should not be used as a message ID. However, it can be a means of filtering out undesired events.

Table 10-1 Example Recognition Rule table

EvtSource	EvtCat/Mesage ID	RecCriteria
TWS	TWS:101	(_$_1 == 101)
_TWS	TWS:102	(_$_1 == 102)
...	...	...

Tip: Event sources that have an entry of TWS are events that are recognized. If the event source has an entry of _TWS, the event is defined but is not currently recognized. The table Adapter..EventParser should be changed with care to activate or deactivate recognition of specific event types.

Figure 10-3 on page 560 diagrams the flow of the output and the input of each module.

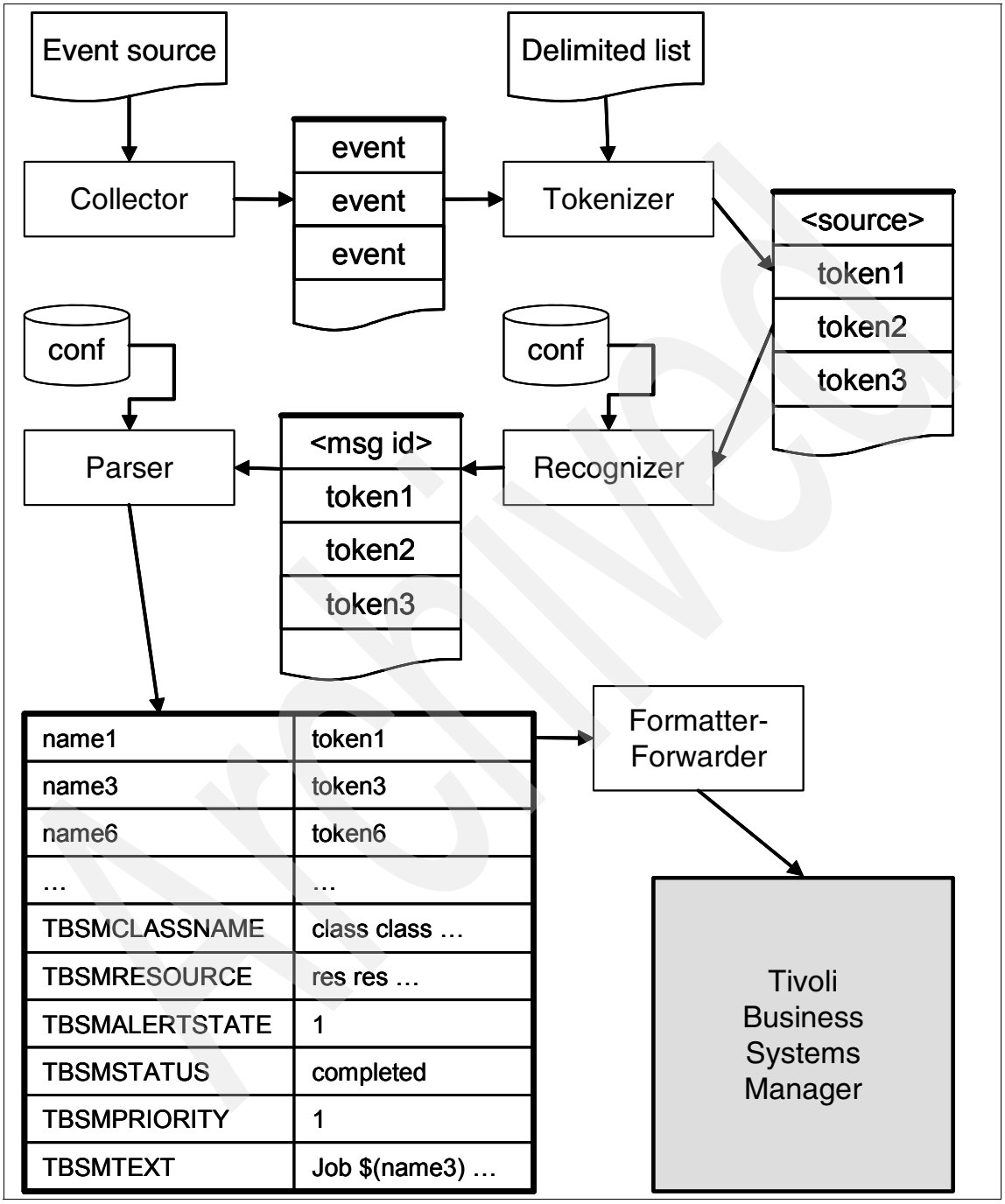

Figure 10-3 Adapter flow diagram

The event parser takes the data from the event recognizer and creates a table for the formatter-forwarder to pass to TBSM. Each message ID from the event parser is a key in another table that returns the following data:

- Parsing Mode
- TBSM Class Name
- TBSM Resource Names
- TBSM Text
- TBSM State
- TBSM Alert
- TBSM Priority

The adapter driver uses the hash table to format and forward the events to TBSM through the formatter-forwarder using the data described above.

10.4 Configuration of the log adapter

Six configuration files are located in the installation directory (typically /opt/Tivoli/IMfTWS):

- AdapterEnv.conf
- DriverPlugIns.conf
- DB_Server.conf
- localadapter.conf
- Log.conf
- TWSConf.conf

These files configure the adapter with data that includes the name of the interface being enabled, the location of the adapter database, where the TBSM database resides, the log adapter information, and what environment variables need to be considered.

Two other files, CollectorInfo.txt and TbsmObjectCache.properties, contain private cache parameters for the adapter driver and are not used for configuration.

10.4.1 AdapterEnv.conf

This configuration file contains environment variables that are used by the adapter driver. You need to update the host name parameter with the short name of the TWS server, without the domain name. The maximum length of the resource name (which includes the host name) is 54 characters. Figure 10-4 shows the Enterprise ITSO_Redbook_Project with events from server milan under it. Example 10-1 shows the updated AdapterEnv.conf file.

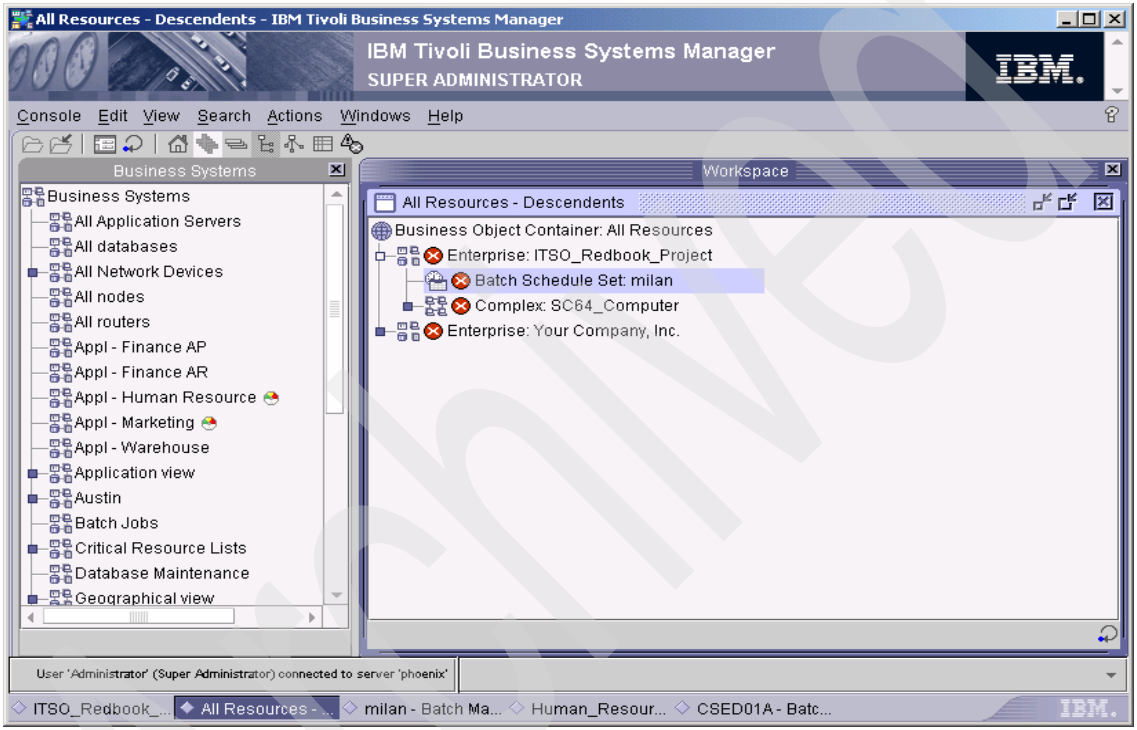

Figure 10-4 ITSO_Redbook_Project Enterprise and the milan Batch Schedule Set

Example 10-1 AdapterEnv.conf

```
[Adapter Env]
ipnw_name_default=UNKNOWN
sep=$(space)
term=~
singlequote=_
prefix=W
Hostname=milan
TWSInstance=m82
...
```

10.4.2 DriverPlugIns.conf

The DriverPlugIns.conf file contains the configuration for the interfaces that the driver implements. The Enterprise parameter in this file requires the Enterprise name under which you would like the batch schedule sets and their resources to be displayed on the TBSM physical hierarchy. Normally, it is the only setting that needs to be changed in this configuration file. Figure 10-4 on page 562 shows the Enterprise ITSO_Redbook_Project with events arriving from server milan. Example 10-2 shows the updated DriverPlugIns.conf file.

Example 10-2 DriverPlugins.conf

```
[PLUG IN]
  PluginName=TWS
  CycleTime=120
  MaxSecBack=-1
  Enterprise=ITSO_Redbook_Project
 #C1=---- COLLECTOR DATA ----
  CollectorMode=inline
  CollectorClassPath=.
  CollectorClassName=com.ibm.tbsmadapter.TWSCollector
 #C2=---- TOKENIZER DATA ----
  TokenizerMode=inline
  TokenizerClassPath=.
  TokenizerClassName=com.ibm.tbsmadapter.TWSTokenizer
 #C3=---- RECOGNIZER DATA ----
  RecognitionMode=inline
  RecognitionClassPath=.
  RecognitionClassName=com.ibm.tbsmadapter.DefaultRecognizer
```

10.4.3 DB_Server.conf

The DB_Server.conf contains configuration settings that need to connect to the TBSM database (via JDBC) where the rules are located. Update all of the appropriate parameters with the information that is required to access the TBSM SQL server (such as user ID, password, and SQL server name.). The user *sa* or equivalent should be the user ID that you enter in this file. Example 10-3 shows the DB_Server.conf file.

Example 10-3 DB_Server.conf

```
[ADAPTER DRIVER]
  Server=tokyo
  User=sa
  Password=xyzzy
  JdbcUrl=jdbc:microsoft:sqlserver://
  JdbcClass=com.microsoft.jdbc.sqlserver.SQLServerDriver
```

Update the JdbcUrl and JdbcClass parameters with the appropriate JDBC information as shown in Example 10-4. The default values point to the default settings for the Microsoft JDBC driver.

Example 10-4 JDBC URL/Class Parameters

```
JdbcUrl=jdbc:microsoft:sqlserver://
JdbcClass=com.microsoft.jdbc.sqlserver.SQLServerDriver
```

Note: Because the file contains the database the user ID and password, it should be encrypted with the following command on UNIX systems:

```
./bin/DBServerEncryptor.sh
```

Use the following command for Windows systems:

```
.\bin\DBServerEncryptor.bat
```

The command generates a file called _DB_Server.conf. The original file DB_Server.conf can be safely deleted when the command completes.

10.4.4 localadapter.config

The localadapter.config files contains data (such as host names) that are related to the TWS machine on which the adapter is running and the host name of the TBSM Common Listener server.

To configure your adapter to communicate with the TBSM Common Listener, make the following changes to localadapter.config:

- Replace `localhost` in the `transport base properties` lines with the local host name or IP address of the system that contains the adapter. In our example, Example 10-5 on page 565, the name is `milan`. You should also change the following fields in this section:
  - transport.local.ip.address
  - transport.request.address
  - transport.response.address

Important: Update the host name parameters in the localadapter.config file with the short host name (without the domain name) or the dotted-decimal IP address only. Ensure that you place the correct host name (adapter or common listener system) in each field.

- Replace `localhost` in the `MQe transport properties` section with the IP address or local host name of the system where the TBSM Common Listener is running. In our example, Figure 10-5 on page 565, the name is `phoenix`.

The field that should be changed is the local host for `transport.server.ip.address`.

- Update the proxy if needed.
- Update the trace.level parameters as necessary. By default, the adapter logs all MQE Transport information automatically. The `transport.trace.level` parameter should be set to `false` (instead of `true`) when the adapter environment is running correctly.

Note: A restart of the adapter is required to reflect changes to the configuration file.

Example 10-5 Parts of localadapter.config to change

```
.
.
.
#   This key is set "true" if the logger is enabled ("on") or "false" if
#   the logger is disabled ("off").

  adapter.trace.enable   = false
  transport.trace.enable = true

#   This key can be low, medium or high:
#      low    = errors, exceptions and warnings
#      medium = low plus info and object creation
#      high   = methods entry/exit

  adapter.trace.level    = high
  transport.trace.level  = high
# transport base properties
.
.
.
transport.class.name       =
com.tivoli.commonlistener.transport.mqe.client.ClientTransportMQe

transport.local.ip.address = milan
transport.request.address  = milan.BASETEST.QM+BASETEST.Q
transport.request.port     = 9890
transport.response.address = milan.BASETEST.QM+BASETEST.Q
transport.response.port    = 9890
.
.
.
   # server
    transport.server.mqe.address = ServerQM+ServerQ
    transport.server.mqe.port    = 8082
```

```
    transport.server.ip.address  = phoenix
.
.
.
```

10.4.5 Log.conf

The Log.conf file contains settings to control logging levels. The Log.conf file should be updated only after the adapter is already running successfully. Like the MQE Transport trace level, the adapter trace level is configured to log all information automatically when initially started. When established, the TYPE_LEVEL1 option should be removed from the Log.conf file to limit the amount of logging information that is stored in the Adapter1.log files.

Tip: The log files can be set to a maximum size and a maximum number of log files to prevent unnecessary waste of disk space. In the localadapter.config file, the `trace.max.file.dim` and `trace.max.files.number` should be adjusted according to the instructions in the file.

The Log.conf file consists of the following parameters

- **max_size *n***, which specifies the maximum file size for the log file called Adapter.log in the log directory
- **max_file_num *n***, which is the maximum number of log files. When the log file reaches the size defined by the max_size parameter, the adapter will create another log file called Adapter*n*.log. where *n* is an incremented number of log files. The max number of log files possible is defined by max_file_num.
- **type_filter *n***, which is the type of the filter. There are currently three different types of filters.
  - **0**, All MaskFilter

    Setting the type_filter to 0 sets the adapter to log events if and only if all filters are satisfied (a boolean AND condition)
  - **1**, anyMaskFilter

    Setting the type_filter to 1 sets the adapter to log events if at least one filter is satisfied (boolean OR condition)
  - **2**, ExcludedMaskFilter

    Setting the type_filter to 2 sets the adapter to log events if and only if the filters are not satisfied (NOT condition)

- **filter**, which are strings separated by space. Possible filters include:

TYPE_INFO	Informational messages
TYPE_WARNING	Warning messages
TYPE_ERROR	Errors
TYPE_FATAL	Critical errors
TYPE_LEVEL1	Debugging messages. TYPE_LEVEL1 is a very expensive type of logging. Use it only when debugging errors.

Attention: At the time of this writing, logging does not work as expected with all type_filter settings. We suggest using only the type 0, which seems to work as an OR condition

Example 10-6 shows an example Log.conf file.

Example 10-6 Log.conf

```
[Log Conf]
max_size=1000
max_file_num=1
type_filter=0
filter=TYPE_INFO TYPE_WARNING TYPE_ERROR TYPE_FATAL TYPE_LEVEL1
```

10.4.6 TWSConf.conf

The TWSConf.conf file contains configuration data that points the adapter to the TWS events file. Update the event log setting with the full path and name of the event.log file that is generated by TWS (*TWSHome*/event.log), as shown in Example 10-7.

Example 10-7 TWSConf.conf

```
[TWSConf]
eventlog=/opt/tws/m82/event.log
dumpfile=./dumpedpointer
```

10.5 Starting the adapter

When you have completed all of the installation and configuration tasks, you can start or stop the adapter daemon using the **adapterInit** command:

```
/opt/Tivoli/IMfTWS/bin/adapterInit [command]
```

Parameters of the **adapterInit** command are:

condstart	Specifies that the adapter should re-process all of the events in the event.log file.
start	Starts the adapter. The adapter will start processing events from the point at which it stopped. If the daemon is already running, a message indicating that fact will be produced.
stop	Stops the adapter, and stores the position of the last monitored event.
debug	Starts the adapter in debug mode.

10.5.1 Starting the adapter at boot time

For UNIX platforms, the adapter daemon is not configured to auto-start. It is left to system administrators to configure their environments to properly start and stop the daemon as the system is booted. In most cases, configuring the adapter to start or stop automatically is as easy as linking the initialization script, which is found in the Intelligent Monitor for TWS bin directory, to the appropriate rc*n*.d directory that used by the system to start and stop programs at startup and shutdown times.

Example 10-8 Installing the adapterInit command to run at boot time

```
cd /etc/rc3.d
ln -s /opt/Tivoli/IMfTWS/bin/adapterInit S93IBMfTWS
ln -s /opt/Tivoli/IMfTWS/bin/adapterInit K93IBMfTWS
```

In a Windows environment, the adapter has a Windows service called *Tivoli BSM Intelligent Monitor for TWS*. The service is created automatically when InstallShield installs the adapter, and it is configured to start automatically.

10.6 Operation

Upon starting, the adapter creates automatically and updates a set of files to keep track of its state and progress.

10.6.1 TbsmObjectCache.properties

The adapter driver creates and expands a file, TbsmObjectCache.properties, with the resources that the adapter driver itself has discovered and, therefore, created in the TBSM repository. When a resource is discovered, it is cached in this file.The key is the concatenation of the classname, the break symbol (|),

and the resource name. For example, if the adapter driver discovers a new resource of class BatchCycles named MySchedSet, it adds a line to the file with key BatchCycles|MySchedSet=*xxx*. The key value is reserved.

Deleting this file and restarting the adapter driver would force a delta discovery of all the resources already discovered. A delta discovery for a given resource can be forced by removing the related line in the TbsmObjectCache.properties file and restarting the adapter driver.

Important: Never edit the TbsmObjectCache.properties file without first stopping the adapter and then creating a backup copy of the file.

To remove an object correctly from both the TbsmObjectCache.properties file and from TBSM, take care to ensure that the items match and are removed together. Example 10-9, from the TbsmObject Cache.properties file, contains the object F100_BSN!PNL_COST. The entry that is highlighted in bold font matches to the schedule that is highlighted for deletion in the close-up of the schedule set list from the Batch Management View in Figure 10-5 on page 570. If you do not remove the object from both places, there will be a mismatch if a new event for the object appears, and the status will not be reflected correctly.

Example 10-9 Data sample from TbsmObjectCache.properties

```
...
...
BatchCycle|milan|m82|F102_CNS|PNL_COST=I
BATCH|milan|m82|F102_CNS|LOAD_XCH|LOAD_XCH-PGMFXCLIENTXR=I
BatchCycle|milan|m82|F100_BSN|PNL_COST=I
BATCH|milan|m82|F100_BSN|CONTROL_EXCP|CONTROL_EXCP-CTRLDIFFEXCP=I
BATCH|milan|m82|F102_CNS|EXCP|EXCP-FILECKSECDBPOS=I
BatchCycle|milan|m82|F100_BSN|REC_JRNL=I
...
...
```

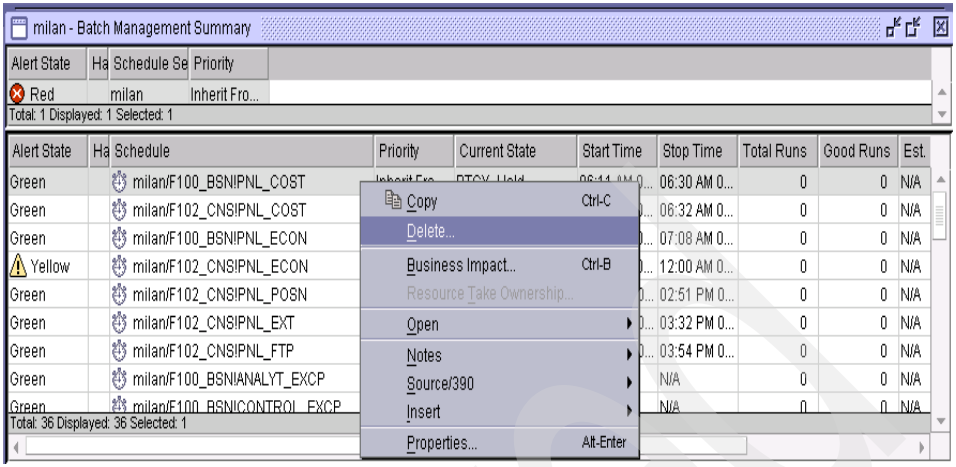

Figure 10-5 Close-up of Batch Schedule deletion

Tip: If any lines are removed from the TbsmObjectCache.properties file, the next time that the adapter driver is started it executes a delta discovery of those resources as soon as it is addressed in an event. Therefore, if a resource is deleted from TBSM, it is good practice to remove the reference to the resource class from the TbsmObjectCache.properties file and restart the adapter, allowing it to perform discovery on the resource class.

10.6.2 CollectorInfo.txt

Any time the adapter starts, it looks for a private file called CollectorInfo.txt and creates the file if it does not exist. The adapter writes a time stamp of the last valid read of the event.log file and uses this time stamp to determine if the event.log has been updated and needs to be read again.

10.6.3 Adapter*n*.log

The local adapter logs various Java class and method statements that are called to the Adapter*n*.log file.

10.6.4 Other tracking files

The following files also have checkpoints and other data reference points: started, dumpedpointer, proc, and proc_check. In a Windows environment, the adapter also creates a file in the log directory called AdapterClientService.log.

This file tracks the Windows service and provides error information that is related to that service.

10.7 Viewing TWS on the TBSM console

This section discusses the ways that TWS objects can be viewed in TBSM. Available views include the physical tree, the batch management view, and the event management view. Each of these views has strengths that make them appropriate for certain audiences. This discussion focuses on views that are available after the objects have been event-discovered to TBSM database.

10.7.1 Viewing the physical tree on the All Resources view

The physical tree is in the All Resources view and begins with the enterprise object ITSO_Redbook_Project. You can drill down the tree to see the batch schedule set milan. Figure 10-6 shows the batch schedule set for server milan on the physical tree, using the All Resources view.

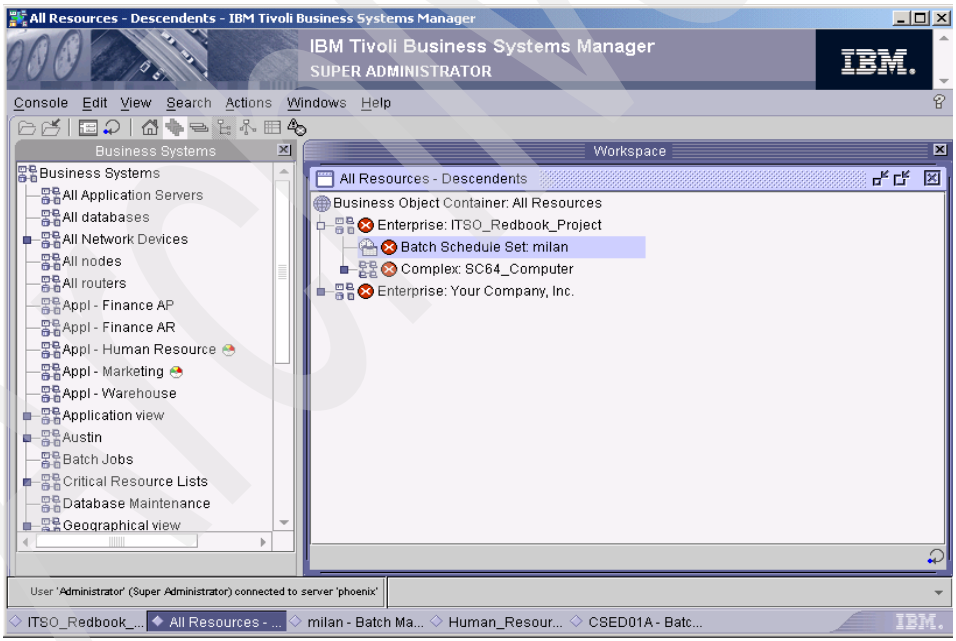

Figure 10-6 Physical Tree on All Resources view

A user can open the properties of the batch schedule set from this view, review child events at the batch schedule level, and then drill further using the properties until the original event on the job is viewed.

10.7.2 The Batch Management view

A better way to view events from the All Resources view is to use the Batch Management Summary view. The Batch Management Summary provides a good view of the batch schedule set, batch schedule and jobs that are scheduled in TWS. Information about customizing this view is available in *IBM Tivoli Business Systems Manager V3.1 Introducing the Consoles*, SC32-9086.

Tip: The appendix in *IBM Tivoli Business Systems Manager V3.1 Introducing the Consoles*, SC32-9086 discusses Key and non-Key jobs. Be aware that for the distributed adapter, key job status in inherited from TWS and the object is treated according to that status. Changing the status of Key or non-Key in TBSM does not change the TWS status, and events are received according to the TWS status of Key or non-Key.

Both Key and non-Key jobshare received by the adapter, and event filtering can be configured by using the Adapter..EventRecognizer table as discussed in 10.3.1, “Adapter internals” on page 557.

Figure 10-7 on page 573 shows a Batch Management Summary for a schedule set that is opened through the job level. The Batch Management Summary view can be opened against any batch schedule set or batch schedule in TBSM. The batch schedule set has three panes that show the batch schedule set, child batch schedules, and their jobs. The batch schedule view has two panes for batch schedules and for jobs.

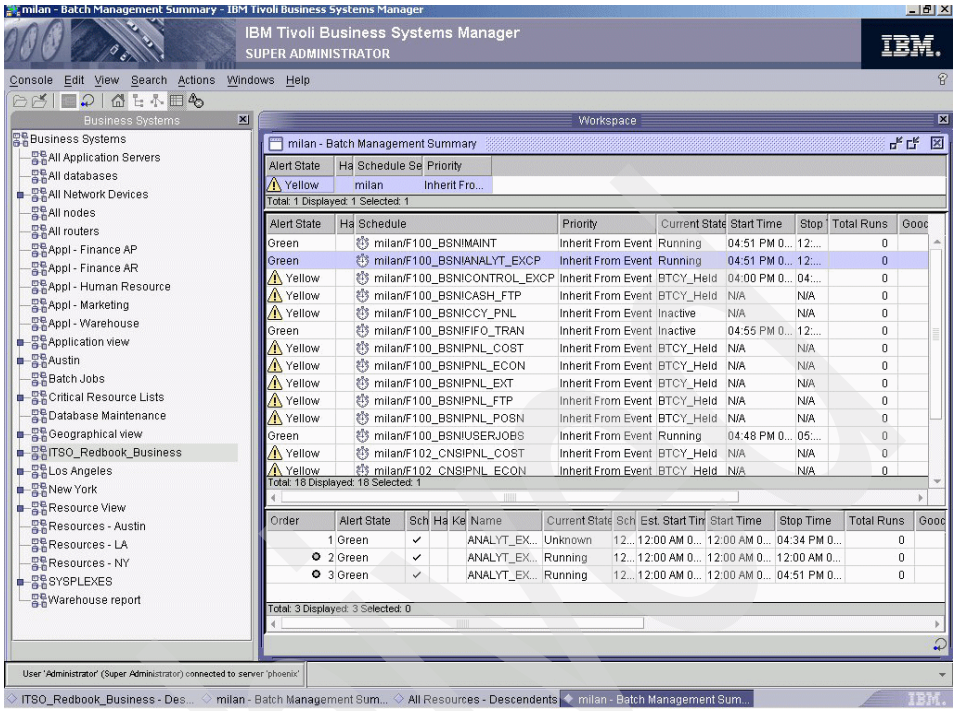

Figure 10-7 Batch Management Summary view

10.7.3 The Event Management view

The Event Management view (Figure 10-8) adds a pane at the bottom of the Batch Management Summary view, allowing direct management of events as they appear in TWS.

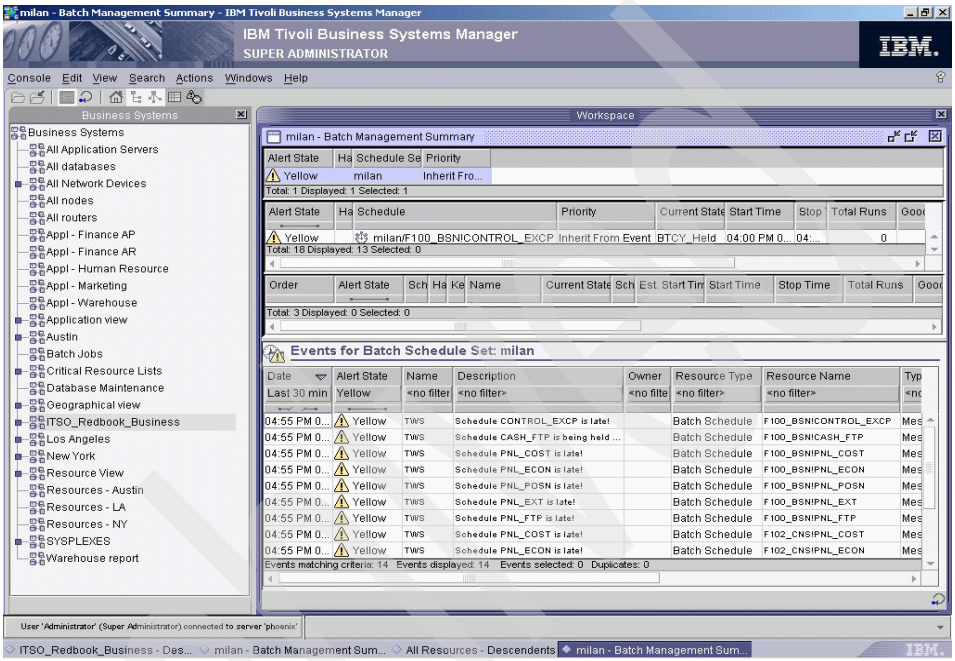

Figure 10-8 Batch Schedule Set view with Event Management view added

The Event Management view can be added by opening the Batch Management Summary and choosing **View | Events** from the main menu. The event view pane that is added at the bottom shows all events that meet the filtered criteria. Figure 10-9 on page 575 shows a close-up of an event viewer pane that is opened from the Batch Management Summary of the line of business ITSO_Redbook_Business. Note that the first filter, `Date` (set to the last 30 minutes), has been broken, so all events are shown.

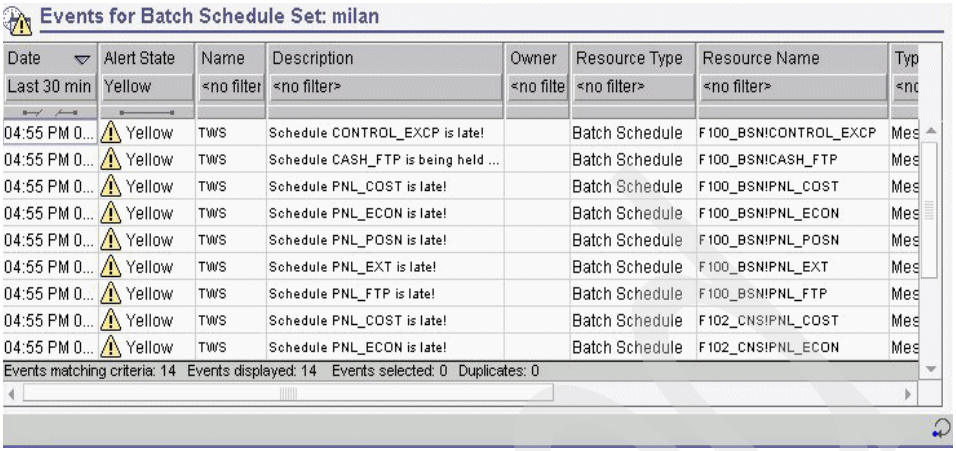

Figure 10-9 Close-up of Event Management pane

10.7.4 Creating lines of business

Objects that are created from TWS include Schedule Set and Schedule objects. They can be drag-and-dropped into lines of business, or they can be added to lines of business by using the TBSM Automated Business Systems method. The Automated Business System process uses a configuration file to match patterns against criteria to create lines of business. Because TWS discovers new schedule sets, schedules, and jobs as they are added, the use of the Automated Business System configuration is a recommended method for placing current and new objects into appropriate lines of business. Use of the Automated Business Systems is discussed in *Tivoli Business Systems Manager V3.1 Planning Guide*, SC32-9088.

10.7.5 Using the Executive Dashboard

After schedule sets or schedules have been placed into lines of business, either through Automated Business Systems or by drag-and-drop, the lines of business can be marked for usage in the Executive Dashboard. Instructions for setting up the Executive Dashboard are in *Tivoli Business Systems Manager V3.1 Introducing the Consoles*, SC32-9086.

10.8 Troubleshooting and testing

This section offers some advice for troubleshooting and testing with the adapter. This section discusses two testing scenarios, testing the adapter prior to having valid TWS connection and deleting and recreating objects in the Tivoli Business Systems database.

10.8.1 Testing the adapter without a TWS connection

You do need not to have TWS installed to test the flow of events from the adapter. All that is needed is a sample TWS event.log file and a proper environment where the adapter can run. With proper configuration, the adapter can process the same events file without the existence of a running TWS environment.

If you see the following error message when running the adapter:

```
Adapter is unable to register with CommonListener
```

Then, you need to:

- Be sure the difference of the system clock time between the TWS server machine and the Business Systems Manager common listener server machine is less than five minutes.
- Check to see that the localadapter.config file has the right value for transport.server.ip.address. When using the host name instead of an IP address, be sure the common listener system is reachable using the given name. In general, if you cannot ping between the common listener server and the adapter server using the host names, the IP Addresses should be used.

Check the Common Listener CL*.log file(s) and look for messages related to `session ID`.

10.8.2 Deleting and recreating objects in the Tivoli Business Systems Manager database

During configuration, the Enterprise name or other configuration data may change and the objects that have been discovered for TWS might need to be deleted and recreated in the TBSM views. This change emphasizes the flexibility of using Automated Business Systems instead of the drag-and-drop method, as the Automated Business Systems discovery automatically repopulates lines of business upon rediscovery.

Important: When any object is deleted from the physical tree (All Resources view), all line of business objects based upon that object must also be deleted from views.

If the physical tree object has been deleted from the database, dependent objects in lines of business should be deleted and rediscovered. This action is critical, because these objects cannot be restored or updated with the same object ID. New events to the deleted physical object and any copies in lines of business will be ignored. If the objects are re-discovered, they have a new object ID in the database and the TbsmObjectCache.properties needs to point to the new objects.

After all discovered TWS objects have been deleted, the adapter should be stopped, the TbsmObjectCache.properties file should be deleted or renamed, and rediscovery should begin as the adapter is restarted. Review the discussion in 10.6.1, “TbsmObjectCache.properties” on page 568 regarding the TbsmObjectCache.properties file. That section contains instructions for when a single or small number of TWS objects are to be deleted and provides more information about the data in the file.

11

Integrating Tivoli Intelligent Orchestrator

This chapter discusses potential product integration scenarios between Tivoli Workload Scheduler (TWS) and Tivoli Intelligent Orchestrator, with possible further integration with Tivoli Provisioning Manager.

Tivoli Intelligent Orchestrator orchestrates automation. It senses why to take action, anticipates when to start provisioning, and prioritizes where to put resources. Tivoli Provisioning Manager coordinates provisioning. It determines what resources to provision and provisions them based on defined business processes. Tivoli Intelligent Orchestrator includes Tivoli Provisioning Manager, a stand-alone product that can be purchased separately as well.

11.1 Reserving batch resources in advance

This scenario uses TWS to assist Tivoli Intelligent Orchestrator in making advance reservations of resources that will be needed at some point in the future for completing work as needed. Figure 11-1 illustrates the integration steps that are involved.

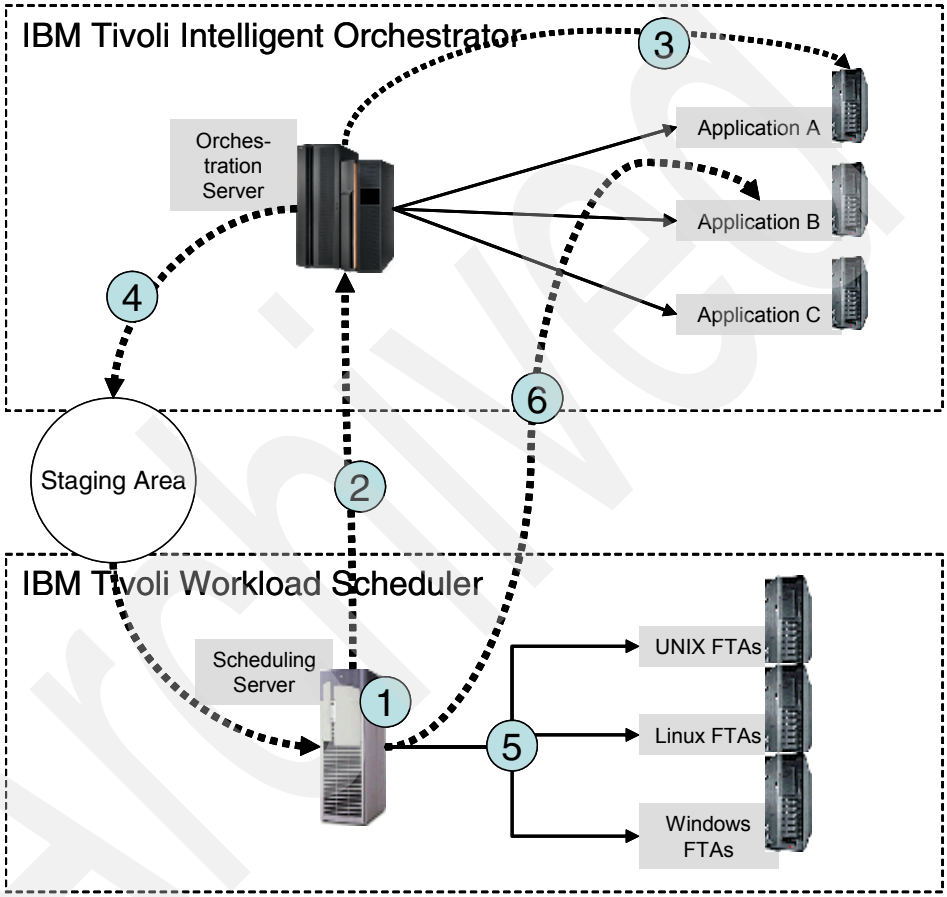

Figure 11-1 Integration with Tivoli Intelligent Orchestrator

The steps that are performed during the use of this proposed integration are:

1. Using a production analysis job, TWS plans the daily workload, estimates its system use requirements, and prepares a resource request.
2. The resource request is sent to Tivoli Intelligent Orchestrator for the daily batch workload requirements.
3. Tivoli Intelligent Orchestrator acquires additional servers, if the affected applications' Service Level Agreements are not violated by doing so, and moves the servers to staging areas.
4. Tivoli Intelligent Orchestrator provisions servers with appropriate software for job scheduling requirements and makes the resources available to TWS.
5. TWS executes daily jobs on the assigned resources.
6. As TWS completes jobs, resources are returned to the applications as needed to be managed by Tivoli Intelligent Orchestrator.

11.2 Applying resources to ensure reliability

This scenario uses Tivoli Intelligent Orchestrator and Tivoli Enterprise Console to provide resource and services to a failed application in the batch production plan while keeping the batch window and associated downtime to a minimum. Figure 11-2 on page 582 illustrates the integration steps that are involved.

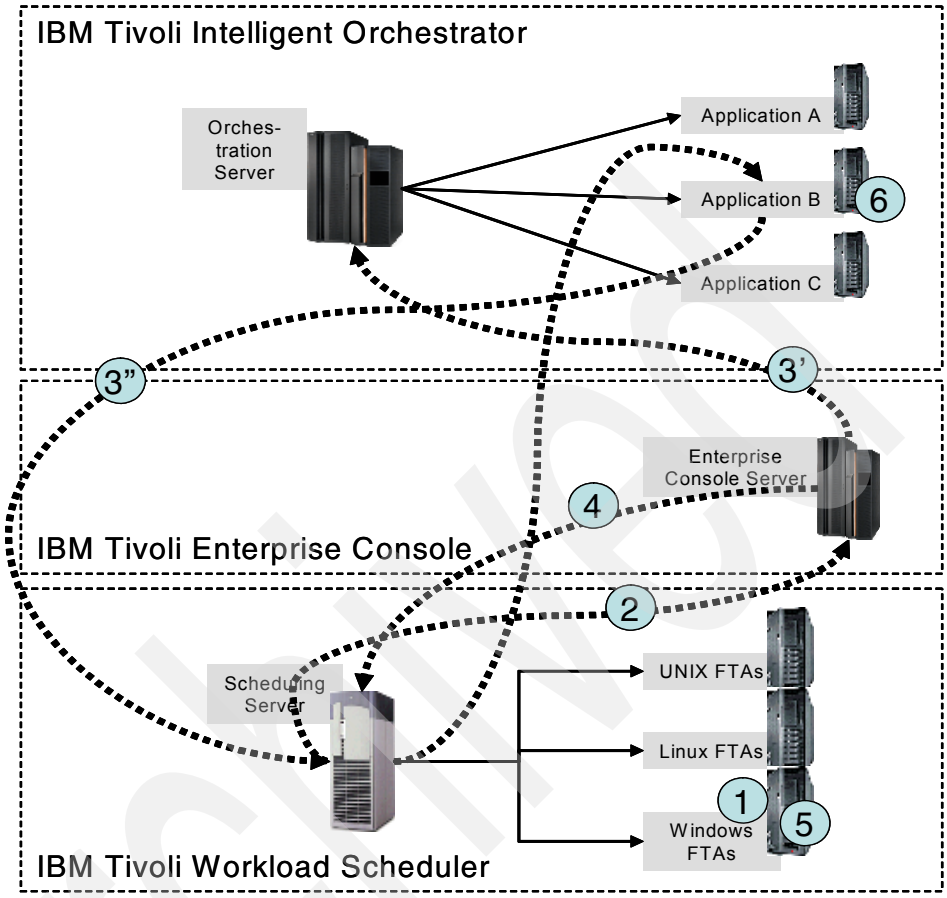

Figure 11-2 Integration with Intelligent Orchestrator and Enterprise Console

The steps that are involved for this integration are:

1. TWS executes a job (which subsequently fails).
2. TWS propagates the failure to Tivoli Enterprise Console, where it is analyzed.
3. Tivoli Enterprise Console executes a workflow for Tivoli Intelligent Orchestrator to provision a new TWS workstation with a newly optimized configuration that can accommodate the now-reduced batch processing window.
4. When the provisioning is complete, Tivoli Enterprise Console notifies TWS to restart the failed job on the new machine.
5. Tivoli Intelligent Orchestrator removes the failing server from the network.

6. When the job successfully completes, Tivoli Intelligent Orchestrator re-provisions the server back to the appropriate application.

11.3 Predictive and automated resource provisioning to ensure reliability

This scenario uses Tivoli Intelligent Orchestrator, Tivoli Monitoring, and Tivoli Enterprise Console to provide predictive resource allocation and services to prevent a failure in the batch production plan. The steps that are performed during the use of this proposed integration are similar to those illustrated in Figure 11-2 on page 582. These steps are:

1. TWS executes a job (which potentially fails owing to inadequate system resources such as processor, memory, and disk).
2. Tivoli Monitoring monitors the resources and propagates the potential resource limitation to Tivoli Enterprise Console.
3. Tivoli Enterprise Console executes a workflow for Tivoli Intelligent Orchestrator to provision additional TWS workstations with a newly optimized configuration that can accommodate the batch processing window.
4. Tivoli Intelligent Orchestrator provisions additional resources as FTAs to help re-distribute the workload ensuring completion within the batch window.
5. When the job successfully completes, Tivoli Intelligent Orchestrator re-provisions the server back to the appropriate application or resource pool.

12

Integrating Tivoli Data Warehouse and Tivoli Service Level Advisor

This chapter describes the integration between Tivoli Workload Scheduler 8.2 and Tivoli Service Level Advisor 2.1 and Tivoli Data Warehouse 1.2.

12.1 Benefits of integrating TWS and TSLA

In today's IT organization, it is an evolutionary journey from a technology focused organization to a business driven organization. IT organizations have implemented several management models to increase their productivity while providing top performing service level agreements (SLA) to their customers.

TWS contains a lot of serviceability information regarding processor use, business flow information, as well as job and schedule runtimes and dependencies. Incorporating this data into SLAs can provide an advantage. By using Tivoli Data Warehouse against the data from TWS, you can then use Tivoli Service Level Advisor to create offerings and SLAs.

With Tivoli Service Level Advisor, you create, track, and manage SLAs between your enterprise and your customers. You can measure, manage, and report on availability and performance aspects of the internal infrastructure of your enterprise. With Tivoli Service Level Advisor, you can quickly and efficiently obtain information to help you manage your IT services. This information enables you to maintain productivity and customer satisfaction, minimize revenue impact, manage costs, and improve planning by assuring offered services. Most customers have a dollar figure associated with their SLAs, and it can literally cause you to loose market share if you do not create the correct baseline.

Figure 12-1 diagrams the creation and life cycle of an SLA.

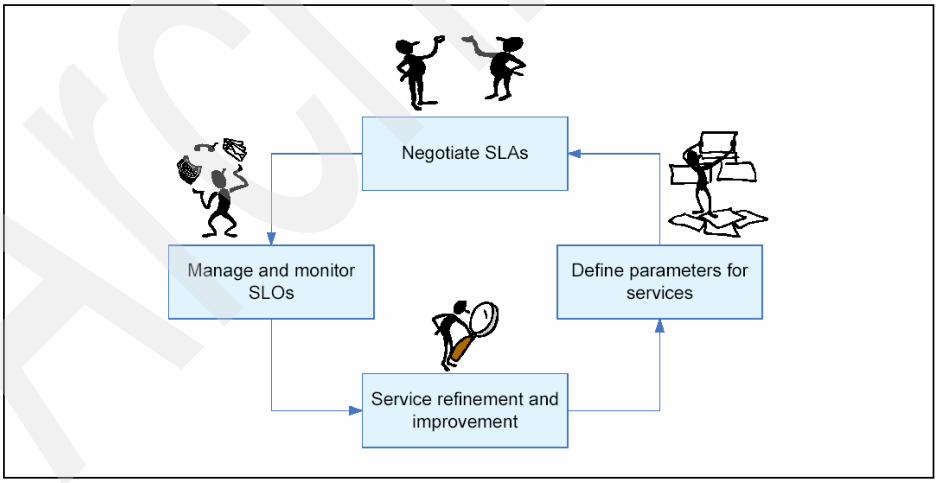

Figure 12-1 Life cycle of an SLA

IBM offers a bundled solution to track and report on business flows that has cohesiveness with SLAs. With Tivoli Data Warehouse as the data collector of

jobs and schedules data and Tivoli Service Level Advisor as the creator and reporter of this data, customers can easily create, manage, and report on the serviceability of their SLAs.

12.2 Tivoli Service Level Advisor

This section provides a basic overview of Tivoli Service Level Advisor, its components, and its functions. The components of Tivoli Service Level Advisor include:

- Service Level Manager (SLM) Server
- Service Level Management (SLM) Reports
- Source Application ETLs (Extract Transform Load)
- IBM WebSphere Advanced Server
- IBM HTTP Server
- Open Data Base Connectivity (ODBC) configuration tool in Windows Administration Tools

12.2.1 Managing your SLAs

An SLA is clearly identified customer and provider deliverables that have:

- Clear terms that are agreed upon by both the customer and the provider
- A fixed period
- A process of reporting and exception handling
- Clearly defined service that is meaningful to the customer
- Measurable, meaningful, and achievable targets

Tivoli Service Level Advisor can provide an understanding of the measurement and Service Level attainment within your organization.

Tivoli Service Level Advisor can also help with:

- Maintaining productivity and customer satisfaction
- Verifying user Service Levels
- Analyzing historical data to predict future Service Levels
- Managing costs and improving planning by assuring offered services
- Measuring, managing, and reporting on availability and performance
- Automating Service Level Management based on Service level objectives
- Evaluating service delivery based on business schedules
- Providing Web-based customer reports

While you might not already have your schedules and jobs in various lines of business, Tivoli Data Warehouse can extract this data, collect measurement data, and summarize it into various report forms. Tivoli Service Level Advisor enables you to:

- Define offerings that are based on different service levels which are based on a set of service level objectives.
- Create customer profiles and associate those profiles with specific service level agreements.
- Provide Web-based evaluation results that can indicate a need for corrective action or ensure that levels of service are being maintained.
- Set up a schedule for the collection, evaluation, and analysis of the data at customized intervals by using TWS
- Analyze the data to detect violations of SLAs for a more predictive analysis.
- Send SLA notifications when violations or trends are detected, so that support personnel can take preventive action and maintain agreed upon levels of performance and availability.

The capabilities of Tivoli Service Level Advisor compliment the data that, in this case, is mined from TWS and is designed to provide metrics to a common data repository. Initially, collecting data is used to form the baseline. With daily retrievals of data from the /schedlog/M* files, this information would populate the Tivoli Service Level Advisor data repository for predictive analysis and to reach a pre-determined threshold.

For instance, JOB2 (printing packing slip for trucking purposes) is depended on JOB1 (order to be picked). We have added a DEADLINE time to this job, because it is critical to the business. If JOB2 is not executed on a timely basis, then we have trucks and people sitting idle and customers who are not receiving goods per the SLA. It was also estimated there would be a loss to the business, approximately USD1.5 million every 15 minutes. Primarily the DEADLINE time is added to a TWS job in order to notify a real-time negative business but also to notify an SLA might be impacted as well. With this information, the customer can then make changes to the SLA itself or its baseline depending on the information.

Table 12-1 on page 589 provides a summary of the benefits and advantages of Tivoli Service Level Advisor.

Table 12-1 Benefits and advantages of Tivoli Service Level Advisor

Benefit	Advantage
Automated SLA evaluation	Eliminates the process of manually reviewing and correlating component level reports again customer SLAs
Improves IT resource productivity	Reduces education and training costs required to support components of SLAs
IBM patent-pending trend analysis identifies IT service delivery problems before they occur	Allows you to take preventive action to maintain Service Levels rather than simply report them
Manage Service Level definitions across IT infrastructure	Increase return on investment of existing systems management tools
The flexibility of Web-based reporting Identifies problem areas and provides executive summary, and detailed operations status of service level agreements	Summarizes the business value of IT resources and can justify cost expenditures
Tivoli Data Warehouse provides open extensible aggregation point for all Tivoli and non-Tivoli systems management data, as well as cross-domain reporting	Leverages the business intelligence for data mining and provides and open interface to include additional monitoring data in SLAs
Maintains customer productivity and satisfaction	Happy customers are repeat customers

In summary, Tivoli Service Level Advisor relies on the monitoring and measuring functions of TWS to gather data through the TWS facilities. Performance and availability data from these applications is kept in a centralized repository using Tivoli Data Warehouse. Tivoli Service Level Advisor operates with Tivoli Enterprise Data Warehouse in managing the data that is used by Tivoli Service Level Advisor. The general flow of monitor data from source application, to TWS, to Tivoli Data Warehouse, and then onto Tivoli Service Level Advisor.

12.3 Tivoli Data Warehouse

This section provides a basic overview of Tivoli Data Warehouse, its components, and functions.

Tivoli Data Warehouse provides the infrastructure for:

- Extract, transform, and load (ETL) processes through the IBM DB2 Data Warehouse
- Schema generation of the central data warehouse
- Historical reporting

Tivoli Data Warehouse consists of a centralized data store where historical data from many management applications can be stored, aggregated, and correlated, as shown in Figure 12-2.

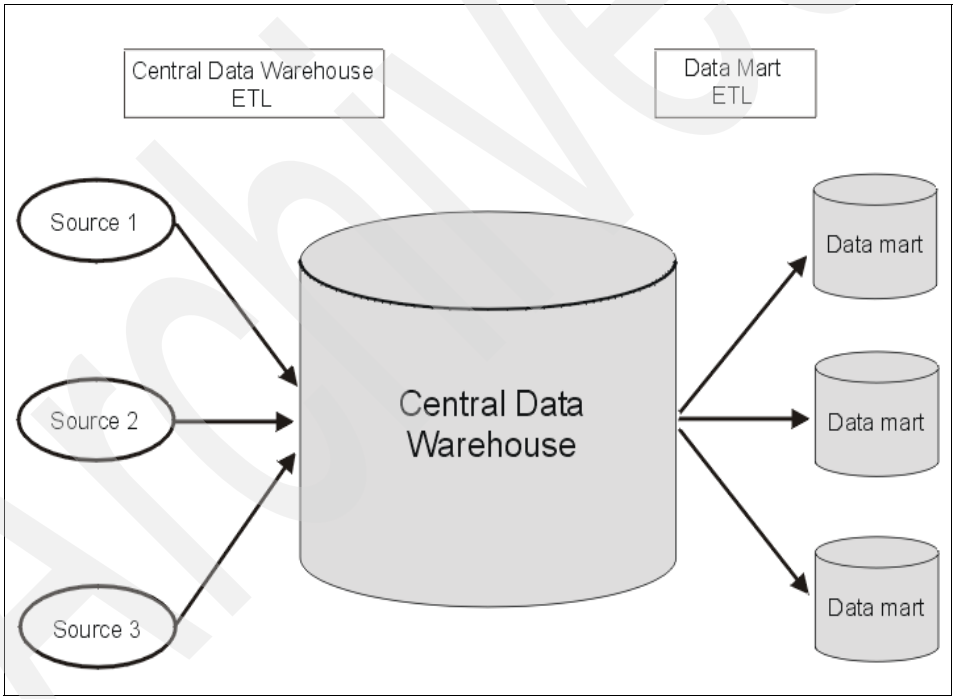

Figure 12-2 Tivoli Data Warehouse overview

The *central data warehouse* uses a generic schema that is the same for all applications. As new components or new applications are added, more data is added to the database. However, no new tables or columns are added in the schema. TWS has the schema code of `AWS`.

A *data mart* is a subset of a data warehouse that contains data that is tailored and optimized for the specific reporting needs of a department or team.

The extract, transform, and load (ETL) process extracts data from a data source and transforms the data into a standard format. It then loads the data into another repository. Figure 12-3 is a high-level diagram of the ETL process.

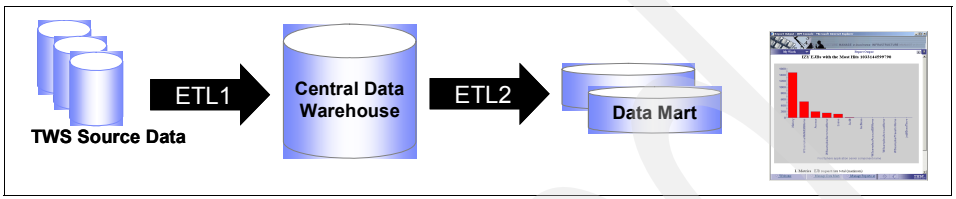

Figure 12-3 ETL process flow

The ETL generally consists of SQL scripts that read data from data sources (for example, previous Symphony files, Central Data Warehouse) and transform and write the data to another database (such as Central Data Warehouse, Data Mart). The *central data warehouse ETL* reads the data from the operational data stores of the application that collects it, verifies the data, makes the data conform to the schema, and places the data into the central data warehouse. The *data mart ETL* extracts a subset of data from the central data warehouse, transforms it, and loads it into one or more star schemas, which can be included in data marts to answer specific business questions.

The ETLs are typically scheduled to run periodically, usually during non-peak hours. If an ETL encounters data that it cannot correctly transform, it creates an entry in an exception table.

12.3.1 Accomplishing service delivery with Tivoli Data Warehouse

In the context of delivering services in a complex IT environment, accomplishing a high level of customer satisfaction requires the IT function of an organization to have a full understanding of and insight into the different aspects of its operation and performance in terms of efficiency and effectiveness. An IT organization must improve service delivery while spending less money. Service Level Management offers IT organizations the ability to deliver the level of service that keeps their businesses competitive.

Running a customer-focused, cost-conscious IT organization is not an easy task. Today, an increasing number of businesses are putting pressure on their IT departments to think of themselves as competitive corporate allies, originating primarily from reasons discussed in the following sections.

Desire to reduce costs

Businesses are constantly seeking ways to reduce costs as well as way to reduce their IT organizations, demanding that IT aligns with business goals not the other way around. The concept of an SLA represents a contract of service delivery between IT and its customers. As a result, IT departments need management services that focus on and support business processes and service delivery rather than just thinking of jobs and schedules.

Use of IT services to gain a competitive advantage

Businesses have recognized that they can gain a competitive advantage when they respond quickly in deploying services, such as online procurement, consumer purchasing mechanisms, and supply-chain delivery, in response to the aggressive goals of their companies. In addition, there is often significant amounts of money at risk.

Increased change management

Competition continues to quicken the pace of change. IT organizations are being forced to improve alignment with their internal customers while developing a competitive mind set using the concept of Service Level Management. When talking about business flows and SLAs, Tivoli Data Warehouse helps determine a successful baseline.

12.4 Installation, configuration, and architecture

When you are installing the three bundled components, TWS Tivoli Data Warehouse, and Tivoli Service Level Advisor, we recommend the following architecture:

- The TWS Master resides on milan.
- The Tivoli Data Warehouse resides on paris.
- A DB2 agent residing on the Master.
- The Tivoli Service Level Advisor on florence.

This design balances the load of processing data and reporting it.

Figure 12-4 on page 593 shows the various pieces of software that were installed to make this scenario work in our environment.

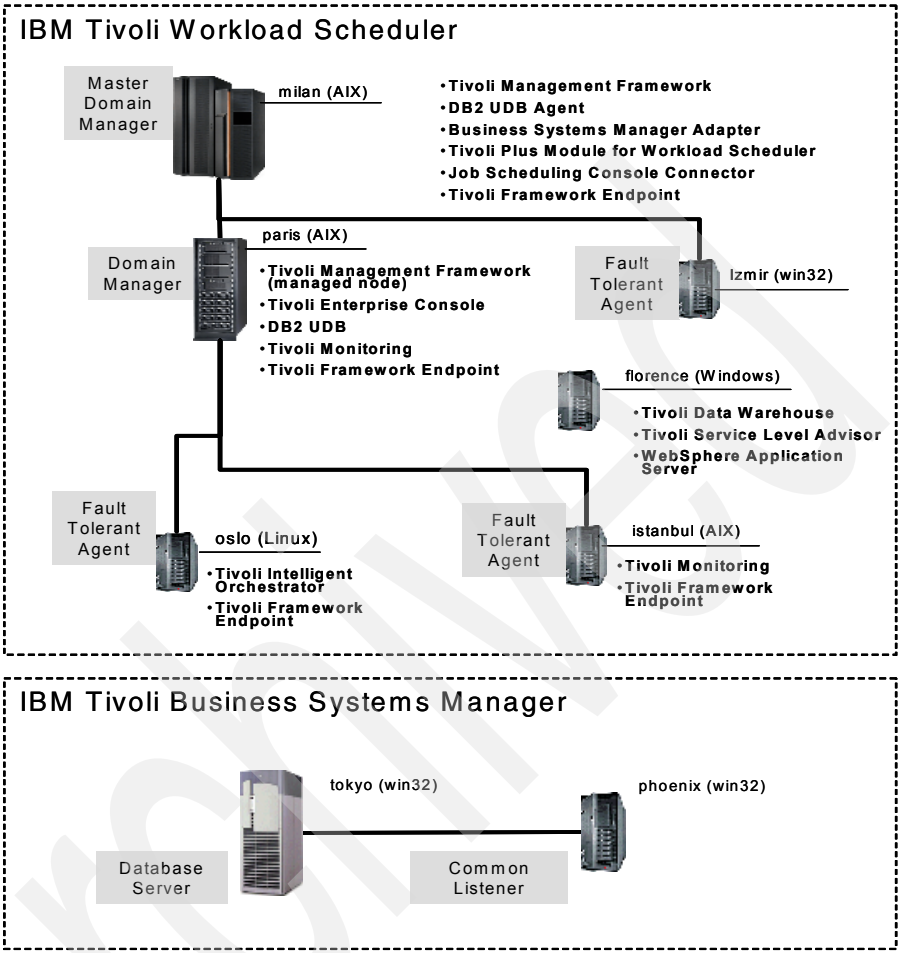

Figure 12-4 High-level architecture

When TWS is installed, you must modify the tws_lauch_archive.pl script. This script reads the M* logs, imports data to the Central Data Warehouse, and holds that data into a staging area. This script resides in the bin directory of the *TWSHome* directory. You need to customize the script as shown in Example 12-1.

Example 12-1 Customizing the tws_lauch_archive.pl script

```
TWS_INSTALL_DIR="E:/TWS";          TWS Installation Directory
DB2USER="db2admin"DB2 User Name
DB2PASSWORD="db2admin"DB@ User Password
DB2_DIR="C:/Program Files/SQLLIB"DB2 Installation Directory
```

This script contains two processes: archiver and import. The archiver process extracts the scheduling history (M* files) from the archived Symphony files and dumps the history into some flat files, (processors, Jobs, Scheds) that are imported into some staging DB2 tables. The import process imports data from those flat files and uploads it into DB2 tables (TWS_WORKSTATION_P, TWS_JOB_P, TWS_JOBSTREAM_P on the AWS schema). See Figure 12-5 for a logical flow.

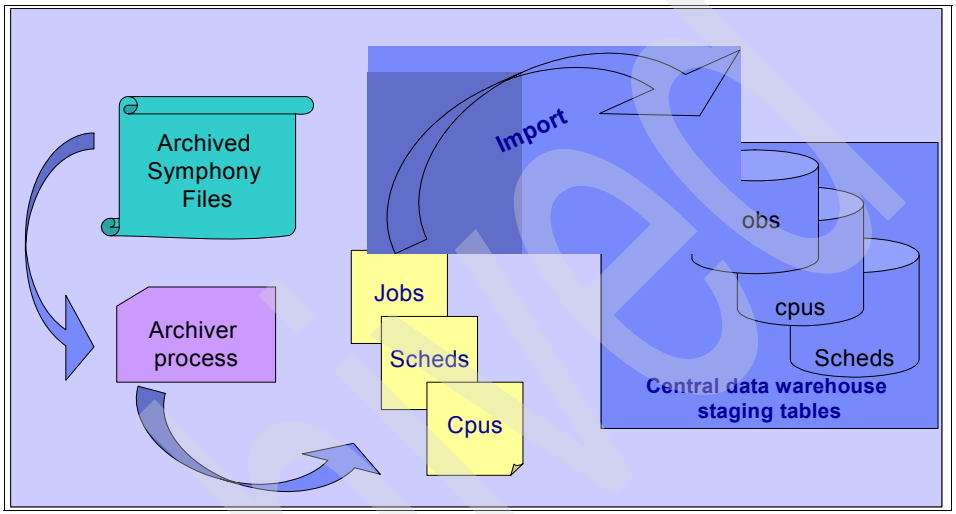

Figure 12-5 Archiver and import process

Because the TWS Master Domain Manager and the Tivoli Data Warehouse control server usually reside on two different machines, for the import process to upload data to the central data warehouse database, a DB2 client must be installed on the TWS Master Domain Manager (see Figure 12-4 on page 593).

Note: All software should be installed prior to running this script. The script will run if the installation is incomplete. However, we recommend that you complete the installation before running the script.

12.4.1 ETL processes

The AWS_c10_ETL_Process performs the central data warehouse ETL function. This process extracts data from staging tables that were filled previously by the tws_launch_archive.pl perl script (that is provided with the TWS). This script runs the archiver process and the **`import`** command.

- The archiver process extracts the last run information of all the jobs and job streams from Symphony files.

- The Import process fills the staging tables with data that was collected by the archiver in the flat files.

The processing flow is as follows:

- The tws_launch_archive.pl perl script runs on the TWS Master workstation. This script extracts the TWS data from the archived plan and saves this data into DB2 staging tables (TWS_WORKSTATION_P, TWS_JOB_P, and TWS_JOBSTREAM_P into the AWS schema) that is used in input by the AWS_c10_ETL_Process to populate the Tivoli Data Warehouse central data warehouse tables.

  The tws_launch_archive.pl script ships in the TWS bin directory. You must customize it to set the proper values for the TWS installation directory and the DB2 client installation directory, user name, and password. You can specify proper values by editing the Installation configuration session into the perl script. You should follow the instructions that are in the script. The tws_launch_archive.pl script, in particular, executes two steps:

  - It launches the Archiver process that reads the Symphony files which are archived in the TWS Schedlog directory and extracts the last run information for all the jobs and job streams, manipulates this data, and then collects it into some flat files.
  - It runs the **`import`** command to transfer data from flat files into DB2 Staging tables.
- The AWS_c10_ETL_Process moves the TWS data from the DB2 staging tables and loads the data into the central data warehouse tables.

You can run these processes repeatedly, typically after a new TWS plan is created. In addition, you can schedule them to run via TWS so that the central data warehouse is updated with new TWS data that is based on the jobs and job streams that have been executed by TWS during the previous Production Plan execution.

With this in mind, it is useful to understand that when a job (job stream) record has been moved into the central data warehouse, it should be not extracted a second time. Therefore, the AWS_c10_ETL_Process is designed so that at each run, it extracts only the jobs that have been executed since the last time it ran. This function of extracting only new data from a data source is referred as *Extract Control.*

To implement the AWS_c10_ETL_Process, the following steps are required:

1. AWS_c10_s040_LoadComp

   This step loads the TWS data from the TWS staging tables and populates the component table in TDW central data warehouse.

2. AWS_c10_40_LoadCompAttr

   This step loads the TWS data from the TWS staging tables and populates the attribute table in TDW central data warehouse.

3. AWS_c10_s040_LoadCompReln

   This step loads the TWS data from the TWS staging tables and populates the CompReln table in TDW central data warehouse

4. AWS_c10_s50_LoadMsmt

   This step loads the TWS data from the TWS staging tables and populates the measurement table in Tivoli Data Warehouse central data warehouse.

Each step produces a log file. You should review these log files for errors. If any of the steps fail, you need to correct the errors as listed in the log file. You then need to run the step again.

Note that AWS is the product code for TWS in the Data Warehouse. You must run each process individually and in the right mode, for example production, test, and so forth. This is initiated from the Data Warehouse Center. It takes a few minutes to execute, and you can schedule all of these process via TWS as well.

Important: The AWS_c10_ETL_Process extracts the scheduling data only from the archived Production plans (M* files) and not from the currently executing production plan.

On the TMR, milan, and the DB2 client, we used the following:

- user ID: `db2inst1`
- password: `db2db2`

To connect to the Tivoli Data Warehouse database server on florence so that you can access the TWH_CDW database to view the TWS data (see Figure 12-5 on page 594):

1. On milan, open a command prompt and change to use db2inst1:

   ```
   su - db2inst1
   ```

2. Connect to TWH_CDW on florence via the DB2 command:

   ```
   db2 connect to twh_cdw user db2admin using db2admin
   ```

Note: DB2 connection was created for the TWH_CDW database on florence from milan.

12.4.2 Tivoli Data Warehouse prerequisites

Before installing the TWS warehouse pack, all fix packs must be installed for TWS, Tivoli Data Warehouse, and Tivoli Service Level Advisor. All supported hardware is reflected in the Release Notes of the three products as follows:

- *IBM Tivoli Service Level Advisor Version 2.1 Release Notes*, SC09-7777
- *Tivoli Data Warehouse Version 1.2 Release Notes*, SC32-1399
- *Tivoli Workload Scheduler Version 8.2 Release Notes (Maintenance Release April 2004)*, SC32-1258

Tivoli Data Warehouse data sources can be run on DB2, Oracle, MS-SQL, Sybase, or Informix databases. Tivoli Data Warehouse Control Database, Central Data Warehouse, and Data Mart can be run only on DB2. Crystal Enterprise (CE) server and client are only supported on Windows environment, but the Web Servers are supported on UNIX and Windows platform.

Figure 12-6 shows the ETL process for TWS data, AWS.

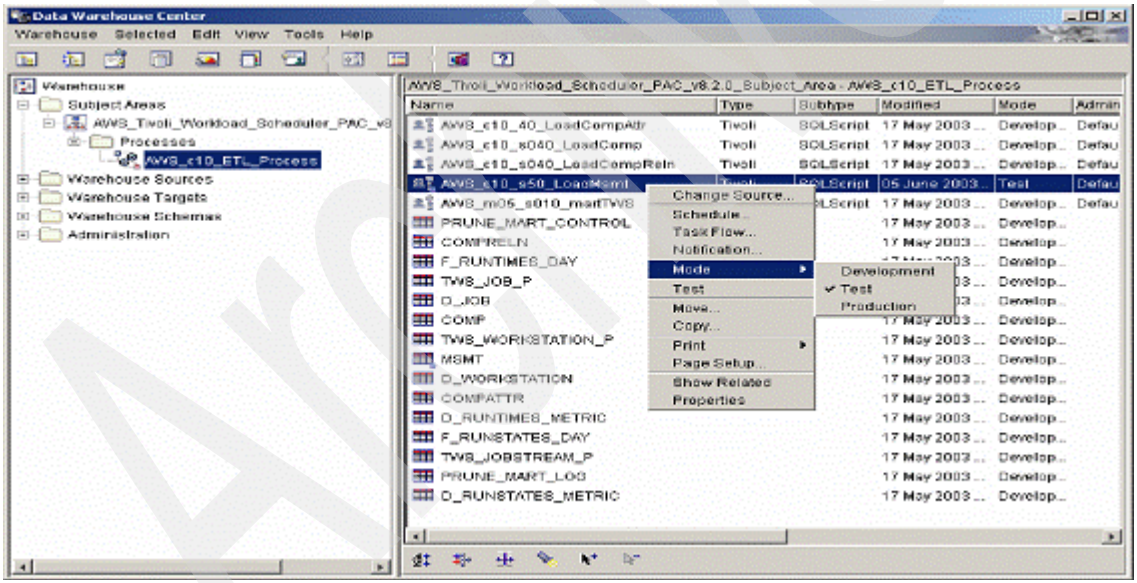

Figure 12-6 ETL process for TWS data, AWS

The Data Mart holds two star schemas, the Daily Run States Data Mart (Figure 12-7 on page 598) and Daily Run Times Data Mart (Figure 12-8 on page 598).

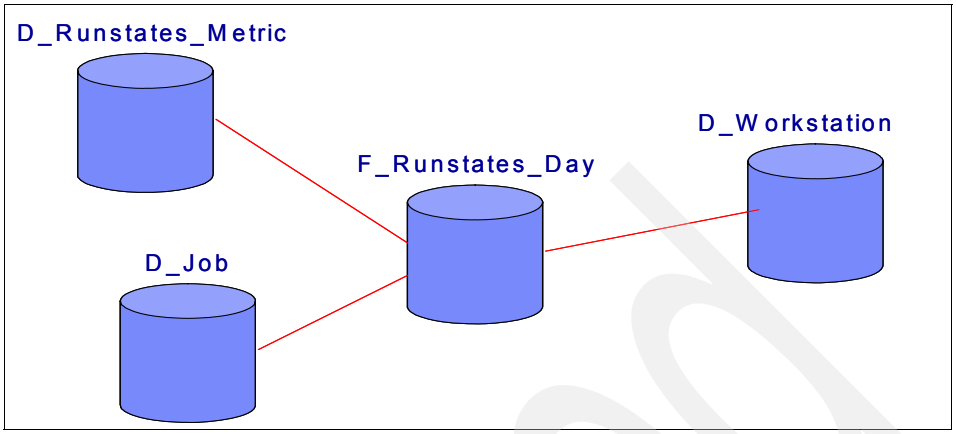

Figure 12-7 D_Runstates - Daily Run States

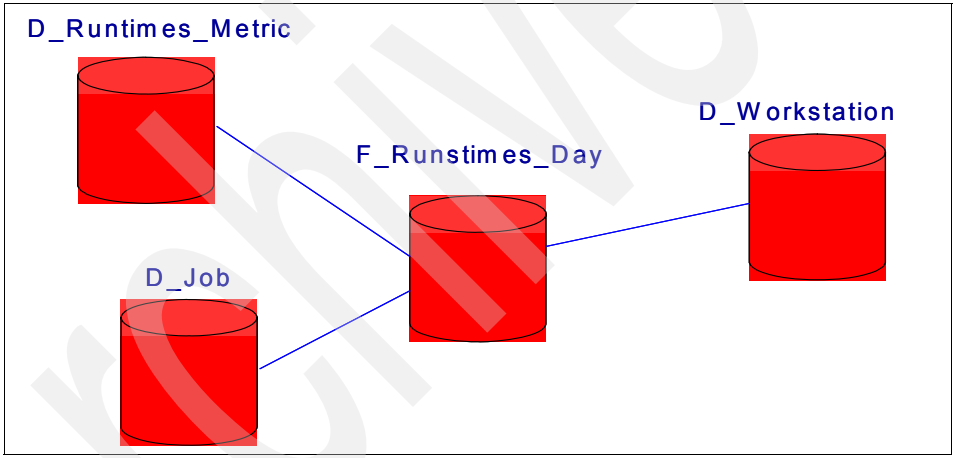

Figure 12-8 D-Runtimes - Daily Run Times

The next step is to login to the Control Center and view the data as shown in Figure 12-9 on page 599. Note that TWS tables have a prefix of `TWH`.

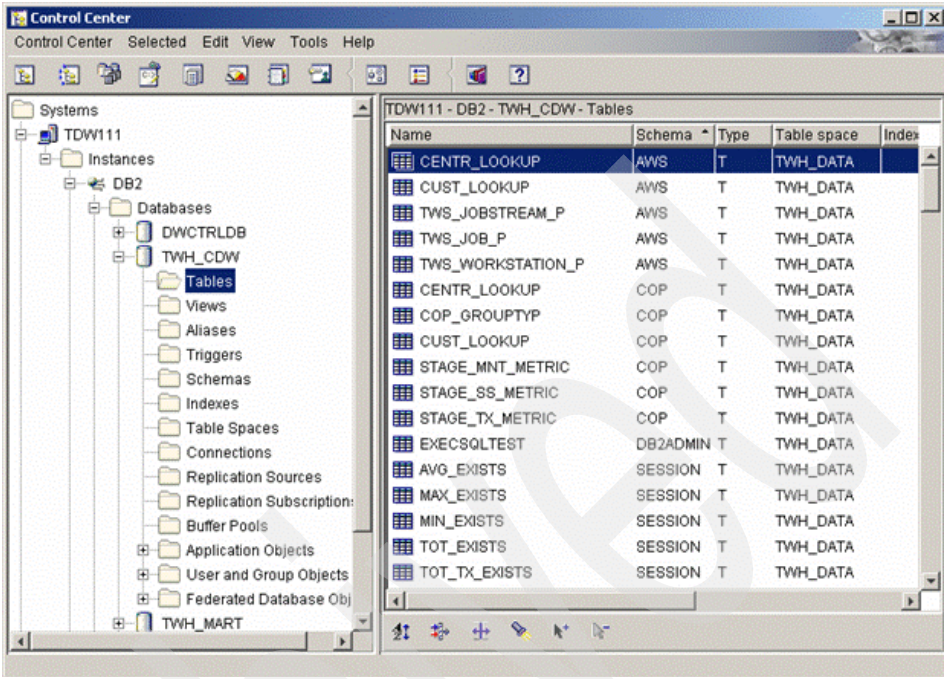

Figure 12-9 Imported data into TWS tables: TWH

Figure 12-10 on page 600 shows the actually data in the data mart for SLA.

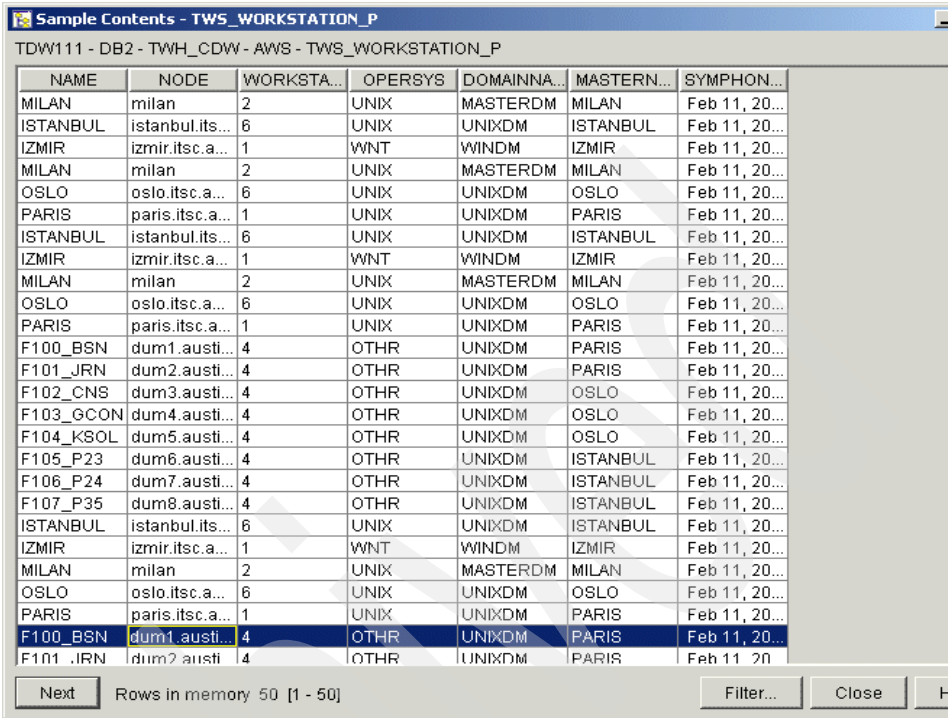

Figure 12-10 TWS data in the data mart now ready for Tivoli Service Level Advisor

In summary, you first need to run the **archiver.pl** script, and then run the import in the Data Warehouse Center. Figure 12-11 on page 601 describes the typical TWS processing and its interaction with the Tivoli Data Warehouse.

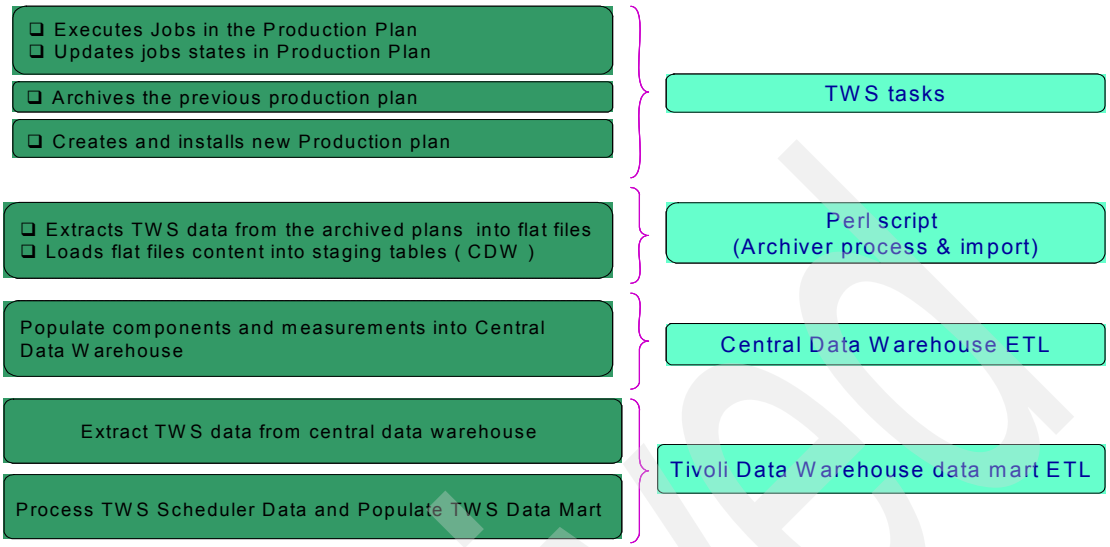

Figure 12-11 Typical TDW processing and the interaction with the Tivoli Data Warehouse

12.4.3 Tivoli Data Warehouse considerations

The warehouse pack must be installed with the db2 user. If that is not the user, you must create a user temporary tablespace for use by the installation program. The user temporary tablespace in each central data warehouse database and data mart is only accessible to that user. If you are installing the warehouse pack using the same database user that installed Tivoli Data Warehouse or if your database user has access to another user temporary tablespace in the target databases, no additional action is required.

If you do not know the user name that was used to install Tivoli Data Warehouse, you can determine whether the tablespace is accessible. You can attempt to declare a temporary table while connected to each database as the user that will install the warehouse pack using the following commands:

```
db2 "connect to TWH_CDW user installing_user using password"
db2"declare gloal temporary table t1 (c1 char(1)) with replace on commit
   preserve rows not logged"
db2 "disconnect TWH_CDW"
db2 "connect to THW_MART user installing_user using password"
db2 "declare global temporary table t1 (c1 char (1)) with replace on commit
   preserve rows not logged "
db2 "disconnect TWH_CDW"
```

In these commands:

- *installing_user* identifies the database user that will install the warehouse pack
- *password* specifies the password for the install user

If the **declare** command is successful, the specified database user can install the warehouse pack. No additional action is required. If the **declare** command fails, run the following DB2 commands to create a new tablespace for the installation in both the central data warehouse database and data mart database:

```
db2 "connect to TWH_CDW user installing_user using password"
db2"create user temporary tablespace usertmps2 managed by system using
   ('usertmp2')"
db2 "disconnect TWH_CDW"
db2 "connect to THW_MART user installing_user using password"
db2 "create user temporary tablespace usertmps3 managed by system using
   ('usertmp3')"
db2 "disconnect TWH_MART"
```

In these commands:

- *installing_user* identifies the database user that will install the warehouse pack
- *password* specifies the password for the install user

12.4.4 Database sizing considerations

The following formula gives approximate considerations for sizing the central data warehouse:

- Consider the component tree (Table Comp) from the root to the farthest tip. Then, give a sample of repeating values for each component based on your environment.
- Consider the number of measurements (msmtTyp table) applicable for each component type (Table MsmtRul).
- Multiple this number by the number of days before the data will be deleted (table Prune_msmt_Control). The default value for daily measurements is 365.
- Multiply this number for the record length of Msmt table (89 bytes).

The following is an example of this formula:

#components x #Msmt_Types x PruneDays x Msmt reclength = msmt_TableSize for one component type

In this formula:

- #components is the estimated number of different instances of a component type, added for all systems for which you are collecting data (for instance, if you have two TWS networks and you estimate 700 jobs in one network and 300 in the second one, your number components will be 1000).
- #Msmt_Type is the number of measurement types applicable for this measurement type (from the MsmtRul table).
- PruneDays is the number of days from the field PMsmtC_Age_inDays in the Prune_Msmt_Control table, before the old data was deleted.
- Msmt reclength is the record length of Msmt table which is 89 bytes.
- Msmt_TableSize is the estimated size for Msmt table.

As an example, if your estimated components are 1000 for which the msmtRul table indicates seven different types, the equation would be:

1000 x 7 x 365 x 89 bytes = 227.395.000 - 216 MB

12.4.5 Tivoli Service Level Advisor overview

Our environment consists of a single machine configuration, where Tivoli Data Warehouse (data source, control database, central data warehouse, and data mart) and Crystal Enterprise (server and Web server) are installed and configured on one machine.

When we have this environment set, we sign into the Tivoli Service Level Advisor console on florence.

WebSphere Application Server must be running in the SLM Report Server host. You can start WebSphere Application Server with the following command:

```
/usr/WebSphere/AppServer/bin/startServer.sh tsla
```

In this command, `tsla` is the WebSphere Application Server name. (The install default can be server1.)

To stop WebSphere Application Server, run the following commands:

```
/usr/WebSphere/AppServer/bin/stopServer.sh tsla
–username <id> -password <pswd>
```

Note: If WebSphere Application Server security is implemented, you need to specify `–username` and `–password` when stopping WebSphere Application Server.

12.4.6 Enabling source application data

There are many Tivoli availability monitoring and performance management applications that might be deployed in your enterprise. The applications that are collecting data and storing it in the Tivoli Data Warehouse database must be registered and enabled in Tivoli Service Level Advisor for service level evaluation. The TWS application name is AWS. To enable source application data:

1. Set up the SLM environment using the ./slmenv.sh script.
2. Register the source application using the following command:

   ```
   scmd sdc registerWarehouseData
   ```

3. Enable the application using the following command:

   ```
   scmd etl enable <avaCode>
   ```

   In this command, avaCode is the three letter code that represents a source application (for example, AMY for Tivoli Monitoring, BWM for Tivoli Monitoring for Transaction Performance, ECO for Tivoli Enterprise Console, and so forth).

12.4.7 Creating a schedule

Schedules are made up of one or more periods that have a start and an end time. Schedules are categorized into business and auxiliary schedules in Tivoli Service Level Advisor.

A *business schedule* can contain one or more auxiliary schedules. *Auxiliary schedules* are used to define the schedule periods that are common to all the business units in the organization. For example, you can include the holidays of the organization where the service levels of the objectives do not matter. Similarly, you can use auxiliary schedules to define a maintenance period as well. You can include one or more auxiliary schedules in a business schedule, but auxiliary schedules cannot contain an auxiliary or a business schedule.

In the following example, we create a business schedule. For this example, the critical period is defined as 9 a.m. to 4 p.m., Monday through Friday. To create a business schedule with this period as critical:

1. Enable hourly evaluation in Tivoli Service Level Advisor.

   Tivoli Service Level Advisor can evaluate the service level objectives (SLOs) every hour, two hours, three hours, four hours, six hours, eight hours, daily, weekly, and monthly. By default only daily, weekly, and monthly intervals are supported. For hourly evaluations supported, run the **scmd** command from a Tivoli Service Level Advisor environment-enabled command window:

   ```
   scmd mem showHourlyFrequencyIntervals enable
   ```

2. To create a schedule, open the Internet Explorer browser window `http://SLMServer:port/SLMAdmin`.
3. Enter user name and password. A window similar to Figure 12-12 opens.

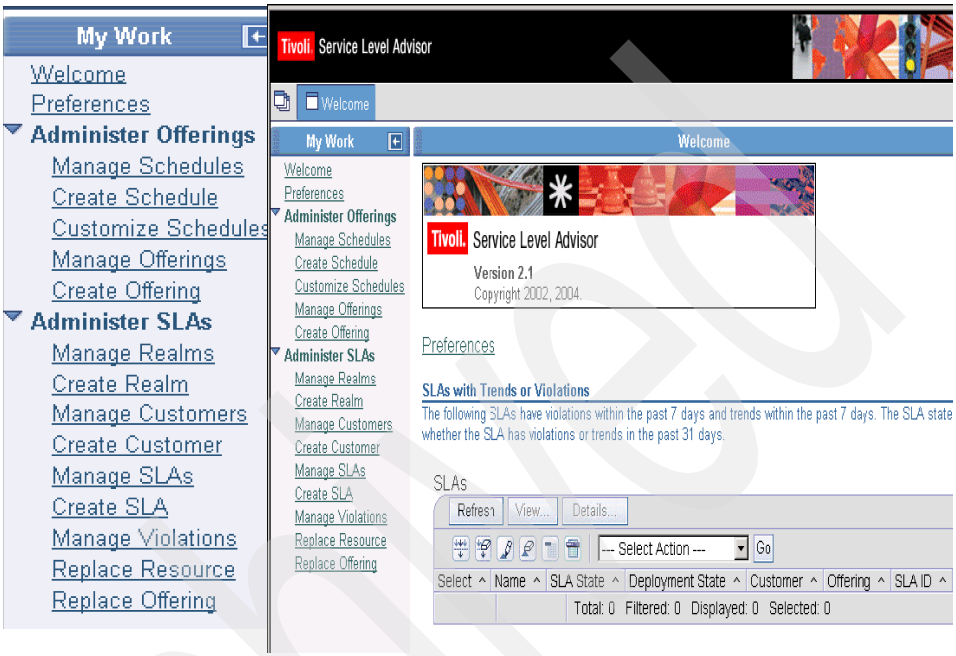

Figure 12-12 Service Level Advisor home page

4. On the Tivoli Service Level Advisor administration console, select **Manage Schedules** → **Create**. The wizard page shown in Figure 12-13 opens.

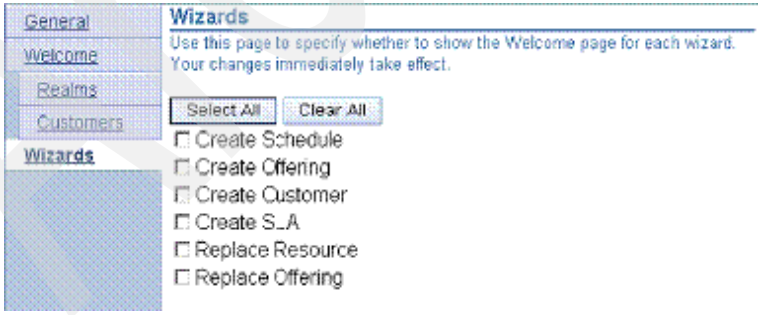

Figure 12-13 Tivoli Service Level Advisor screen for creating a schedule

5. Name the schedule and specify it as type *Business Schedule*, as shown in Figure 12-14.

Figure 12-14 Tivoli Service Level Advisor schedule window

6. Define the business schedule periods as follows:
   a. Under the Define the schedule state to be active during unspecified periods option, select **Standard**.
   b. Select **Create a new schedule period**.
   c. Mark this period as *Critical*.
   d. Select the start time as **9:00** and end time as **15:59**.
   e. Select the frequency as **Weekly**.
   f. Clear the selection for **Saturday** and **Sunday**.
   g. Click **Next** and select **Finish** to complete the schedule creation.

12.4.8 Creating offerings

When data is loaded in the data mart databases, you can configure SLOs using Tivoli Service Level Advisor. An *offering* is a template that is used to describe a service, with agreed upon service levels, that forms the basis for the SLAs in which it is ultimately included. Offerings can be differentiated to provide service level choices to customers, such as Gold, Silver, and Bronze services, or any other naming convention that suggests a unique level of service.

An offering is associated with a business schedule that is defined with one or more schedule periods. Each schedule period is associated with a unique schedule state, such as PEAK, PRIME, STANDARD, OFF HOURS, and others, and you can configure each of these states to represent a unique level of service for that schedule period.

Tivoli Service Level Advisor allows you to define the business schedule for various periods and different states and to define the SLOs in an offering. When an offering is completed, it can be published for inclusion in an SLA.

For example, we can create an offering where the resources are all schedules on systems called paris and milan where we are looking at metrics to include start delay of more then *xxxx* seconds based on breached times. Figure 12-15, Figure 12-16 on page 608, and Figure 12-17 on page 608 show sample screens with these settings.

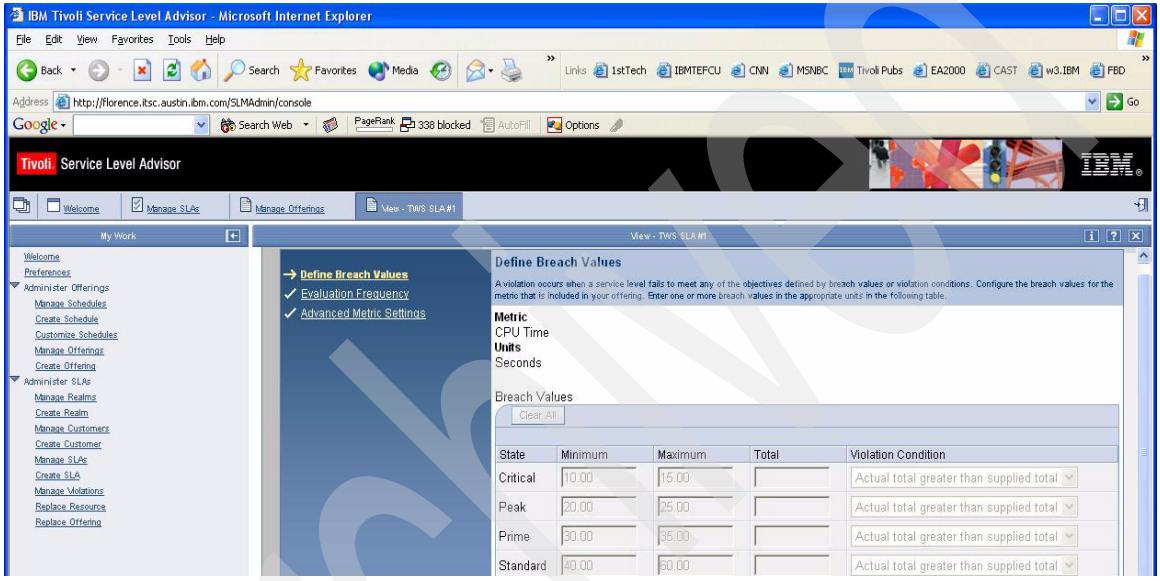

Figure 12-15 Setting up the breach data for your offering

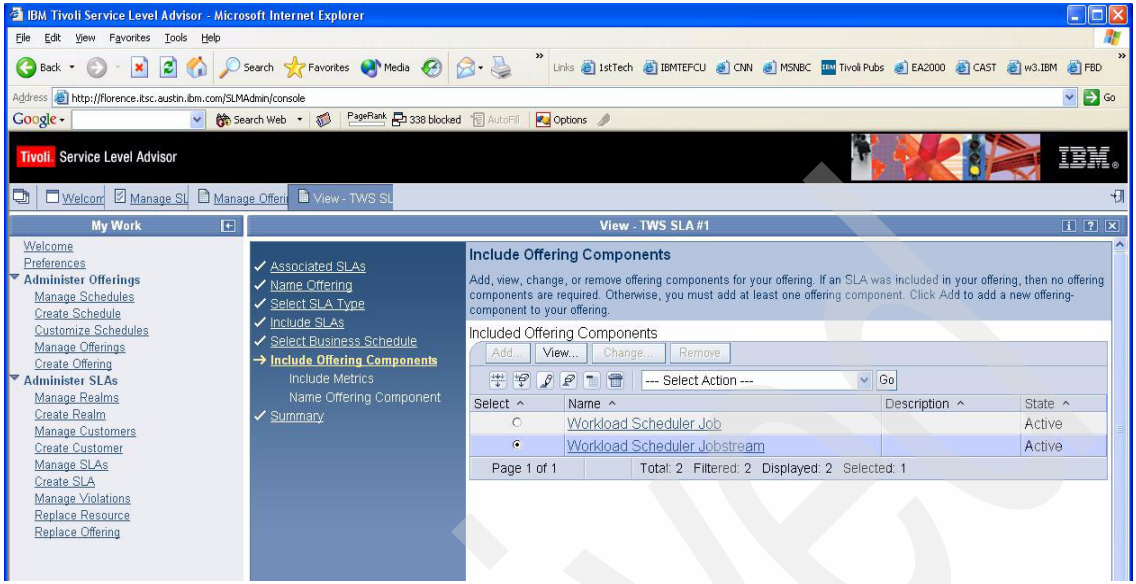

Figure 12-16 TWS components for Tivoli Service Level Advisor reporting

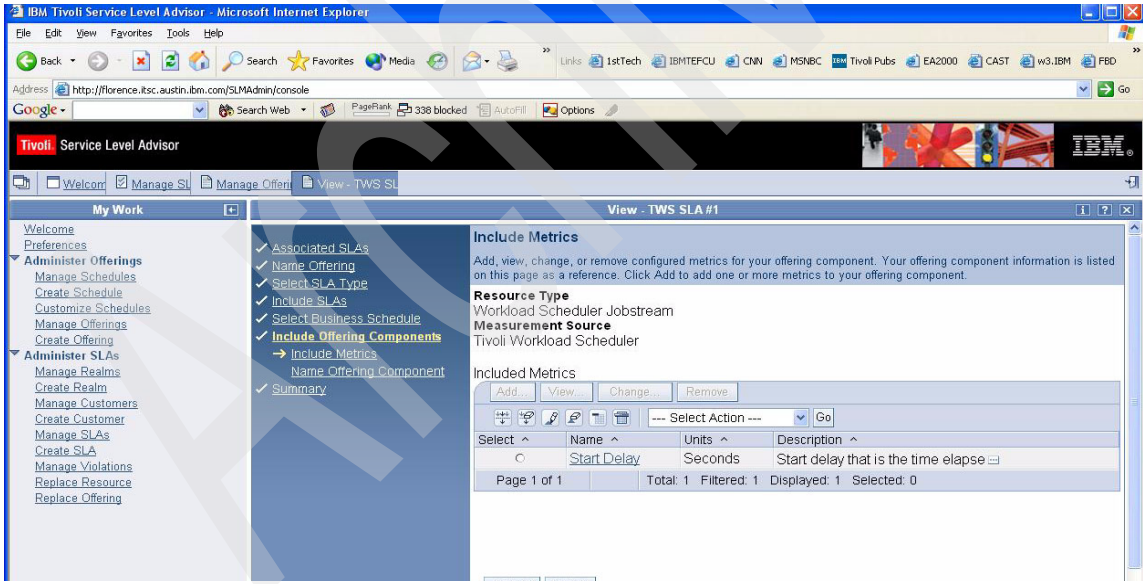

Figure 12-17 TWS Metric of Start Delay of XXX seconds

12.4.9 Creating SLAs

An SLA is an agreement between your enterprise and your customers on expected levels of service for services provided by your enterprise. The SLA is an association between a specific customer, an offering that represents a specific level of service, and a resource or set of resources that are monitored and managed to adhere to the desired level of service.

You must define a customer prior to setting up an SLA. Before setting up a customer information for an SLA, you need to define one or more realms to organize and segment your customers into groups. Customers in the same or different organizations, companies, geographic regions, and so forth can be grouped into realm that segregate one customer's data and reports from another. The realm can also be a line of business.

An SLA includes all the related information of defined schedules, offerings, customers realms, as well as the particular resources for example, processor name, processor use, execution of job, and so forth.

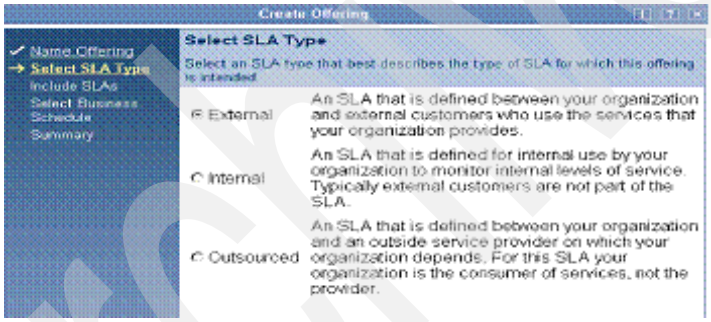

Figure 12-18 Creating a SLA type

12.4.10 Evaluating data for violations and trends

After defining schedules, offerings, customer, and realms and associating them with appropriate resources for a SLA, you must submit the SLA for deployment. Then the SLA is then processed and activated.

After the SLA is deployed, Tivoli Service Level Advisor can begin evaluating measurement data from Tivoli Data Warehouse that has been transferred to the SLM Measurement Data Mart (dyk_dm). SLAs that are submitted to Tivoli Service Level Advisor include schedule information that determines how often evaluations and trend analyses are performed, on a daily, weekly, monthly interval, or multiple times per day if the hourly evaluation frequency has been enabled.

Tivoli Service Level Advisor also allows intermediate evaluations. An intermediate evaluation must occur at a higher frequency than the regular SLO evaluations. For example, you can define a monthly SLO evaluation that performs evaluation and trend analysis for the entire month, along with daily intermediate evaluations that perform a separate evaluation during each day of the month.

When a violation or a trend toward a violation of an SLA is detected, Tivoli Service Level Advisor can create an external notification. When the support personnel receive the notification, they can take immediate corrective action to decrease outage times and improve their ability to guarantee levels of service. The notifications can take any of the following forms:

- An e-mail message
- A Tivoli Enterprise Console event
- An SNMP trap

When you have determined the resource type, including the metrics and the name of the offering, you can then publish the offering. When the new offering is available, you then create a realm and a customer before you can create an SLA.

12.4.11 Creating a realm and a customer

Tivoli Service Level Advisor provides mechanisms called realms and customers that segregate data to ensure that reporting information is made available only to the appropriate people. The highest level of segregation is called a *realm*. A realm contains one or more customers. For example, you can create a realm for all customers in the United States and another realm for customers in Europe. You might also create a realm for customers in a particular line of business within your organization or another grouping that makes sense for your enterprise. Customers can be associated with more than one realm.

The second level of segregation is called a *customer*. A customer must be associated with at least one realm. When SLAs are defined in Tivoli Service Level Advisor, they are associated with both realms and customers.

When Tivoli Service Level Advisor users are given access to reporting functionality, they are given permission to access specific realms and customers. They are unable to view data related to realms or customers for which they have not been granted permissions.

Before you create a customer, a realm must exist. Realms are a grouping of users, line of business, or SLAs by geography. To create realms in Tivoli Service Level Advisor:

1. Launch the Tivoli Service Level Advisor Administration Console.
2. From the portfolio, select **Create Realm**.
3. Enter an appropriate name and optionally provide a description.

After you create the realm, you can create a customer:

1. Launch the Tivoli Service Level Advisor Administration console.
2. Select **Create Customer**.
3. Provide the customer name and a description. For example, we entered the customer name `TSLA1SVT`. Enter a description for the customer and click **Next**.
4. Because we must relate this customer to a realm, click **Add**.
5. Choose the appropriate realm. In this example, customer name `TSLA1SVT` belongs to the realm `KC1SVT`. Click **Next**.
6. Click **Next** again to reach the Summary page.
7. On the Summary page, click **Finish** to finalize the customer creation.

Figure 12-19 shows that we have a customer named `TSLA1SVT` in a realm called `KC1SVT`.

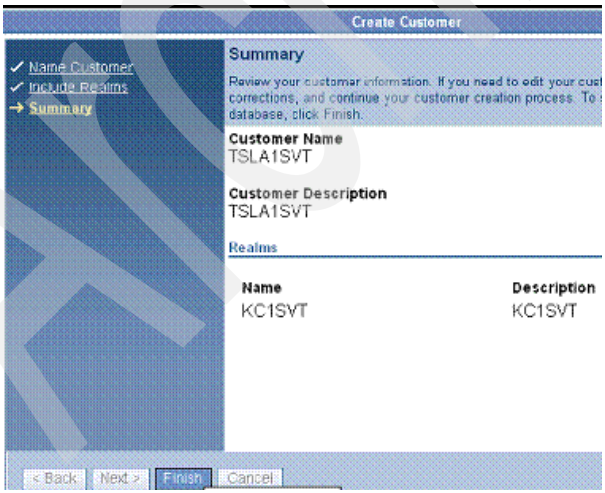

Figure 12-19 Finished product of final step of creating a SLA

12.4.12 Creating the SLA

To create the SLA:

1. Launch the Tivoli Service Level Advisor Administration console.
2. In the portfolio, select **Create SLAs**.
3. In the Name SLA panel, name the SLA. Optionally provide a description. Click **Next**.

   Note: Naming standards should be implemented for effectively identifying your SLAs.

4. In the next panel, select an existing customer to be associated with the SLA, such as TSLA1SVT. Click **Next**.

   Note: Instead of using an existing customer, you can create a new customer.

5. In the Select Offering panel, select the offering(s) to be part of the SLAs definitions, such as Workload Scheduler Job and Workload Scheduler Jobstream. Click **Next**.
6. In the Include Resources panel, click **Add**.
7. In the Select Resource List Type panel (Figure 12-20), define the type of resources to add to the SLA. The Dynamic Resource List is used to group resources and create filter. Static resources are used for particular resources that are to be added. You might want to use dynamic resource list, for constantly changing environments. Click **Next**.

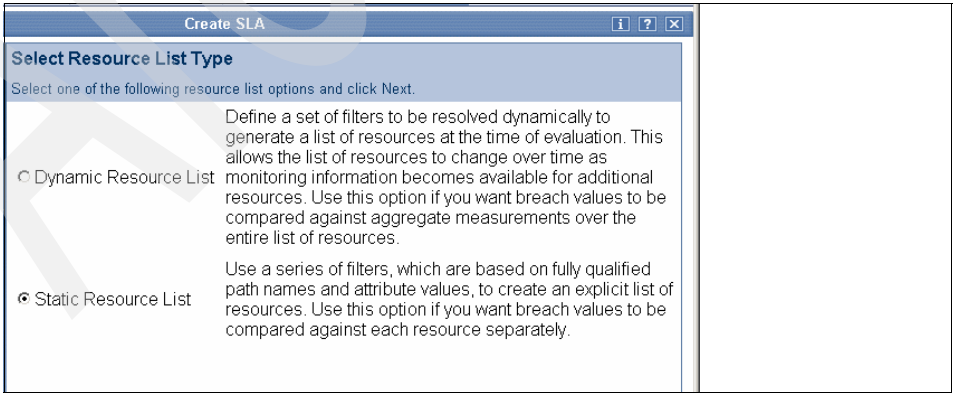

Figure 12-20 Selecting Resource List Types

8. For Dynamic Resource List, you need to create a filter. So, select **Preview current evaluation of filters**, then click **Create Filter**, and put an asterisk (*) for all resources. Name the resource. Use descriptive and meaningful names.
9. The Select SLA Start Data panel displays. The start date of the SLA is used to evaluate the previous monitoring data to verify the SLOs instantaneously. If there is no data, choose the default date (the current date). Optionally, select the time zone in which the SLA is evaluated. Click **Recalculate the First Evaluation** to refresh the first evaluation date, depending on the SLA start date.
10. The summary of the SLA creation displays. Click **Finish** to complete the SLA creation. You go back to the initial Manage SLAs screen and see the new SLA.

12.4.13 SLA reports

The SLA reports reveal where the SLAs have exceeded their threshold and the impact. Figure 12-21 shows an example report.

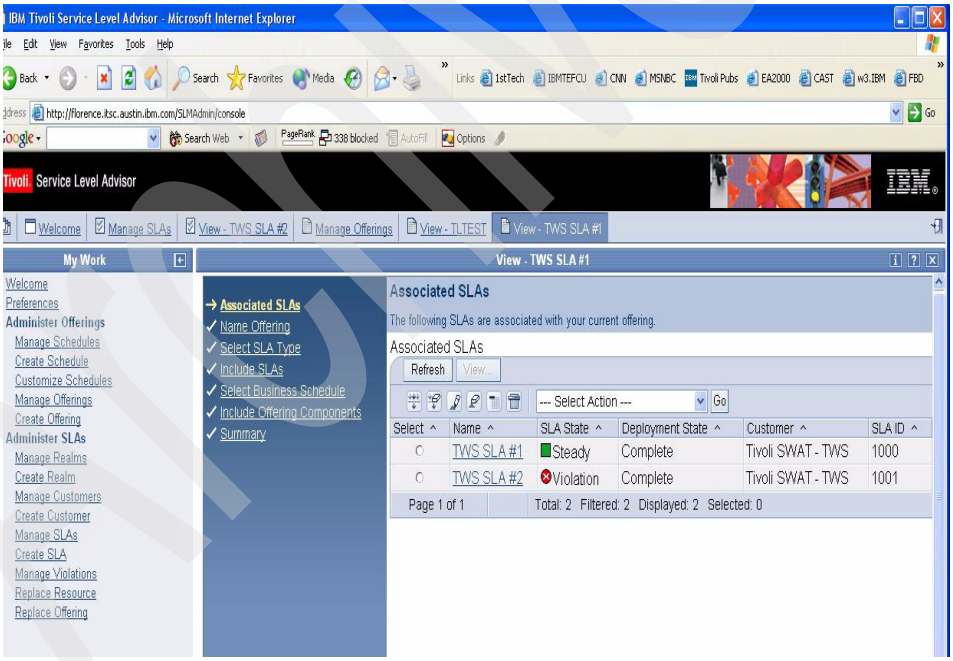

Figure 12-21 Example of a Service Level Agreement reports

13

Integrating Tivoli NetView Distributed

This chapter explains the integration of Tivoli Workload Scheduler on UNIX with Tivoli NetView on AIX. It gives a brief overview of Tivoli NetView and then describes how you can integrate Tivoli Workload Scheduler (TWS) with Tivoli NetView and use TWS and NetView integration to monitor your scheduling environment.

TWS runs on UNIX, Linux, and Windows. This chapter, however, discusses only the integration provided on the UNIX platform.

13.1 Tivoli NetView overview

Tivoli NetView (or NetView) is an IBM Tivoli availability tool that emphasizes monitoring network resources.

13.1.1 Network management basic terminology

Basic terminology that is commonly used in network management activities include:

Manager Manager is a software application that monitors and controls a network. A manager collects, processes, stores, and displays network data. Tivoli NetView is a manager.

Agent Agent is a software application that is responsible for reporting on and maintaining information related to one or more devices in the network. An agent gives network information to a manager.

Protocol Protocol is a set of rules that determine how individual network devices communicate with each other.

SNMP Simple Network Management Protocol (SNMP) enables managers to ask agents to retrieve and change information about network devices. NetView uses SNMP to manage TCP/IP networks. Because SNMP has low network overhead, it is an inexpensive way to gather network statistics. It is also ideal for real-time monitoring.

MIB Management information base (MIB), a collection of many pieces of information called MIB objects, that are located in the network device. The MIB objects can be accessed and sometimes changed by the agent at the manager's request. This is how the Tivoli NetView program manages network devices.

13.1.2 NetView as an SNMP manager

NetView uses SNMP to gather information about the resources that it manages and serves as an SNMP manager. The basic functions of NetView are:

- Discovering the devices in the network.
- Collecting and storing data about network conditions.
- Issuing and responding to notifications about network conditions.
- Issuing commands that cause actions at network nodes.

In addition to these basic functions, NetView has extended its management flexibility to provide both network management and systems management as it becomes more tightly integrated with Tivoli's other management applications such as Tivoli Workload Scheduler.

NetView keeps the dynamically updated topology and customization information in its database. It uses maps to maintain customizable views of the database. Each network object in the map is seen as a symbol. In other words, a symbol is a visual representation of a network object. NetView has two user interfaces. The NetView GUI on the server and the Java-based Web browser interface. The topology maps are the same in the NetView GUI on the server and the Java-based Web browser interface.

For more information about NetView, see *NetView for UNIX User's Guide for Beginners Version 7.1*, SC31-8891.

13.2 What the integration provides

The integration between NetView and TWS provides systems management of TWS as an application from the NetView console. TWS is treated like any other application with information being collected by SNMP agents on the TWS workstations and forwarded to NetView. NetView uses this information to update submaps that are specific to TWS.

The integration provides SNMP managers and agents for TWS as well as the TWS MIB. TWS uses the NetView databases to store workstation and process information, which is used to generate maps. It can send and receive SNMP events and can signal events using SNMP traps. The TWS MIB can be viewed from the NetView console. The integration includes the following:

- A number of TWS-specific submaps have been defined to illustrate where TWS is running in the network and its status. This shows the TWS network topology. There are two sets of submaps:
  - The submaps showing the status of the TWS managed jobs or job streams in a TWS network. The maps reflect job or job stream status changes.
  - The submaps showing the status of the TWS processes on a given workstation.
- The provided TWS management events include:
  - Notification of stopped or failed TWS processes
  - Notification of abended or failed jobs
  - Notification of a broken link between the TWS workstations

- Some TWS tasks can be launched from the NetView console. The menu actions include:
  - Start TWS processes.
  - Stop TWS processes.
  - Run the console manager (**conman**) on any workstation in the TWS network.

The following sections explain the components of the integration.

13.3 How the integration works

The integration is accomplished by the TWS NetView software, which is delivered and installed as part of TWS on UNIX.

Important: The TWS and NetView integration is only provided for NetView running on AIX. NetView/NT is not supported for integration. Also, at least one TWS workstation running on UNIX is required for the integration.

The integration consists of manager and agent software. The manager runs on the NetView server nodes, and the agent runs on the managed nodes. The manager (mdemon) polls its agents (magent) periodically to obtain information about scheduler processing. If the information returned during a poll is different from that of the preceding poll, the color of a corresponding symbol is changed to indicate a state change, for example, from green (normal) to red (critical) or yellow (marginal).

The agents also generate SNMP traps to inform the manager of asynchronous events, such as job abends, stuck jobs, and restarted scheduler processes. Although polling and traps are functionally independent, the information that accompanies a trap can be correlated with symbol state changes. If, for example, a scheduled job abends, the symbol for the workstation changes color, and a job abend trap is logged in the NetView event log. By scanning the log, you can quickly isolate the problem and take appropriate action.

The muser process runs commands issued by a NetView user, and updates the user's map. An muser is started for each NetView user whose map has the TWS and NetView integration application activated.

13.4 Integration architecture and our environment

Figure 13-1 illustrates the environment that we used for the NetView and TWS integration.

Note: This environment is a different from the environment that we have used for the other integration scenarios in this book.

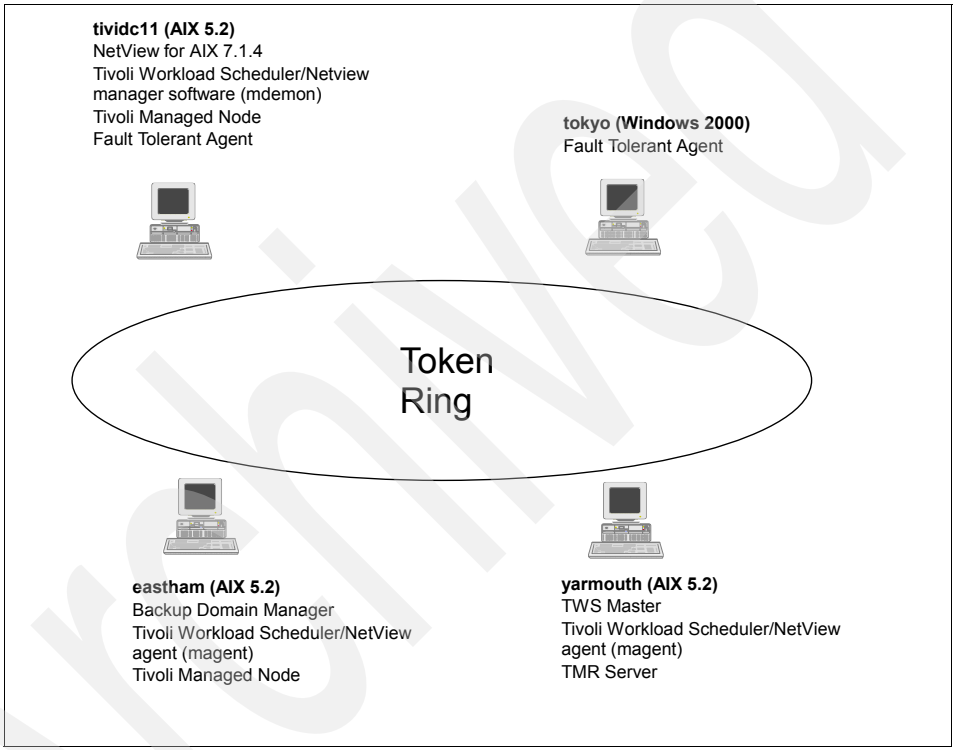

Figure 13-1 Our distributed TWS environment

Note that the TWS and NetView integration manager software (**mdemon**) should be installed on the NetView for AIX machine, which is also called the *management node* in the integration terminology.

Note: In NetView 4.1 and later, the management node functions can be distributed across a server and one or more clients. In that case, you have to install TWS and **magent** on the client NetView machines as well. We do not use NetView clients in this scenario. However, if you need more information about how to install and configure magent on NetView clients, refer to *IBM Tivoli Workload Scheduler Reference Guide Version 8.2*.

In the same way, the TWS and NetView integration agent (**magent**) software should be installed on *at least* one TWS UNIX workstation. This is called a *managed node* (not to be confused with the Tivoli Managed Node) in the integration terminology. The integration agent should be installed on either the master or the backup domain manager — that is, a fault tolerant agent with `fullstatus` and `resolvedep` enabled in its definition.

Important: You cannot install **magent** on TWS workstations other the UNIX. However, you can manage other workstations with NetView in the same way as UNIX workstations, provided that at least one **magent** is installed in the TWS network.

A group of nodes that are configured as a TWS network and whose job scheduling status is managed from a NetView management node is called a *Managed Workload Scheduler Network*. There must be at least one managed node in a Managed Workload Scheduler Network.

In our environment, we installed one management node (tividc11) and two managed nodes (yarmouth, which is also the master, and eastham, the backup domain manager).

Installation tips:

- It is a good policy to configure both TWS master and TWS backup domain managers as managed nodes. If you are using only one managed node, *which is not the Tivoli Workload Scheduler master*, you have to set `OPTIONS=MASTER` in the BmEvents configuration file on the managed node. For more explanation of the BmEvents configuration file refer to *IBM Tivoli Workload Scheduler Reference Guide Version 8.2*, SC32-1274.
- Management nodes (server and client) need not be members of Managed Workload Scheduler Networks; however, making them so simplifies your configuration.
- The best location for a management node is a TWS FTA workstation. Choosing a TWS master for this purpose is generally not recommended, especially for busy TWS networks, due to the additional overhead of the NetView application (although it does have one benefit of minimizing the TWS and NetView integration manager-agent polling traffic).

13.5 Installing and customizing the integration software

The integration software is installed as part of the TWS installation on UNIX. This section shows how to customize the software to enable the integration. Installation and customization steps are also covered in detail in *IBM Tivoli Workload Scheduler Reference Guide Version 8.2*, SC32-1274.

Important: Before beginning the following steps for integrating TWS and NetView, make certain that TWS for UNIX is properly installed on your environment (Master Domain Manager, Backup Master, Domain Manager, Backup Domain Manager, and Fault Tolerant Agents).

All installations use the *TWSHome*/OV/customize script. You can view this script if you want to see what it actually does. You can also run this script with the `–noinst` option to review the changes.

If you have issues with the installation, you can back out using the *TWSHome*/OV/decustomize script.

You should run the *TWSHome*/OV/customize script on both management nodes, including the NetView server — and clients, if any — and managed nodes (TWS/UNIX workstations).

13.5.1 Installing mdemon on the NetView Server

To install mdemon on the NetView server-tividc11:

1. Run **conman shutdown** to stop all TWS processes.
2. Log in as root.
3. Run the following:

   ```
   /bin/sh /tivoli/TWS/D/tws-d/OV/customize -uname tws-d
   ```

   In this command, `/tivoli/TWS/D/tws-d/` is the TWS home directory and `tws-d` is the maestro user ID.

Important notes:

1. If you are installing on a NetView client instead of a server, you use the `-client` keyword as follows:

   ```
   /bin/sh /tivoli/TWS/D/tws-d/OV/customize -client hostname
   ```

   Where `hostname` is the NetView Server.

   By default, the customize script assumes that you are installing on a server.
2. Do not forget to use the `uname` parameter if your maestro user ID is different from the default user ID, which is maestro.

The output of the script is shown in Example 13-1.

Example 13-1 Output of the script

```
280505  modifying files for tws-d in /tivoli/TWS/D/tws-d
AWS22280506  BmEvents.conf.orig already exists
AWS22280506  StartUp.OV.orig already exists
AWS22280506  Maestro.s.c.orig already exists
AWS22280506  decustomize.orig already exists
AWS22280506  H.conf.D.orig already exists
AWS22280506  H.conf.T.orig already exists
AWS22280506  H.timeout.T.orig already exists
AWS22280506  H.traps.T.orig already exists
AWS22280506  Mae.aix.app.orig already exists
AWS22280506  Mae.hp.app.orig already exists
AWS22280506  Mae.mgmt.lrf.orig already exists
AWS22280506  Mae.actions.orig already exists
AWS22280506  mae.traps.hp.orig already exists
AWS22280506  maestroEvents.orig already exists
AWS22280507  Copying the appropriate application registration file.
AWS22280508  Copying sample filters
AWS22280509  Copying the field registration
AWS22280510  Compiling the field registration
```

```
AWS22280511  Making the help directories
AWS22280512  Copying the dialog help texts
AWS22280513  Copying the task help texts
AWS22280514  Copying the function help texts
AWS22280515  Copying the local registration
ovaddobj - Static Registration Utility
Successful Completion
AWS22280517  Adding OVW path to TWS .profile
AWS22280519  tws-d .profile has replaced the old file.
The previous version is in /tivoli/TWS/D/tws-d/.profile.old.
AWS22280520  Adding traps to trapd.conf
Trap uTtrapReset has been added.
Trap uTtrapProcessReset has been added.
Trap uTtrapProcessGone has been added.
Trap uTtrapProcessAbend has been added.
Trap uTtrapXagentConnLost has been added.
Trap uTtrapJobAbend has been added.
Trap uTtrapJobFailed has been added.
Trap uTtrapJobLaunch has been added.
Trap uTtrapJobDone has been added.
Trap uTtrapJobUntil has been added.
Trap uTtrapJobCancel has been added.
Trap uTtrapJobCant has been added.
Trap uTtrapSchedAbend has been added.
Trap uTtrapSchedStuck has been added.
Trap uTtrapSchedStart has been added.
Trap uTtrapSchedDone has been added.
Trap uTtrapSchedUntil has been added.
Trap uTtrapGlobalPrompt has been added.
Trap uTtrapSchedPrompt has been added.
Trap uTtrapJobPrompt has been added.
Trap uTtrapJobRecovPrompt has been added.
Trap uTtrapLinkDropped has been added.
Trap uTtrapLinkBroken has been added.
Trap uTtrapDmMgrSwitch has been added.
AWS22280524  IMPORTANT
AWS22280525  Be sure to add an appropriate entry to each TWS cpu's rhost
This could be '"<manager> <user>"' if a user other than tws will be
running the management station.  Or '"<manager>"' if the TWS user
AWS22280528  will be managing from the management station.
AWS22280529  see documentation on remsh and .rhosts for further information.
AWS22280530  will install the agent reporting to tividc11
AWS22280531  changing ownership and permissions on agent
AWS22280532  Processing the agent files
AWS22280534  /etc/snmpd.peers has been replaced the old file is
/etc/snmpd.peers.old
AWS22280536  /etc/snmpd.conf has been replaced the old file is
/etc/snmpd.conf.old
AWS22280537  Recompile the defs for this machine
```

```
AWS22280539  /etc/mib.defs has been replaced the old file is /etc/mib.defs.old
AWS22280541  Refreshing snmpd
0513-095 The request for subsystem refresh was completed successfully.
AWS22280542  Setting up the agent files
AWS22280543  Copying the configuration files.
AWS22280545  /tivoli/TWS/D/tws-d/StartUp has been replaced
AWS22280523  changing ownership and permissions on programs
AWS22280546  You should now restart NetView/Openview and TWS
```

3. Execute the startup script to restart the TWS netman process using:

   ```
   /tivoli/TWS/D/tws-d/StartUp
   ```

4. The **mdemon** is started with NetView as part of the **ovstart** sequence, so start the NetView deamon as follows:

   ```
   /usr/OV/bin/ovstop
   /usr/OV/bin/ovstart
   ```

   As an alternative, you can use the smitty panels by selecting **Smitty** → **Commands Applications and Services** → **Tivoli NetView** → **Control** → **Stop all running daemons**, and then **Restart all stopped daemons**.

Tip: The run options of **mdemon** are included in the /usr/OV/lrf/Mae.mgmt.lrf file.

13.5.2 Installing magent on managed nodes

To install **magent** on eastham:

1. Run **conman shutdown** to stop all TWS processes.
2. Log in as root.
3. Run the following:

   ```
   /bin/sh /tivoli/TWS/D/tws-d/OV/customize -uname tws-d -manager tividc11
   ```

   In this command, `/tivoli/TWS/D/tws-d/` is the TWS home directory, `tws-d` is the user ID, and `tividc11` is the NetView server.

4. Execute the startup script to restart the TWS netman process and **magent** using the following:

   ```
   /tivoli/TWS/D/tws-d/StartUp
   ```

You can repeat the same steps on yarmouth.

13.5.3 Customizing the integration software

After running the scripts on the management and managed nodes, the following additional steps are necessary:

- Configuring user access.
- Defining the submaps in NetView.
- Loading the TWS MIB.
- Restarting TWS on all systems.

Configuring user access

There are two elements to the configuration:

- Configuring the remote access for the user.
- Defining the user to TWS.

When you define the user that manages TWS from NetView, the user must have access to run commands from the NetView server. You define the user by adding the name of the NetView server to the users $HOME/.rhosts file on each managed node. In our environment, the user was root. So, we edited the /.rhosts file to add the following entry:

```
tividc11.itsc.austion.ibm.com root
```

The second step is to add this user to the TWS security file. Because we used the root user, we do not need to perform this step. (Root user and the user that you perform the TWS installation, also known as the maestro user, are included in the security file by default.) However, if you are using any other user, you have to modify the security file using the **makesec** command. For more information about how to add a user to the security file, refer to the *Tivoli Workload Scheduler Planning and Installation Guide Version 8.2*, SC32-1273.

Defining TWS maps in NetView

To define the TWS map in NetView, on the NetView server:

1. Run **netview** to launch the NetView console.
2. Select **File** → **New Map**.
3. Give a name to the new map, such as TWS in the Name field.
4. Select **Maestro-Unison Software Inc. (c)** in the Configurable Applications field, as shown in Figure 13-2 on page 626.

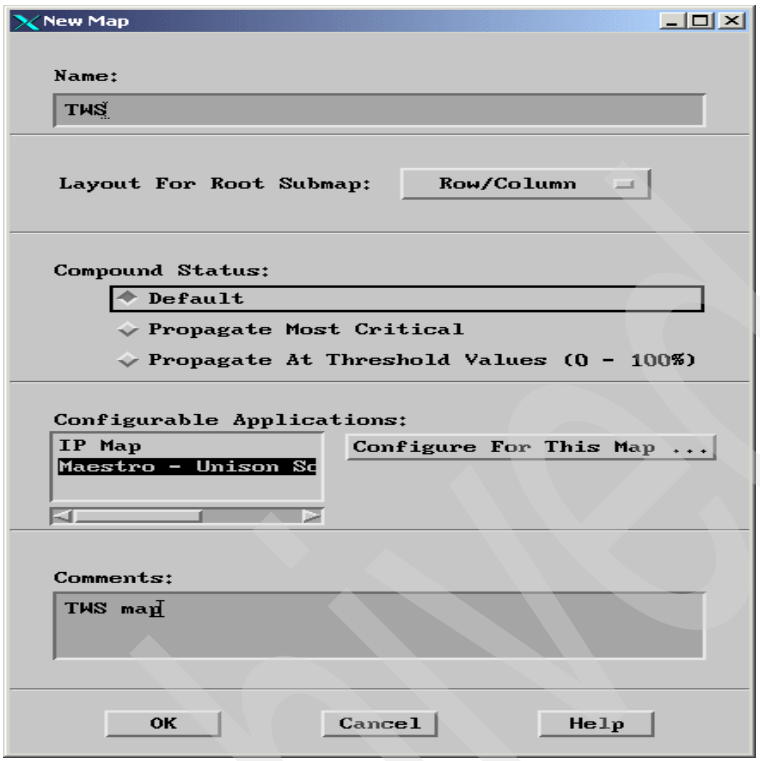

Figure 13-2 Selecting Maestro - Unison Software Inc. (c)

5. Click **Configure For This Map...** to specify options for the TWS map.
6. When the Configuration dialog opens, set the Enable Maestro for this map option to **True**. Leave all other options, the commands run under the TWS menu items, at their default values (Figure 13-3 on page 627).

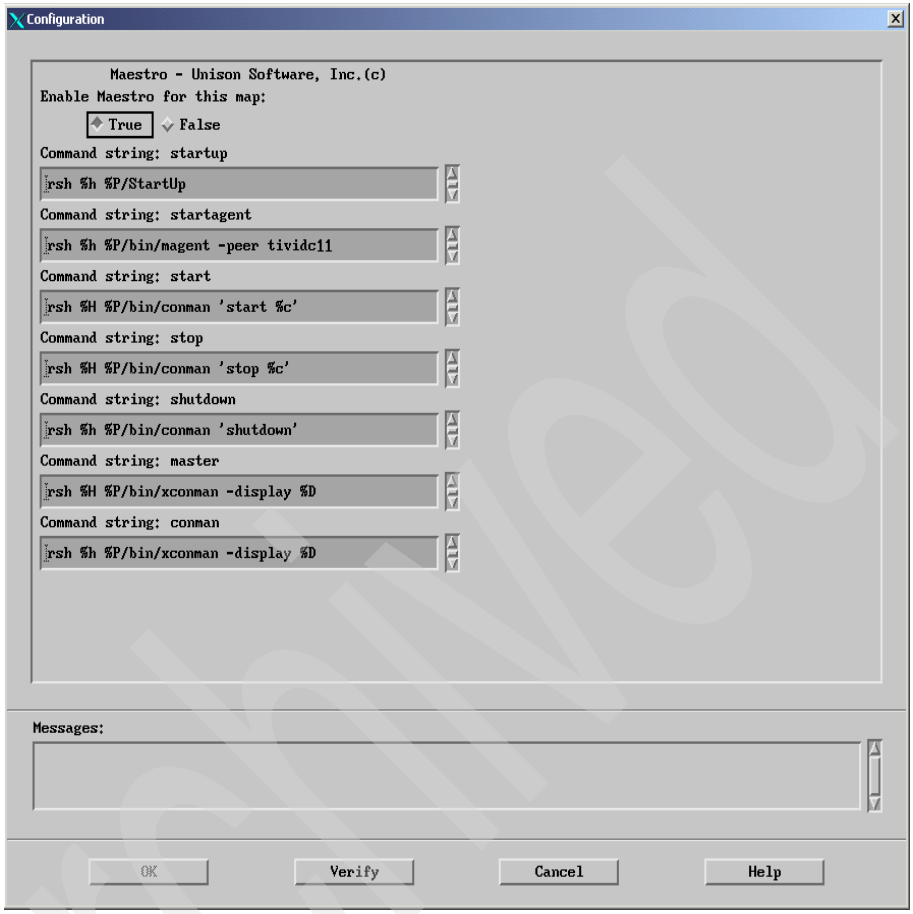

Figure 13-3 Configuring the TWS map

7. To complete the addition of the new map, click **OK**. The new map, TWS, is shown in Figure 13-4 on page 628.

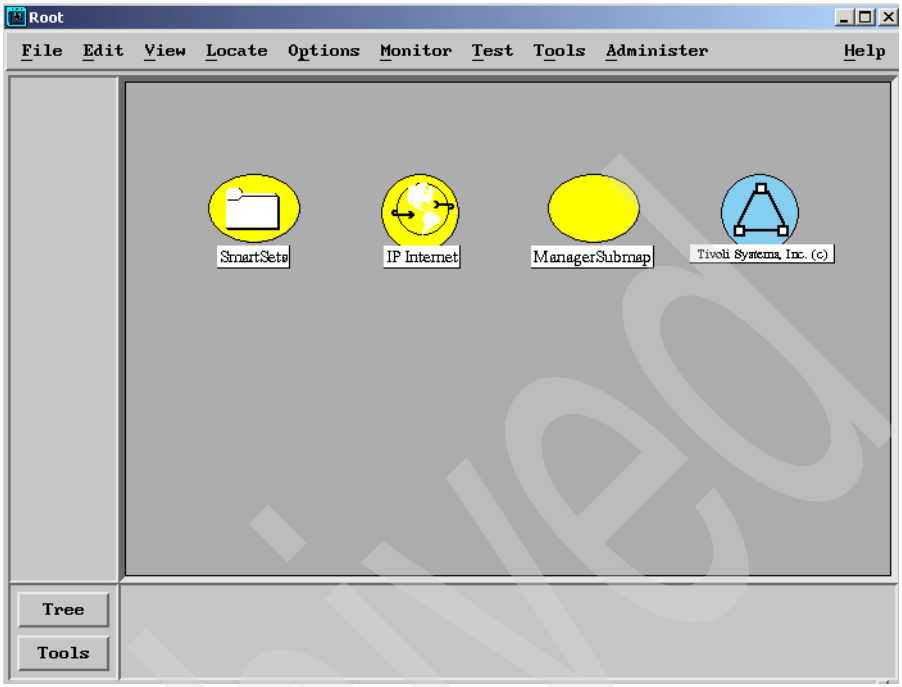

Figure 13-4 NetView Root map with TWS icon added

Note: The TWS map (Tivoli Systems,Inc. (c) labeled icon) is shown first in blue because NetView has not yet received any information from the TWS application. After the polling cycle, it changes to green.

8. Add the TWS MIB by selecting **Options** → **Load/Unload MIBs** → **SNMP** from the NetView GUI.

   Although this is not a mandatory step, it is highly recommended. If you add the MIB, you are able to use the MIB browser application in NetView.

9. Because the TWS MIB is not loaded by default, select **Load** as shown in Figure 13-5 on page 629.

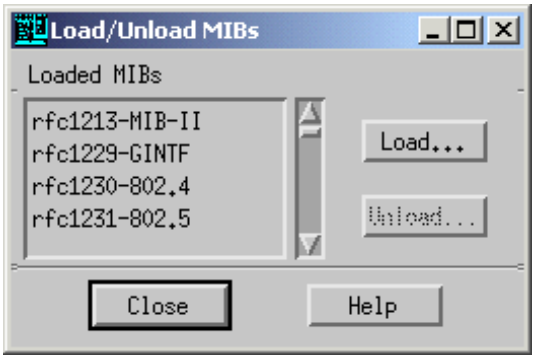

Figure 13-5 Loaded MIBs by default

10. When the Load MIB From File dialog opens, find and select the Maestro MIB (/usr/OV/snmp_mibs/Maestro.mib) from the list and select **OK** (Figure 13-6).

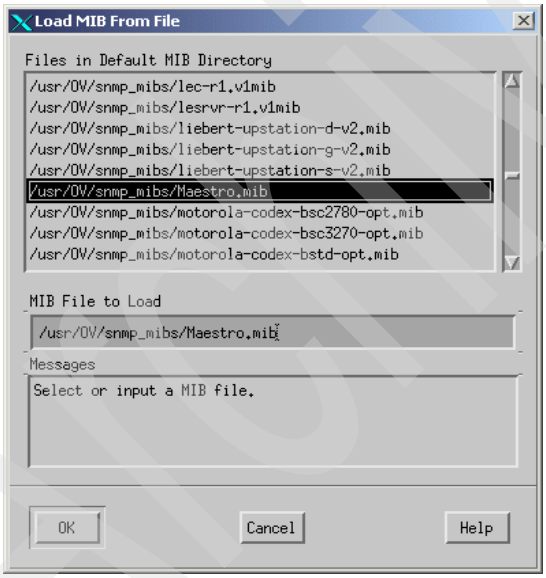

Figure 13-6 Loading Maestro MIB file

11. Click **Close** to close the Load/Unload MIBs dialog box.

Restarting TWS on all systems

When you have completed the changes, you can restart TWS by logging on to the TWS master, yarmouth in this case, and issuing the **`conman start @`** command. The `@` symbol is the TWS wildcard for all nodes.

13.6 Operating TWS and NetView

After installing and customizing TWS and NetView, you can use this integration to manage your scheduling environment.

13.6.1 Discovery of the TWS Network

The first phase of managing the TWS network is discovering what nodes are running TWS and adding submaps and icons to the NetView display. With a single TWS network and the NetView server node as a TWS node, as we have, this is done automatically. With multiple TWS networks, or where the NetView server is not in the Managed Workload Scheduler Network, there are additional setup steps as outlined in *IBM Tivoli Workload Scheduler Reference Guide Version 8.2*, SC32-1274.

The discovery adds submaps and icons for:

- The TWS network. These are contained under a new root map icon.
- The TWS processes that are running on each node. These are contained under the node icons along with the interfaces and other node components.

13.6.2 Creating TWS maps

To create the TWS maps in the NetView Root map:

1. Launch the NetView console, if it is not opened already.

   Figure 13-7 on page 631 shows the Root map. After the installation of the integration software, this map has an icon that is labelled Tivoli Systems,Inc. (c). This application symbol represents all discovered TWS networks. The color of the icon represents the aggregate status of all the TWS networks.

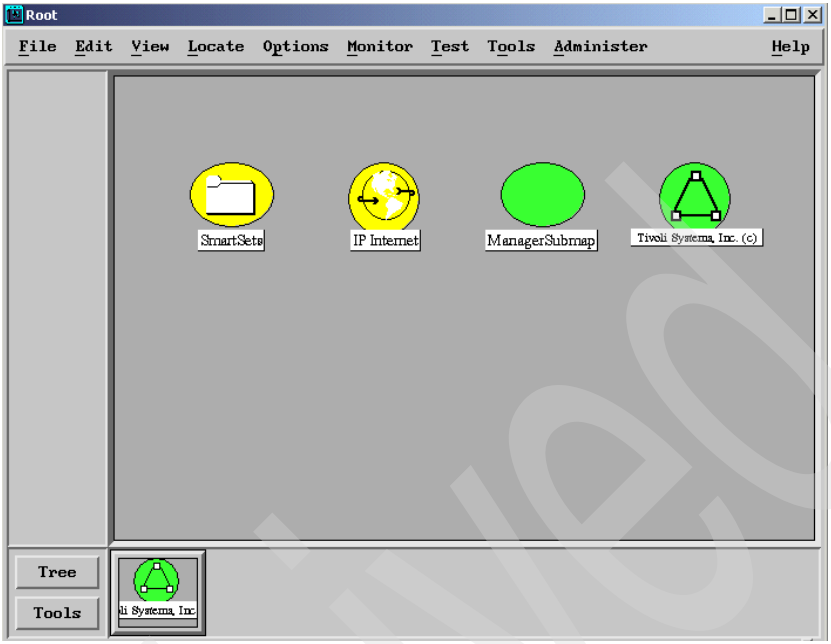

Figure 13-7 NetView Root map

2. Choose the TWS symbol by double-clicking it.

   There is only one TWS network in this application submap, labelled YARMOUTH-D:Maestro (as shown in Figure 13-8 on page 632). If there were multiple TWS networks, there would be multiple icons, each labelled with the specific TWS network name. The icon color represents the status of all workstations and links that comprise the TWS network.

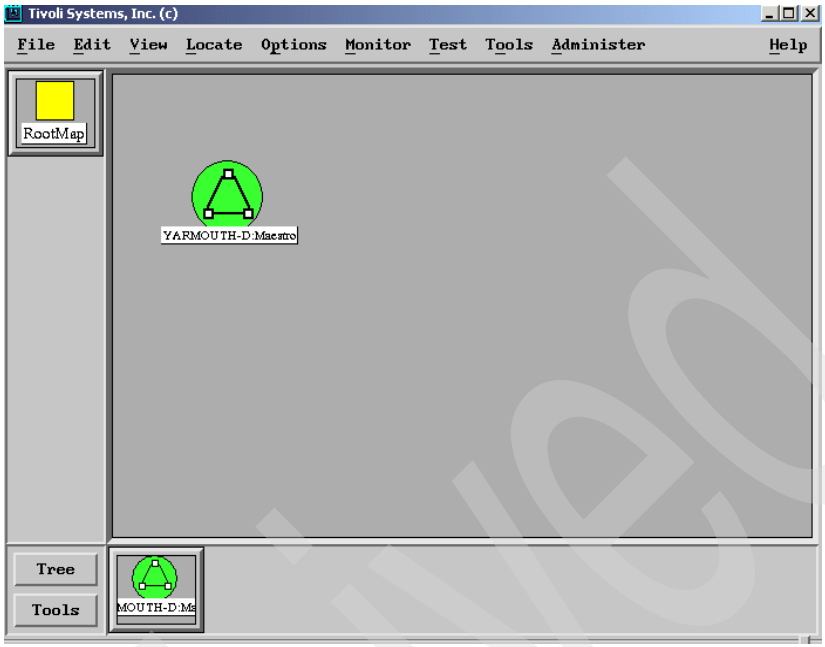

Figure 13-8 TWS main map

3. Open the TWS network icon to show all workstations and links in the network, as shown in Figure 13-9 on page 633.

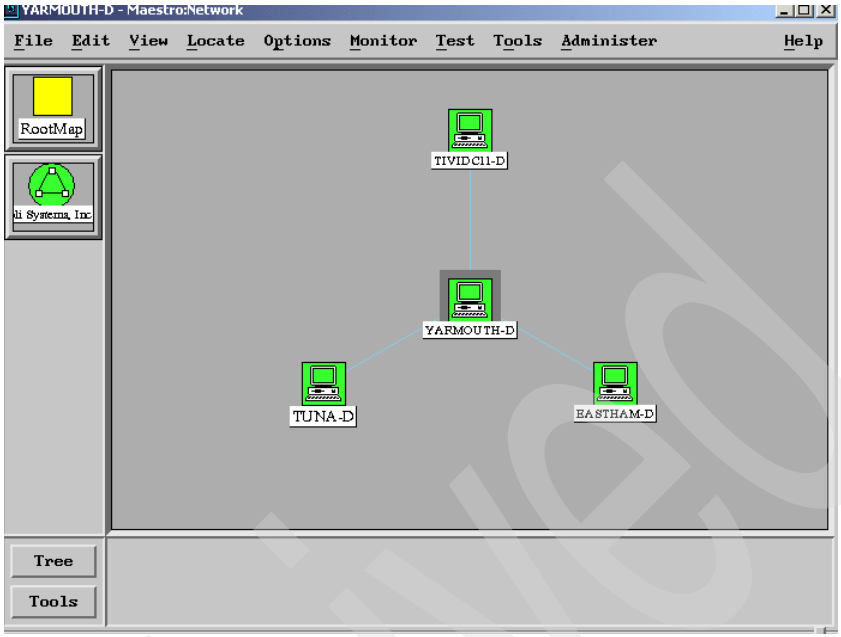

Figure 13-9 TWS map showing all workstations

Figure 13-9 is equivalent to the diagram of our TWS network (Figure 13-1 on page 619). There are three TWS nodes, with yarmouth being the master node. If there were more nodes, these nodes would be arranged in a star pattern around the master node.

Each node symbol represents the job scheduling on that workstation. The color represents the status of the job scheduling. If a trap is received indicating a change in status of a job scheduling component, the icon color changes. The links represent the TWS workstation links with the color representing the status of the workstation link.

In addition to monitoring TWS status, you can run several tasks from this window.

13.6.3 Menu actions

To use the TWS and NetView integration menu actions, select **Workload Scheduler** from the Tools menu, as shown in Figure 13-10.

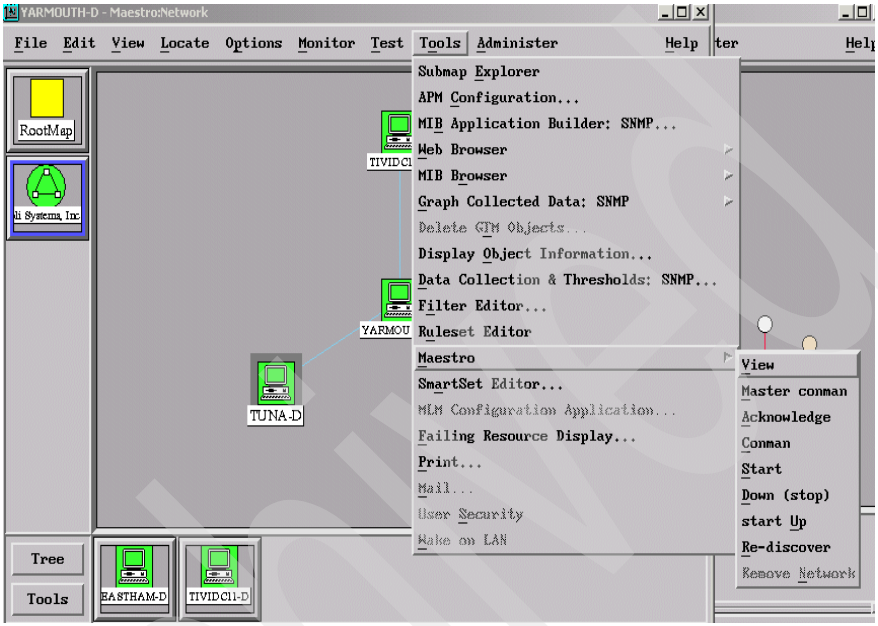

Figure 13-10 Launching Maestro menu from Tools

These actions are also available from the object context menu by clicking a symbol with mouse button three (or right mouse button, if you are using a two-button mouse).

The following sections describe each of the menu actions.

View

Selecting **View** opens a child submap for a TWS and NetView integration symbol. Choosing View after selecting a workstation symbol on the TWS network submap opens the monitored processes submap. Choosing View after selecting a workstation symbol on the IP node submap returns to the TWS network submap.

Master conman

Selecting **Master conman** runs the **conman** program on the TWS master. Running the program on the master permits you to execute all conman commands (except **shutdown**) for any workstation in the TWS network.

Acknowledge

Selecting this option acknowledges the status of the selected TWS and NetView integration symbols. When acknowledged, the status of a symbol returns to normal. It is not necessary to acknowledge critical or marginal status for a monitored process symbol. The process returns to normal when it is running again. Critical or marginal status of a TWS workstation symbol should be acknowledged either before or after you have taken some action to remedy the problem. The process will not return to normal otherwise.

Conman

Selecting **Conman** runs the **conman** program on the selected TWS workstations. Running the program on a workstation other than the master permits you to execute all conman commands on that workstation only.

Start

Selecting **Start** issues a **conman start** command for the selected workstations. By default, the command for this action is:

```
remsh %H %P/bin/conman 'start %c'
```

Down (stop)

Selecting **Down (stop)** issues a **conman stop** command for the selected workstations. By default, the command for this action is:

```
remsh %H %P/bin/conman 'stop %c'
```

Start up

Selecting **Start up** executes the TWS StartUp script on the selected workstations. By default, the command for this action is:

```
remsh %h %P/StartUp
```

Rediscover

Selecting **Rediscover** locates new agents and new TWS objects and updates all TWS and NetView integration submaps.

Note: You need to run the rediscover function each time you change the TWS workstation configuration.

For example, to stop the **batchman** process on tividc11 from the NetView console:

1. Select **tividc11** and then right-click the icon. You see the Maestro menu among the side menu options.
2. Select **Maestro** and then select **Down (stop)**, as shown in Figure 13-11. This stops the **batchman** process on tividc11.

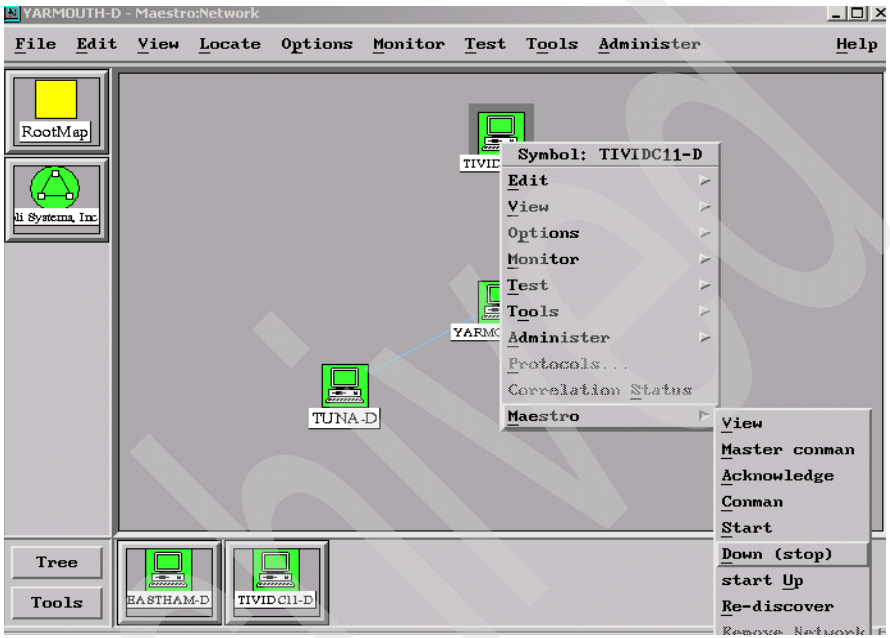

Figure 13-11 Stopping conman

3. You can verify that the **batchman** process has stopped by checking the status on tividc11, as shown in Example 13-2.

Example 13-2 Check status on tividc11

```
tividc11:/tivoli/TWS/D/tws-d>conman status
TWS for UNIX (AIX)/CONMAN 8.2 (1.36.2.16)
Licensed Materials  Property of IBM
5698-WKB
(C) Copyright IBM Corp 1998,2001
US Government User Restricted Rights
Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Installed for group 'TWS-Distributed'.
Locale LANG set to "En_US"
Schedule (Exp) 02/08/05 (#10) on TIVIDC11-D. Batchman down. Limit: 10, Fence: 0, Audit Level: 1
TWS for UNIX (AIX)/CONMAN 8.2 (1.36.2.16)
Licensed Materials  Property of IBM
5698-WKB
```

There are no additional submaps to TWS maps. If you want to see the TWS process information, you need to browse the node symbol under the main IP submap, which we are going to explain next.

13.6.4 Discovering Maestro process information

To find the Maestro process information, drill down to the submap representing the node. In NetView, you can use the Locate function to find a node on the IP map. To discover the TWS processes in tividc11:

1. In the NetView console select **Locate** → **Objects** → **By Symbol Label** as shown in Figure 13-12.

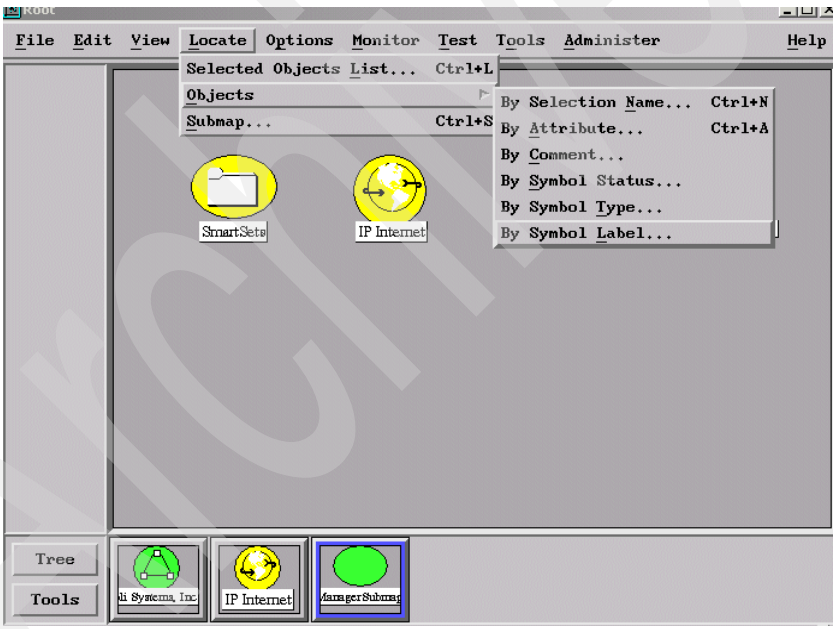

Figure 13-12 Locate function

2. Type `tividc11` in the Symbol Label field and then select **Apply** for NetView to locate the node for you.
3. Select the node and click **Open**, as shown in Figure 13-13 on page 638.

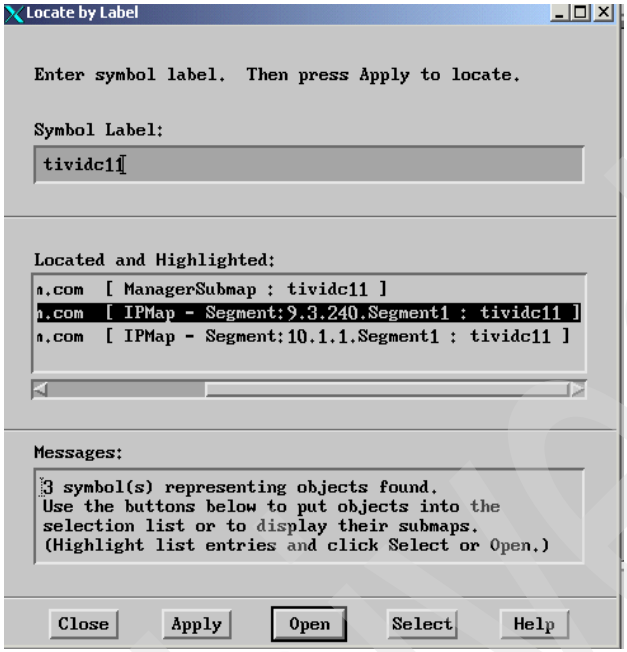

Figure 13-13 Locating tividc11 node

The map that contains the tividc11 node opens, as shown in Figure 13-14 on page 639.

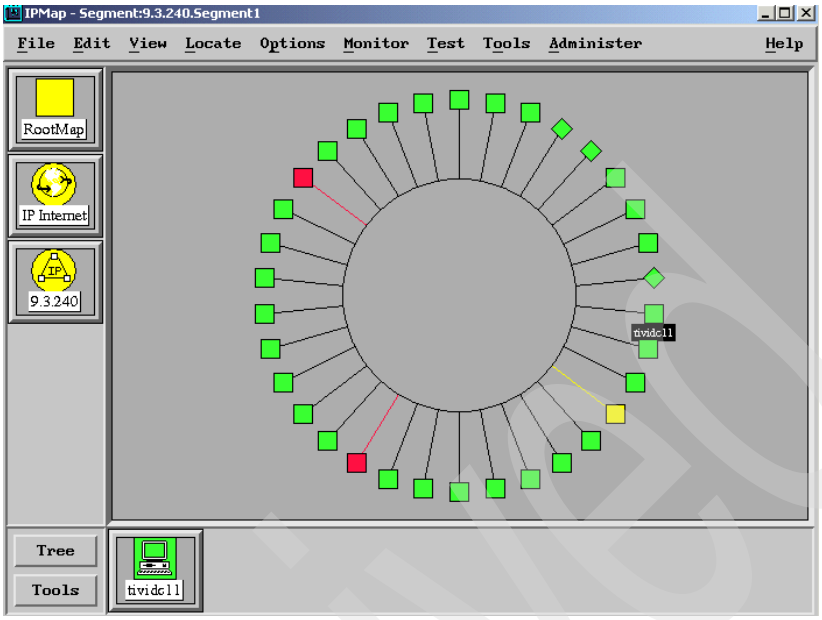

Figure 13-14 tividc11 on the map

4. Select **tividc11** to open up the processes on this node, as shown in Figure 13-15 on page 640.

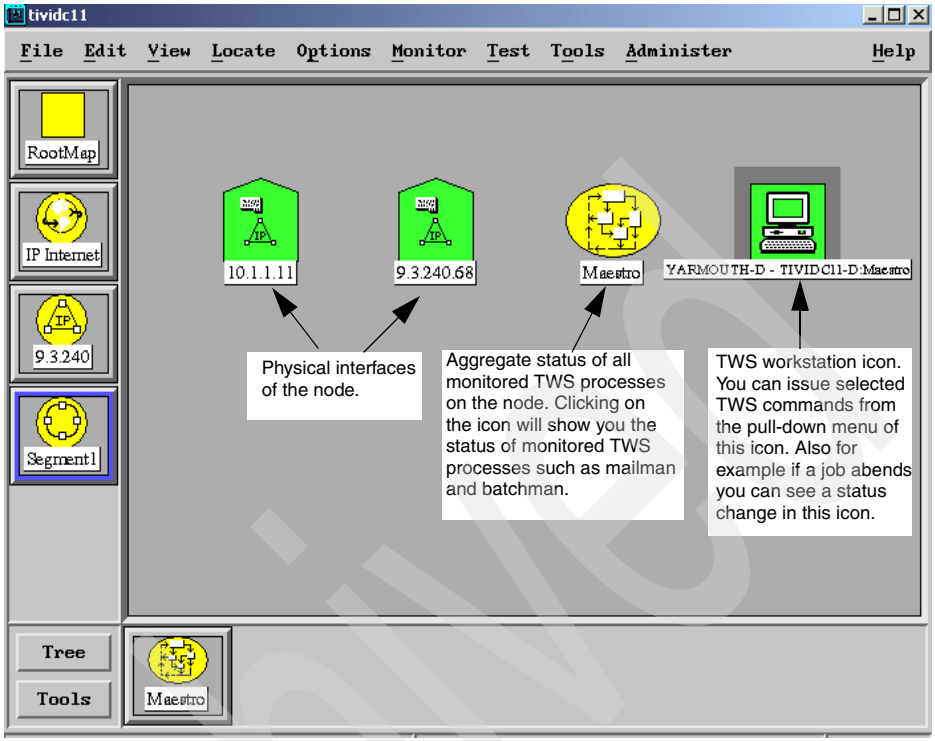

Figure 13-15 tividc11 map

This submap shows the following icons:

- The TWS workstation (YARMOUTH-D - TIVIDC11-D:Maestro icon). The label is in the format TWS master:FTA name:Maestro.
- The TWS software (Maestro icon). It represents all TWS processes running on the node. The color indicates the aggregate status of all the monitored processes on a TWS workstation.
- Two different physical interfaces for this node.

Note that the Maestro icon is yellow, which means that there is a problem. Opening the TWS software icon shows all TWS processes on the node, as shown in Figure 13-16 on page 641.

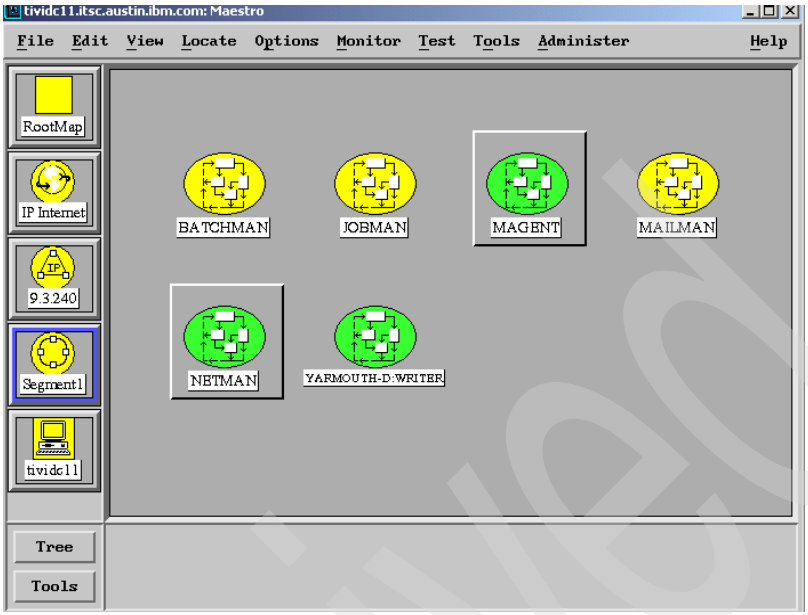

Figure 13-16 TWS processes

Figure 13-13 on page 638 shows that BATCHMAN, JOBMAN, and MAILMAN icons are yellow, which means that they are stopped. Remember that we have previously issued a **`stop conman`** command from the NetView console.

Tip: Yellow means a marginal problem or warning. Red means a severe problem. For example, if we kill the **`mailman`** process manually using the **`kill`** command (simulating an abend), the **`mailman`** process icon would be red.

You can restart the processes by:

- Running the **`conman start`** command on the node (or from the Maestro master).
- Using the menus that were added with the NetView integration software installation.

Many of these items are grayed-out when selecting this menu for the process icon because many options are not relevant to a Maestro process. One of the grayed-out items is the **start** command, which we need to use.

5. To access the **`start`** command, you need to go up a level. The easiest way to do this action is to click the gray area on the map (not on icons) with the third mouse button or right mouse button on a 2-button mouse and select **Show Parent** as shown in Figure 13-17 on page 642.

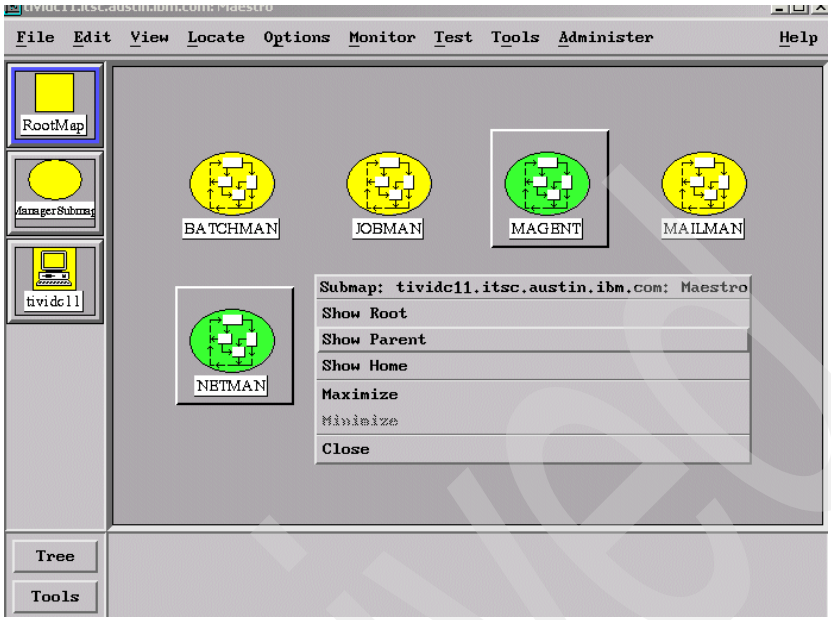

Figure 13-17 Show Parent dialog

This opens the parent map for you.

6. Select the **YARMOUTH-D-TIVIDC11-D:Maestro icon** with the third mouse button or right mouse button on a 2-button mouse and select **Start**, as shown in Figure 13-18 on page 643.

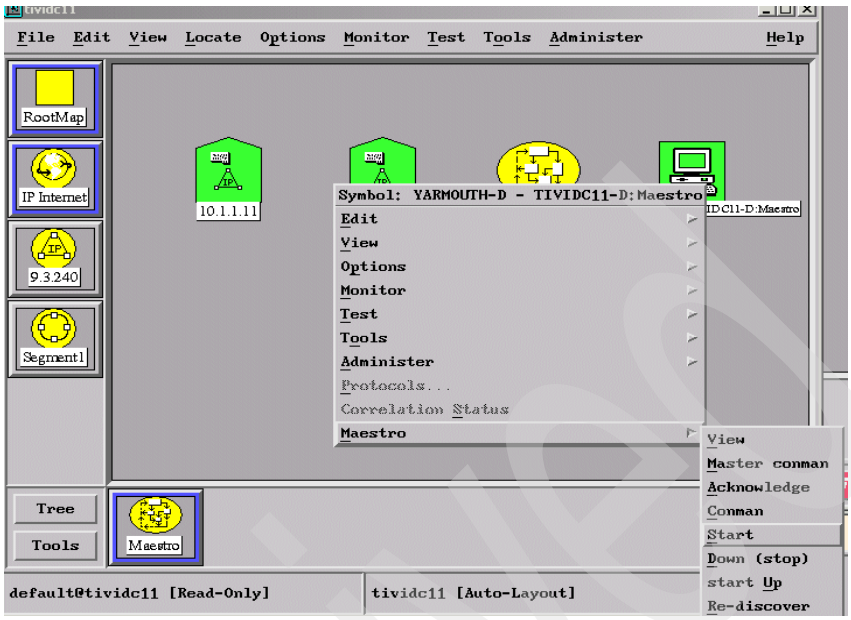

Figure 13-18 Issuing Start command

7. After the polling cycle (which is 60 seconds by default) all TWS processes should be started on tividc11. When you select the **Maestro** icon again, you should see all processes started, and all colors should turn green, as shown in Figure 13-19 on page 644.

Tip: The rate is defined in the /usr/OV/lrf/Mae.mgmt.lrf file on the management node. To change the rate:

1. Edit the file to add the **-timeout** option to the **mdemon** command line.
2. Delete the old registration by running the **ovdelobj** command.
3. Register the manager by running the **ovaddobj** command and supplying the name of the lrf file.

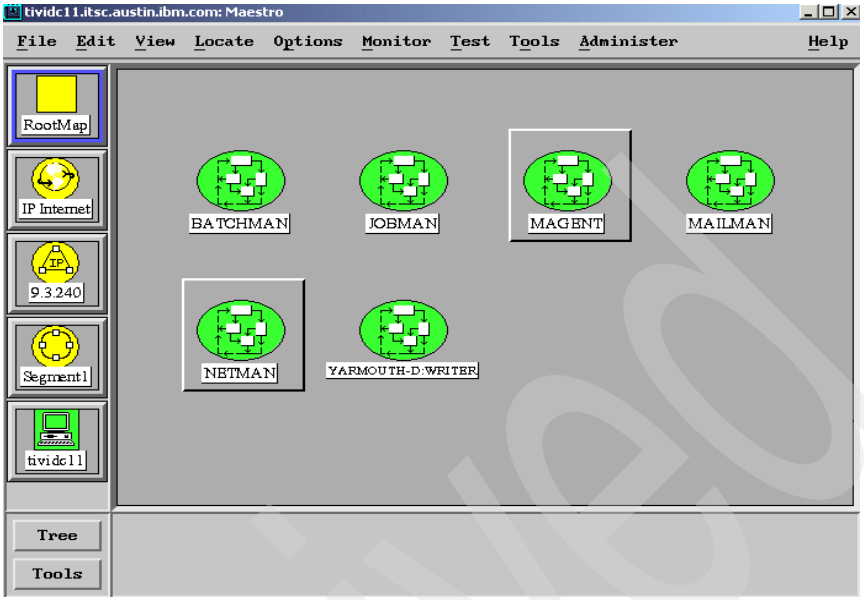

Figure 13-19 All processes up

13.6.5 Monitoring job schedules from NetView

In addition to giving the status information of TWS processes, TWS and NetView can also inform you about the abended or failed jobs and broken links between the TWS workstations.

In order to test this function, we created a scenario where two jobs (ABENDJOB and ABENDJOB2) were abended on tividc11. These jobs can be seen from the Job Scheduling Console in abend states (Figure 13-20 on page 645).

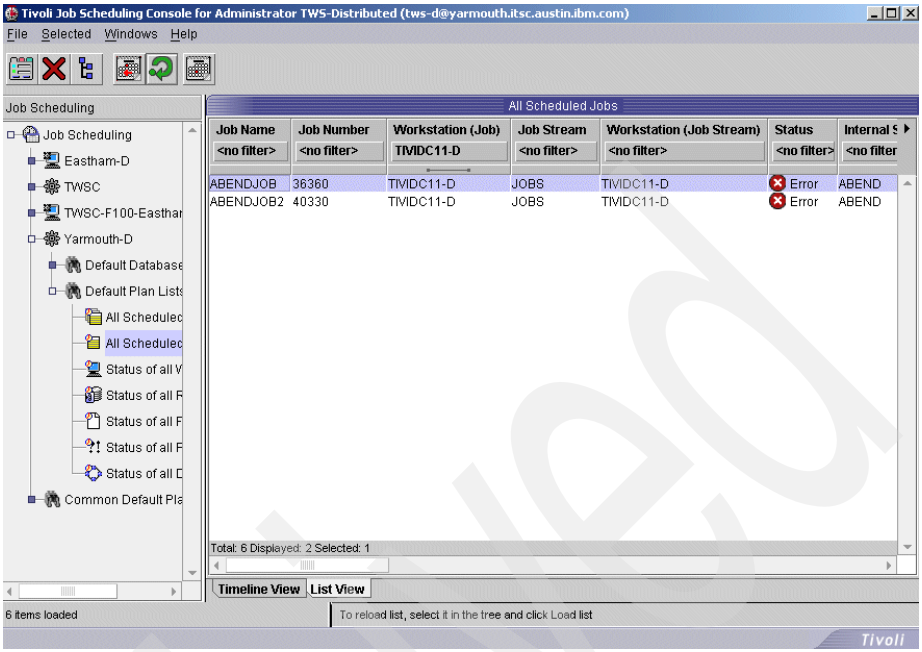

Figure 13-20 Job abended on tividc11

When we checked the NetView console, we saw that the tividc11 icon in the TWS map turned red (Figure 13-21 on page 646) and this also has turned the Tivoli Systems Inc icon yellow (shown at the left side).

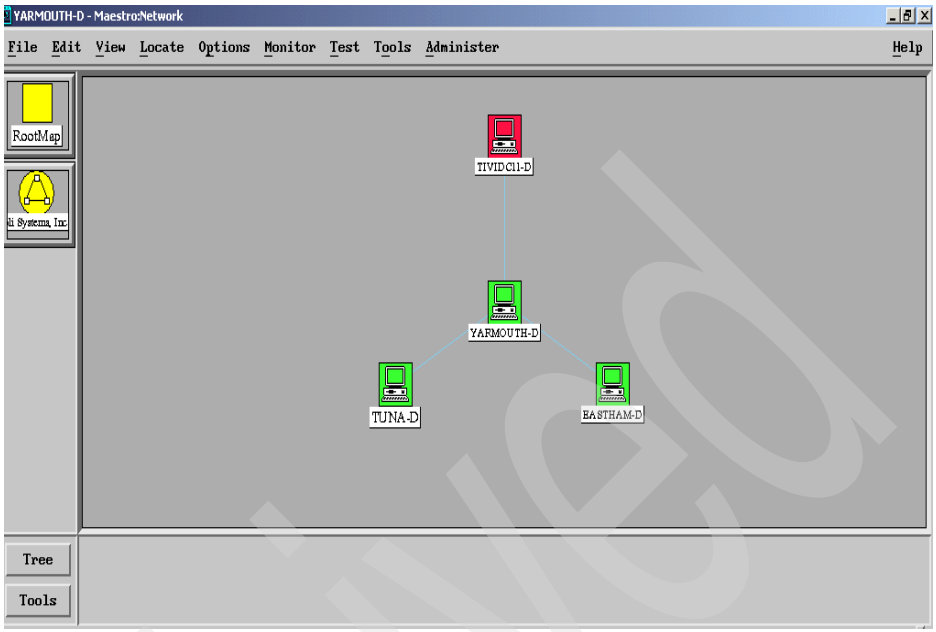

Figure 13-21 Status changed on tividc11

Note that when the schedules complete successfully after initial failures, the NetView icons are not restored to a normal green state. The display looks the same as Figure 13-21 because the TWS and NetView does not change the symbol state with the incoming success traps by default.

You have two options to turn it back to the green state:

- You can configure the TWS and NetView to send events that are not sent by default (mostly successful completion events or warning messages, a list is given in Appendix B, “Event file message formats” on page 663). However, in this case, you have to be careful about the extra network traffic you are introducing by doing so.
- You can use the Acknowledge function in NetView:

  a. Select **tividc11** and click it with the third mouse button.

  b. Select **Options** and then **Acknowledge**.

13.6.6 TWS and NetView configuration files

On each managed node, the selection of events and how they are reported is controlled by setting variables in two configuration files:

- The BmEvents configuration file controls the reporting of job scheduling events by the **mailman** and **batchman** production processes. These events are passed on to the agent, which can convert them to SNMP traps, depending on the settings in its configuration file. The BMEvents.conf file is located in the /*TWSHome*/OV directory.

  Example 13-3 shows the BMEvents.conf file on yarmouth, our TWS master.

Example 13-3 BmEvents.conf file

```
# cat BmEvents.conf
# @(#) $Header:
/usr/local/SRC_CLEAR/maestro/JSS/maestro/NetView/RCS/BmEvents.co
nf,v 1.6 1996/12/16 18:19:50 ee viola_thunder $
# This file contains the configuration information for the BmEvents module.
#
# This module will determine how and what batchman notifies other processes
# of events that have occurred.
#
# The lines in this file can contain the following:

# OPTIONS=MASTER|OFF

# MASTER This tells batchman to act as the master of the network and
# information on all cpus are returned by this module.
#
# OFF This tells batchman to not report any events.
#
# default on the master cpu is to report all job scheduling events
# for all cpus on the Maestro network (MASTER); default on other cpus
# is to report events for this cpu only.

# LOGGING=ALL|KEY

# ALL This tells batchman all the valid event numbers are reported.
#
# KEY This tells batchman the key-flag filter is enabled
#
# default is ALL for all the cpus

# EVENTS = <n> ...

# <n> is a valid event number (see Maestro.mib traps for the valid event
# numbers and the contents of each event.
#
```

```
# default is 51 101 102 105 111 151 152 155 201 202 203 204 251 252 301

# These can be followed with upto 5 different notification paths in the
# following format:
# PIPE=<filename> This is used for communicating with a Unison Fifo file.
#  The format of this is a 4 byte message len followed by the message.
# FILE=<filename> This is for appending to the end of a regular file.
#  This file will be truncated on each new processing day.
# MSG=<filename%-.msg>  This is used for communicating with a Unison Message
# file.

# To communcate with Unison snmp agent, the following is required:
# PIPE=/tivoli/TWS/D/tws-d/MAGENT.P
PIPE=/tivoli/TWS/D/tws-d/MAGENT.P

# To communcate with OperationsCenter using the demonstration log file
#  encapsulation the following is required:
# FILE=/tivoli/TWS/D/tws-d/event.log
```

- The MAgent configuration file controls reporting by the TWS and NetView integration agent, magent. Events selected in this file are turned into SNMP traps, which are passed to NetView by the TWS and NetView integration manager, mdemon, on the management node. The traps can also be processed by other network management systems. The MAgent.conf file is located in the */TWSHome*/OV directory.

  Example 13-4 shows the MAgent.conf file on yarmouth, our TWS master.

Example 13-4 MAgent.conf file

```
# cat MAgent.conf
# @(#) $Header:
/usr/local/SRC_CLEAR/maestro/JSS/maestro/NetView/RCS/MAgent.conf
,v 1.6 1996/12/16 18:20:50 ee viola_thunder $
# This file contains the configuration information for the snmp agent.

# OPTIONS=MASTER|OFF

# MASTER This tells the agent to act as the master of the network and
# information on all cpus are returned by this module.
#
# OFF This tells the agent not report any events, i.e., no traps are sent.
#
# default is MASTER on the master cpu, and OFF on other cpus

# EVENTS = <n> ...

# <n> is a valid event number (see Maestro.mib traps for the valid event
# numbers and the contents of each event.
```

```
#
# default is 1 52 53 54 101 102 105 111 151 152 155 201 202 203 204 252 301

# These can be followed with changes to the critical processes in the
# following format:
# +<name> [<pidfilename>] If this is a maestro process then the pidfilename
#  is optional.
# +SENDMAIL /etc/sendmail.pid
# +SYSLOG   /etc/syslogd.pid
#
# -<name> To eliminate a maestro process from the critical processes.
# -@:WRITER to eliminate all writers
# -SERVER@ to eliminate all servers
# -MAGENT to eliminate the snmp agent.
#
```

13.6.7 TWS and NetView events

The TWS and NetView events are listed in Appendix B, “Event file message formats” on page 663. All of the listed events can result in SNMP traps that are generated by the TWS and NetView integration agents.

Tip: Whether traps are generated is controlled by the two different settings:

- The EVENTS parameter in the BmEvents configuration file:

  This parameter lists the events to be reported. If the EVENTS parameter is omitted, the events reported by default are: 51 101 102 105 151 152 155 201 202 203 204 251 252.

- The EVENTS parameter in the MAgent configuration file:

  This parameter lists the events to be sent as SNMP traps. With the exception of events 1, 52, and 53, traps are not generated unless the corresponding events are turned on in the BmEvents configuration file.

  For example, the following settings enable the events numbers 1, 52, 53, 54, and101 to be sent to NetView.

  EVENTS parameter in the MAgent configuration file:

  ```
  EVENT=1,52,53,54,101:
  ```

  EVENTS parameter in the BMEvents configuration file

  ```
  EVENT=54,101
  ```

13.6.8 Configuring TWS to send traps to any SNMP manager

TWS has the capability to send traps to any SNMP manager, not only NetView. The added function with NetView is for TWS workstations and processes to be seen in the NetView Console graphically and the capability to issue commands on TWS workstations from the NetView console. However, if you are using an SNMP manager application other than NetView, you can still configure TWS to send SNMP traps to these management applications.

To configure TWS to send SNMP traps on the master workstation:

1. Edit the BmEvents.conf file in the /*TWSHome*/OV directory to include the desired SNMP traps.
2. Determine whether to use a named pipe to send the information to the application or whether the application will parse a text file.
   a. For the pipe option, uncomment the PIPE option and name the file accordingly if a specific name is required to provide the output in a Unison FIFO file whose format is a four-byte message length followed by the message. For example:

      ```
      PIPE=/TWSHome/MAGENT.P
      ```

   b. For the file option, uncomment the FILE option and name the file accordingly. The output is an ASCII file that has the trap number in the first field with varying numbers of fields following the trap number, depending upon the trap. For example:

      ```
      FILE=/TWSHome/event.log
      ```

Tip: The difference between using the FILE option versus the PIPE option is that the FILE option creates a flat ascii file which you can then parse to get the snmp traps to your application. The PIPE option, on the other hand, creates a fifo file with the traps in it and the management node should have to read the file as a pipe. We found that the FILE option is easier to implement.

3. After making the configuration changes above, run the customize script that is in the /*TWSHome*/OV directory to enable Maestro to trap the events. This command adds a line to start magent on the master pointing to the peer (which should be the host name of the workstation on which the SNMP management node is running or the host name of the SNMP manager).

   ```
   /bin/sh customize -manager hostname-of-host-receiving-traps
   ```

   For example, in order to send TWS events to a TEC server named tecitso, you can run the customize script as follows:

   ```
   /bin/sh customize -manager tecitso
   ```

Note: In addition to running this script, you need to install and configure the Tivoli Enterprise Console SNMP adapter on the TWS master to send SNMP events directly to Tivoli Enterprise Console. Refer to Tivoli Enterprise Console manuals for more information about customizing the Tivoli Enterprise Console SNMP adapter.

The other way to the send TWS events to Tivoli Enterprise Console is to use the TWS Tivoli Enterprise Console adapter that is shipped with the TWS Plus Module. Also, if you are using NetView and Tivoli Enterprise Console together, you can send TWS events from NetView to Tivoli Enterprise Console, as well. This configuration does not require the Tivoli Enterprise Console SNMP adapter to be installed.

4. To enable the **magent** daemon, stop maestro with the following command:

   ```
   conman stop;wait and then shut;wait
   ```

5. Run Startup to restart **netman** and **magent**.
6. Issue a link command as follows:

   ```
   conman link @;noask
   ```

Part 4

Right tool for the job

The preceding chapters of this book have covered various integration points with Tivoli Workload Scheduler. This final section looks at the scenarios that have been presented and discusses how to put these scenarios together for sample larger picture configurations.

A

Using the right tool for the job

With so many ways to integrate products with Tivoli Workload Scheduler (TWS), which is the best set of Tivoli products to manage your business? Each business should be evaluated to find the best combination of Tivoli management products to enhance productivity in a timely and understandable manner.

Tivoli software provides intelligent management software to help manage IT resources. As IT infrastructure has become more complex, it also has become more important to have timely responsiveness to problems, to be able to keep up with a constantly changing environment, to remain compliant with laws and regulations, and to manage costs. Each product that is included in this book is capable of helping with one or more of these goals. This chapter looks at the scenarios that have been presented and discusses how to put these scenarios together for sample larger picture configurations.

Product categorization

Tivoli products provide on demand management and automation of an IT infrastructure. The products that this manual uses are only a subset of the Tivoli products that are available. For example, security products were not included in the individual scenarios but could easily be part of an installation.

Tivoli Availability products

Tivoli Availability products ensure that your IT environment is functioning efficiently. The following products were reviewed in scenarios in this book:

- Tivoli Information Manager for z/OS
- Tivoli OMEGAMON
- Tivoli Monitoring
- Tivoli Framework

Tivoli Business Service Management products

Tivoli Business Service Management products demonstrate and leverage links between your business processes and the actual IT environment. Using these products organizes your IT organization to follow business priorities and increases the agility and responsiveness of the IT organization. The following Business Service Management products were reviewed in scenarios in this book:

- Tivoli Business Systems Manager
- Tivoli Business Systems Manager for z/OS
- Tivoli Enterprise Console
- Tivoli NetView
- Tivoli NetView for z/OS
- Tivoli Service Level Advisor

Tivoli Orchestration products

Tivoli Orchestration products continually monitor and measure demand and load in the distributed environment, ensuring that servers, applications, and middleware maintain performance levels as resources become available and are needed. We used Tivoli Intelligent Orchestrator in the scenarios in this book.

Tivoli Provisioning products

Tivoli Provisioning products allow IT organizations to be able to provide the right resources as needed, when needed. The Provisioning products that we used in scenarios in this book include:

- Tivoli Configuration Manager
- Tivoli System Automation for z/OS

Tivoli Storage and Optimization products

Tivoli Storage and Optimization products allow you to protect your environment and maximize its utility. This protection is an important part of keeping an IT environment flexible and responsive to company needs. The Storage and Optimization products that we used in scenarios in this book include:

- Tivoli Decision Support for z/OS
- Tivoli Storage Manager

Non-Tivoli products

This book also used the following non-Tivoli products:

- Tivoli Workload Manager
- DFSMSrmm
- JOB/SCAN

Integrating scenarios

For each scenario, we used TWS or TWS for z/OS paired with one or more other products. How do you begin to consider which products should be used together?

The following sections discuss some of the extended scenarios that our team considered. These scenarios are in no way is a definitive list of ways to combine products. Your service provider can help you customize the correct combination of products for your environment.

Availability and Business Systems Monitoring in a distributed or end-to-end TWS environment

Some products have a natural relationship. Tivoli Framework, Tivoli Monitoring, Tivoli NetView, Tivoli Enterprise Console, and Tivoli Business Systems Manager,

for example, provide a very complete method of monitoring both TWS and the hardware and software that are part of the distributed TWS environment.

The use of Tivoli Framework, along with Tivoli Monitoring and Tivoli Enterprise Console, gives your environment powerful monitoring tools and the ability to automate responses in a distributed environment. You can use Tivoli NetView to monitor hardware and network devices, and this data can be fed to either Tivoli Enterprise Console (for correlation and possible automation) or to Tivoli Business Systems Manager directly. Feeding Tivoli Monitoring events through Tivoli Enterprise Console, NetView events either through Tivoli Enterprise Console or directly to Tivoli Business Systems Manager, and TWS Distributed or TWS for z/OS (or end-to-end) directly to Tivoli Business Systems Manager creates a large body of information that you can use to create lines of business in Tivoli Business Systems Manager that reflect the true state of FTA systems, the FTA job streams and jobs, and network conditions.

Provisioning and Business Systems Monitoring in a distributed or end-to-end environment

NetView is capable of monitoring the status of distributed servers and other network objects. You can deploy it to gather status on new systems that can become FTAs in a growing TWS environment. Tivoli Configuration Manager can then be used to install the FTAs. The NetView data can continue to be used to have feedback on the environment where TWS operates.

Provisioning and Storage Optimization in a z/OS environment

One important result of using multiple Tivoli products is that many of the products are capable of ensuring that the other products are performing as expected. An example of this is using RODM to monitor the state of a DB/2 database (up/down) to ensure that Tivoli Decision Support for z/OS reports or data collects run only after DB/2 is available. You would then use a TWS special resource to monitor the state of DB/2 via the RODM object state. See *Tivoli Workload Scheduler for z/OS 8.2 Managing the Workload*, SC32-1263, for more information.

Provisioning in a distributed environment

You can configure Tivoli Configuration Manager to ensure that scripts and programs that are launched by the jobs on an FTA exist and are the current version before the jobs launch.

Availability and Business Systems Management in a z/OS environment

You can configure TWS, Tivoli Business Systems Manager, and Infoman to work together to ensure a prompt response if there is a problem , such as a job stream or application not completing. If an event occurs in the TWS environment, you can configure Tivoli Business Systems Manager to respond by cutting a trouble ticket to Infoman, ensuring that the support team has the relevant information quickly so that they can fix the problem.

Using multiple interfaces

TWS can interface to one or more of these products at the same time. Depending on the needs of the business, it might be advantageous to interface through one product to another, rather than directly. Some examples of this type of indirect integration are:

- Tivoli Business Systems Manager through Tivoli Monitoring and Tivoli Enterprise Console
- Tivoli Service Level Advisor through Tivoli Business Systems Manager
- Tivoli Enterprise Console through Tivoli NetView

Table A-1 on page 660 and Table A-2 on page 661 show possible integration points.

Note: These lists are not exhaustive. There are many possibilities for integration between TWS Distributed and TWS for z/OS and Tivoli and other IBM products.

Table A-1 Possible zSeries IBM Tivoli product integrations

Possible IBM Tivoli product integrations	**Tivoli Workload Scheduler for z/OS**	**Tivoli Information Management**	**Tivoli NetView for z/OS**	**Tivoli System Automation**	**Tivoli Business Systems Manager**	**Tivoli Decision Support**	**Diversified JOB/SCAN**
Tivoli Workload Scheduler for z/OS	✓	✓	✓	✓	✓	✓	✓
Tivoli Information Management	✓	✓	✓	✓	✓	✓	
Tivoli NetView for z/OS	✓	✓	✓	✓	✓	✓	✓
Tivoli System Automation	✓	✓	✓	✓			
Tivoli Business Systems Manager	✓	✓	✓		✓	✓	
Tivoli Decision Support	✓	✓	✓		✓	✓	
Diversified JOB/SCAN	✓		✓				✓

Table A-2 Possible IBM Tivoli product integrations

Possible IBM Tivoli product integrations	**Tivoli Workload Scheduler**	**Tivoli Monitoring**	**Tivoli Enterprise Console**	**Tivoli NetView**	**Tivoli Service Level Advisor**	**Tivoli Business Systems Manager**	**Tivoli Intelligent Orchestrator**	**Tivoli Storage Manager**
Tivoli Workload Scheduler	✓	✓	✓	✓	✓	✓	✓	✓
Tivoli Monitoring	✓	✓	✓	✓	✓	✓		
Tivoli Enterprise Console	✓	✓	✓	✓	✓	✓	✓	✓
Tivoli NetView	✓	✓	✓	✓		✓		
Tivoli Service Level Advisor	✓	✓	✓		✓	✓		✓
Tivoli Business Systems Manager	✓	✓	✓	✓	✓	✓		✓
Tivoli Intelligent Orchestrator	✓		✓				✓	
Tivoli Storage Manager	✓		✓		✓	✓		✓

B

Event file message formats

This appendix contains a list of messages and the formats of the messages that are produced by the batch management process (**batchman**) of Tivoli Workload Scheduler (TWS) for integrating with other products. This chapter also describes how to customize the **batchman** events by editing the BmEvents.conf file.

Customizing the batchman events configuration file

The **batchman** events configuration file is named *TWSHome*/BmEvents.conf. You use it to configure the TWS production event logging processes on each workstation that has an agent installed or on the master domain manager. You can change the following parameters in this file.

OPTIONS

The OPTIONS parameter is as follows:

```
OPTIONS=MASTER|OFF
```

If set to MASTER, this workstation report s all job scheduling events for all workstations in the TWS network. If set to OFF, the job scheduling events are reported only for the workstation on which the file is configured. If omitted, it defaults to MASTER on the master workstation, and OFF on other workstations. The value should be set to MASTER for the master domain manager workstation and OFF for the other workstations.

Tip: A workstation other than the master domain manager can be used as a reporting workstation for all events in the TWS network. To provide this capability, the workstation definition must have FULLSTATUS ON and the BmEvents.conf parameter OPTIONS=MASTER.

LOGGING

The LOGGING parameter is as follows:

```
LOGGING=ALL|KEY
```

If the value is KEY, the key flag filter mechanism is enabled. Events are sent only for key jobs and job streams (those marked with the KEYJOB or KEYSCHED keywords or "Is monitored" in the Job Scheduling Console.) Refer to Table B-1 on page 666 to find which events are filtered by the key flag. If the value is ALL, events are sent for all jobs and job streams, regardless of the key flag settings.

EVENTS

The EVENTS parameter is as follows:

```
EVENTS = <n>
```

This is the list of events to report to the external monitoring tool. Enter event numbers to report, separated by spaces. If omitted, the default events are:

```
EVENTS = 51 101 102 105 151 152 155 201 202 203 204 251 252 301
```

Note: Event 51 causes the mailman and batchman processes to report the fact that they were restarted. Events 1, 52, and 53 are not valid in this file.

If the EVENT parameter is included, it completely overrides the defaults. To remove only event 102 from the list, for example, enter the following:

Example: B-1 Selecting events

```
EVENT=51 101 105 151 152 155 201 202 203 204 251 252 301
```

MSG

The MSG parameter is as follows:

```
MSG=<msg_path>
```

The name and the path of a message file where batchman and mailman will write the events for the Tivoli Business Systems Manager listener agent to read. More than one message file can be added.

PIPE

The PIPE parameter is as follows:

```
PIPE=<pipe_path>
```

The name and the path of a message file (actually, a named pipe) where batchman and mailman will write the events for the SNMP management agent to read. More than one pipe can be added.

SYMEVNTS

The SYMEVNTS parameter is as follows:

```
SYMEVNTS=YES|NO
```

If the value is YES, batchman reports job status events immediately after the generation of the plan. It is valid for only key-flagged jobs with LOGGING=KEY. If the value is set to NO, no report is given. NO is the default.

FILE

The FILE parameter is as follows:

```
FILE=filename
```

If set, job scheduling events are written to an ASCII log file specified by the FILE parameter. You can add more than one file.

FILE_NO_UTF8

The FILE_NO_UTF8 parameter is as follows:

`FILE_NO_UTF8=`*`filename`*

If set, the job scheduling events are written in the local language file specified by this parameter. You can add ore than one file.

Batchman events by number

Events are produced into the file specified in the BmEvents.conf file, as described in 1.2, “IBM TWS solution scenario” on page 25. Each event in the log file is produced in a specific format described in this appendix. Each event is prefixed by an event number.

Table B-1 describes each event. If the *Key filter* column contains Yes, the LOGGING=KEY setting in BmEvents.conf causes the message to be produced only when the associated job or job stream is flagged as *Monitored*.

Table B-1 Batchman events

Event	When the event is produced	Key filter
51	TWS process started	No
52	TWS process no longer present	No
53	A monitored process ended abnormally	No
54	A connection between an Extended Agent and its host was lost	No
101	Job abended (ABEND)	No
102	Job ended in error (ERROR)	No
103	Job launched (EXEC)	Yes
104	Job completed (SUCC, DONE, ABENP, PEND)	Yes
105	Job latest start time (UNTIL time) has expired	No
106	Job submitted (ADD)	No
107	Job has been canceled (CANCL)	No
108	Job dependencies have been met (READY)	Yes
109	Job is on hold (HOLD status)	Yes
110	Job is in restart status	Yes

Event	When the event is produced	Key filter
111	Job failed to launch (FAIL)	No
112	Job is in success pending state (SUCCP)	Yes
113	Job is awaiting external dependency (EXTRN)	Yes
114	Job is being introduced to the system (INTRO)	Yes
115	Job is suspended (SUSP)	Yes
116	Job is waiting (WAIT)	Yes
117	Job is wait-deferred (WAITD)	Yes
118	Job is “to be scheduled” (SCHED)	Yes
119	Job property modified	Yes
120	Job has not completed and is past its deadline time	Yes
121	Job latest start time expired - continue option selected	Yes
122	Job latest start time expired - cancel option selected	Yes
151	Job stream abended (ABEND)	No
152	Job stream cannot continue (STUCK)	No
153	Job stream started (EXEC)	Yes
154	Job stream finished (SUCC)	Yes
155	Job stream latest start time expired (UNTIL)	Yes
156	Job stream submitted (ADD)	No
157	Job stream canceled (CANCL)	No
158	Job stream dependencies met (READY)	Yes
159	Job stream on hold (HOLD)	Yes
160	Job stream awaiting external dependency (EXTRN state)	Yes
161	Job stream will be canceled pending dependency resolution (CANCP cancel pending)	Yes
162	Job stream property modified	Yes
163	Job stream is late	Yes
164	Job stream latest start time expired - continue option selected	Yes

Event	When the event is produced	Key filter
165	Job stream latest start time expired - cancel option selected	Yes
201	Global prompt displayed	No
202	Job stream prompt displayed	Yes
203	Job prompt displayed	Yes
204	Job recovery prompt displayed	No
251	Communication link between workstations closed	No
252	Communication link between workstations failed	No
351	Domain manager has been switched	No

Event message parameters

There are 11 types of events that are produced by the **batchman** process. Each type has a different set of positional parameters that describe the event. Depending on the type of event, some fields might contain a zero or -32768 when that field does not apply to the event being produced.

Process reset events

A process reset event is produced when the critical TWS process changes. Process reset events 52 to 54 are produced by the NetView management agent, not batchman, as shown in Table B-2.

Table B-2 Process reset events (51 to 54)

Position	Description
1	Event number
2	Process name (batchman)
3	Local workstation name
4	Master workstation name

Job events

Positional parameters for job-related events are listed in Table B-3. Note that event 111 (job fail) has a slightly different format.

Table B-3 Job events (101-110, 112-118)

Position	Description
1	Event number
2	Job stream workstation
3	Job stream name
4	Job name
5	Job workstation
6	Job number
7	Job status, indicated by an integer: 1 (READY), 2 (HOLD), 3 (EXEC), 5 (ABEND), 6 (SUCC), 7 (CANCL), 8 (DONE), 13 (FAIL), 16 (INTRO), 23 (ABENP), 24 (SUCCP), 25 (PEND).
8	Job name (may be different than field 4 if the job is rerun with “FROM” or “STEP” options)
9	Job login name
10	Script or command (spaces are substituted with the octal equivalent, \040)
11	“Every” interval (or -32768 when no “every” is defined)
12	Recovery status, indicated by an integer: 1 (stop), 2 (stop after recovery job), 3 (rerun), 4 (rerun after recovery job), 5 (continue), 6 (continue after recovery job), 10 (this is the rerun of the job), 20 (this is the run of the recovery job).
13	Event time stamp. This is the local time on workstation where the job event occurred. It is expressed as: yyyymmddhhmmss00 (that is, year, month, day, hour, minute, second, hundredths - always zeroes).
14	Recovery prompt number, or zero if there is no prompt
15	Job record number
16	Key flag (1 = key job, 0= non-key job)
17	Effective start time
18	Estimated start time
19	Estimated duration

Position	Description
20	Deadline time
21	Return code
22	Original job stream name (will be different than field 3 when job streams carry forward)
23	First job record number (related to field 15, changes when jobs are rerun or cycled by “every” range)

Job events 111 and 204

When a job fails, the event messages produced by events 111 and 204 are slightly different than the other job events (see Table B-4). Event 111 contains the text message explaining the reason for the job failure and event 204 contains the text message of the recovery prompt.

Table B-4 Job events (111, 204)

Position	Description
1	Event number
2	Job stream workstation
3	Job stream name
4	Job name
5	Job workstation
6	Job number
7	Job status, indicated by an integer: 1 (READY), 2 (HOLD), 3 (EXEC), 5 (ABEND), 6 (SUCC), 7 (CANCL), 8 (DONE), 13 (FAIL), 16 (INTRO), 23 (ABENP), 24 (SUCCP), 25 (PEND).
8	Job “real” name (may be different than field 4 if the job is rerun with “FROM” or “STEP” options)
9	Job login name
10	Script or command name (spaces are substituted with the octal equivalent \040)
11	“Every” interval (or -32768 when no “every” is defined)
12	Recovery status, indicated by an integer: 1 (stop), 2 (stop after recovery job), 3 (rerun), 4 (rerun after recovery job), 5 (continue), 6 (continue after recovery job), 10 (this is the rerun of the job), 20 (this is the run of the recovery job).

Position	Description
13	Event time stamp. This is the local time on workstation where the job event occurred. It is expressed as: yyyymmddhhmmss00 (that is, year, month, day, hour, minute, second, hundredths - always zeroes).
14	Recovery prompt number, or zero if there is no prompt
15	Error message or recovery prompt text - this field may contain spaces, and is terminated with three plus signs and a tab character (“+++\t”)
16	Record number
17	Key flag (1 = key job, 0= non-key job)
18	Effective start time
19	Estimated start time
20	Estimated duration
21	Deadline time
22	Return code
23	Original job stream name (will be different than field 3 when job streams carry forward)
24	First job record number (related to field 15, changes when jobs are rerun or cycled by “every” range)

Job properties event

The job properties change event is produced when any job changes. Table B-5 describes the positional parameters produced by the event. The format of this event might change slightly depending on the type of change. The property value changes depending on the property type.

Table B-5 Job properties event (119)

Position	Description
1	Event number
2	Job stream workstation
3	Job stream name
4	Job name
5	Job workstation
6	Job number

Position	Description
7	Property type (see Table B-6 on page 672 below)
8	Property value
9	Current job record number
10	Key flag
11	First job record number (related to field 9)
12	Job name (may be different than field 4 if the job is rerun with “FROM” or “STEP” options)
13	Original job stream name (will be different than field 3 when the job stream carries forward)
14	Time stamp

The job properties event can be sent several times for the same job. Each time the event is issued the Property type and Property value changes from the previous 119 event, as shown in Table B-6.

Table B-6 Job properties types and values

Property type (field 7)	Property value (field 8)
1	Estimated completion time expressed as: yyyymmddhhmmss00 (that is, year, month, day, hour, minute, second, hundredths - always zeroes).
2	Start time
3	Stop time
4	Duration (minutes)
5	Termination property
6	Key status

Global prompt event

Table B-7 describes the positional parameters that are produced by the global prompt event.

Table B-7 Global prompt events (201)

Position	Description
1	Event number
2	Prompt name
3	Prompt number
4	Prompt text (may include spaces)

Job stream prompt

Table B-8 describes the positional parameters that are produced by a job stream prompt being asked.

Table B-8 Job stream prompt events (202)

Position	Description
1	Event number
2	Job stream workstation
3	Job stream name

Job prompt

Table B-9 describes the positional parameters that are produced by a job or job recovery prompt being asked.

Table B-9 Job prompt events (203)

Position	Description
1	Event number
2	Job stream workstation
3	Job stream name
4	Job name
5	Job workstation
6	Prompt number
7	Prompt message (may include spaces)

Job stream events

Table B-10 describes the positional parameters for job stream events.

Table B-10 Job stream events (151-161, 163-164)

Position	Description
1	Event number
2	Job stream workstation
3	Job stream name
4	Job stream status, indicated by an integer: 1 (READY), 2 (HOLD), 3 (EXEC), 4 (STUCK), 5 (ABEND), 6 (SUCC),7 (CANCL)
5	Record number
6	Key flag
7	Original job stream name (will be different than field 3 when job streams carry forward)
8	Event time stamp. This is the local time on workstation where the job event occurred. It is expressed as: yyyymmddhhmmss00 (that is, year, month, day, hour, minute, second, hundredths - always zeroes).

Job stream properties event

The job stream properties change event is produced when any job stream changes. Table B-11 describes the positional parameters that are produced by the event. The format of this event might change slightly depending on the type of change. The property value changes depending on the property type.

Table B-11 Job stream property change events (162)

Position	Description
1	Event number
2	Job stream workstation
3	Job stream name
4	Property type (2 = start time, 3 = stop time, 4 = duration)
5	Property value (related to field 4)
6	Record number
7	Original job stream name (will be different than field 3 when job streams
8	Time stamp

Link status events

Table B-12 describes the positional parameters for link status events.

Table B-12 Link status events (251-252)

Position	Description
1	Event number
2	Workstation name
3	Link state

Switch manager events

Table B-13 describes the positional parameters for domain switch manager events.

Table B-13 Switch manager events (301)

Position	Description
1	Event number
2	New manager workstation name
3	Domain name
4	Time stamp

C

JOB/SCAN examples

This appendix provides example output from JOB/SCAN and the source code for the JOB/SCAN exit that is demonstrated in this book (see Chapter 7, “Integrating JOB/SCAN” on page 283). The purpose of the exit is to tailor messages produced by JOB/SCAN using information from Tivoli Workload Scheduler (TWS) for z/OS.

See *JOB/SCAN Standards User Guide* (shipped with the product) for more information about writing exits.

JOB/SCAN exit sample TWSSTD1

TWSSTD1 searches the JOB/SCAN error message table looking for message number DSS4050 (Data set not found). If the message exists, it calls TWSSPRES to see if there is a Special Resource matching data set name (see Example C-1). If there is, then message DSS4050 is removed and replaced with a warning message about the job's requirement of a special resource.

To invoke this exit change the JOB/SCAN PARMLIB member JSOPT02 and include the following line:

```
EXIT NAME               =TWSSTD1,'TWSC'
```

The second parameter corresponds to the name of the TWS for z/OS subsystem.

Example: C-1 JOB/SCAN Standards exit - TWSSTD1

```
IDENTIFICATION DIVISION.
       PROGRAM-ID.    TWSSTD1.
       AUTHOR.        DIVERSIFIED SOFTWARE SYSTEMS, INC.
       INSTALLATION.  LAST MODIFIED ON 02/15/05.
       DATE-WRITTEN.  02/15/05.
       DATE-COMPILED.
       REMARKS.
               THIS MODULE WILL SUPPRESS A DSS4050 ERROR MESSAGE
               THAT IS ISSUED FOR A DATASET ALREADY DEFINED AS
               A SPECIAL RESOURCE.

       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       01  WORK-FIELDS.
           10  TWS-SUBSYSTEM     PIC X(04) VALUE SPACES.
           10  TWS-LUNAME        PIC X(17) VALUE SPACES.
           10  DELIM-1           PIC X.
           10  DELIM-2           PIC X.
       01  DATA-SET-NAME         PIC X(44).
       LINKAGE SECTION.
       COPY CEPL.
      *
       PROCEDURE DIVISION USING EXIT-PARAMETER-LIST.
      * ---------------------------------------------------------------
      *
           SET     EPLE-SUB TO 1.

           IF      EPLC-CALL-DD-ENTRY
                   PERFORM CALL-DD-ENTRY  THRU CALL-DD-ENTRY-EXIT
                   GO TO   SECTION-BRANCH-EXIT.
```

```
     IF      EPLC-CALL-INITIAL
             PERFORM CALL-INITIAL   THRU CALL-INITIAL-EXIT
             GO TO   SECTION-BRANCH-EXIT.

 SECTION-BRANCH-EXIT.
     GOBACK.

* ---------------------------------------------------------------
*
 CALL-INITIAL SECTION.
     UNSTRING EPLC-USER-PARM DELIMITED BY ','
              INTO TWS-SUBSYSTEM DELIMITER IN DELIM-1
                   TWS-LUNAME    DELIMITER IN DELIM-2.

     IF  TWS-SUBSYSTEM = SPACES
         DISPLAY 'NO TWS SUBSYTSTEM SPECIFIED IN USER PARMS'
         DISPLAY 'NO SPECIAL RESOURCE CHECKING DONE'.

 CALL-INITIAL-EXIT.
     EXIT.

* ---------------------------------------------------------------
*
 CALL-DD-ENTRY SECTION.
*
     IF TWS-SUBSYSTEM = SPACES
        GO TO CALL-DD-ENTRY-EXIT.

     MOVE 'DSS4050' TO    EPLM-MESSAGE-NUMBER
     MOVE +2        TO    EPLM-REQUEST
     CALL 'JO6YTBL' USING EPL-JOB-SCAN-ERRORS
                          EPLM-DATA
*
* RETURN CODE = 0 MEANS KEY WAS FOUND IN ERROR MESSAGE TABLE
* EPLM-ACTION-FLAG 'S' = SUPPRESS
* REQUEST CODE     +17 = UPDATE ENTRY
* SUPPRESS DSSI ERROR MESSAGE DSS4050
*
     IF  RETURN-CODE = 0
         MOVE EPLD-DSNAME TO    DATA-SET-NAME
         CALL 'TWSSPRES'  USING TWS-SUBSYSTEM
                                TWS-LUNAME
                                DATA-SET-NAME
         IF  RETURN-CODE = 0
*
*            RETURN CODE 0 MEANS IT IS A SPECIAL RESOUCE
*            SO DELETE THE ORIGINAL MESSAGE AND ISSUE
*            A WARNING ABOUT TWS WAITING
```

```
*
                MOVE 'S'             TO EPLM-ACTION-FLAG
                MOVE +17             TO EPLM-REQUEST
                MOVE +4              TO EPLM-SEVERITY
                CALL 'JO6YTBL' USING EPL-JOB-SCAN-ERRORS
                                     EPLM-DATA

                MOVE  '101 TWS WILL WAIT FOR SPECIAL RESOURCE.'
                      TO EPLC-ERROR-MESSAGE
                MOVE 4 TO EPLC-SEVERITY
                PERFORM Z999-SET-ERROR

                STRING '102 TWS RESNAME = "' DELIMITED BY SIZE
                       DATA-SET-NAME DELIMITED BY SPACES
                       '"' DELIMITED BY SIZE
                       INTO EPLC-ERROR-MESSAGE
                MOVE 0 TO EPLC-SEVERITY
                PERFORM Z999-SET-ERROR.
      *
       CALL-DD-ENTRY-EXIT.
           EXIT.
       COPY CERR.
```

JOB/SCAN TWS interface module TWSSPRES

TWSSPRES is invoked by TWSSTD1 to look up Special Resource names in the TWS for z/OS data base (see Example C-2).

Example: C-2 JOB/SCAN interface module to extract Special Resource names

```
TWSSPRES CSECT
TWSSPRES   AMODE 24
        SAVE  (14,12),,'TWSSPRES &SYSDATE &SYSTIME DSSI'
        LR    R12,R15             ESTABLISH BASE REGISTER
        USING TWSSPRES,R12        TELL ASSEMBLER
        LA    R14,SAVEAREA        CHAIN SAVE AREAS
        ST    R14,8(,R13)         MINE INTO MVS
        ST    R13,4(,R14)         MVS INTO MINE
        LR    R13,R14             MINE IS NOW CURRENT
*
*       get parms
*
        L     R2,0(R1)            ADDRESS OF SUBSYSTEM FIRST PARM
        MVC   SUBSYS,0(R2)        GET SUBSYSTEM NAME
        L     R2,4(R1)            RESOURCE NAME SECOND PARM
        MVC   LUNAME,0(R2)        SAVE IT
        L     R2,8(R1)            RESOURCE NAME SECOND PARM
```

```
         MVC   TRGTNAME,0(R2)      SAVE IT
         LOAD  EPLOC=EQQYCOM
         ST    R0,EQQYADDR
*
*        init eqqycom
*
         MVC   ACTION,=CL8'INIT    '  INDICATE INIT DESIRED.
         MVC   RESOURCE(4),SUBSYS
         MVC   NAMELIST(8),=CL8'      '
         CLI   LUNAME,C' '
         BNH   NOLU
         MVC   NAMELIST(8),=CL8'LUNAME'
         LA    R2,LUNAME
         ST    R2,VALULIST
NOLU     DS    0F
         LA    R1,PARMLIST        SET UP PARM ADDRESS FOR EQQYCOM.
         L     R15,EQQYADDR
         BALR  R14,R15
         LTR   R15,R15            ANY PROBLEMS?
         BNZ   RC20               CONTINUE
*
*        list special resource
*
         MVC   ACTION,=CL8'LIST    '  INDICATE LIST DESIRED.
         MVC   RESOURCE,=CL8'SRCOM   '  INDICATE RESOURCE NAME.
         MVC   NAMELIST(8),=CL8'RESNAME ' RESOURCE NAME
         LA    R2,TRGTNAME        GET TARGET NAME
         ST    R2,VALULIST        SAVE THE NAME ADDRESS
         LA    R1,PARMLIST        SET UP PARM ADDRESS FOR EQQYCOM.
         L     R15,EQQYADDR
         BALR  R14,R15
         C     R15,RETCODE
         ST    R15,SAVERC
         BNE   RC24
*
*        terminate eqqycom
*
         MVC   ACTION,=CL8'TERM    '  INDICATE TERMINATION DESIRED.
         LA    R1,PARMLIST        SET UP PARM ADDRESS FOR EQQYCOM.
         L     R15,EQQYADDR
         BALR  R14,R15
         DELETE  EPLOC=EQQYCOM
         L     R15,SAVERC         GET LIST RETURN CODE
         B     RETURN
RC24     DS    0H
         LA    R15,24
         B     RETURN
RC20     DS    0H
         LA    R15,20
```

```
RETURN   DS    OH
         L     R13,4(,R13)          RESTORE MVS SAVE AREA
         RETURN (14,12),RC=(15)     RETURN TO MVS
*
PARMLIST DC    A(ACTION)            ACTION CODE
         DC    A(RESOURCE)          RESOURCE CODE
         DC    A(DATAPTR)           DATA AREA ADDRESS
         DC    A(NAMELIST)          ARGUMENT NAME LIST
         DC    A(VALULIST)          ARGUMENT VALUE LIST
         DC    A(COMBLOCK)          COMMUNICATION BLOCK
LISTEN   DC    A(RETCODE)           RETURN CODE
*
ACTION   DC    CL8' '               * ACTION CODE
RESOURCE DC    CL8' '               * RESOURCE NAME
DATAPTR  DC    F'0'                 * DATAAREA POINTER
NAMELIST DC    CL80' '              * 10 ARGUMENT NAMES
VALULIST DC    10F'0'               * ADDRESSES OF 10 VALUES
COMBLOCK DC    F'0'                 * COM BLOCK POINTER
RETCODE  DC    F'0'                 * RETURN CODE
SUBSYS   DC    CL4' '
LUNAME   DC    CL17' '
TRGTNAME DC    CL44' '
SAVEAREA DC    18F'0'
saverc   DC    F'0'
EQQYCOM  DC    CL8'EQQYCOM '
EQQYADDR DC    A(*-*)
         LTORG
         YREGS
         END
```

JOB/SCAN sample output

Example C-3 shows a Structured JCL Listing from JOB/SCAN. This is the output from a JCL scan that contains the executing JCL as it was submitted by TWS for z/OS with all variables resolved, a merged JCL listing containing procedure JCL and control cards, and all error, warning, and information messages from the scan. These messages are included both in place within the JCL and as a summary at the end.

Example: C-3 JOB/SCAN sample output

```
JOB/SCAN            STRUCTURED JCL LISTING    6.2.4A     2005-02-17  17:51:34
------------------------------------------------------------------------------
//PJGLOO2O JOB (A,P,GL),'JONES',CLASS=A,MSGCLASS=X,  GL MASTER UPDATE
//             MSGLEVEL=(1,1)
//      JCLLIB ORDER=TWS.INST.JOBLIB
//*%OPC SCAN
//*>OPC SETFORM OCDATE=(MM/DD/YY)
//*>OPC SETVAR TACC=(OCDATE - 3WK)
//*>OPC SETFORM OCDATE=(DD/MM/CCYY)
//*>OPC SETVAR TAUD=(OCDATE - 27WD)
//*>OPC TABLE NAME=GLDLYPOST
//*************************************************************
//JSO1O    EXEC  PPGLOO2O,TYPE=WEEKLY
$$SYSUT1  DD  DSN=FC.GLPROD.INPUT.FILE,DISP=OLD
$$PSO3O.DATECARD DD *
070589    050217       1 1 3 NNN USRCODE=GLEDGER
$$PSO5O.DATECARD DD *                                DATE CARD OVERRID
070589                          TYPE 5 CONTROL
ACCDATE=01/27/05
AUDITDATE=17/01/2005
  P1//PPGLOO2O PROC INDX=FC,TYPE=D,
  P1//            APPLIC=GLPROD,
  P1//            SPTRN=2O,SPROLL=5OO,DVOL=VOLWEEK2,
  P1//            GEN3='+1',GEN4='+1',GEN5='+1'
  P1//PSO1O    EXEC  PGM=IEBGENER                 COPY THE KEYTAPE TO DISK
  P1//SYSPRINT DD  SYSOUT=A
  P1//SYSIN    DD  DUMMY
  P1//SYSUT1   DD  DUMMY                          (PJGLTRO1) FROM DATA PREP
  P1$/SYSUT1  DD  DSN=FC.GLPROD.INPUT.FILE,DISP=OLD
**WARNING - DSS91O1W -  TWS WILL WAIT FOR SPECIAL RESOURCE.
*ADVISORY - DSS91O2A -  TWS RESNAME = "FC.GLPROD.INPUT.FILE"
  P1//SYSUT2   DD  DSN=&&TEMP,                     WORKING INPUT DATA FILE
  P1//             DISP=(,PASS),
  P1//             UNIT=SYSDA,
  P1//             SPACE=(TRK,(5,1)),
  P1//             DCB=BLKSIZE=6160
  P1//SYSUDUMP DD  SYSOUT=A
```

```
P1//PS020    EXEC  PGM=SORT,                SORT BY ACCT AND UPDATE ACCT
P1//            REGION=512K
P1//STEPLIB  DD  DSN=FC.GLPROD.LOAD,
P1//            DISP=SHR
P1//SORTLIB  DD  DSN=SYS1.SORTLIB,                  +RD SORT RESIDENCE
P1//            DISP=SHR
P1//SORTPRT  DD  SYSOUT=(J,,9),         (PJGLRD05) SORT STATISTICS REPORT
P1//            COPIES=2
P1//SYSPRINT DD  SYSOUT=A
P1//SYSIN    DD  DSN=&INDX..&APPLIC..CNTL(G201001S),          SORT CTL
P1--SYSIN    DD  DSN=FC.GLPROD.CNTL(G201001S),          SORT CTL
P1//            DISP=SHR
  ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----
  SORT CONTROL 7
P1//SORTIN   DD  DSN=*.PS010.SYSUT2,                   COPIED KEYTAPE
P1//            DISP=(OLD,DELETE)
P1//SORTOUT  DD  DSN=&INDX..&APPLIC..MDGL0027(+1),          KEY TRANS
P1--SORTOUT  DD  DSN=FC.GLPROD.MDGL0027(+1),          KEY TRANS
P1//            DISP=(,CATLG,DELETE),
P1//            UNIT=SYSDA,
P1//            SPACE=(6160,(&SPTRN.0,&SPTRN),RLSE),
P1--            SPACE=(6160,(200,20),RLSE),
P1//            VOL=REF=&INDX..&APPLIC..&DVOL,
P1--            VOL=REF=FC.GLPROD.VOLWEEK2,
P1//            DCB=(FC.DSCB,RECFM=FB,LRECL=80,BLKSIZE=6160)
P1//SORTWK01 DD  UNIT=SYSDA,
P1//            SPACE=(6160,&SPTRN.0)
P1--            SPACE=(6160,200)
P1//SORTWK02 DD  UNIT=SYSDA,
P1//            SPACE=(6160,&SPTRN.0)
P1--            SPACE=(6160,200)
P1//SORTWK03 DD  UNIT=SYSDA,
P1//            SPACE=(6160,&SPTRN.0)
P1--            SPACE=(6160,200)
P1//PS030    EXEC  PGM=PXGLEDIT,                EDIT TRANSACTION FILE
P1//            TIME=5,REGION=468K,
P1//            PARM=&TYPE
P1--            PARM=WEEKLY
P1//STEPLIB  DD  DSN=FC.GLPROD.LOAD,              THE PRODUCTION LIB
P1//            DISP=SHR
P1//SYSUDUMP DD  SYSOUT=A                                BAD MESSAGES
P1//ERRORS   DD  SYSOUT=A                         CUSTOMER INPUT ERROR
P1//MASTIN   DD  DSN=&INDX..&APPLIC..MDGL0157(0),             (GLMSTR)
P1--MASTIN   DD  DSN=FC.GLPROD.MDGL0157(0),           (GLMSTR)
P1//            DISP=SHR
P1//MASTOUT  DD  DSN=&INDX..&APPLIC..MDGL0157(+1),       EDIT G/L MASTER
P1--MASTOUT  DD  DSN=FC.GLPROD.MDGL0157(+1),       EDIT G/L MASTER
P1//            DISP=(,CATLG,DELETE),
P1//            UNIT=TAPE,
```

```
P1//             DCB=(&INDX..DSCB,BLKSIZE=8000)
P1--             DCB=(FC.DSCB,BLKSIZE=8000)
P1//TRANS    DD  DSN=&INDX..&APPLIC..MDGL0027(&GEN3),
P1--TRANS    DD  DSN=FC.GLPROD.MDGL0027(+1),
P1//             DISP=SHR
P1//DATECARD DD  DUMMY                          PROVIDE DATE CARD 4OVERRIDE
P1$/PS030.DATECARD DD *
----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+-
070589    050217       1 1 3 NNN USRCODE=GLEDGER
P1//*
P1//PS040    EXEC  PGM=PXGLUPDT,                            UPDATE MASTER
P1//             REGION=678K,TIME=15,
P1//             PARM=&TYPE
P1--             PARM=WEEKLY
P1//STEPLIB  DD  DSN=FC.GLPROD.LOAD,                  PROD LOAD LIBRARY
P1//             DISP=SHR
P1//SYSUDUMP DD  SYSOUT=A
P1//ERRORS   DD  SYSOUT=A                          CUSTOMER DIAGNOSTIC
P1//ROLLTRIN DD  DSN=&INDX..&APPLIC..MDGL0539(0),           PEND TRANS
P1--ROLLTRIN DD  DSN=FC.GLPROD.MDGL0539(0),          PEND TRANS
P1//             DISP=SHR
P1//ROLLTOUT DD  DSN=&INDX..&APPLIC..PEND.TRANS,     NEW PENDING TRANS
P1--ROLLTOUT DD  DSN=FC.GLPROD.PEND.TRANS,     NEW PENDING TRANS
P1//             DISP=(,CATLG,DELETE),
P1//             UNIT=SYSDA,
P1//             SPACE=(6160,(&SPROLL.0,&SPROLL),RLSE),
P1--             SPACE=(6160,(5000,500),RLSE),
P1//             VOL=REF=&INDX..&APPLIC..&DVOL,
P1--             VOL=REF=FC.GLPROD.VOLWEEK2,
P1//             DCB=(&INDX..DSCB,BLKSIZE=6160)
P1--             DCB=(FC.DSCB,BLKSIZE=6160)
P1//MASTTRAN DD  DSN=&INDX..&APPLIC..MDGL0157(&GEN4),            CURRENT
P1--MASTTRAN DD  DSN=FC.GLPROD.MDGL0157(+1),             CURRENT
P1//             DISP=SHR
P1//CONTROL  DD  DSN=&INDX..&APPLIC..CNTL(G201003S),
P1--CONTROL  DD  DSN=FC.GLPROD.CNTL(G201003S),
P1//             DISP=SHR
P1//PS050    EXEC  PGM=PXGLSMRY,COND=(8,GT,PS040),        RECAP RUN ERROR
P1//             REGION=230K,
P1//             PARM=&TYPE
P1--             PARM=WEEKLY
*ADVISORY - DSS3230A - EXEC COND CODE MATCH FOR: JS010 PS050
*ADVISORY - DSS3231A - MATCHING COND CODE: 8 GT PS040
P1//STEPLIB  DD  DSN=FC.GLPROD.LOAD,                  PROD LOAD LIBRARY
P1//             DISP=SHR
P1//SYSUDUMP DD  SYSOUT=A
P1//ERRORS   DD  SYSOUT=A                          CUSTOMER DIAGNOSTIC
P1//MASTER   DD  DSN=&INDX..&APPLIC..MDGL0539(&GEN5),
P1--MASTER   DD  DSN=FC.GLPROD.MDGL0539(+1),
```

```
  P1//             DISP=SHR
  P1//REPORT1  DD  SYSOUT=(H,,6666)                      DETAILED ACTIVITY
  P1//REPORT2  DD  SYSOUT=(H,,6666)                    INTERNAL CROSS-AUDIT
  P1//DATECARD DD  DUMMY
  P1$/PS050.DATECARD DD *                                DATE CARD OVERRID
  ----+----1----+----2----+----3----+----4----+----5----+----6----+----7----+-
  070589                              TYPE 5 CONTROL
  ACCDATE=01/27/05
  AUDITDATE=17/01/2005
  P1//         PEND
//
*** ERROR SUMMARY AND COUNTS ***
    3  ADVISORY LEVEL
    1  WARNING LEVEL
    0  ERROR LEVEL
    4  TOTAL ISSUED        4 SUPPRESSED.
     THE HIGHEST SEVERITY CODE ENCOUNTERED WAS    04.
     THE  STDS  PGM  CALLED FOR THIS RUN WAS "TWSSTD1".
REL LINESV MSG.NO.    ERROR MESSAGE
---------- --------   ------------------------------------------------------
    29   4 DSS9101W -  TWS WILL WAIT FOR SPECIAL RESOURCE.                   P
    30   0 DSS9102A -  TWS RESNAME = "FC.GLPROD.INPUT.FILE"                  P
   127   0 DSS3230A - EXEC COND CODE MATCH FOR: JS010 PS050                  P
   128   0 DSS3231A - MATCHING COND CODE: 8 GT PS040                         P
XPJ60K  - END OF REPORT  JOB/SCAN      6.2.4A (C) DIVERSIFIED SOFTWARE SYSTEMS
```

D

Additional material

This book refers to additional material that you can download from the Internet as described in this appendix.

Locating the Web material

The Web material associated with this book is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

ftp://www.redbooks.ibm.com/redbooks/SG246648

Alternatively, you can go to the IBM Redbooks Web site at:

ibm.com/redbooks

Select **Additional materials** and open the directory that corresponds with the book form number, SG246648.

Using the Web material

The additional Web material that accompanies this book includes the following files:

File name	*Description*
SG246648.zip	In this zipped file there are three directories: TDS zip files directory contains zipped JCL samples and REXX programs that are used for the TDS 'EXTRA OPC Component'. There is also file called Download Notes that explains the installation. TWSPlusModule directory contains the custom code that is used in the book for TWS Plus module. ITM-Integration directory contains sample custom resource modes used in the book for monitoring TWS.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space: 10 MB minimum
Operating System: Windows/Linux

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zipped file into this folder.

Abbreviations and acronyms

ACF	Adapter Configuration Facility
APG	application group
API	Application Programing Interface
CE	Crystal Enterprise
CLIST	Command list
DM	Domain Manager
FTA	Fault tolerant agent
FTP	File transfer program
FTW	Fault tolerant workstation
GID	Group Identification Definition
GUI	Graphical user interface
HSM	Hierarchical Storage Manager
IBM	International Business Machines Corporation
IDCAMS	Access Method Services
IDCAMS	Access Method Services
IMS	Information Management System
ISPF	Interactive System Productivity Facility
ITSO	International Technical Support Organization
JCL	Job control language
JES	Job Execution Subsystem
JSC	Job Scheduling Console
JSS	Job Scheduling Services
LCF	Lightweight Client Framework
MDM	Master Domain Manager
MIB	Management Information Base
MLOG	Controller Message Log
MN	Managed Node
NJE	Network Job Entry
ODBC	Open Data Base Connectivity
ODM	Workstations Domain Policy object entry
OPC	Operations, Planning and Control
PID	Process id
PIF	Program interface
PTF	Program temporary fix
RACF	Resource Access Control Facility
RLS	Record Level Sharing
RMM	Removable Media Manager
RODM	Resource Object Data Manager
SCLM	Software Configuration Library Manager
SMF	System Management Facility
SMP	System Modification Program
SNMP	Simple Network Management protocol
SRC	Stored Response Chain
TBSM	Tivoli Business Systems Manager
TEC	Tivoli Enterprise Console
TMF	Tivoli Management Framework
TMR	Tivoli Management Region
TSLA	Tivoli Service Level Advisor
TSO	Time Sharing Option
TWS	Tivoli Workload Scheduler
WLM	Workload Manager
X-agent	Extended Agent
XCF	Cross-system coupling facility

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 694. Note that some of the documents referenced here may be available in softcopy only.

- *End-to-end scheduling with Tivoli Workload Scheduler*, SG24-6022
- *IBM Tivoli Workload Scheduler and Content Manager OnDemand to Provide Centralized Job Log Processing*, SG24-6629
- *Event Management Best Practices*, SG24-6094
- *TEC Implementation Examples*, SG24-5216
- *An Introduction to Tivoli Enterprise*, SG24-5494
- *Tivoli Enterprise Performance Tuning Guide*, SG24-5392
- *Troubleshooting Tivoli Using the Latest Features*, SG24-6614
- *Maintaining Your Tivoli Environment*, SG24-5013
- *Implementing Automated Inventory Scanning and Software Distribution After Auto Discovery*, SG24-6626
- *IBM Tivoli Monitoring Version 5.1.1 Creating Resource Models and Providers*, SG24-6900
- *IBM Tivoli Monitoring Version 5.1: Advanced Resource Monitoring*, SG24-5519
- *Troubleshooting Tivoli Using the Latest Features*, SG24-6614

Other publications

These publications are also relevant as further information sources:

- *Information Management for z/OS User’s Guide*, SC31-8756
- *Tivoli Decision Support for OS/390 Administration Guide Version 1.6*, SH19-6816

- *z/OS MVS JCL Reference*, SA22-7597
- *z/OS UNIX System Services User's Guide*, SA22-7801
- *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning Version 8,2*, SC32-1265
- *Tivoli Decision Support for OS/390 Guide to the Reporting Dialog Version 1.6*, SH19-6842
- *IBM Tivoli Workload Scheduler for z/OS Version 8.2 Installation Guide,* SC32-1264
- *IBM Tivoli Workload Scheduler for z/OS Version 8.2 Customization & Tuning*, SC32-1265
- *IBM Tivoli Workload Scheduler for z/OS Version 8.2 Managing the Workload,* SC32-1263
- *IBM Tivoli Business Systems Manager Version 3.1 Administrator's Guide,* SC32-9085
- *IBM Tivoli Business Systems Manager Version 3.1 Introducing the Consoles,* SC32-9086
- *IBM Tivoli Business Systems Manager Version 3.1 Introducing the Consoles,* SC32-9086
- *IBM Tivoli Business Systems Manager Version 3.1 Installation and Configuration Guide,* SC32-9089
- *OPC Automation Programmer's Reference and Operator's Guide*, SC33-7046
- *System Automation for z/OS Defining Automation Policy*, SC33-7039
- *IBM Tivoli Workload Scheduler Version 8.2 Release Notes (Maintenance Release April 2004),* SC32-1258
- *IBM Tivoli Workload Scheduler Version 8.2 Plus Module User's Guide Version 8.2*, SC32-1276
- *IBM Tivoli Workload Scheduler Planning and Installation Guide Version 8.2*, SC32-1273
- *IBM Tivoli Workload Scheduler Reference Guide Version 8.2,* SC32-1274
- *Tivoli Decision Support for OS/390 Guide to the Reporting Dialog Version 1.6*, SH19-6842
- *IBM Tivoli NetView for z/OS Version 5.1 Automated Operations Network Customization Guide*, SC31-8871
- *IBM Tivoli NetView for z/OS Version 5.1 Automation Guide*, SC31-8853
- *IBM Tivoli Enterprise Console User's Guide Version 3.9*, SC32-1235

- *IBM Tivoli Enterprise Console Rule Developer's Guide Version 3.9*, SC32-1234
- *IBM Tivoli Enterprise Console Rule Set Reference*, SC32-1282
- *IBM Tivoli Workload Scheduler Plus Module User's Guide Version 8.2*, SC32-1276
- *IBM Tivoli Service Level Advisor Version 2.1 Release Notes*, SC09-7777
- *Tivoli Data Warehouse Version 1.2 Release Notes*, SC32-1399
- *NetView for UNIX User's Guide for Beginners Version 7.1*, SC31-8891
- *IBM Tivoli Resource Model Builder User's Guide Version 1.1.3*, SC32-1391
- *IBM Tivoli Monitoring User's Guide Version 5.1.2,* SH19-4569
- *IBM Tivoli Monitoring Resource Model Reference Guide Version 5.1.2*, SH19-4570

Online resources

These Web sites and URLs are also relevant as further information sources:

- The documentation for IBM Tivoli Enterprise Console

  http://publib.boulder.ibm.com/tividd/td/EnterpriseConsole3.9.html
- The documentation for IBM Tivoli Management Framework

  http://publib.boulder.ibm.com/tividd/td/ManagementFramework4.1.1.html
- The documentation for IBM Tivoli Workload Scheduler

  http://publib.boulder.ibm.com/tividd/td/WorkloadScheduler8.2.html
- The documentation for IBM Tivoli Monitoring

  http://publib.boulder.ibm.com/tividd/td/Monitoring5.1.2.html
- Tivoli products end-of-support list

  http://www-306.ibm.com/software/sysmgmt/products/support/eos.html
- IBM Tivoli Management Framework 4.1.1 Fix Pack 2 URL

  ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_4.1.1/4.1.1-TMF-FP02/
- IBM Tivoli Monitoring 5.1.2 Fix Pack 3 URL

  ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_5.1.2/5.1.2-ITM-FP03/

- IBM Tivoli Monitoring Web Health Console 5.1.1 Fix Pack 3 URL

  ftp://ftp.software.ibm.com/software/tivoli_support/patches/patches_5.1.2/5.1.2-ITM-FP04/

- Microsoft Services for UNIX version 3.5 product URL

  http://www.microsoft.com/windows/sfu/

- Microsoft Windows Scripting Host service URL

  http://msdn.microsoft.com/scripting/

- Perl information site

  http://www.perl.com/

- Python information site

  http://www.python.org/

- The utility convert existing IBM Tivoli Distributed Monitoring version 3.7 monitors into resource models

  http://www-1.ibm.com/support/docview.wss?uid=swg24004683

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

IBM Support and downloads

ibm.com/support

IBM Global Services

ibm.com/services

Index

D

E

F

G

H

I

J

K

L

M

R

S

T

Integrating IBM Tivoli Workload Scheduler with Tivoli Products

(1.0” spine)
0.875”<->1.498”
460 <-> 788 pages

Integrating IBM Tivoli Workload Scheduler with Tivoli Products

Integrate Tivoli Workload Scheduler with other IBM products

Experiment with real-life scenarios

Learn the benefits of integration

This IBM Redbook explains the benefits and technical merits of integrating IBM Tivoli Workload Scheduler Distributed and IBM Tivoli Workload Scheduler for z/OS with other IBM products. Scheduling is a mission critical process for any company. However, when you talk about scheduling, you are really talking about an ecosystem. Each solution is a building block that adds value to the overall solution. With IBM Tivoli Workload Scheduler, you can collect and add data to and from each component.

This book discusses these integration points and provides detailed scenarios on how to integrate IBM Tivoli Workload Scheduler with these types of applications.

Because workload management is widely considered the nucleus of the data center, there are numerous opportunities for you to integrate IBM Tivoli Workload Scheduler with other products. This book addresses just some of these many opportunities. In terms of integration with IBM Tivoli Workload Scheduler, do not limit yourself to the products that this book discusses. Integration points discussed in this book should give you an idea of the potential value that IBM Tivoli Workload Scheduler integration can provide for your company.

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

SG24-6648-00 ISBN 0738491225