Tivoli. software

IBM Tivoli Workload Scheduler 8.2 for z/OS Scheduler's Workshop

Instructor Training Guide

S150-1749-00

Trademarks

IBM® is a registered trademark of International Business Machines Corporation.

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

AIX		GDDM
Hiperbatch	IBM	IMS
MVS	NetView	OS/390
OS/400	RACF	Tivoli
TME	VTAM	z/OS

Notes is a trademark or registered trademarks of Lotus Development Corporation and/or IBM Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

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

August, 2003 Edition

The information contained in this document has not be submitted to any formal IBM test and is distributed on an “as is” basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer’s ability to evaluate and integrate them into the customer’s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will result elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk. The original repository material for this course has been certified as being Year 2000 compliant.

Contents

Trademarks

The reader should recognize that the following terms, which appear in the content of this training document, are official trademarks of IBM or other companies:

IBM® is a registered trademark of International Business Machines Corporation.

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

AIX®		GDDM®
Hiperbatch™	IBM®	IMS™
MVS™	NetView®	OS/390®
OS/400®	RACF®	Tivoli®
TME®	VTAM®	z/OS™

Notes is a trademark or registered trademarks of Lotus Development Corporation and/or IBM Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

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

Notes is a trademark or registered trademarks of Lotus Development Corporation and/or IBM Corporation in the United States, other countries, or both.

Linux is a registered trademark of Linus Torvalds in the United States and other countries.

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

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

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

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

Instructor Course Overview

This course teaches basic TWS for z/OS-driven scheduling skills. Students use ISPF dialogs as the primary interface to the TWS for z/OS databases and plans. The TWS distributed environment and the Job Scheduling Console are mentioned and referenced throughout the course. One unit and a hands-on exercise cover how to install and use the Job Scheduling Console.

The course also introduces some of the more advanced TWS for z/OS features. Extensive hands-on exercises are used to reinforce topics covered in the units of instruction.

There are twelve units of instruction and eleven student exercises in this course. Basic TWS for z/OS scheduling concepts are introduced and skills developed in units 1 through 9 and exercises 1 through 10

Units 10, 11, and 12 cover some of the advanced scheduling features of TWS for z/OS.

This is a summary of the Units and Exercises:
Course Units:

1. Introduction and Overview
2. Workstations
3. Calendars and Periods
4. Applications - Part 1 of 4
   Applications - Part 2 of 4 - Defining Run Cycles
   Applications - Part 3 of 4 - Defining Operations
   Applications - Part 4 of 4 - Job Descriptions
5. Dynamic Feedback
6. The TWS for z/OS Plans
7. Special Resources
8. Using the Restart and Cleanup Function
9. The Job Scheduling Console (JSC)
10. Handling Unplanned Work
11. JCL Automation
12. Automatic Recovery

Exercises:

1. Creating Workstations
2. Creating Calendars and Periods
3. Creating Applications Part 1 of 3
4. Creating Applications Part 2 of 3 - Run Cycles
5. Creating Applications Part 3 of 3 - Operations
6. Creating Job Descriptions

7. Current Plan Operations
8. Creating and Using Special Resources
9. Using the Restart and Cleanup Function
10. Setting up and Using the JSC
11. Automated Job Tailoring

Refer to the instructor exercises guide for a description of the student exercises.

Course Description

Tivoli Workload Scheduler for z/OS Scheduler's Workshop

Duration: 4 days

Purpose

This course teaches how to use IBM TWS for z/OS dialogs and the Job Scheduling Console to define, implement, and manage end-to-end batch scheduling in an enterprise.

Through classroom-delivered instruction and extensive hands-on exercises, this course equips students with the skills to analyze scheduling requirements, and to create and schedule units of work for processing in z/OS (OS/390) and distributed environments.

Audience

Persons whose job responsibilities include using IBM TWS for z/OS to plan and schedule batch workload in their installation. Technical support personnel who are responsible for the implementation and support of IBM TWS for z/OS should also attend this course. This includes z/OS system programmers, technical support staff, schedulers, scheduling analysts, and scheduling administrators.

Prerequisites

Students should possess basic TSO/ISPF navigational skills, and some basic understanding of the z/OS operating environment.

Objectives

After completing this course, you should be able to:

- Install and use the Job Scheduling Console
- Use IBM TWS for z/OS dialogs or the Tivoli Job Scheduling Console to:
  - Create objects in the IBM TWS for z/OS databases to implmement scheduling policies. This includes *workstations, applications, application groups,* and *special resources*
- Use IBM TWS for z/OS dialogs to:
  - Create scheduling *calendars and periods*
  - Develop IBM TWS for z/OS long-term and current plans

- Use IBM TWS for z/OS dialogs or the Tivoli Job Scheduling Console to:
  - Control and monitor scheduled work
  - Add *unplanned work* to the current plan
- Analyze run requirements, and schedule applications and application groups
- Use the IBM TWS for z/OS *restart* and *cleanup* function for data set clean up and job restarts
- Use *special resources* to manage the use of limited resources and control the start of selected work
- Use IBM TWS for z/OS TSO authorized commands such as OPSTAT and SRSTAT to improve and control workflow
- *Use Event Triggered Tracking* and Dataset Triggering to handle unplanned work
- Describe IBM TWS for z/OS automated *Job Tailoring* features which include:
  - *Automatic editing* of JCL, scripts and command files
  - The use of *date arithemetic* for date calculations in jobs
  - IBM TWS for z/OS JCL variables
- Describe the IBM TWS for z/OS Automatic Recovery function.

Agenda

Day 1

(00.30) Welcome and Administration
(02.30) Unit 1 - Introduction and Overview
(01.10) Unit 2 - Workstations
(00.40) Exercise 1: Creating Workstations
(00.30) Unit 3 - Calendars and Periods
(00.30) Exercise 2: Creating Calendars and Periods

Day 2

(01.15) Review of Topics and Exercises from day 1
(00.55) Unit 4 - Applications and Job Descriptions - Topic 1: General Information
(00.10) Exercise 3: Creating Applications Part 1 of 3
(01.25) Unit 4 - Applications and Job Descriptions - Topic 2: Defining Run Cycles
(00.20) Exercise 4: Creating Applications Part 2 of 3 -Run Cycles
(00.45) Unit 4 - Applications and Job Descriptions - Topic 3: Defining Operations
(01.00) Exercise 5: Creating Applications Part 3 of 3 -Operations
(00.15) Unit 4 - Applications and Job Descriptions - Topic 4: Job Descriptions
(00.10) Exercise 6:Creating Job Descriptions
(00.20) Unit 5 - Dynamic Feedback
(01.50) Unit 6 - The TWS for z/OS Plans

Day 3

(01.00) Review of Topics and Exercises from day 2
(01.30) Exercise 7: Current Plan Operations
(01.00) Unit 7 - The Job Scheduling Console (JSC)
(00.30) Exercise 8: Installing and Using the JSC
(00.30) Unit 8 - Using the Restart and Cleanup Function
(00.30) Exercise 9: Using the Restart and Cleanup Function

Day 4

(01.00) Review of Topics and Exercises from day 3
(00.35) Unit 9 - Special Resources
(01.00) Exercise 10: Creating and Using Special Resources
(01.00) Unit 10 - Handling Unplanned Work

Unit 1. Introduction and Overview

What This Unit is About

This unit introduces the terminology, components, functions, and features of IBM Tivoli Workload Scheduler (TWS) for z/OS.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe the basic functions of IBM TWS for z/OS
- List the main IBM TWS for z/OS databases
- List the names of the IBM TWS for z/OS plans
- Describe the purpose of the IBM TWS for z/OS plans

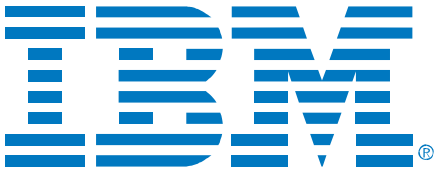
How You Will Check Your Progress

Accountability:

- Checkpoint questions

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 1: Introduction

Tivoli software

Figure 1-1. Introduction TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- Describe the basic functions of TWS for z/OS
- List the main TWS for z/OS databases
- List the names of the TWS for z/OS plans
- Describe the purpose of the TWS for z/OS plans

Tivoli software

Figure 1-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

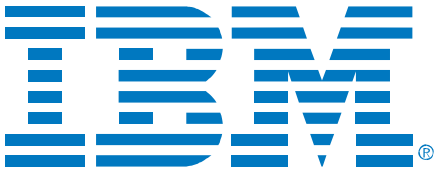
Purpose — To list the objectives of this unit.

Details — Mention that this unit plays a double role. It introduces TWS for z/OS, and some of its functionality and terminology. It also provides a very brief excursion into some of what will be covered in the course, and introduces some TWS for z/OS terminology. Items mentioned in this unit will be covered in greater detail in other units.

Additional Information —

Transition Statement —

1.1 What Is TWS for z/OS



Topic 1.1

What Is TWS for z/OS?

Tivoli software

Figure 1-3. What is TWS for z/OS TM402.0

Notes:

Instructor Notes:

Purpose — Introduce the tool, give a brief history, identify what it does, identify it's companion components in the suite of scheduling products, and introduce some terminology.

Details —

Additional Information —

Transition Statement — The Tivoli Workload Scheduler (TWS) suite is made up of these products:

- TWS for z/OS
- TWS
- The Tivoli Job Scheduling Console (JSC)

IBM Tivoli Workload Scheduler Suite

- IBM Tivoli Workload Scheduler for z/OS
- IBM Tivoli Workload Scheduler
- IBM Tivoli Job Scheduling Console

Figure 1-4. IBM Tivoli Workload Scheduler Suite TM402.0

Notes:

The IBM Tivoli Workload Scheduler (TWS) suite is made up of these products:

- IBM Tivoli Workload Scheduler for z/OS (TWS for z/OS) - This component of the suite is used to schedule batch work on z/OS systems and manage the enterprise workload.
- IBM Tivoli Workload Scheduler (TWS) - This component is used to schedule and manage work on distributed, non-z/OS systems.
- IBM Tivoli Job Scheduling Console (JSC) - This is a Java-based graphical user interface to objects and plans managed by TWS for z/OS, and TWS.
- Using TWS for z/OS, a user can define workload management policies in a series of databases, then use these policies to plan and control batch workload for the enterprise.

Instructor Notes:

Purpose — Define Tivoli Workload Scheduler for z/OS, identify its companion products, and what they are used to do.

Details —

- TWS for z/OS - This is used to schedule batch work on z/OS systems and manage the enterprise workload.
- TWS - This product can be used to schedule and manage work on distributed, non-z/OS systems.
- The Tivoli Job Scheduling Console (JSC) - This provides a common graphical user interface to Tivoli Workload Scheduler for z/OS (TWS for z/OS), and Tivoli Workload Scheduler (TWS) functions.

Additional Information —

Transition Statement — Tivoli Workload Scheduler for z/OS plans your production workload schedule by producing both high-level and detailed plans for all batch work in your enterprise.

Enterprise Workload Management

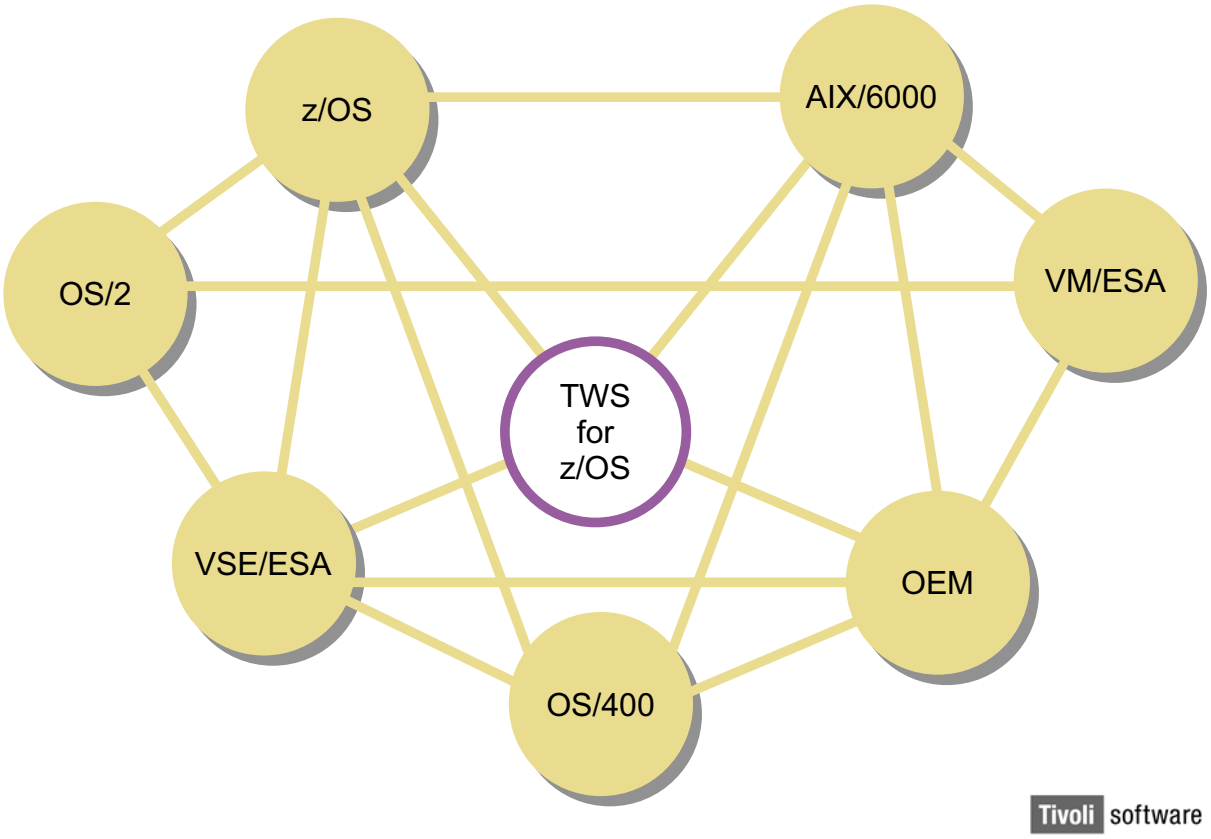


Figure 1-5. Enterprise Workload Management TM402.0

Notes:

In addition to managing work on z/OS systems, TWS for z/OS is capable of end-to-end scheduling. Through its application programming interface (API), TWS for z/OS can communicate with, and provide data to application programs running on compliant platforms.

TWS for z/OS supports tracker agents for scheduling on distributed systems which include platform systems such as OS/400, UNIX, AIX, Windows NT, DEC, Sun/OS, and OS/2.

A strong feature of this tool is its ability to interface with fault tolerant agents (FTA) which can be used to bridge a TWS domain of distributed systems to a TWS z/OS system.

Instructor Notes:

Purpose — To show the power and platform unification capabilities of TWS for z/OS.

Details —

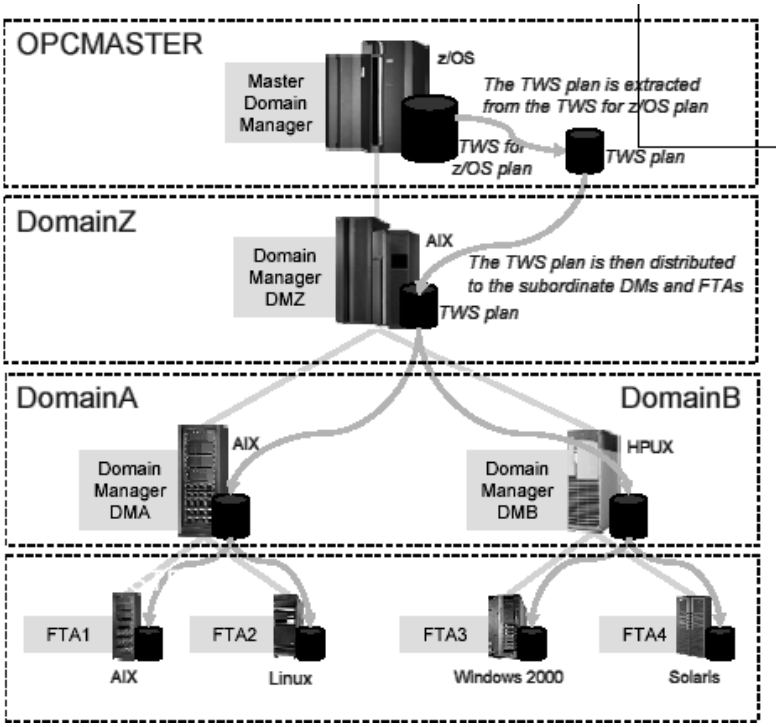
- Mention in its former life as OPC/ESA, the product managed workload on multiple operating platforms using trackers and tracker agents.
- Today TWS for z/OS manages the workload much more efficiently using a function called End-to-end Scheduling to schedule work in the distributed environment.

Additional Information — Customers can no longer order tracker agent software.

Transition Statement —

- A strong feature of this tool is its ability to interface with fault-tolerant agents (FTA) which are used in the distributed systems domain.

The End-to-End Solution: Configuration



Tivoli software

Figure 1-6. The End-to-End Solution: Configuration TM402.0

Notes:

End-to-end scheduling is implemented by directly connecting a TWS domain manager to TWS for z/OS. The domain manager functions as the broker system for the entire distributed network by resolving all its dependencies. TWS for z/OS handles its own jobs and notifies the domain manager of all the status changes of the TWS for z/OS jobs that involve the TWS plan.

The domain manager sends its updates (in the form of events) to the master domain manager (the TWS for z/OS controller). The TWS for z/OS controller uses the TWS events to update the current plan.

In this configuration the domain manager, and all the distributed agents, recognize TWS for z/OS as the master domain manager and notify it of all the changes occurring in their own plan.

The TWS plan (Symphony) file is created by TWS for z/OS when the new current plan created.

Instructor Notes:

Purpose — To introduce the End-to-End Scheduling concept.

Details — Mention some of the advantages of using E2E.

- Less overhead on the controller
- Fault tolerance
- Scalability of the distributed environment to almost infinity.

Additional Information —

Transition Statement — The next visual lists some distributed environment terminology.

Tivoli Workload Scheduler for z/OS Provides

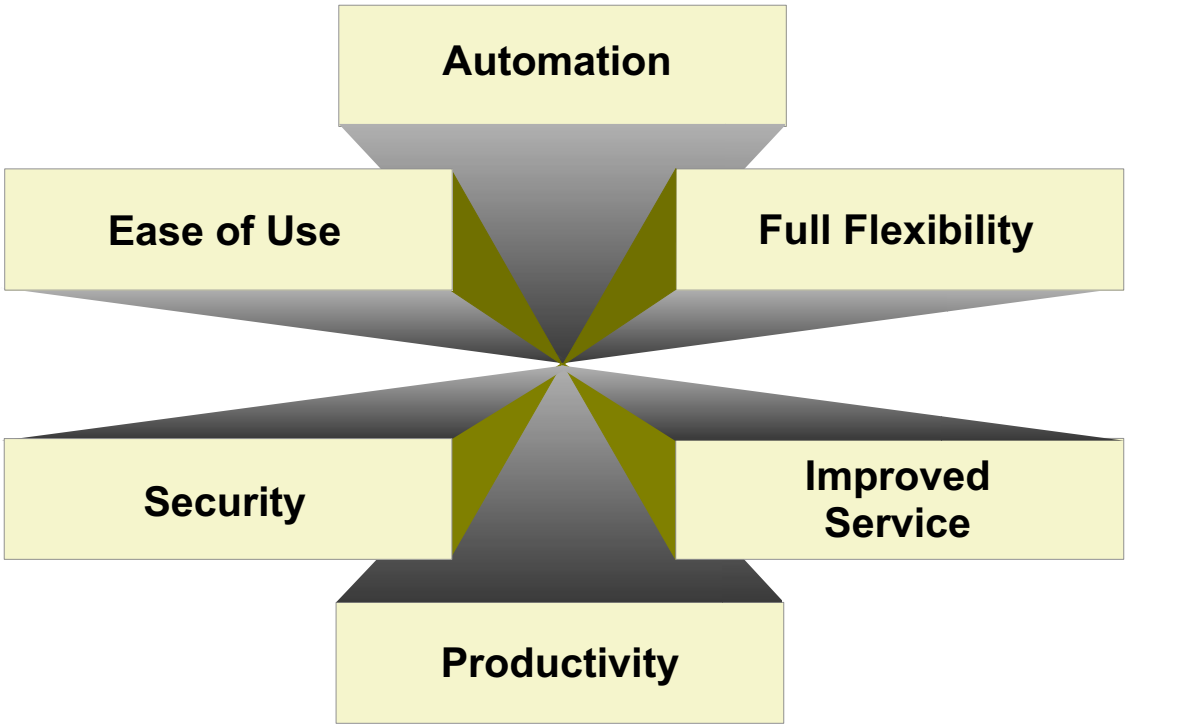


Figure 1-7. Tivoli Workload Scheduler for z/OS Provides TM402.0

Notes:

TWS for z/OS provides Automation, Productivity, Improved Service, Ease of Use, and Flexibility of in the scheduling environment by using defined policies to:

- Prepare long-range and daily plans of work to be run.
- Schedule the work, taking into account job priorities, needed resources and their availability, any user-specified start and end times, and dependencies between jobs.
- Automatically submit work according to the schedule.
- Automatically report events and provide current status information.
- Allow dynamic rescheduling and generate trial plans.
- Provide automatic restart and recovery functions.
- Allow manual intervention and control when necessary.

Instructor Notes:

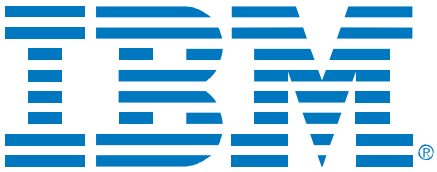
Purpose — To highlight the key values provided by TWS for z/OS.

Details —

Additional Information —

Transition Statement — Many components working together give TWS for z/OS its values. The next topic introduces those components and give a brief glimpse at how they are connected.

1.2 Basic Components and Connections



Topic 1.2

Basic Components and Connections

Tivoli software

Figure 1-8. Basic Components and Connections TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

TWS for z/OS Components

- Base component
  - Tracker
- Features
  - Controller
  - Non-MVS tracker agents
  - Fault-tolerant agents
  - Data store
  - APPC/MVS protocol server
  - TCP/IP protocol server

Tivoli software

Figure 1-9. TWS for z/OS Components TM402.0

Notes:

Tivoli Workload Scheduler for z/OS consists of a base product, the tracker, and a number of features.

A tracker started task must be installed on every z/OS system on which TWS for z/OS is being used to monitor and control production workload. The tracker's primary function is the collection of job-tracking events. However, it can be used to submit jobs and start started tasks.

The *controller* feature controller manages the TWS for z/OS dialogs, the TWS for z/OS databases and plans, and can communicate with up to 16 trackers. Only one active controller is required in a TWS for z/OS configuration.

Other features include:

- A variety of non-z/OS *tracker agents* - used to manage production workload on non-z/OS distributed platforms. Tracker agents perform duties similar to those of a tracker. Although the existing OPC tracker agents are supported in TWS for z/OS 8.1, they cannot be ordered as TWS for z/OS features.

- *End-to-end scheduling* which enables the control of work on distributed platforms from a TWS for z/OS controller. This requires the use of *fault tolerant agents (FTA)*.
- The *Data Store* - Restart and Cleanup function.
- The *APPC/MVS Server* - Allows access to controller functions from a controlled system.
- The *TCP/IP Server* - This is an intermediate component that connects the TWS for z/OS controller to the TWS domain, and the Job Scheduling Console.

Instructor Notes:

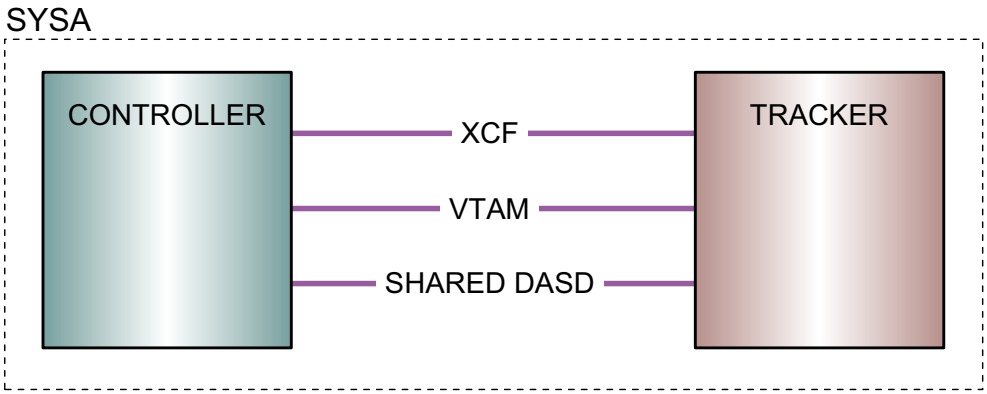
Purpose — To introduce the many components of TWS for z/OS.

Details — See student notes.

Additional Information —

Transition Statement — A tracker must transmit job execution and completion information to its controller so the next visual shows the methods used to connect the components.

Controller-to-Tracker Connections



Tivoli software

Figure 1-10. Controller to Tracker Connections TM402.0

Notes:

TWS for z/OS controllers and trackers can be connected by any one of these methods:

- Shared DASD
- XCF (Cross-system Coupling Facility) communication links
- VTAM link

Connection between the controller and z/OS trackers is required for the transmission of work and the return of job-tracking events.

It is recommended that you use either XCF- or VTAM- connected systems to improve performance. Due to the high volume of I/O activity involved with Shared DASD connections, it is considered the worst performer of the three methods.

Instructor Notes:

Purpose — Show the three types of tracker-controller connections.

Details — See student notes.

Additional Information —

Transition Statement —

Controller-to-Tracker Agent Connections

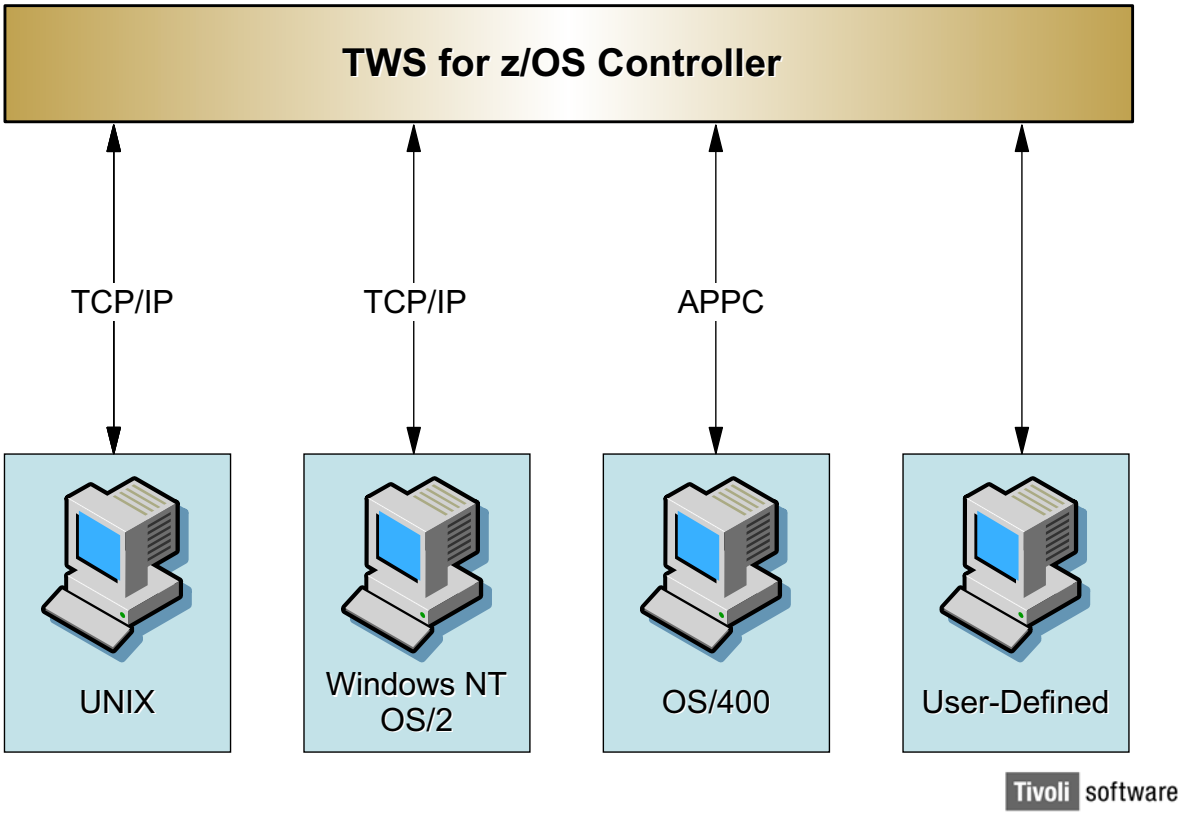


Figure 1-11. Controller-to-Tracker Agent Connections TM402.0

Notes:

TCP/IP is the protocol used to connect all non-OS/400 tracker agents to the TWS for z/OS controller.

Advanced Program-to-Program Communication (APPC) is used for the IBM OS/400 tracker agent.

Users can also create their own tracker agent for other operating systems using the TWS for z/OS open interface.

Instructor Notes:

Purpose — Introduce controller to tracker-agent connections.

Details — See student notes.

Additional Information —

Transition Statement —

Controller to Fault-Tolerant Agents

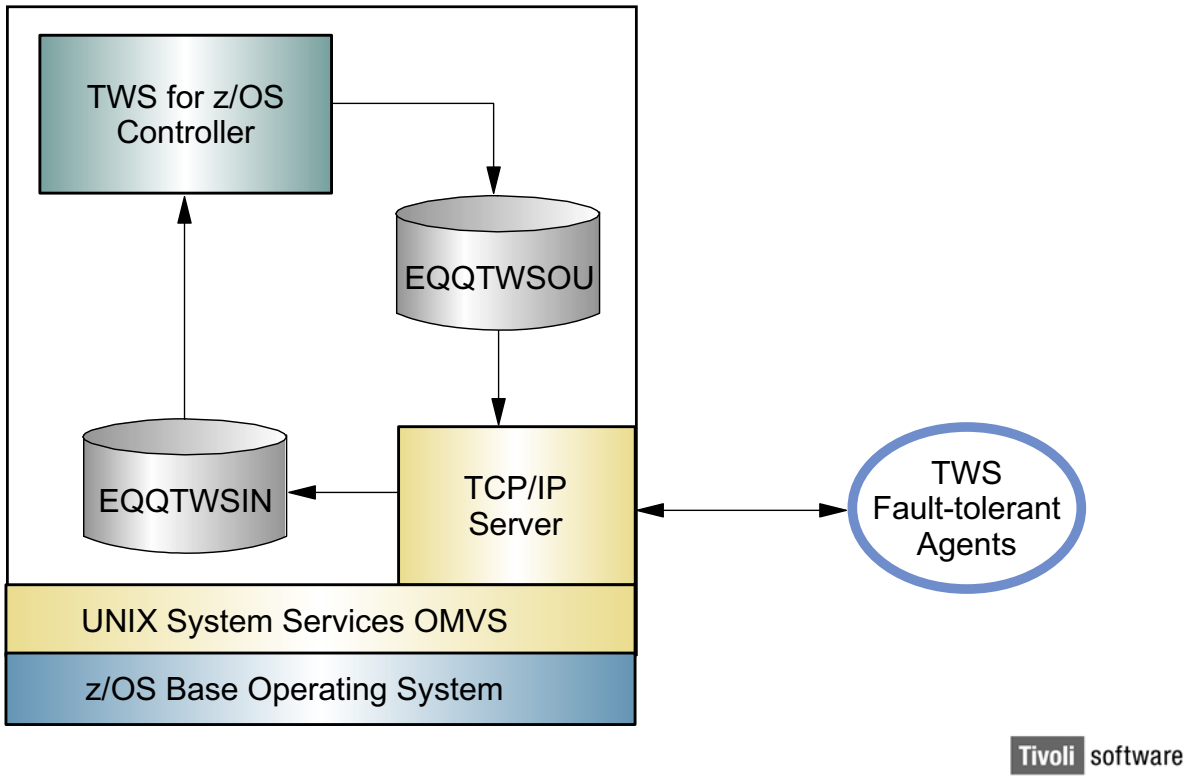


Figure 1-12. Controller to Fault-Tolerant Agents TM402.0

Notes:

On the TWS for z/OS controlling system, UNIX System Services and TCP/IP must be installed and operational. A TWS for z/OS TCP/IP server must be started for end-to-end scheduling.

The controller uses the server to communicate events to the distributed agents. The server will start multiple tasks and processes using the UNIX System Services.

The tasks started by the server perform a number of functions which include queuing and translating events to and from the controller. The **EQQTWSIN** data set is used to queue distributed agent events sent by the server to the controller while the **EQQTWSOU** data set is used for events going in the opposite direction.

Instructor Notes:

Purpose — Show how fault-tolerant agents in a distributed environment communicate with the controller.

Details — See student notes.

Additional Information —

Transition Statement — The next visual shows a simple TWS domain.

Distributed Environment - TWS Terminology

- Domain
- Domain Manager (DM)
- Fault-tolerant Agent (FTA)
- Standard Agent (SA)
- Extended Agent (XA)
- Backup Domain Manager

Tivoli software

Figure 1-13. Distributed Environment - TWS Terminology TM402.0

Notes:

- **Domain** A domain is a named group of TWS workstations consisting of one or more workstations and a domain manager acting as the management hub.
- **Domain Manager** The management hub in a domain. All communications to and from the agents in a domain are routed through the domain manager.
- **Fault-tolerant Agent (FTA)** A workstation capable of resolving local dependencies and launching its jobs in the absence of a domain manager.
- **Standard Agent** A workstation that launches jobs only under the direction of its domain manager.
- **Extended Agent** A logical workstation definition that enables the launch and control jobs on other systems and applications, such as Baan, Peoplesoft, Oracle Applications, SAP, and MVS JES2 and JES3.
- **Backup Domain Manager** A fault-tolerant agent capable of assuming the responsibilities of its domain manager.

Instructor Notes:

Purpose — Introduce TWS terminology.

Details — In an E2E environment, the TWS for z/OS controller is the master domain manager. Currently, only one TWS FTA can connect to the master domain manager

Additional Information —

Transition Statement — TWS for z/OS offers many values. The key values are shown in the next visual.

TWS Domain Example

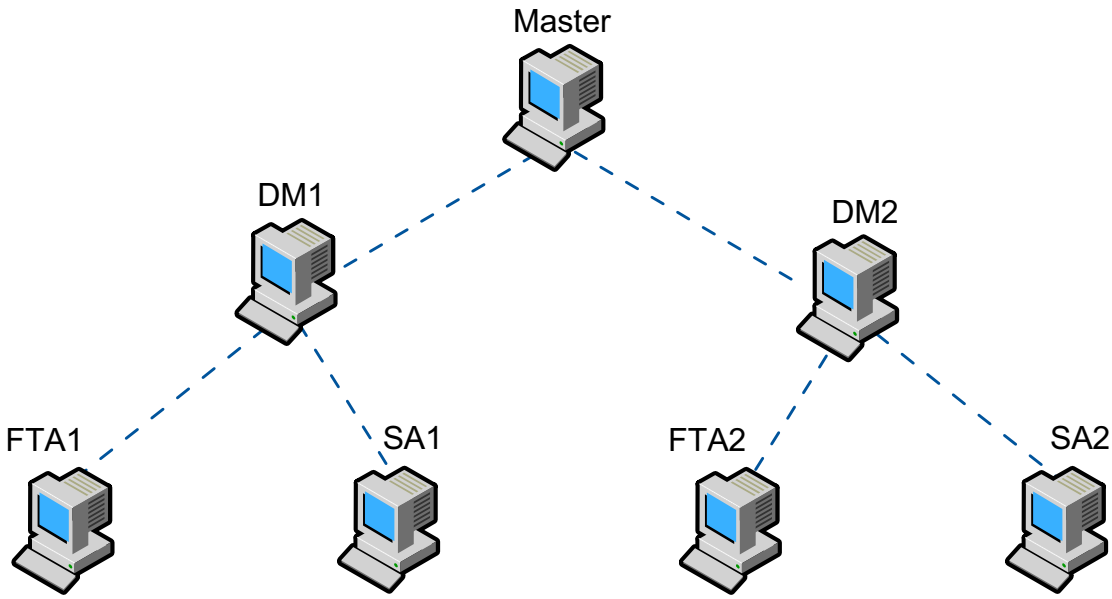


Figure 1-14. TWS Domain Example TM402.0

Notes:

A TWS network contains at least one domain, the ***master domain***, in which the master domain manager is the management hub. Additional domains can be used to divide a widely distributed network into smaller, locally managed groups.

In a single domain configuration, the master domain manager maintains communications with all of the workstations in the TWS network.

In a multidomain configuration, the master domain manager communicates with the workstations in its domain, and subordinate domain managers.

The subordinate domain managers, in turn, communicate with the workstations in their domains and subordinate domain managers.

Using multiple domains reduces the amount of network traffic by reducing the communications between the master domain manager and other computers. Multiple domains also provide fault-tolerance by limiting the problems caused by losing a domain manager to a single domain. To limit the effects further, backup domain managers can be designated to take over if their domain managers fail.

Instructor Notes:

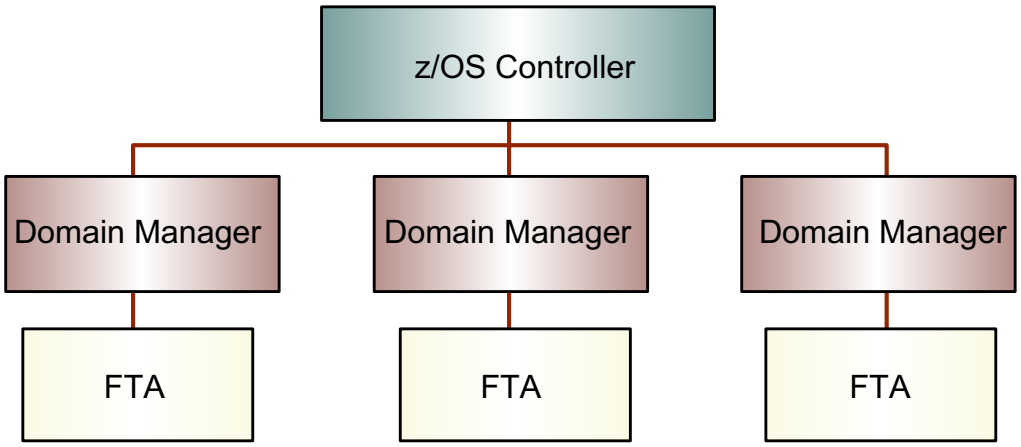
Purpose — Show an example of a TWS domain and the advantage of using fault-tolerant agents.

Details — See student notes.

Additional Information —

Transition Statement —

Controller to Multiple Domain Manager Connection



Tivoli software

Figure 1-15. Controller to Multiple Domain Manager Connection TM402.0

Notes:

Multiple domain managers can be connected to the controller, allowing greater flexibility, scalability and performance.

This type of configuration requires TWS for z/OS version 8.2 or higher to be installed on the master domain manager.

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Data Store Connections

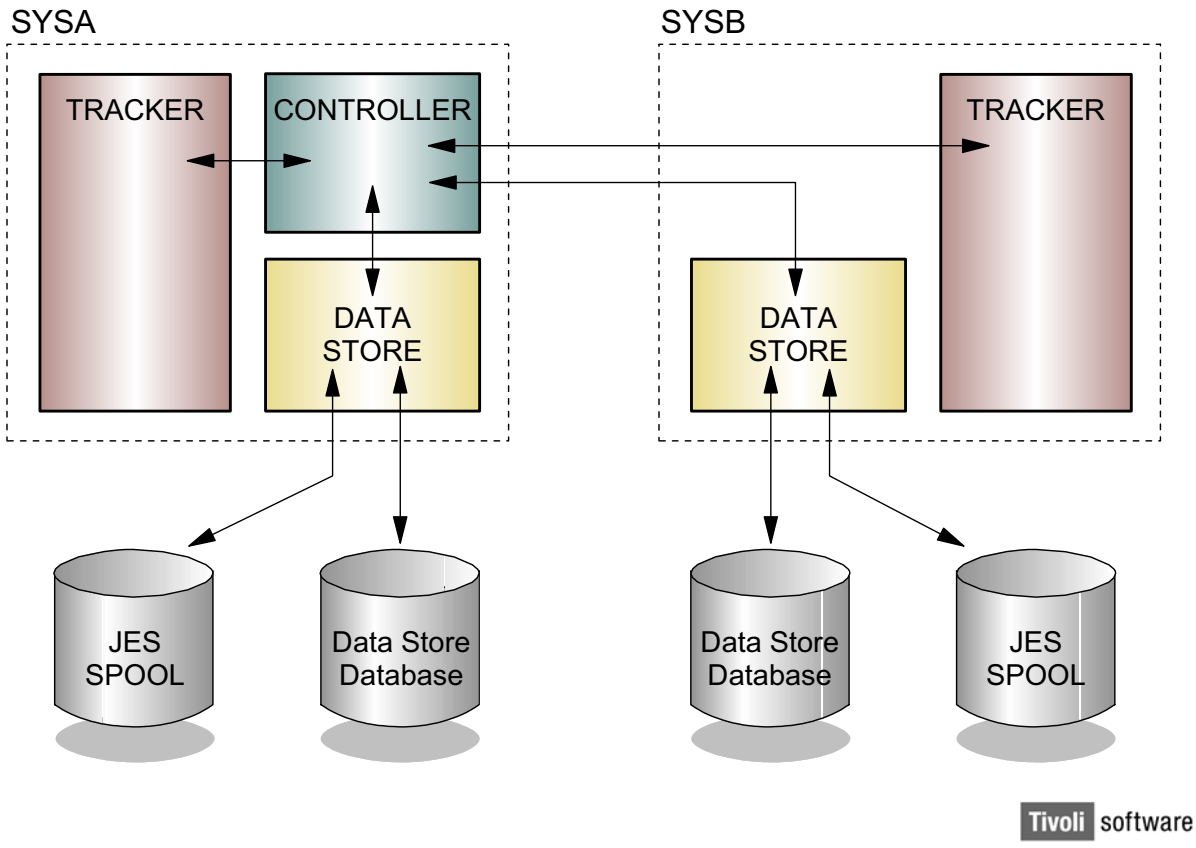


Figure 1-16. Data Store Connections TM402.0

Notes:

Data store is a separate address space. Its function is to collect structured (steps and datasets) and, optionally, unstructured (SYSOUT) information for all submitted jobs.

Data store is required in order to use the restart and cleanup functions which include:

- Restart at the job or step level
- Data set clean up
- JOBLOG retrieval

The controller can be connected to data store via **XCF or VTAM/SNA only**.

Instructor Notes:

Purpose — Introduce data store and the part it plays in job restarts and data set cleanup.

Details — See student notes.

Additional Information —

Transition Statement —

APPC Server Connections

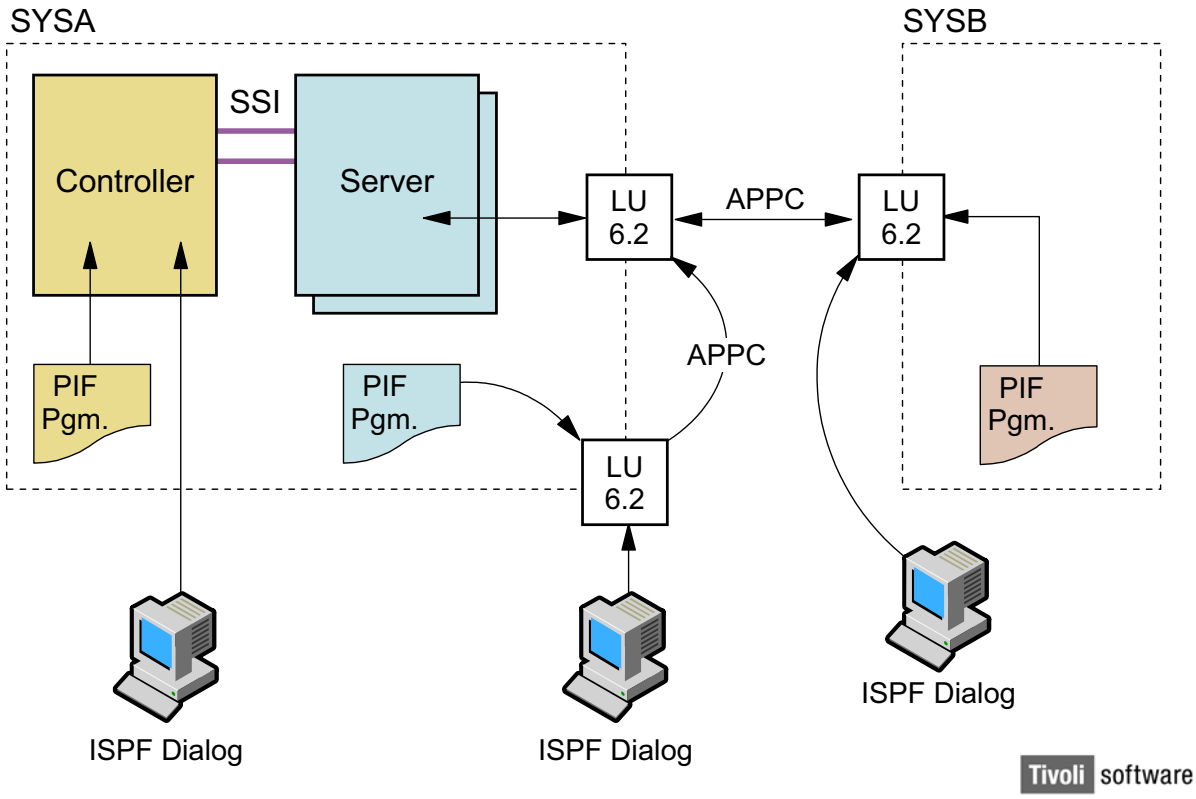


Figure 1-17. APPC Server Connections TM402.0

Notes:

A TWS for z/OS APPC/MVS server enables access to controller functions from any z/OS system connected to the controlling system. The server runs in its own address space.

This access is available to both dialog users and programs using the TWS for z/OS program interface (PIF).

Connections to the server use Advanced Program-to-Program Communication (APPC).

Instructor Notes:

Purpose — Show how dialog users connect to the controller from a controlled system or a non-TWS for z/OS system.

Details —See student notes.

Additional Information —

Transition Statement —

Job Scheduling Console Connections

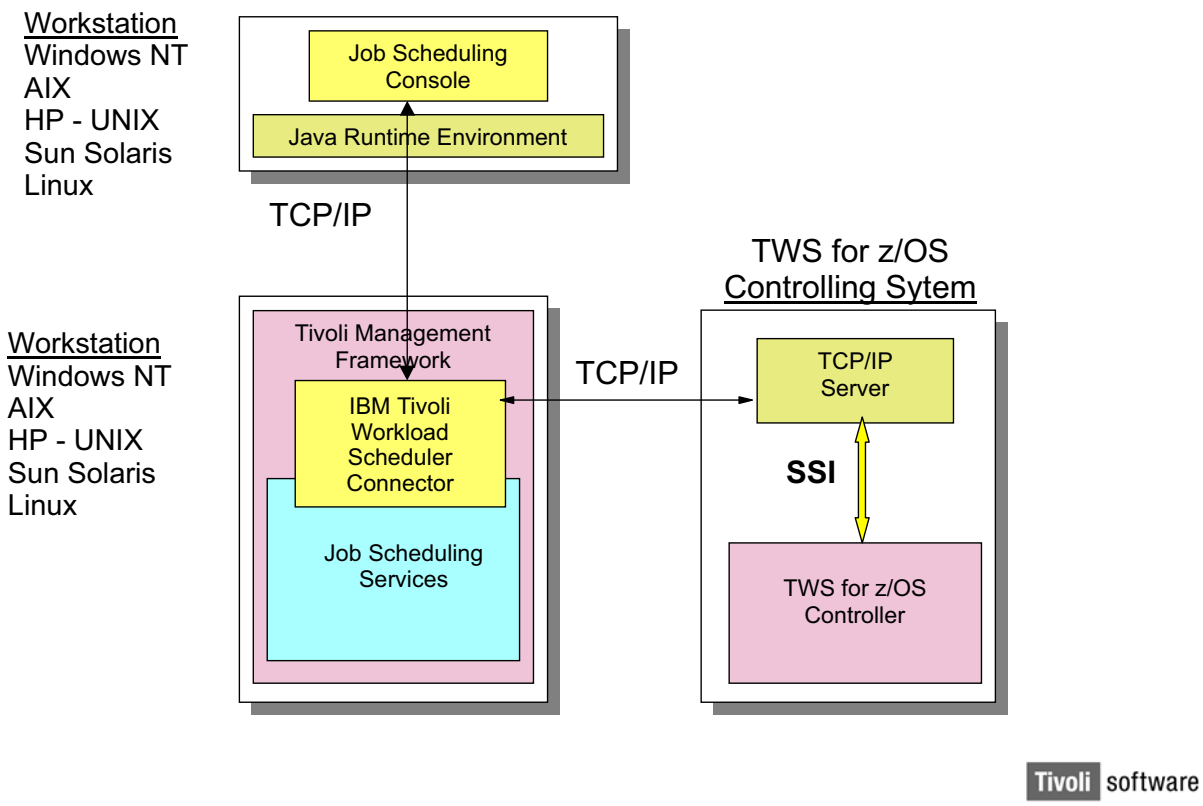


Figure 1-18. Job Scheduling Console Connections TM402.0

Notes:

The Job Scheduling Console (JSC) connects to an intermediary component, the JSC Connector through TCP/IP.

The JSC Connector manages the traffic between the Job Scheduling Console and the TCP/IP server on the TWS for z/OS controlling system. The Connector must be installed on either a Tivoli Managed Region (TMR) server or a Tivoli managed node; and it connects to the TCP/IP server via TCP/IP.

The TCP/IP Server routes requests from the JSC user to the TWS for z/OS controller via the z/OS Subsystem Interface (SSI).

Instructor Notes:

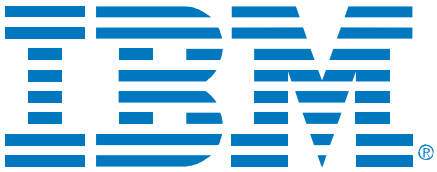
Purpose — Introduce the graphic user interface to TWS for z/OS

Details —

Additional Information —

Transition Statement —

1.3 Functional Overview



Topic 1.3

Functional Overview

Tivoli software

Figure 1-19. Functional Overview TM402.0

Notes:

Instructor Notes:

Purpose — Give a big picture of what the scheduler does and how to accomplish the workload management tasks.

Details —

Additional Information —

Transition Statement —

Major Functions of TWS for z/OS

- Policy management
  - Define, display, and modify policies
- Workload planning
  - Jobs, subsystems, and manual activities
- Monitoring and control
  - Drives workload according to policies
  - Automatic modification of input streams
  - Dialog for manual activities
  - Maintains current plan with detailed work unit information
  - ISPF dialogs
  - Job scheduling console

Tivoli software

Figure 1-20. Major Functions of TWS for z/OS TM402.0

Notes:

The major business functions of TWS for z/OS can be separated into three distinct categories:

- Policy management: This allows users to define business policies in a series of databases.
- Workload planning: This includes long-term and daily planning functions for all activities and processes in the production workload day. It makes proactive management of the scheduled workload possible.
- Monitoring and control: This function drives the workload according to the predefined policies while maintaining flexibility to allow the running of unscheduled work. Monitoring and control is done both automatically by TWS for z/OS, and by end users through ISPF dialogs and/or and or the Job Scheduling Console.

Instructor Notes:

Purpose — Identify the major functions of TWS for z/OS.

Details —

Additional Information —

Transition Statement — Let us take a more detailed look at each of the major functions.

Policy Management and Workload Planning

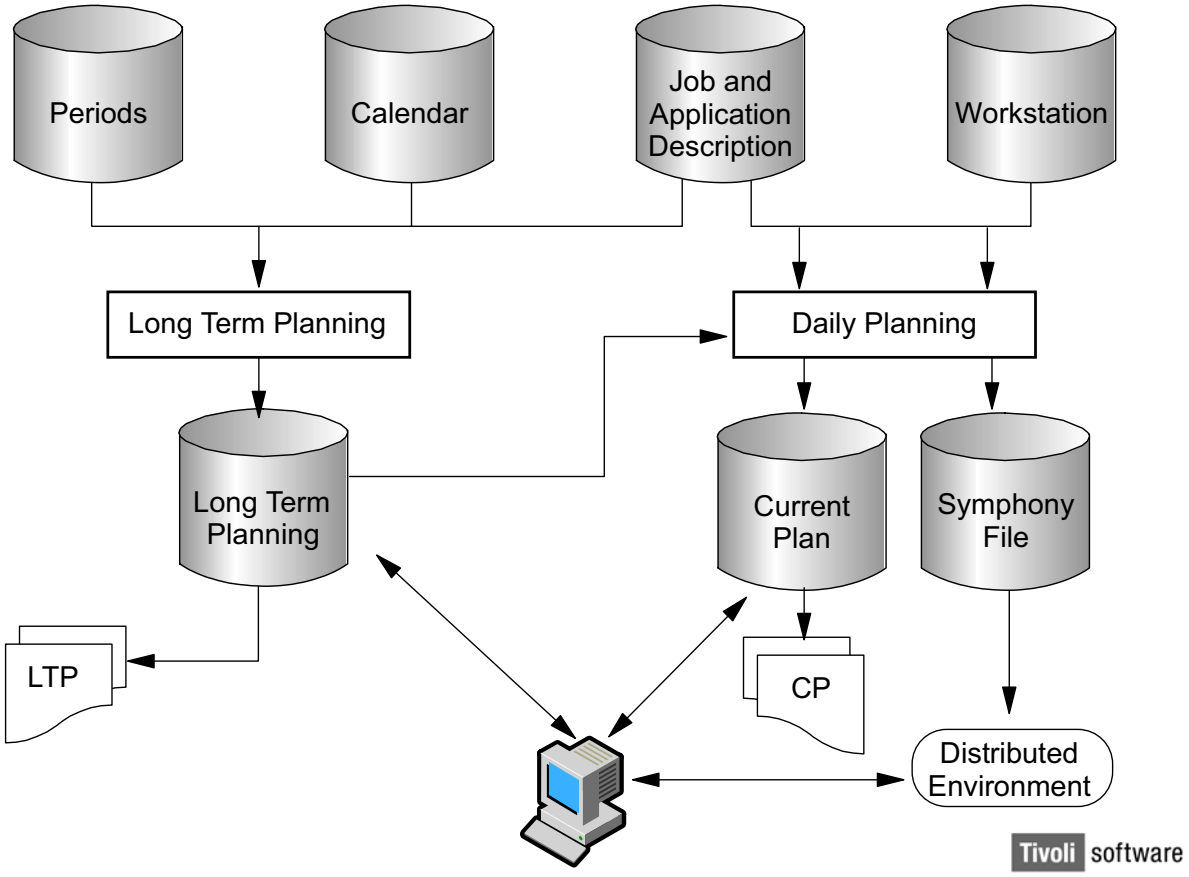


Figure 1-21. Policy Management and Workload Planning TM402.0

Notes:

Users enter their business policies into TWS for z/OS databases. This is known as modeling the data center or describing the environment.

The workstation database holds descriptions of logical places used for production tasks which include automated processing and activities related to the automated processing.

The calendar database contains user-defined TWS for z/OS calendars. A TWS for z/OS calendar is used to identify normal business days known as work days, and non-business days known as free days.

The periods database is used to hold templates of user-defined business processing cycles. These templates are used in conjunction with calendars to generate run-dates for units of work know as applications.

TWS and TWS for z/OS units of work – applications/job streams are defined in the applications database. An application is a description of a unit of production work. It can contain up to 255 related activities at a variety of workstations, and run policies (scheduling cycles) for the work.

The TWS for z/OS planning functions are batch processes which use input from the policy databases to construct operating plans. The operating plans known as the long-term plan (LTP), the current plan (CP), and the symphony file (optional) are key-sequenced VSAM files. In these files, each record represents an instance of an application and is called an occurrence. The occurrences are keyed by application name, run date, and a user-specified time.

The contents of the LTP and CP can be viewed online. The symphony file is an optional file which you generate in an end-to-end scheduling environment.

Instructor Notes:

Purpose — Show the three-step process used to build the schedule.

Details —

1. Describe the environment by creating objects in the databases.
2. Generate a long-term plan.
3. Create a current plan.

Additional Information —

Transition Statement —

Long-Term Planning Overview

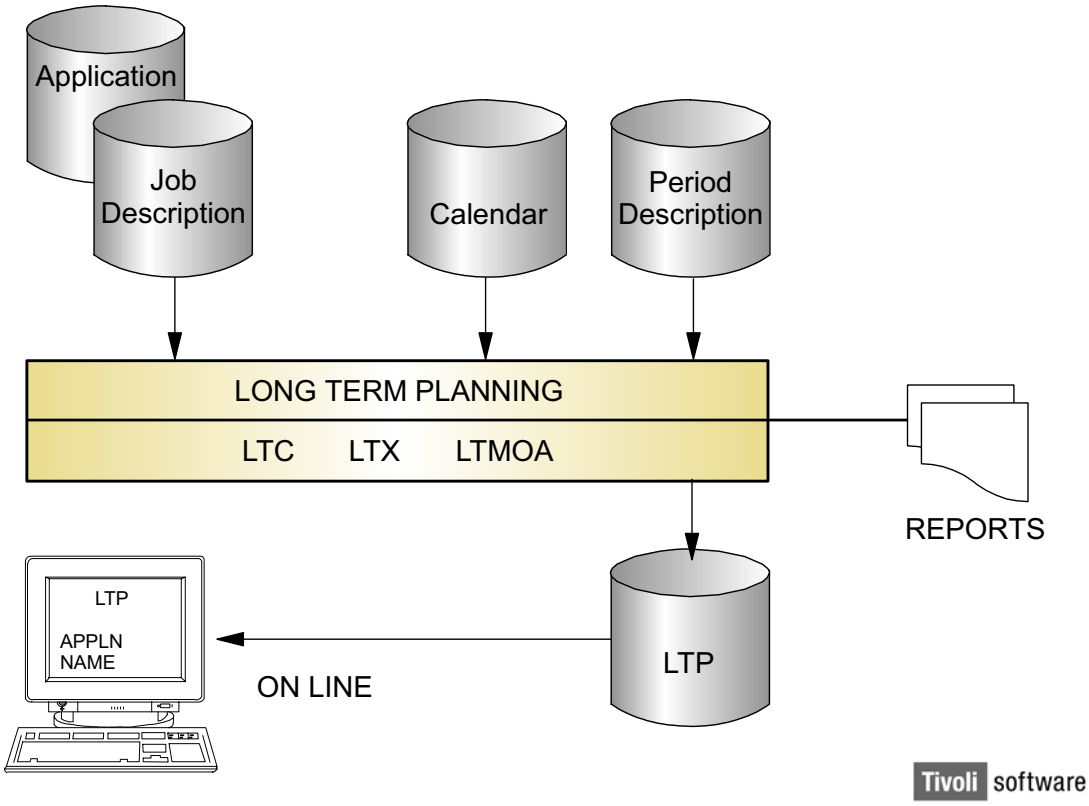


Figure 1-22. Long Term Planning Overview TM402.0

Notes:

The LTP provides a high-level view of the scheduled workload. The LTP file EQQLTDS lists scheduled occurrences chronologically and can contain up to four years of anticipated work. It also lists any dependencies that exist between occurrences.

The long-term planning functions use information from the application, calendar, and period databases to generate occurrences.

The three long-term planning batch functions are:

1. Create: Creates a new LTP. Typically run when setting up TWS for z/OS the first time or after a long-term plan refresh.
2. Extend: Extends the existing LTP to include additional days in the plan and picks up any scheduling changes made in the databases.
3. Modify: You use this function to modify, add, and delete occurrences in the existing plan after making scheduling changes in the databases. This function does not change the end date of the long-term plan.

Instructor Notes:

Purpose — High-level view of the long-term planning batch functions

Details — See student notes.

Additional Information —

Transition Statement — The primary function of the long-term plan is to provide input for the daily planning process.

Daily Planning Overview

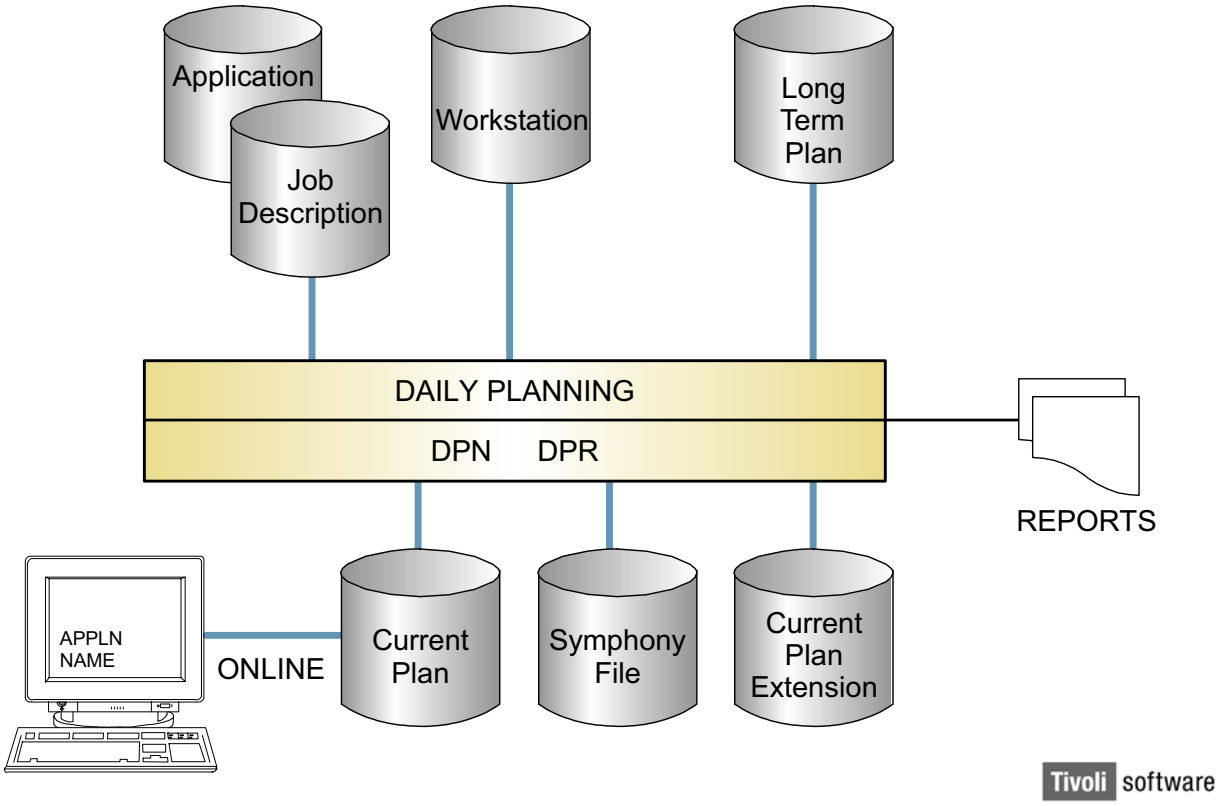


Figure 1-23. Daily Planning Overview TM402.0

Notes:

The current plan is the heart of TWS for z/OS processing. The current plan is produced by the daily planning batch function -- daily plan next (DPN).

The DPN function copies LTP occurrences that fall within a user-specified time period into the current plan. The minimum extension period. The minimum time allowed is one minute while the maximum is 21 days. The CP files contain a detailed view of the workload for the specified window of time.

The second daily planning batch function -- daily plan replan (DPR) allows users to delete completed occurrences from the existing current plan, and update the long-term plan. In the long-term plan file, the batch process updates the records of the completed occurrences.

Instructor Notes:

Purpose — Show the batch functions of the daily planning process.

Details — See student notes.

Additional Information —

Transition Statement — We will now look at the next major TWS for z/OS functions.

Monitoring and Controlling the Workload

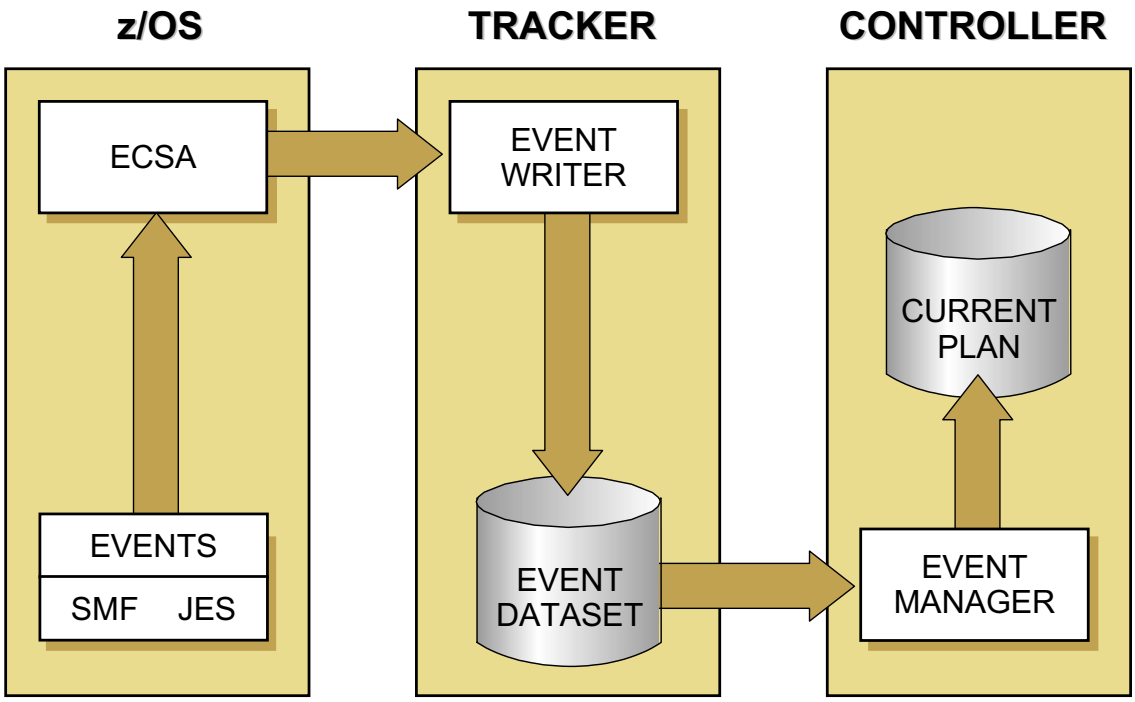


Figure 1-24. Monitoring and Controlling the Workload TM402.0

Notes:

Tivoli Workload Scheduler for z/OS automatically drives the production workload by monitoring the flow of work; and by directing the processing of jobs so that it follows the defined business priorities in the current plan.

One TWS for z/OS component, the *tracker* retrieves JES and SMF information from the *ECSA* (Extended Common Service Area) of the z/OS system. The *event writer*, a tracker subtask writes the retrieved information known as *job-tracking* events to the event data set.

The events are sent to or retrieved by the second TWS for z/OS component, the *controller*. A controller subtask, the *event manager* uses the information to effect status updates of work in the current plan.

Instructor Notes:

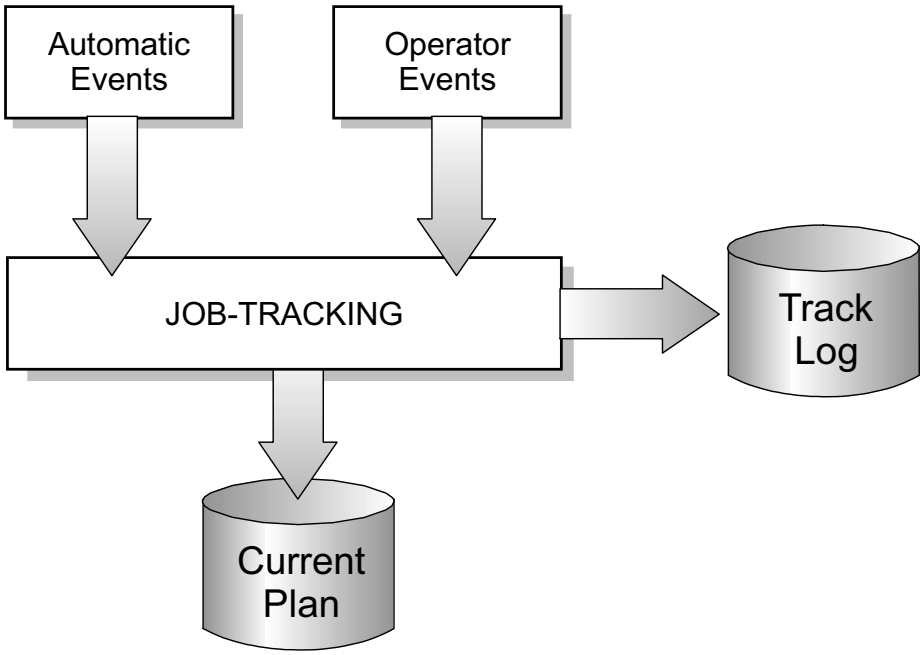
Purpose — Introduce job-tracking.

Details — See student notes.

Additional Information —

Transition Statement —

Job-Tracking



Tivoli software

Figure 1-25. Job-Tracking TM402.0

Notes:

The *job-tracking* function is vital to workload monitoring and control of work. It is used to update the status of operations in the current plan.

Tivoli Workload Scheduler for z/OS uses *event records* to track the workload. Most event records are generated automatically as a job progresses through the system, but they can also be generated manually.

For TWS for z/OS recovery, and for reporting purposes, job-tracking saves all events on job-tracking logs.

Instructor Notes:

Purpose — Define job-tracking.

Details — See student notes.

Additional Information —

Transition Statement —

Functional Hierarchy

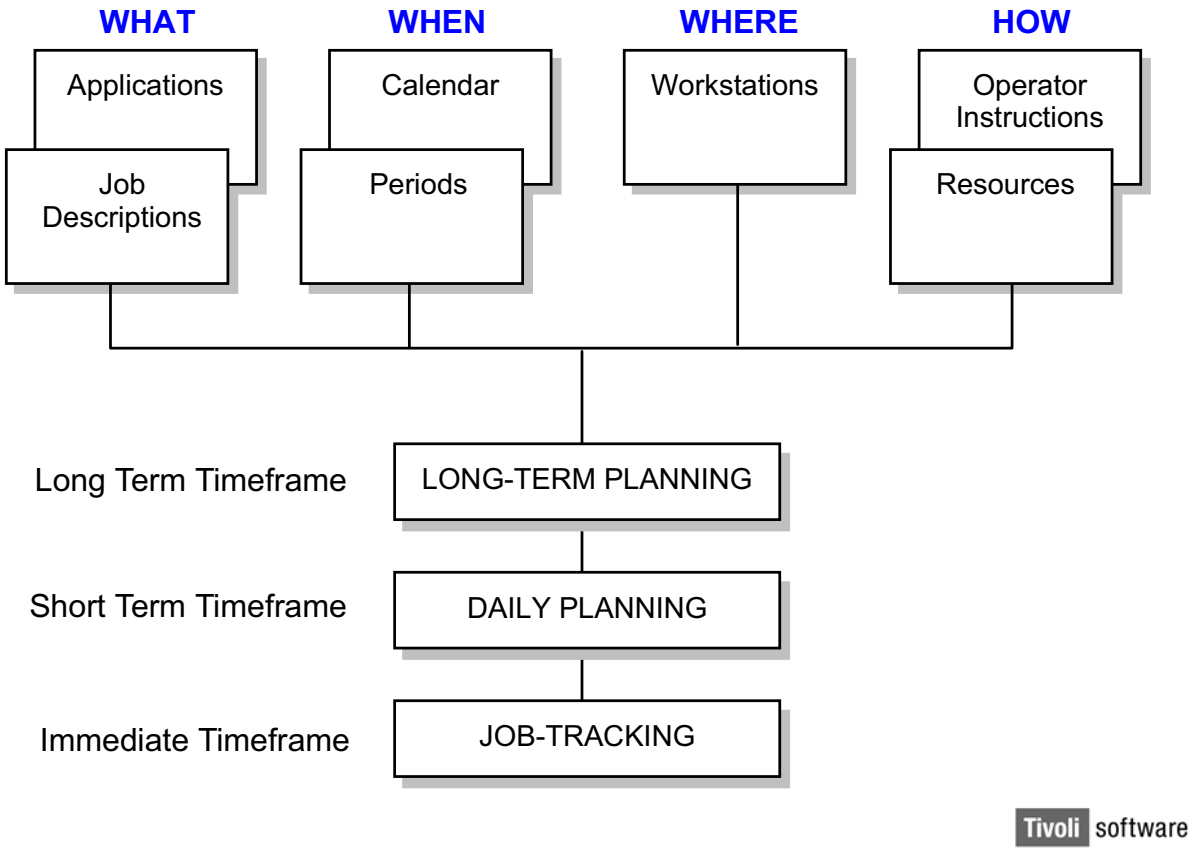


Figure 1-26. Functional Hierarchy TM402.0

Notes:

The databases contain information that together define your business batch-processing operational model. It describes your environment.

TWS for z/OS uses the information you define in the databases to generate occurrence records in the long-term plan file.

Daily planning uses the long-term plan as an input source to create the current plan. The daily planning process also retrieves detailed information about each operation, calculates planned start times, and checks the availability of workstations and required resources.

To reflect the status of each operation, the job-tracking function uses real-time event information to update occurrences in the current plan.

Instructor Notes:

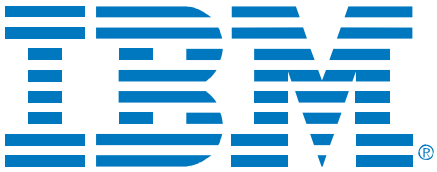
Purpose — Show the relationship between the databases and the plans.

Details — See student notes.

Additional Information —

Transition Statement —

1.4 Describing Your Environment



Topic 1.4

Describing Your Environment

Tivoli software

Figure 1-27. Describing Your Environment TM402.0

Notes:

Instructor Notes:

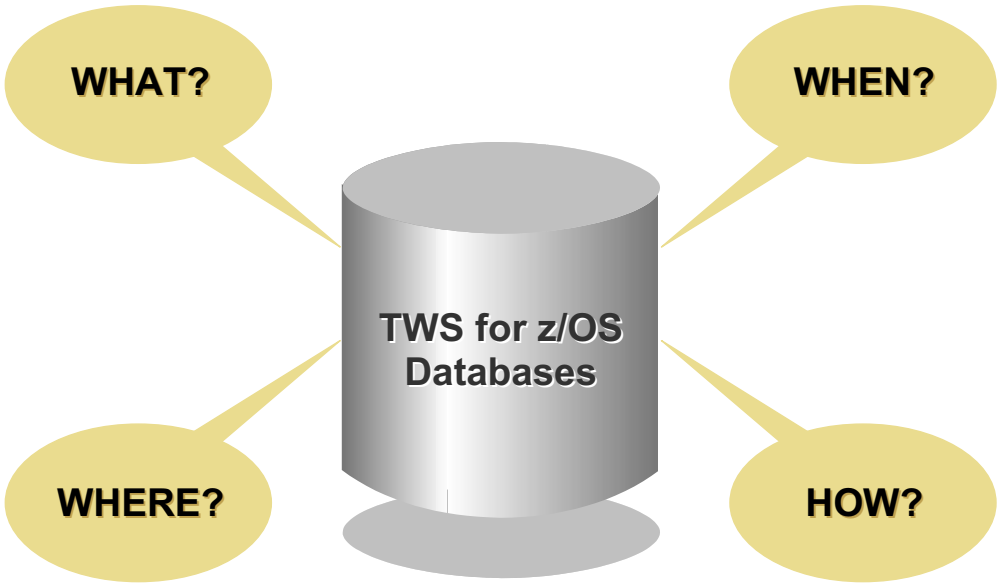
Purpose — Use this topic to introduce the class to what TWS for z/OS requires for scheduling, and how to supply the data.

Details —

Additional Information —

Transition Statement —

TWS for z/OS Needs to Know



Tivoli software

Figure 1-28. TWS for z/OS Needs to Know TM402.0

Notes:

The basic batch-production policies are defined in the TWS for z/OS workstation, calendar, period, application, special resources, and operator instruction databases.

The words

- What [work?]
- When [to run it?]
- Where [to run it?] and
- How [to run it?]

relate to TWS for z/OS databases when describing your batch-scheduling environment.

What - In the Application and Job Description databases, you define the details of the batch work.

Where - You define the logical and physical places to run work in the Workstation database.

When - You define your business calendar and processing cycles in the Calendar and Period databases.

How - Start and restart instructions for operations in the current plan can be defined in the Operator Instruction database. You can also define TWS for z/OS JCL variables in a JCL variables database. You can use JCL variables to automatically edit jobs prior to submission.

Instructor Notes:

Purpose — Link the concept of What, When, Where, and How to databases To identify basic information needed to develop a schedule of work.

Details — See student notes.

Additional Information —

Transition Statement —

Workstations

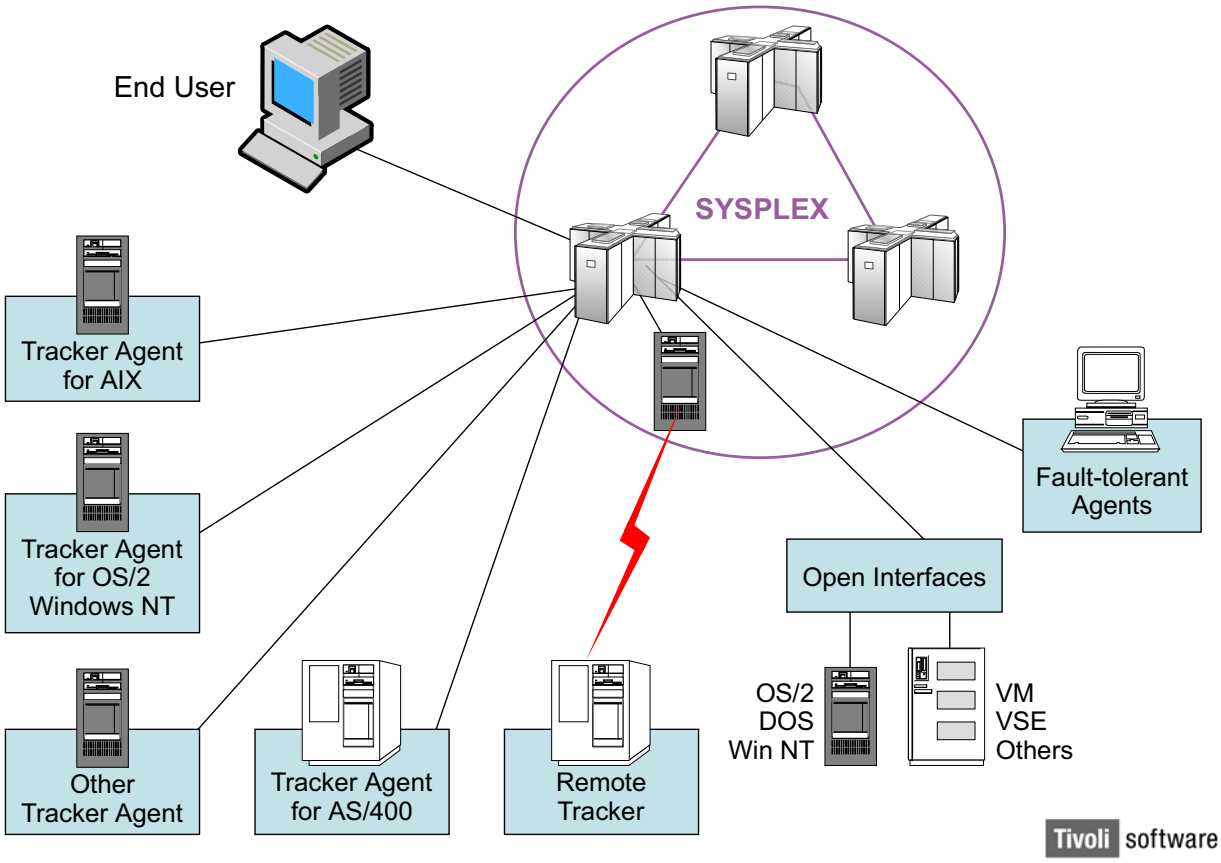


Figure 1-29. Workstations TM402.0

Notes:

TWS for z/OS supports a range of work process types, called *workstations*, which control the processing of production workload tasks.

Each operation that TWS for z/OS tracks, whether a job, started task, or some other activity, must be associated with a workstation.

Each workstation supports one type of activity.

Instructor Notes:

Purpose — Introduce the concept of workstations.

Details — See student notes.

Additional Information —

Transition Statement —

Calendars

Mon.	Tue.	Wed.	Thu.	Fri.	Sat.	Sun.
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Figure 1-30. Calendars TM402.0

Notes:

You use calendars to define standard and nonstandard processing days for your business. Standard processing days are known as *work* days. Nonstandard processing days are known as *free* days. TWS for z/OS supports *multiple calendars* for enterprises where different departments have different work days and free days.

Instructor Notes:

Purpose — Introduce calendars

Details — See student notes.

Additional Information —

Transition Statement — Calendars provide the foundation for scheduling but you specify the run frequency by using templates that represent business processing cycles.

Business Cycles - Periods

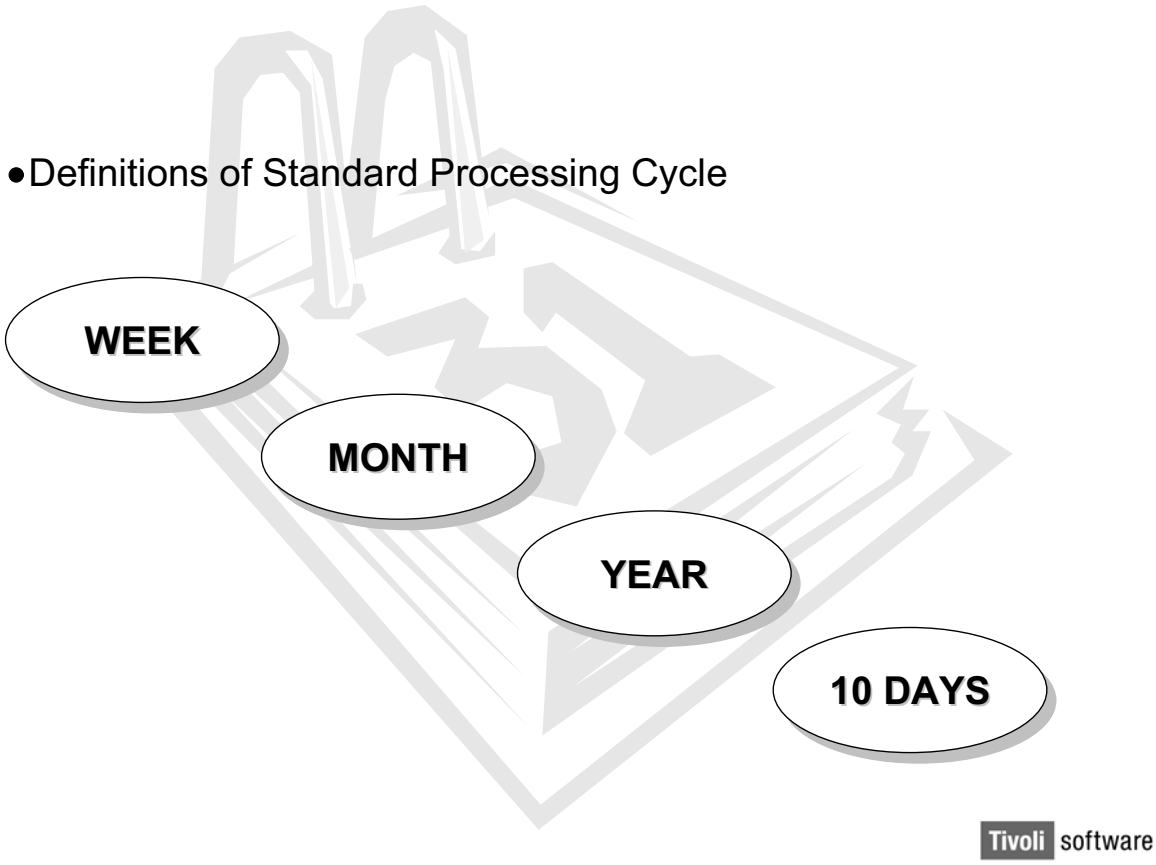


Figure 1-31. Business Cycles - Periods TM402.0

Notes:

A period is a user-defined processing cycle or pattern.

TWS for z/OS uses processing cycles, or periods and calendars to calculate the run-dates for applications.

Instructor Notes:

Purpose — Introduce periods.

Details — See student notes.

Additional Information —

Transition Statement —

Work to Be Done: Applications and Job Descriptions

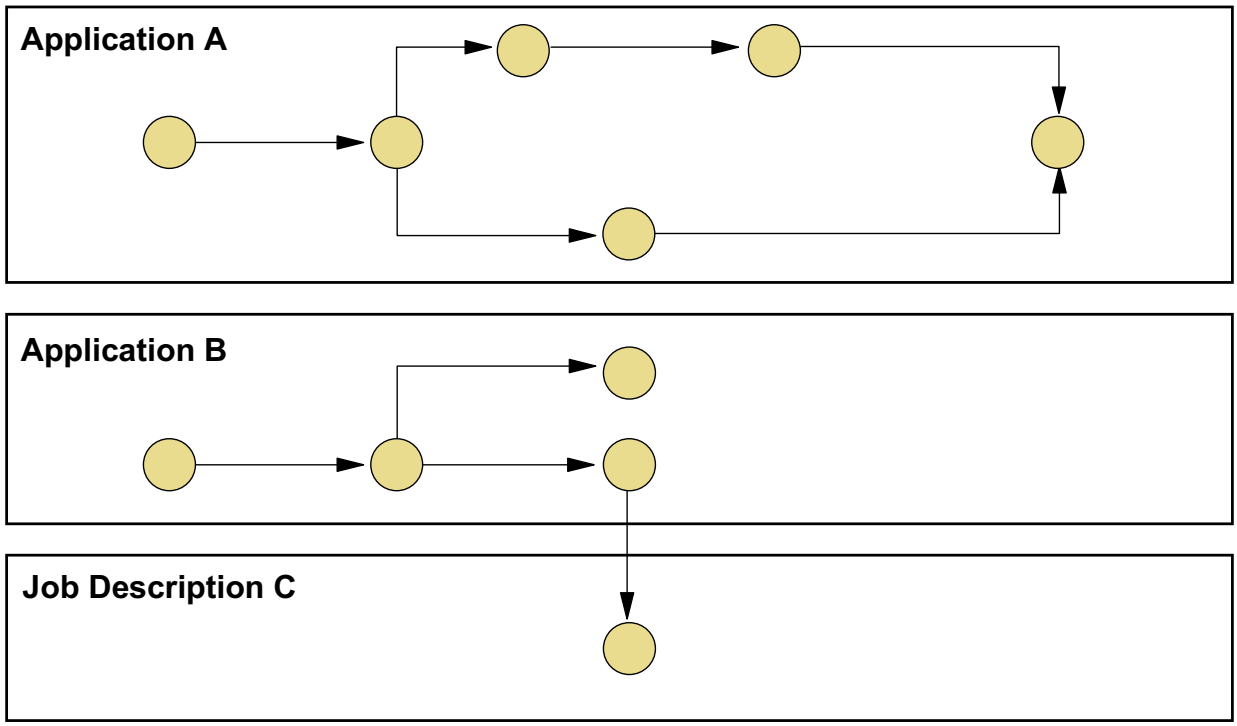


Figure 1-32. Work to Be Done: Applications and Job Descriptions TM402.0

Notes:

An application is a unit of work that you can schedule. It contains one or more operations.

In applications with multiple operations, you must define dependencies between the operations to specify the run sequence.

When creating an application, you can also define scheduling criteria for the application. The scheduling criteria is called a run cycle.

To achieve a desired result, you can define multiple run cycles for an application.

Instructor Notes:

Purpose — Introduce applications.

Details — See student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- Describe the basic functions of TWS for z/OS
- List the main TWS for z/OS databases
- List the names of the TWS for z/OS plans
- Describe the purpose of the TWS for z/OS plans

Tivoli software

Figure 1-33. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 2. Workstations

What This Unit is About

This unit introduces the concept of IBM Tivoli Workload Scheduler for z/OS workstations, describes their function, and how to create them.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe the functions of IBM TWS for z/OS workstations
- List the types of IBM TWS for z/OS workstations
- Create IBM TWS for z/OS workstations

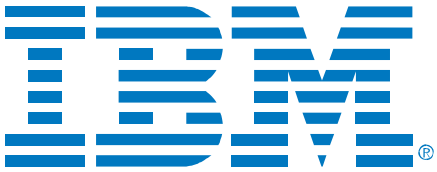
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 2: Workstations

Tivoli software

Figure 2-1. Workstations TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- Describe the functions of TWS for z/OS workstations
- List the types of TWS for z/OS workstations
- Create TWS for z/OS workstations

Tivoli software

Figure 2-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

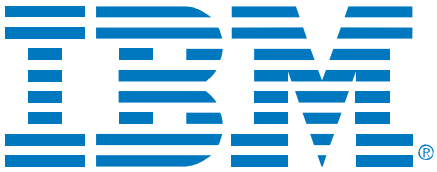
Purpose —

Details —

Additional Information —

Transition Statement —

2.1 Introducing TWS for z/OS Workstations



Topic 2.1

Introducing TWS for z/OS Workstations

Tivoli software

Figure 2-3. Introducing TWS for z/OS Workstations TM402.0

Notes:

Instructor Notes:

Purpose — To describe the concept of workstations, and how they are used. This unit is followed by a hands-on exercise in which the students create three workstations.

Details —

Additional Information —

Transition Statement —

Workstations - Where Work Is Done

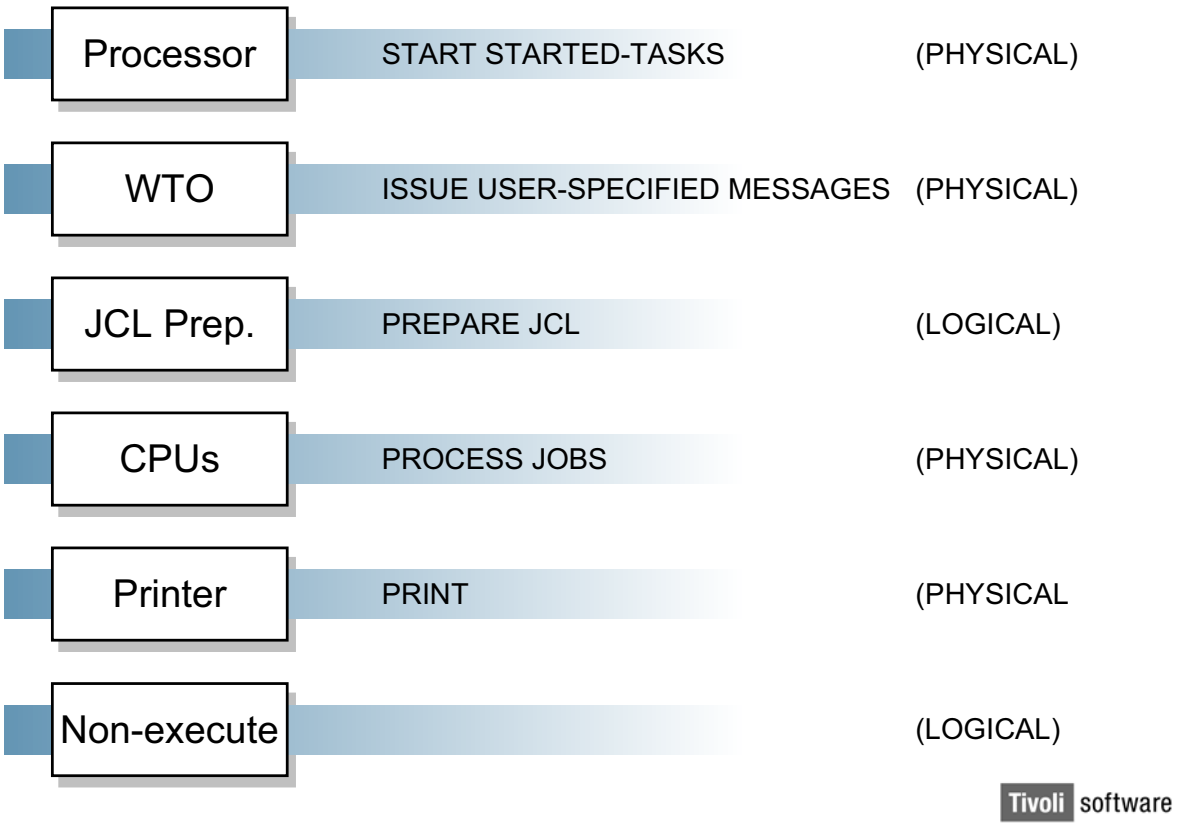


Figure 2-4. Workstations: Where Work is Done TM402.0

Notes:

TWS for z/OS supports a range of work process types, called *workstations*. Workstations map the processing needs of any task in the production workload.

Each workstation supports one type of activity. This gives you the flexibility to schedule, monitor, and control any type of DP activity, including the following:

- Job setup, both manual and automatic
- Job submission
- Started-task actions
- Issuing WTO messages
- Print jobs
- Manual preprocessing or post processing activity

For TWS for z/OS planning and control purposes the work to be performed is defined as individual operations, and these operations take place at workstations.

Each operation that TWS for z/OS tracks, whether a job, started task, or some other activity, must be associated with a workstation. The workstation is the point of control.

TWS for z/OS tracks jobs as they are processed at workstations and dynamically updates the plan with real-time information on the status of jobs.

Users can browse or modify the status of work at workstations by using :

- The Ready List dialog
- The Status-of-all-workstations graphic user interface of the job scheduling console

Instructor Notes:

Purpose — To reinforce the fact the TWS for z/OS workstations are logical entities map the processing needs of any task in the production workload.

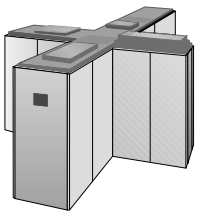
This unit is followed by a hands-on exercise in which the students create three workstations.

Details — See student notes.

Additional Information —

Transition Statement —

Workstation Types



Computer

- Supports:
  - Jobs
  - Started-tasks

Printer

General

- Supports:
  - Job setup
  - Write to Operator
  - Other tasks

Tivoli software

Figure 2-5. Workstation Types TM402.0

Notes:

There are three types of workstations:

- Computer workstations
- Printer workstations
- General workstations

Computer workstations are required to execute jobs. On z/OS or OS/390 platform systems, computer workstations can also be used to start started-tasks. Computer workstations for the distributed environment are defined with a special attribute.

Printer workstations are used on z/OS platform systems only.

General workstations support a myriad of tasks which include manual activities, issuing write-to-operator messages, and non-processing (dummy) operations..

Instructor Notes:

Purpose — Identify the three types of workstations.

Details — See student notes.

Additional Information —

Transition Statement —

Computer Workstations

They define destinations and attributes for:

- Job submission on -
  - z/OS systems
  - Non-z/OS systems
    - Tracker Agents
    - Fault-tolerant Agents
- Started-tasks on -
  - z/OS systems ONLY

Tivoli software

Figure 2-6. Computer Workstations TM402.0

Notes:

The majority of operations in a batch-production schedule are batch jobs. These operations are run on computer workstations where they are normally started automatically when all requirements are satisfied.

Computer workstations can also be used to start started tasks. Batch jobs and started tasks are automatically tracked to completion.

At job-submission, computer workstation has the STARTED TASK, STC option on the workstation general information panel set to N and can submit jobs in z/OS and non-z/OS environments. A started-task computer workstation has the STARTED TASK, STC option on the workstation general information panel set to Y. Operations you specify to run on this workstation are treated as started tasks, not as jobs.

When TWS for z/OS submits a job, or starts a started-task in the z/OS environment it gives control to the z/OS operating system and JES. TWS for z/OS keeps track of what happens to the job or started task and acts accordingly, but TWS for z/OS cannot affect how the job or started task executes.

You must keep the job-statements for z/OS jobs in partitioned data sets that are allocated to the TWS for z/OS controller. The JCL for started tasks is stored in the same way as JCL for jobs but, instead of submitting the job to the internal reader on the destination system, TWS for z/OS puts it into an allocated procedure library -EQQSTC, and issues the z/OS START command.

TWS for z/OS continues to support OPC tracker agents on certain non-z/OS operating platforms. These are defined the same as z/OS job submission, computer workstations with each having a defined destination. The job statements for these workstations must reside in a job library data set allocated to the TWS for z/OS controller.

A fault-tolerant agent is a computer workstation which is configured to schedule jobs on a distributed agent. You use a fault-tolerant workstation to schedule jobs on computers in the IBM Tivoli Workload Scheduler network. The workstation is defined with the FTA WORK STATION attribute on the workstation general information panel set to Y. The workstation must also be defined in the CPUREC initialization statement in the TWS for z/OS parameter library.

Like z/OS, and tracker agent jobs, you can store job statements for the distributed environment in TWS for z/OS-allocated, job-library datasets. This is an option with advantages as well as disadvantages. You must carefully weigh each case when deciding. This option is defined at the job level in the unit or work. By default the job statements for FTA workstations are stored on the workstations themselves.

Instructor Notes:

Purpose — Describe the function of computer workstations and identify the options and attributes to define computer workstations.

Details —See student notes.

Additional Information —

Transition Statement —

General Workstations

- Non-reporting
- Manual interaction
  - JCL preparation
  - Manual completion
- Issue user-defined messages
  - MLWTO
  - MessageID EQQW775I

Figure 2-7. General Workstations TM402.0

Notes:

A general workstation lets you control operations that are normally not controlled automatically.

A non-reporting, general workstation is often used for dummy operations - operations that do not require any processing. Operations at a non-reporting workstation are set to completed status as soon as they become eligible to be started. No job, started task, or WTO is submitted.

TWS for z/OS also uses two special general workstations to allow users to schedule tasks such as:

- Set up JCL for jobs and started tasks

  A job setup general workstation has the **JOB SETUP** option on the general information panel set to **Y**. The job setup workstation lets users prepare job or started-task JCL manually before execution.

  The job setup operation must be an immediate predecessor of the computer workstation operation for the job.

- Issue WTO messages, possibly to trigger action by NetView

  A WTO general workstation has the **WTO** option on the general information panel set to **Y**. A WTO workstation lets users schedule the issuing of write-to-operator (WTO) messages at the designated destination.

  The messages are multiline WTO (MLWTO) messages with the message ID of EQQW775I.

Instructor Notes:

Purpose — Describe general workstations, and their uses.

Details —See student notes.

Additional Information —

Transition Statement —

Workstation Reporting Attributes

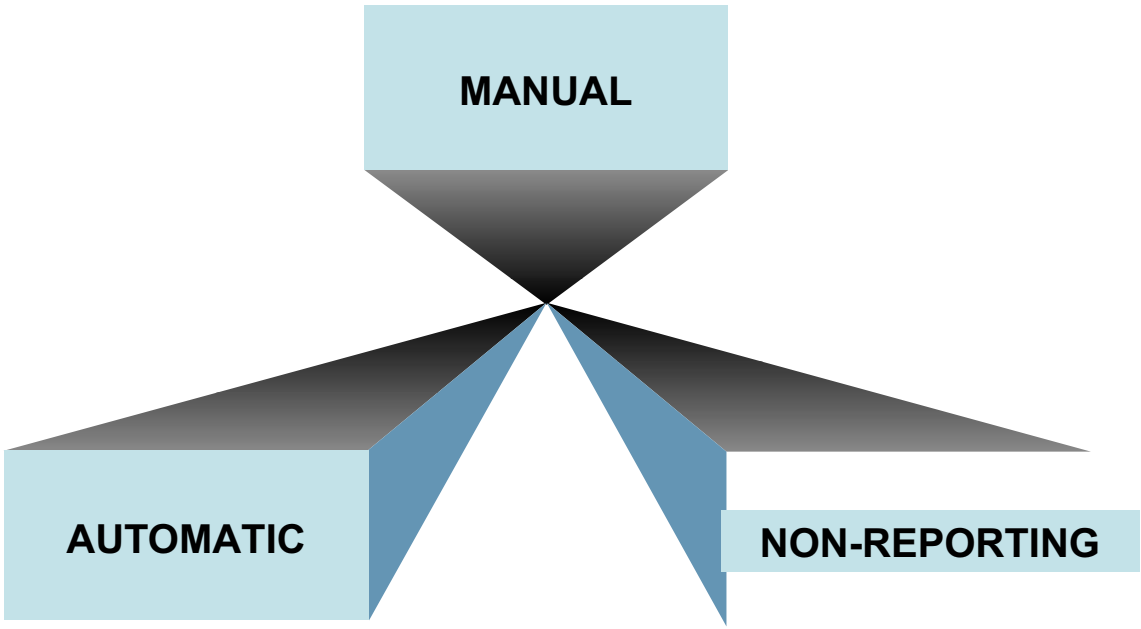


Figure 2-8. Workstation Reporting Attributes TM402.0

Notes:

Every operation in the current plan is assigned a status. The status of an operation describes its current condition. When all processing for an operation is finished, the operation is assigned status C (complete).

Before an operation reaches completion, it will have many different statuses as it progresses through the system. The sequence of statuses that an operation is assigned and the mechanism used for reporting status updates depends on the *reporting attribute* of the workstation on which the operation is defined.

There are four workstation-reporting attributes:

- Automatic

  The status change of operations is normally reported automatically, in response to event records created by the JES and SMF exits in TWS for z/OS. You can also change the status of these operations using the dialog panels.

- Completion only

This attribute is used for workstations on which users wish to control assignment of the complete (C) status for operations scheduled on these workstations. Typically, operations defined on workstations with this attribute require manual interaction.

- Start and Completion

  This attribute is used for workstations on which users wish manually edit job-statements prior to submission.

- Non-reporting

  You can use workstations with this attribute for operations that do not require any processing. For example, the operations can be used to hold the dependencies for successors, or the operation may represent milestones in the processing.

Instructor Notes:

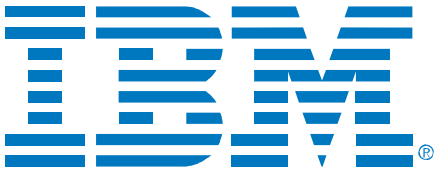
Purpose — Describe workstation attributes and how they affect the statuses of operations in the current plan.

Details — See student notes.

Additional Information — Spend some time reviewing the attributes section in chapter 2 of the *Planning and Scheduling the Workload* manual.

Transition Statement —

2.2 Creating Workstations



Creating TWS for z/OS Workstations

Topic 2.2

Tivoli software

Figure 2-9. Creating Workstations TM402.0

Notes:

Instructor Notes:

Purpose — To introduce TWS for z/OS dialog used to create workstations.

Details —See student notes.

Additional Information —

Transition Statement —

General Information

```
EQQWCGEP ----- CREATING GENERAL INFORMATION ABOUT A WORK STATION -----
Command ===>

Enter the command R for resources or A for availability or M for access method
above, or enter/change data below:

WORK STATION NAME    ===> cpu1
DESCRIPTION           ===> Main JES processor ____________
WORK STATION TYPE    ===> c          G General, C computer, P Printer
REPORTING ATTR       ===> a          A Automatic, S Manual start and completion
                                     C Completion only, N Non reporting
FTA Work Station     ===> N          FTA Work Station, Y or N
PRINTOUT ROUTING     ===> SYSPRINT   The ddname of daily plan printout data set
SERVER USAGE         ===> b          Parallel server usage, B, N, P, or C

Options:
 SPLITABLE           ===> N          Interruption of operation allowed, Y or N
 JOB SETUP           ===> N          Editing of JCL allowed, Y or N
 STARTED TASK, STC   ===> N          Started task support, Y or N
 WTO                 ===> N          Automatic WTO, Y or N
 DESTINATION         ===> _____      Name of destination
Defaults:
 TRANSPORT TIME      ===> 00.00      Time from previous work station HH.MM
 DURATION            ===> 00.05.00   Duration for a normal operation HH.MM.SS
```

Figure 2-10. General Information TM402.0

Notes:

The workstation-creation, general-information panel is the primary workstation definition panel.

- WORK STATION NAME: A four-character alphanumeric name for the object in the workstation database. The first character must be an alphabetic character.
- DESCRIPTION: A 24-character user description.
- FTA Work Station: Specify **Y** for a distributed-agent, computer workstation used in an end-to-end scheduling environment.
- SERVER USAGE: Specifies whether TWS for z/OS should consider the availability of parallel servers when generating a current plan or submitting jobs. The valid options are **B**, **N**, **P**, and **C.**
- SPLITTABLE: Used to allow the interruption of started operations. This option is not applicable to computer workstations. It is typically used for job-setup workstations where you prepare JCL for submission. If the preparation of the JCL is interrupted by

the preparer issuing the TSAVE command, the operation is given status I, interrupted. Preparation of the JCL can continue at a later time.

- JOB SETUP: Specify **Y** when defining a job-setup workstation.
- DESTINATION: This option is used for computer and general-WTO workstations to specify where the controller must send the job, started-task, or message. This can be:
  - A VTAM logical unit (LU) name in VTAM configurations.
  - An XCF member name in a monoplex or sysplex configuration.
  - A user-defined DD name of a submit/release data set in a shared-DASD configuration.
  - An IP name when defining a workstation for a tracker agent.
  - Blank, when the job, started-task, or message is destined for the system on which the controller is running.

  The destination field is not used for fault-tolerant workstations.
- Defaults DURATION: An optional field in which an estimated processing time for operations run on the workstation, is specified. TWS for z/OS uses the estimated processing time when creating the current plan, to work out a timetable for the operations.
- To run scheduled work, TWS for z/OS workstations must be available (A) and online in the current plan.

Instructor Notes:

Purpose — Describe the attributes and options on the general information panel.

Details —See student notes.

Additional Information —

Transition Statement —

Workstation Availability

```
EQQWMAVL ------------ AVAILABILITY OF A WORK STATION ----- ROW 1 TO 8
Command ===>                                               Scroll ===>

Work station               : CPU1          Main JES processor

Enter the ALL command above to get all open time intervals or
change data in the rows, and/or enter any of the following row commands:
I(nn)-Insert, R(nn)-Repeat, D(nn)-Delete
C-Close a day/date, S-Define open intervals for day/date

Row        Day of week or      Status        Description of day
cmd        YY/MM/DD
''         STANDARD_________   DEFINED       _________________________
''         MONDAY___________   STANDARD      _________________________
''         TUESDAY__________   STANDARD      _________________________
''         WEDNESDAY________   STANDARD      _________________________
''         THURSDAY_________   STANDARD      _________________________
''         FRIDAY___________   STANDARD      _________________________
C'         SATURDAY_________   STANDARD      _________________________
''         SUNDAY___________   STANDARD      _________________________

******************** BOTTOM OF DATA ***********************************
```

Figure 2-11. Workstation Availability TM402.0

Notes:

To be available for work, a workstation must be open, active, and connected. By controlling workstation availability, you control the running of operations defined on the workstation. TWS for z/OS establishes the availability of a workstation by using the *open intervals* in the workstation description database.

Open intervals are times of day when the workstation will process work.

Define open intervals for each day of the week or simplify the process by defining normal operating intervals as a STANDARD day. The ideal method for defining open intervals is to define open intervals for STANDARD and the days of the week will use those definitions.

On the Workstation Availability panel, deviations from the standard availability can be specified for specific dates. This feature can be used to identify a partial-day, workstation closure for a planned maintenance window, or to close the workstation for an entire day.

Fault-tolerant workstations do not use the Workstation Availability feature because they are always available.

Instructor Notes:

Purpose —Help students understand the workstation availability concept and how to define availability.

Details —See student notes.

Additional Information —

Transition Statement —

Open Intervals

```
EQQWMOTL -------------- OPEN TIME INTERVALS FOR ONE DAY ----- Row 1 to 1 of 1
Command ===>                                                    Scroll ===>

Work station               : CPU1          Main JES processor
Day or specific date       : TUESDAY
All work station closed    : NO

Change data in the rows, and/or enter any of the following commands:
I(nn)-Insert, R(nn)-Repeat, D(nn)-Delete

Row       Open time interval       Parallel     Resources   Alternate
cmd       HH.MM  -  HH.MM          servers      R1   R2     Work station
''        ______    ______           00         00   00      ________

*************************** BOTTOM OF DATA ********************************
```

Figure 2-12. Open Intervals TM402.0

Notes:

On this panel, define the times when the workstation parallel servers are available to process work. If no parallel servers are available, no work is run at the workstation.

The defaults for STANDARD are:

```
Row Open time interval Parallel Resources Alternate
cmd  HH.MM -   HH.MM    servers  R1 R2     Work Station
''   00.00     24.00    99       99 99     ____
```

You can specify different Open-time-intervals with varying numbers or Parallel-servers within a 24-hour period.

Use this panel to specify alternate workstations for the workstation you are creating. You can specify an alternate workstation where TWS for z/OS reroutes operations if the normal workstation becomes unavailable. Operations that have not yet started on an inactive workstation can be rerouted to an alternate workstation.

The values specified on this panel should reflect the resources of your system that are available to TWS for z/OS.

Instructor Notes:

Purpose — Show how to define open intervals for workstations and how to define alternate workstations.

Details —See student notes.

Additional Information —

Transition Statement —

Parallel Servers

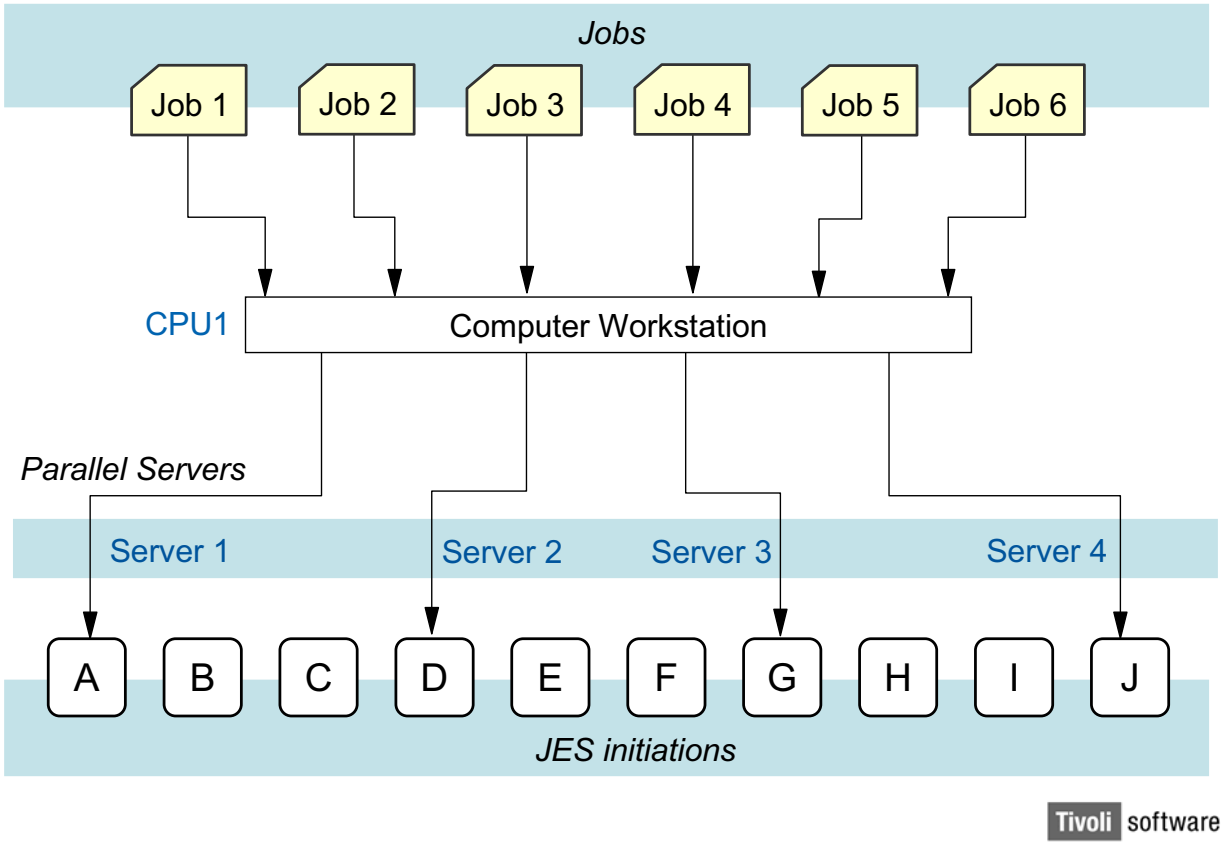


Figure 2-13. Parallel Servers TM402.0

Notes:

The number of parallel servers a workstation owns limits the number of operations that can be in started status at any time. Parallel servers are not used for fault-tolerant workstations.

At a computer workstation, a parallel server can represent a JES initiator. Operations defined to a computer workstation must allocate at least one parallel server.

This example shows jobs 1 through 6 were defined on a computer workstation, CPU1, which has four parallel servers. Server usage B was specified for this workstation. This means that CPU1 can start only four of the six waiting jobs, even though there are several JES initiators free to run the jobs that are waiting.

Instructor Notes:

Purpose —To reinforce the concept of parallel servers and show how they relate to JES initiators.

Details —

Additional Information —

Transition Statement —

Recommendations

- Keep the numbers low
- Use a single computer WS where possible
- Avoid the use of unnecessary manual WS
- Do not specify job setup unnecessarily
- Change existing WS definitions with care
  - Reporting attribute
    - Changing open intervals without changing input arrival and deadline times may have an effect on many applications, and make the plan unworkable
  - Do not close a WS on the standard day
- Such changes are not visible on, and do not affect the long-term plan

Tivoli software

Figure 2-14. Recommendations TM402.0

Notes:

It is not necessary to define a large number of workstations to efficiently run a large work load.

In shared spool z/OS environments, you might be able to process all your jobs on a single, job-submission computer workstation. Consider using job class to control where the job is executed. If necessary, you can define one job-submission computer workstation per system.

One each of job-setup, manual-completion-only, and non-reporting workstations should suffice.

Workstation definitions can be changed during the current plan period, but note that altering open intervals and/or resource numbers in flight could affect work in progress, so take care.

Instructor Notes:

Purpose — Highlight some factors that should be considered when creating workstations for an installation.

Details —See student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- Describe the functions of TWS for z/OS workstations
- List the types of TWS for z/OS workstations
- Create TWS for z/OS workstations

Tivoli software

Figure 2-15. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 3. Calendars and Periods

What This Unit is About

This unit introduces IBM TWS for z/OS calendar and calendar periods, describes their functions, and how to create them.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe the purpose of IBM TWS for z/OS calendars
- Create an IBM TWS for z/OS calendar
- Describe the function of IBM TWS for z/OS calendar periods
- Create IBM TWS for z/OS periods

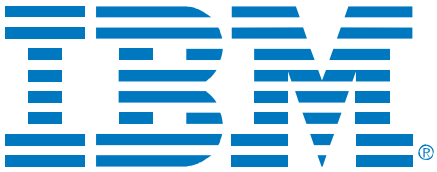
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 3: Calendars and Periods

Tivoli software

Figure 3-1. Calendars and Periods TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- Describe the purpose of TWS for z/OS calendars
- Create a TWS for z/OS calendar
- Describe the function of TWS for z/OS calendar periods
- Create TWS for z/OS periods

Tivoli software

Figure 3-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

Purpose —

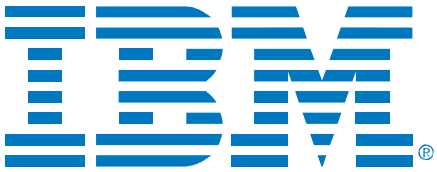
Details — After completing this unit, you should be able to:

- Describe the purpose of TWS for z/OS calendars
- Create a TWS for z/OS calendar
- Describe the function of TWS for z/OS calendar periods
- Create TWS for z/OS periods

Additional Information —

Transition Statement —

3.1 Calendars



Topic 3.1

Calendars

Tivoli software

Figure 3-3. Calendars TM402.0

Notes:

Instructor Notes:

Purpose —Introduce this topic.

Details —The Calendar database contains one or more Calendar definitions each containing a list of free and work days.

Additional Information —

Transition Statement —

What Is a TWS for z/OS Calendar?

Tivoli. software

Figure 3-4. What is a TWS for z/OS Calendar? TM402.0

Notes:

The calendar specifies normal-processing days, weekends, and public holidays recognized by a business. Normal-processing days are called *work* days and the others are called *free* days.

Work (business) days: Days on which there is no deviation in the normal, processing schedule. These are typically Monday through Friday.

Free (non-business) days: Days on which you do not process some of the work in the normal-processing schedule.

More than one calendar can be defined. You do this for systems that span geographical areas with different public holidays. Although you may have several calendars, always call your most commonly used - default calendar, DEFAULT.

The default calendar is identified in a TWS for z/OS initialization statement for batch services, such as extending the long-term plan.

Instructor Notes:

Purpose —Identify the role and contents of TWS for z/OS calendars.

Details —See student notes.

Additional Information —

Transition Statement —

Function of a Calendar

```
--------------------------- LIST OF GENERATED DATES ---------------------------
Command ===>                                                 Scroll ===> CSR
                                                           Goto Year ===>
Rule        : RULE1      RUN EVERY WORKDAY OF THE WEEK
Calendar    : DEFAULT            Work day end time: 00.00
Interval    : 020101   - 051231      Free day rule: 4

      January          2002  February         2002  March            2002
      Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
         01 02 03 04 05 06                 01 02 03                 01 02 03
      07 08 09 10 11 12 13  04 05 06 07 08 09 10  04 05 06 07 08 09 10
      14 15 16 17 18 19 20  11 12 13 14 15 16 17  11 12 13 14 15 16 17
      21 22 23 24 25 26 27  18 19 20 21 22 23 24  18 19 20 21 22 23 24
      28 29 30 31           25 26 27 28           25 26 27 28 29 30 31

      April            2002  May              2002  June             2002
      Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su  Mo Tu We Th Fr Sa Su
      01 02 03 04 05 06 07        01 02 03 04 05                 01 02
      08 09 10 11 12 13 14  06 07 08 09 10 11 12  03 04 05 06 07 08 09
      15 16 17 18 19 20 21  13 14 15 16 17 18 19  10 11 12 13 14 15 16
      22 23 24 25 26 27 28  20 21 22 23 24 25 26  17 18 19 20 21 22 23
      29 30                 27 28 29 30 31        24 25 26 27 28 29 30
```

Tivoli software

Figure 3-5. Function of a Calendar TM402.0

Notes:

TWS for z/OS uses the calendar to determine when applications are scheduled, and to calculate dates for automatic JCL tailoring.

This is an example of a list of generated dates. It is the result of an online test of a scheduling rule in an application.

The long-term planning process uses the calendaring objects, and user-specified scheduling rules to determine the days on which an application should be scheduled.

You specify the calendar in the application definition. If no calendar is specified TWS for z/OS will try to locate a calendar in this sequence:

1. The defined default calendar.
2. The calendar used for online services such as testing a rule with GENDAYS. You specify this calendar name on the option 0.2 dialog panel.
3. A calendar with the name DEFAULT.
4. If the DEFAULT calendar does not exist, all days are considered work days.

Instructor Notes:

Purpose — To describe how calendars are used by TWS for z/OS.

Details —See student notes.

Additional Information —

Transition Statement — Let us see how you create a TWS for z/OS calendar.

Creating a Calendar

Week, Day or Date	Comments	Status
Monday		W
Tuesday		W
Wednesday		W
Thursday		W
Friday		W
Saturday		F
Sunday		F
021225	Christmas Day	F
021226	Boxing Day	F
030101	New Years Day	F

Tivoli software

Figure 3-6. Creating a Calendar TM402.0

Notes:

- Specify the workday and time as **00.00**
- Specify each normal weekday workday with a status of **W** (work day)
- Specify each non-business day with a status of **F** (free day)
- Override the status of a day by using a specific date. Dates take precedence over generally defined days of the week.

Instructor Notes:

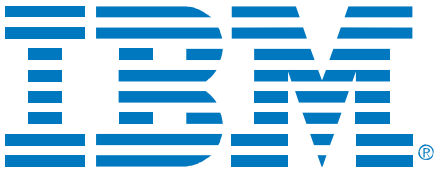
Purpose —Show how to create a calendar.

Details —See student notes.

Additional Information —

Transition Statement — Calendars simply identify work days and free days. To generate the run days for a unit of work, a processing cycle must be used together with the calendar. The next topic covers the creation of processing cycles.

3.2 Calendar Periods



Topic 3.2

Calendar Periods

Tivoli software

Figure 3-7. Calendar Periods TM402.0

Notes:

Instructor Notes:

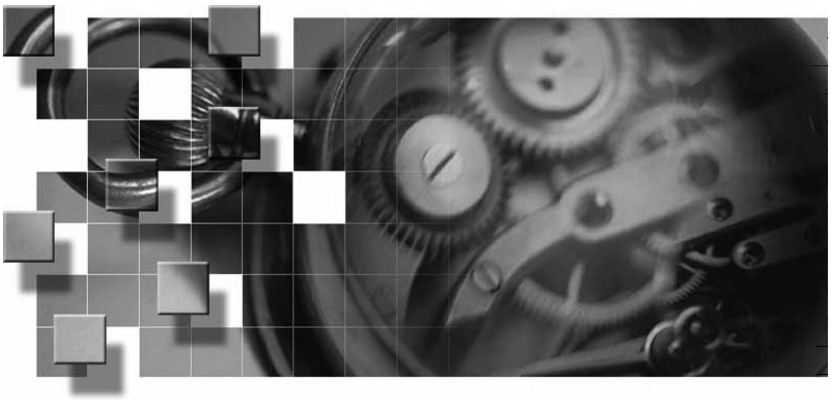
Purpose —Introduce this topic

Details —The Period database contains user-defined business or processing cycles.

Additional Information —

Transition Statement —

What Is a Calendar Period



The Period

Tivoli. software

Figure 3-8. What is a TWS for z/OS Calendar Period TM402.0

Notes:

A calendar period can be considered a user-defined template which represents a business's processing cycle for one or more of its TWS for z/OS applications.

The Period database holds definitions of business cycles which are used to schedule work.

Instructor Notes:

Purpose —Describe the term calendar period.

Details —See student notes.

Additional Information —

Transition Statement — The next visual will help to clarify the function of a calendar period.

When Work Is to Be Scheduled

Different applications run at different times, for example:

- Ordering - daily
- Sales - weekly
- Payroll - monthly
- Inquiries - regular intervals
- Accounting - monthly, quarterly
- Tax, annually

We can term these business processing cycles

Figure 3-9. When Work Is to Be Scheduled TM402.0

Notes:

Many applications run on a daily, weekly, or monthly basis. Others may run at less frequent intervals. Some may run on an irregular basis, perhaps once a week for the first quarter, twice a week for the next, thrice for the next, and four times a week for the last.

The majority of production work runs to regular schedules. TWS for z/OS *rules* provide a simple way to define run schedules for applications.

Most common application schedules can be defined by using rules which use predefined cycles, but for unusual requirements user-defined periods may be needed.

The run-frequency of an application can be defined by rules or a combination of user-defined periods and specified run days. Run frequencies are known as *run cycles*.

An application may refer to more than one period.

Instructor Notes:

Purpose —Clarify the need for periods and the function of a period.

Details —For standard processing cycles, such as daily, weekly, monthly, yearly, and so forth, you will not need to define calendar periods, because these are built-in cycles already defined in TWS for z/OS.

Additional Information —

Transition Statement —Let us see the built-in cycles and how you use them.

Cycle Specification

```
EQQRULEP --------------------MODIFYING A RULE -----------------------------------
Command ==>

Enter the GENDAYS command to display the dates generated by this rule
Enter S and user data in the fields below to define a rule

Application   : PMAY1
Rule          : RULE1

--- Frequency ---          --- Day ---      --- Cycle Specification ---
------------------------------------------------------------------------------
       _ Only           |  _ Day         |  _Week        _ January     _ July
       _ Every          |  _ Free day    |  _ Month      _ February    _ August
                        |  _ Work day    |  _ Year       _ March       _ September
 _ First    _ Last      |  _ Monday      |               _ April       _ October
 _ Second   _ 2nd Last  |  _ Tuesday     |               _ May         _ November
 _ Third    _ 3rd Last  |  _ Wednesday   |               _ June        _ December
 _ Fourth   _ 4th Last  |  _ Thursday    |  Week number  __  __  __  __  __  __
 _ Fifth    _ 5th Last  |  _ Friday      |  Period name  __________  __________
   __  __     __  __    |  _ Saturday    |               __________  __________
   __  __     __  __    |  _ Sunday      |
   __  __     __  __    |                |  Shift default origin by __ days
------------------------------------------------------------------------------
```

Figure 3-10. Cycle Specification TM402.0

Notes:

When using rules to define the run cycle for an application, you can use the MODIFYING A RULE panel to specify the run days, and processing cycle.

The Cycle Specification column lists the available TWS for z/OS defined scheduling cycles. These cycles can satisfy most requirements, but for unusual schedules, user-defined periods can be specified in this column.

Instructor Notes:

Purpose — Introduce TWS for z/OS defined cycles.

Details —See student notes.

Additional Information —

Transition Statement —Now that you have seen the need for business processing cycles, let us learn about the types of periods that you can create.

Types of Periods

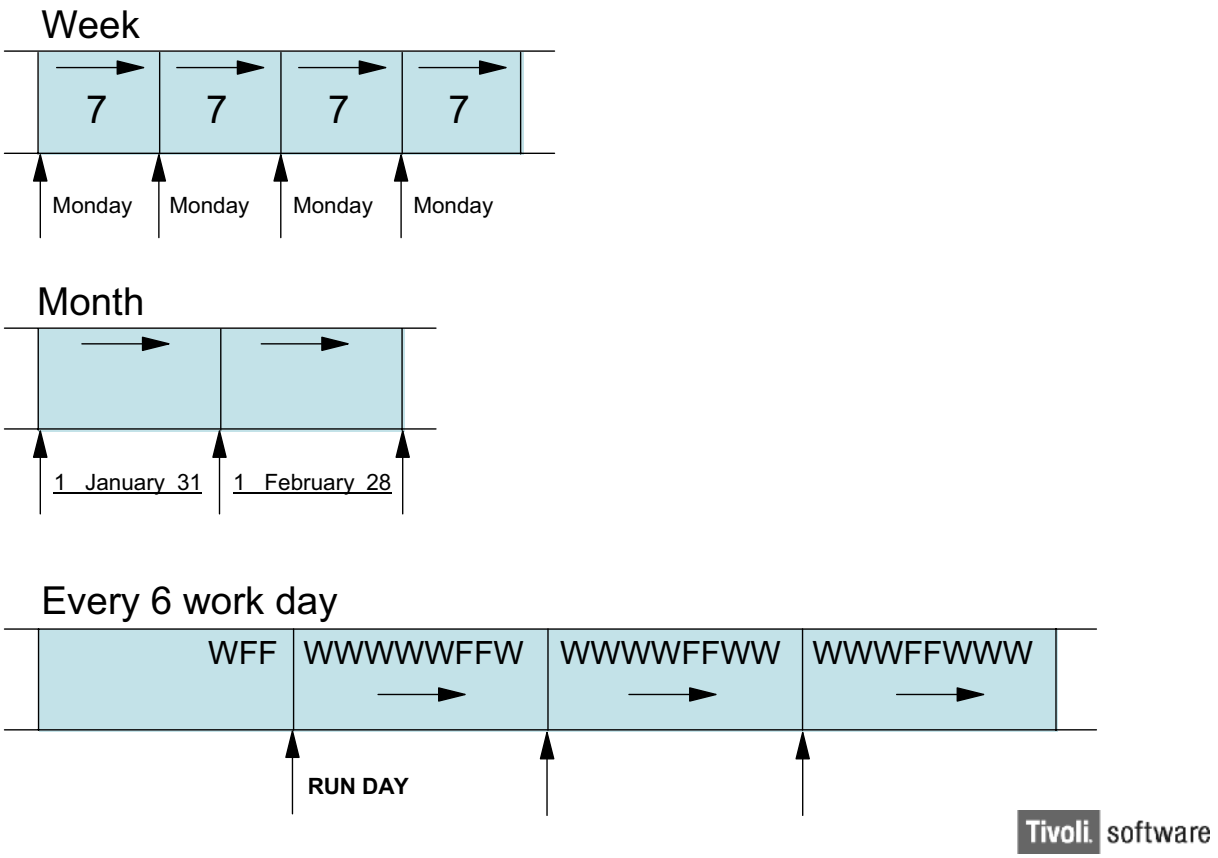


Figure 3-11. Types of Periods TM402.0

Notes:

There are two types of Periods:

- Cyclic - those of a constant length, for example, a week, a day.
- Non-Cyclic - those of variable length, for example, Calendar month, a year.

Cyclic periods can be defined to include all days or workdays only when scheduled days are being calculated.

TWS for z/OS counts from the start date for the number of days specified for the length; then it resets its counter to 1 before starting to count again.

Non-Cyclic periods are ideal templates for unique, specialized business processing cycles. A good example is an academic semester. Academic semesters have varying intervals, so the origin date of each interval must be specified.

Instructor Notes:

Purpose —Identify the types of periods and describe usage results.

Details —See student notes.

Additional Information —

Transition Statement —

Using Period Offsets

OFFSET 4

Period	WEEK	M T W T F S S
Period	MONTH	1 2 3 4 5 6 7

OFFSET -1

Period	MONTH	25 26 27 28 29 30 31 1 2

Tivoli software

Figure 3-12. Using Period Offsets TM402.0

Notes:

- Offsets are used to specify the actual run-days for the application.
- Offsets are relative to the first day of the period for positive offsets, and relative to the last day of the period for negative offsets.
- An offset of 1 means the application will run on the first day of the period. An offset of 4 will mean run on the fourth day of the period.
- Negative offsets count back from the last day of the period. A -1 specifies the last day of the period. This is very useful for scheduling work to run on the last day of the month without worrying about the length of the month.

Instructor Notes:

Purpose —Show how to specify run-days using period offsets.

Details —See student notes.

Additional Information —

Transition Statement —

Creating a Period

```
EQQTCRPL --------------- CREATING A CALENDAR PERIOD --------- ROW 1 TO 1 OF 1
Command ===>                                                  Scroll ===> CSR

Enter/change data below and in the rows,
and/or enter any of the following row commands:
I(nn)-Insert, R(nn),RR(nn)-Repeat, D(nn),DD-Delete

PERIOD NAME       ===> ________ Name of period
PERIOD TYPE       ===> __      A = Cyclic based on all days, W = Cyclic
                               based on work days, N = non-cyclic
DESCRIPTION       ===> ________________________________________
                               Descriptive text of the period
INTERVAL          ===> 000     Number of days between cyclic run period
VARIABLE TABLE    ===> _______________ JCL vaiable table id


Row      Interval     Interval
cmd      origin       end
''       ________     ________

******************************** BOTTOM OF DATA ****************************
```

Figure 3-13. Creating a Period TM402.0

Notes:

Cyclic Period

To define a cyclic period specify:

- The period name
- The type:
  - **A** - to include all days
  - **W** - to count workdays only
- The period length - **Interval**
- The start date for the period - **Interval Origin Date**.

Non-Cyclic Period

To define a noncyclic period specify:

- The period name
- The type - **N**
- An Interval of zero (**0**)
- The interval start dates - **Interval Origin Date**
- The interval end dates
- - **Interval End Date** (Optional)

Instructor Notes:

Purpose —How to create the different types of periods.

Details —See student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- Describe the purpose of TWS for z/OS calendars
- Create a TWS for z/OS calendar
- Describe the function of TWS for z/OS calendar periods
- Create TWS for z/OS periods

Tivoli software

Figure 3-14. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 4. Applications and Job Descriptions

What This Unit is About

This unit has four topics which cover how to create and schedule IBM TWS for z/OS applications, application groups, and job descriptions.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List the segments that make up an Application
- List the types of Applications
- Create Applications with or without Run Cycles

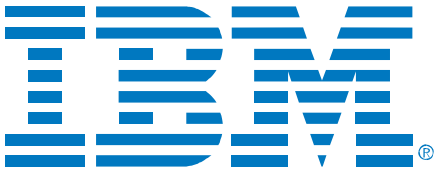
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 4: Applications

Tivoli software

Figure 4-1. Applications TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- List the segments that make up an application
- List the types of applications
- Create applications with or without run cycles

Tivoli software

Figure 4-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

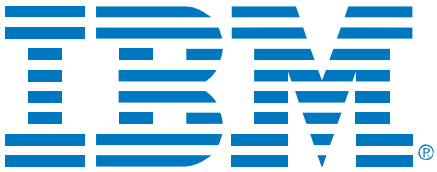
Purpose —

Details —

Additional Information —

Transition Statement —

4.1 What Are Applications?



Topic 4.1

What Are Applications

Tivoli software

Figure 4-3. What Are Applications TM402.0

Notes:

Instructor Notes:

Purpose —

Details —This unit contains four topics. The first three topics identify the three major segments of an application. The first topic introduces the TWS for z/OS concept of applications as the units of work that you schedule; it identifies the types of applications, and clearly spells out what is required to create the general information for each type.

To reinforce the contents of the topic, administer the appropriate hands-on exercise before starting the second topic.

At the end of the second topic, administer the run cycles exercise, and follow the third topic with the operations exercise.

The job description exercise should follow the fourth topic.

Additional Information —

Transition Statement —

Application Descriptions (AD) Work to be Done (1 of 2)

- Tasks managed by TWS for z/OS are called operations
- An operation can be:
  - A job to be executed
  - JCL preparation
  - Batch-scheduling related activity
- An AD can hold up to 255 operations
- Each operation must be associated with a workstation

Figure 4-4. Application Descriptions (AD): Work to be Done (1 of 2) TM402.0

Notes:

- In OPC terms, an *application* is a set of related jobs or tasks.
- Applications can range from a single computer operation with no dependencies, to a set of tasks at computer, general, and printer work stations with complex links both within the application and externally with other applications.
- The tasks that you want to control, such as running a job, issuing a WTO, or preparing JCL for a job, are called *operations*.
- *Application* is the name of the basic unit of work that you can schedule in TWS for z/OS.
- An application can hold a maximum of 255 operations.
- The operations in an application are normally run together, but you have the flexibility to add or remove operations.
- Since the workstation is the point of control in TWS for z/OS, each operation is run at a defined workstation.

Instructor Notes:

Purpose —Clearly define the term application in the context of TWS for z/OS.

Details —See student notes.

Additional Information —

Transition Statement —Let us see what are the other characteristics of a TWS for z/OS application.

Application Descriptions (AD) Work to be Done (2 of 2)

- Each operation in an AD has a unique sequence number (operation number)
- A workstation name and the operation number form the Operation ID
- Operations in an AD are run in a specified sequence
- An AD can be scheduled according to known business processing cycles

Figure 4-5. Application Descriptions (AD): Work to be Done (2 of 2) TM402.0

Notes:

- You must link all operations in a TWS for z/OS application to satisfy dependency relationships.
- You use an *operation number* (unique sequence number) as part of the identification tag for an operation in an application. Operation numbers must be in the range of 1 to 255.
- The workstation name and operation number form the *operation ID*.
- Operation numbers are used to define the dependency links between the operations in an application.
- TWS for z/OS runs the operations in the sequence dictated by the defined dependency relationships, that is, predecessors followed by successors.
- You can schedule dates and times when an application is run.

Instructor Notes:

Purpose —Clearly define the term application in the context of TWS for z/OS.

Details —See student notes.

Additional Information —

Transition Statement —

Basic Structure of an Application

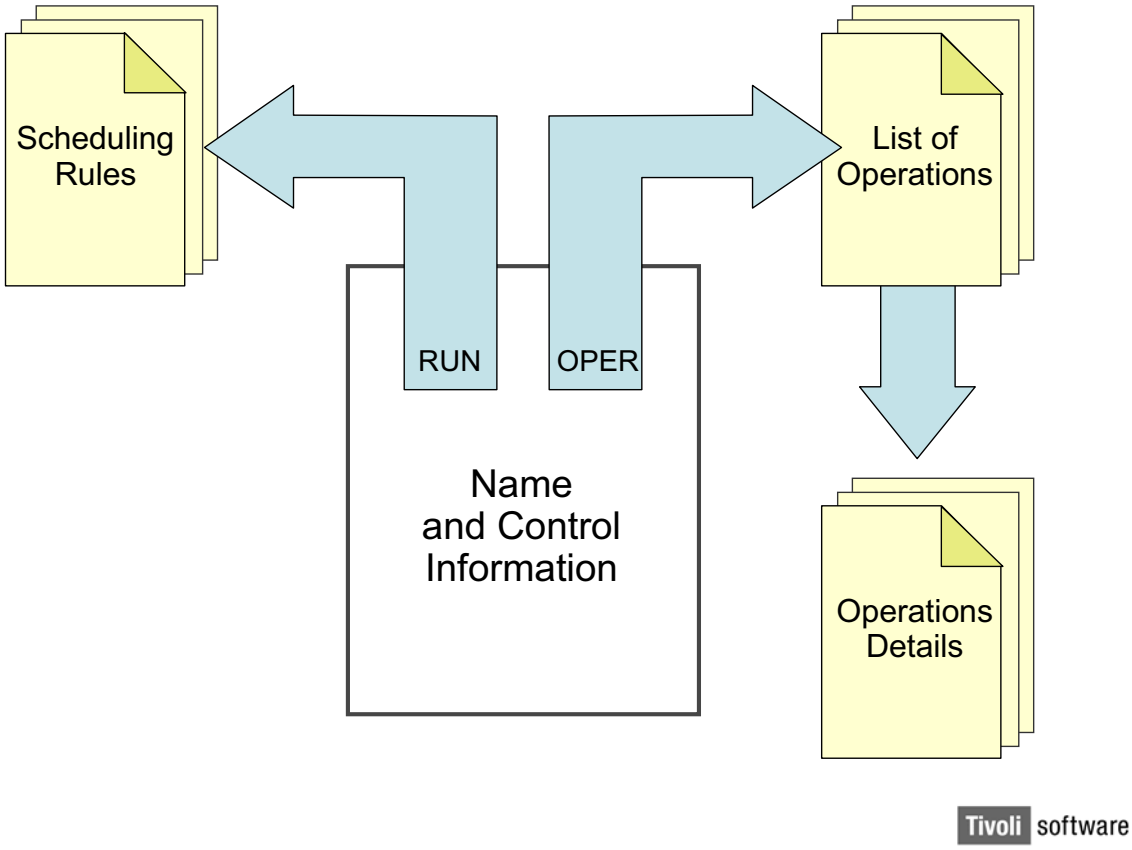


Figure 4-6. Basic Structure of an Application TM402.0

Notes:

The three basic segments of an application are:

- *General (Common) Information* - contains the application name and other application control information, such as the calendar, owner ID and submission priority.
- *Run Cycle* - contains the scheduling rules that define when the application will run.
- *Operation* - contains the jobs, tasks, and other activities that actually make up the batch work.

Applications which contain multiple operations or are linked to other applications will have a *dependency* segment as well.

All applications have a defined general information segment. Based on application type and the use of run-cycles, the other segments may be empty or may not be used.

The first application-creation, dialog panel you see when creating an application is the general information. On this panel are the two commands you use to launch the other two segments.

From the operations dialog panel, you can launch an Operations Details dialog to specify additional requirements for each operation.

Instructor Notes:

Purpose —Identify the base segments of an application.

Details —See student notes.

Additional Information —

Transition Statement —

Types of Applications

- Application Descriptions
  - Type A
  - Contains operations
- Application Groups (Group Definitions)
  - Type G
  - A **container** for Application Descriptions

Tivoli software

Figure 4-7. Types of Applications TM402.0

Notes:

The two types of applications are:

- **A**: Standard Application Descriptions - These hold the work you want TWS for z/OS to manage.
  - Hold the operations to be run
  - Can have data in all segments
  - Have a 255 operation limit
  - Can be scheduled individually
  - Can be scheduled as part of a group.
- **G**: Application Groups (Group Definitions) - These are **containers** for standard application descriptions.
  - Can have data in the general information and run-cycle segments only
  - Can be used to hold the scheduling rules for multiple applications.

Instructor Notes:

Purpose —Describe the types of applications and their characteristics.

Details —See student notes.

Additional Information —

Transition Statement —

Application Group and Its Members

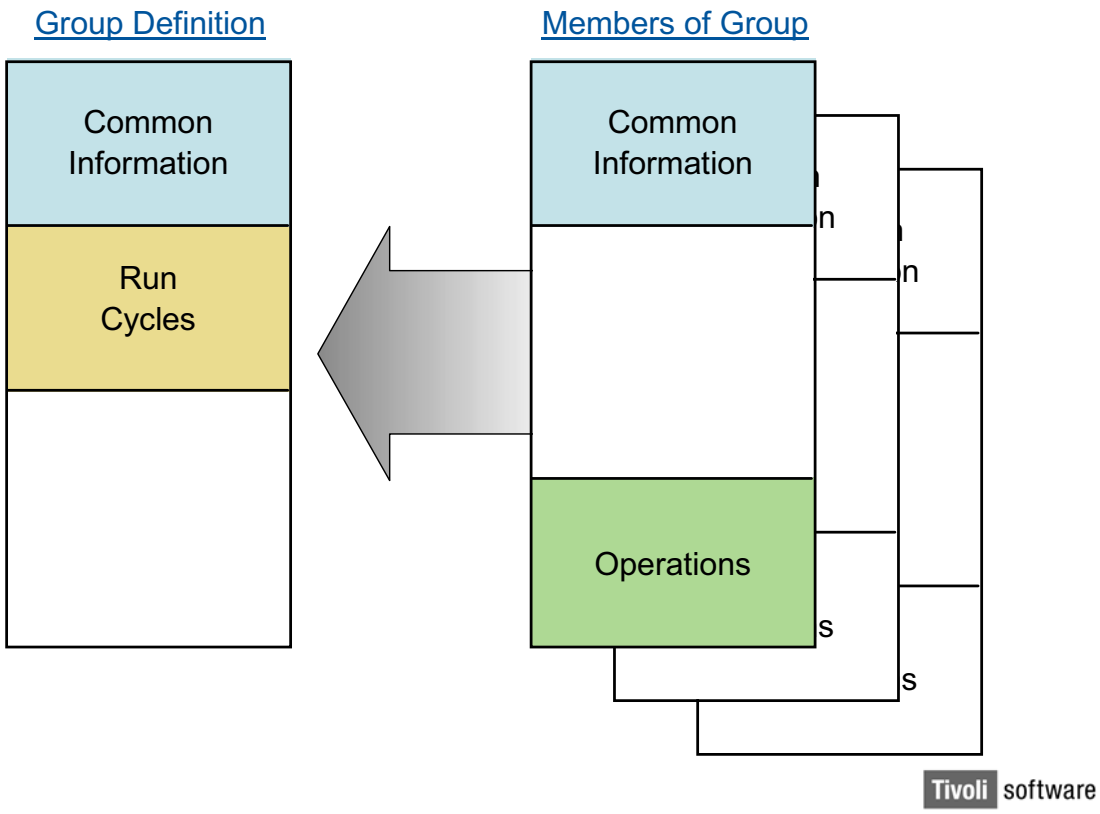


Figure 4-8. An Application Group and its Members TM402.0

Notes:

The Group Definition in this example contains data in the general information and run cycle segments.

- Data in the General Information segment include:
  - Application ID
  - Application type
  - Owner ID
  - Calendar ID
- Data in the Run Cycle segment include:
  - Scheduling rules

By doing this, you avoid having to specify the same calendar and run policy information for each application.

The Application Descriptions in this example are members of the group definition in the example. They contain data in the General Information and Operations segments

- Data in the General Information segment of the members include:
  - Application ID
  - Application type
  - Owner ID
  - Priority
  - Group Definition
- There is no data defined in the Run Cycle segment of the members
- Data in the Operations segment include:
  - Operations

Instructor Notes:

Purpose —Explain the relationship between an application group and its members.

Details —In this unit we have described the AD as consisting of three basic parts. This visual shows those parts and the roles they play. The three ADs on the right no run cycles but are members of the definition on the left which holds the scheduling rules.

Additional Information —

Transition Statement —

Advantages of Using Application Groups

- Time saver when creating and maintaining applications
- A handy and easy way to group related applications
- Easy way to add multiple-related applications to the current plan
- Protection against accidental deletion or manual completion in the current plan

Figure 4-9. Advantages of Using Application Groups TM402.0

Notes:

- You can group applications and job descriptions that are always scheduled together by creating them as members of an *application group*.
- It is easier to create and maintain a single group definition with run cycle information for its member applications, than it is to maintain the run cycles for multiple applications.
- You can also use groups in the Modify Current Plan panel, to add, delete, and complete all or part of an application group in the current plan.
- Applications in a group are protected against accidental deletion or modification of occurrences in the current plan. To delete or manually complete an application occurrence that is part of a group in the current plan, you must first remove it from the group.

Instructor Notes:

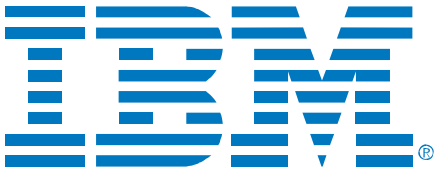
Purpose —To show the advantages of application groups.

Details —See student notes.

Additional Information —

Transition Statement —

4.2 Creating an Application



Topic 4.2

Creating an Application

Tivoli software

Figure 4-10. Creating an Application TM402.0

Notes:

Instructor Notes:

Purpose —This topic shows the basic requirements for creating general information for applications and application groups. A hands-on exercise follows the topic.

Details —

Additional Information —

Transition Statement —

Defining General Information

- Application ID (R)
- Type (R)
- Owner ID (R)
- Priority (R for Type A)
- Status (R)
- Valid from (R)
- Calendar
- Group Definition
- Descriptive text

Tivoli software

Figure 4-11. Defining the General Information TM402.0

Notes:

Fields on the General Information panel:

- Application ID - up to 16 alphanumeric characters. The first character must be alpha. (Required)
- Type - **A** or **G** (Required)
- Owner ID - up to 16 alphanumeric characters. The first character must be alpha. It should be used to identify a department rather than an individual. Never specify your TSO user ID. Can be used to control access to the application. (Required)
- Priority - used by TWS for z/OS to determine submission priority for operations in the application. This field is not used in group definitions. 1 = low, 2–7 = medium, 8 = high, 9 = urgent. It is recommended that you assign medium priority 5 to all application descriptions. (Required)
- Status - **A**ctive or **P**ending. Active means the application can be included in the long-term planning process or manually added to the current plan. Pending prevents inclusion in the LTP and adding to the CP. (Required)

- Valid From - by default it is the creation date of the application; but it can be used to create multiple copies of the application in the database. (Required)
- Calendar - used to calculate the run-days for the application. It is used when scheduling this application. If you have a default calendar in your profile this will appear here.
- Group Definition - used to identify the name of the group to which you wish to add the application.
- Descriptive text - use to document associated fields.

Instructor Notes:

Purpose —Explain the fields on the general information panel.

Details —See student notes.

Additional Information —

Transition Statement —

General Information by Application Type

- Type = **A**
  - Application ID
  - Owner ID
  - Priority
  - Status
  - Valid from
  - Type
  - Calendar ID (if not group member)
  - Group Definition (If a group member)
- Type = **G**
  - Application ID
  - Owner ID
  - _____
  - Status
  - Valid from
  - Type
  - Calendar ID Optional)
  - _____

Figure 4-12. General Information by Application Type TM402.0

Notes:

- The fields in the type A column are used in Standard Application Descriptions.
- The fields in the type G column are used in application Group Definitions. The blank lines indicate that the field cannot be specified.

Instructor Notes:

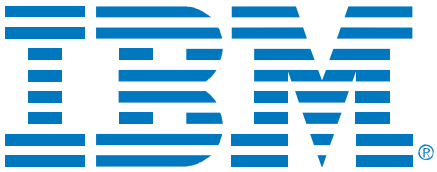
Purpose —To show the GI panel fields that apply to the different types of applications.

Details —See student notes.

Additional Information —

Transition Statement —

4.3 Defining Run Cycles



Topic 4.3

Defining Run Cycles

Tivoli software

Figure 4-13. Defining Run Cycles TM402.0

Notes:

Instructor Notes:

Purpose —To teach how to define run cycles.

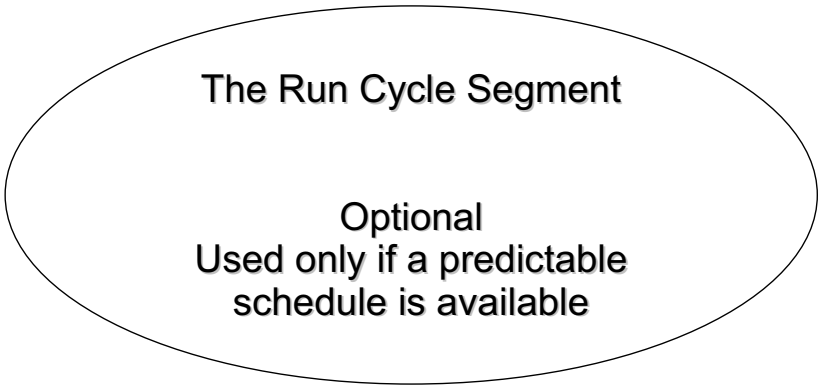
Details —The run-cycle segment holds scheduling rules for the application description or group definition.

This segment will contain data only if there is a known and predictable schedule. In member applications of application groups, this segment will be empty.

Additional Information —

Transition Statement —

The Run Cycle Segment



Tivoli software

Figure 4-14. The Run Cycle Segment TM402.0

Notes:

The run-cycle segment holds scheduling rules for the application description or group definition.

This segment will contain data only if there is a known and predictable schedule. In member applications of application groups, this segment will be empty.

Instructor Notes:

Purpose —Introduce run cycles and what they do.

Details —See student notes.

Additional Information —

Transition Statement —

Coding Run Cycles

You specify:

- Period/Rule + Run days = relative calendar dates
- Input Arrival Time = part of occurrence key
- Deadline Time
- Type of Run Cycle
  - Normal = When to run (N/R)
  - Negative = When not to run (X/E)
- Free Day Rule = handles free-day scheduling policy
- Effective dates of Run Cycle
- Variable Table name (Optional)
- Descriptive text

Tivoli software

Figure 4-15. Coding Run Cycles TM402.0

Notes:

On the run-cycles definition panel, you specify:

- ***Period/Rule*** -- business cycle or unique name of a scheduling *Rule*
  - **Run Days** - if using a period, you specify relative offsets in the period to define the run days.
    If using a *Rule name*, you select the run days from lists on a TWS for z/OS Rules panel.
- ***Input Arrival Time*** -- time associated with the occurrence. It is one of the fields that make up the occurrence key.
- ***Deadline***
  - ***day*** -- can be up to 99 days from the date the occurrence is added to the current plan.
  - ***time*** --time by which all operations in the application occurrence must be completed.
- ***Type*** of Run Cycle
  - Normal -- When to run (**N/R**)
  - Negative -- When not to run (**X/E**)
- Free ***day rule*** -- handles free-day, scheduling policy

- Effective dates of Run Cycle - time-span for inclusion in the long-term plan.
  - ***In effect*** - default is application creation date
  - ***Out of Effect*** - default is end date of TWS for z/OS calendar
- ***Variable Table name*** (Optional)
- Descriptive text

Instructor Notes:

Purpose —Describe the fields on the run cycles panel.

Details —See student notes.

Additional Information —

Transition Statement —Each occurrence in the LTP is unique. The next visual shows the fields that form the occurrence key.

The Run Cycles Definition Panel

```
--------------------------------- RUN CYCLES ---------------------- Row 1 of 1
Command ===>                                                 Scroll ===> CSR

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Specify run days/Modify rule

Application            : SM4P#DLYW#OPC01  CPEXTEND 24 HRS WRKDAYS
     Name of                                 In       Out of
Row  period/rule Input Deadline       F day effect   Effect
cmd  Text        HH.MM day HH.MM Type rule  YYMMDD   YYMMDD   Variable table
'''' RULE1___    06.00 01  06.00 R    4     981012__ 711231__ _________________
     RUN EVERY WORKDAY OF THE WEEK_____________________

****************************** Bottom of data ********************************
```

Figure 4-16. The Run Cycle Definition Panel TM402.0

Notes:

- The RUN CYCLES panel is displayed when you enter the RUN command on the general information panel.
- On this panel you define the scheduling rules for the application or application group. The scheduling rules are known as run cycles.
- There are two methods available for creating run cycles:
  - Rules (*rule-based*)
  - Periods with offsets (*offset-based*)
- The fields on this panel are:
  - **Name** of the period or rule must be specified. The Type field indicates whether you are creating a rule-based or offset-based run cycle. For offset-based, the period specified here must be already be defined in the period database.
  - **Input arrival time** is a time of day to be associated with each run of the application.
  - **Deadline day and time** is the number of days offset from the input arrival day, and time by which all operations in application must be completed.
  - **Type** defines whether the name refers to a rule or a period and whether the run cycle is scheduling or suppressing a run.

  - **R** - a rule used to schedule a run(rule-based)
  - **E** - a rule used to suppress a run(rule-based)
  - **N** - a period used to schedule a run(offset-based)
  - **X** - a period used to suppress a run(offset-based)
- **Free Day Rule** specifies what action TWS for z/OS must take when the application occurrence is scheduled on a free day.
- **In Effect/Out of Effect date** is used to define when the run cycle should be used. The fields will default to today's date and the highest system date when you hit enter.
- **Variable Table** is used to associate a variable table for the resolution of JCL variables used in jobs in the application.

- You can define multiple run cycles on this panel.
- Each run cycle generates an occurrence of the application.
- During long-term planning, the run cycles are evaluated in a top to bottom sequence.

Instructor Notes:

Purpose —Explain the fields on the run cycle panel and to explain scheduling choices.

Details —See student notes.

Additional Information —

Transition Statement —

The Free Day Rules

Month Run Cycle

Sat + Sun Free days

Offset 6 + 10

Date	1	2	3	4	5	6	7	8	9	10	11	12
Day	M	T	W	T	F	S	S	M	T	W	T	F
Rule 4										*		
Rule 3						*				*		
Rule 2								*		*		
Rule 1					*					*		
Rule E								*				*

Tivoli software

Figure 4-17. The Free-day Rules TM402.0

Notes:

- Free day rules are used by TWS for z/OS to handle scheduling on days that are defined as free days in the calendar associated with the application.
- You define one of five free day rules for each run cycle:
  - **1** Reschedule to run on the closest workday before the free day
  - **2** Reschedule to run on the closest workday after the free day
  - **3** Run on the free day
  - **4** Do not run on the free day (default for rule-based run cycles)
  - **E** Disregard free days when counting offsets to generate the run date. (default for offset-based run cycles)

This example shows the scheduled runs generated when each free-day rule is used.

Instructor Notes:

Purpose —Explain and demonstrate the functions of Free day rules.

Details —See student notes.

Additional Information —

Transition Statement —

Selecting a Run Frequency

- Run Frequency options are:
  - ONLY
  - EVERY
- In the Cambridge International Dictionary of English their meanings are listed as:
  - every (ALL)
  - every (REPEATED)
  - every (GREATEST)
  - only (SINGLE OR FEW)
  - only (LIMIT)
  - only (BUT)
- Keep the first two meanings of EVERY and ONLY in mind when selecting run frequencies.

Tivoli software

Figure 4-18. Selecting a Run Frequency TM402.0

Notes:

Instructor Notes:

Purpose —Explain the choices for selecting a run frequency.

Details —See student notes.

Additional Information —

Transition Statement —

Specifying Run Days

```
------------------------------ MODIFYING A RULE -------------------------------
 Command ===>

 Enter the GENDAYS command to display the dates generated by this rule
 Enter S and user data in the fields below to define a rule

 Application   : SM4P#DLYW#OPC01           CPEXTEND 24 HRS WRKDAYS
Rule           : RULE1      RUN THIRD WORKDAY OF THE MONTH

  --- Frequency ---          --- Day ---     --- Cycle Specification ---
-------------------------------------------------------------------------------
        S Only           |  _ Day       |    Week       _ January    _ July
          Every          |  _ Free day  |  S Month      _ February   _ August
                         |  S Work day  |  _ Year       _ March      _ September
   First     _ Last      |  _ Monday    |               _ April      _ October
   Second    _ 2nd Last  |  _ Tuesday   |               _ May        _ November
 S Third     _ 3rd Last  |  _ Wednesday |               _ June       _ December
 _ Fourth    _ 4th Last  |  _ Thursday  |  Week number  __ __ __ __ __ __
 _ Fifth     _ 5th Last  |  _ Friday    |  Period name  ________ ________
   ___ ___     ___ ___   |  _ Saturday  |               ________ ________
   ___ ___     ___ ___   |  _ Sunday    |
   ___ ___     ___ ___   |              |  Shift default origin by  ___ days
-------------------------------------------------------------------------------
```

Tivoli software

Figure 4-19. Specifying Run Days TM402.0

Notes:

The Rules panel allows scheduling requirements to be defined by selecting at least one item from each the three columns:

- FREQUENCY
- DAY
- CYCLE SPECIFICATION

- In the FREQUENCY column, you must select Only or Every. Every implies running at a repeated interval defined by Frequency.

  You can also select from or specify numbers in the FIRST and LAST lists in the Frequency column.

- In the DAY column, you make one or more selections to specify the run days for the application.
- The CYCLE SPECIFICATION column presents lists of TWS for z/OS supplied cycle templates or patterns. These allow users to define the processing cycle for their business applications.

- You must make one or more unambiguous selections in the CYCLE SPECIFICATION column. Selecting MONTH and one of the listed months is an ambiguous selection. The WEEK cycle in this column starts on Mondays.
- You can shift the start day of a cycle by using the SHIFT DEFAULT ORIGIN date field.
- In the CYCLE SPECIFICATION column, you can specify a user-defined period if the listed cycles fail to meet your needs.

Instructor Notes:

Purpose — Describe the selection of options for specifying run days.

Details —See student notes.

Additional Information —

Transition Statement —

Verifying the Relative Dates

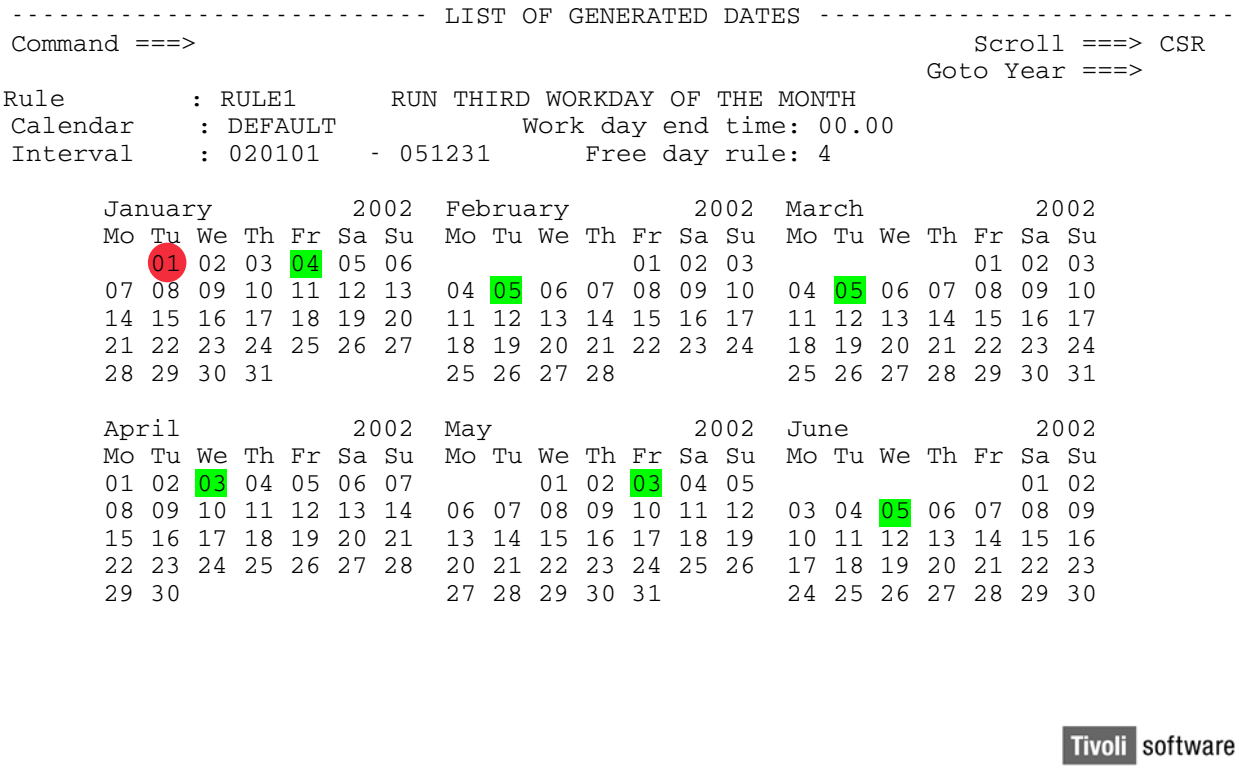


Figure 4-20. Verifying the Relative Dates TM402.0

Notes:

- After selecting frequency, day and cycle specifications, users can enter the GENDAYS command to display a calendar with the run days generated for the rule.
- This is an example of the GENDAYS output.
- If no calendar is specified in the application, the Gendays function uses the calendar specified on the option 0.2 dialog panel.
- The Gendays function displays the run days for one run cycle at a time.

Instructor Notes:

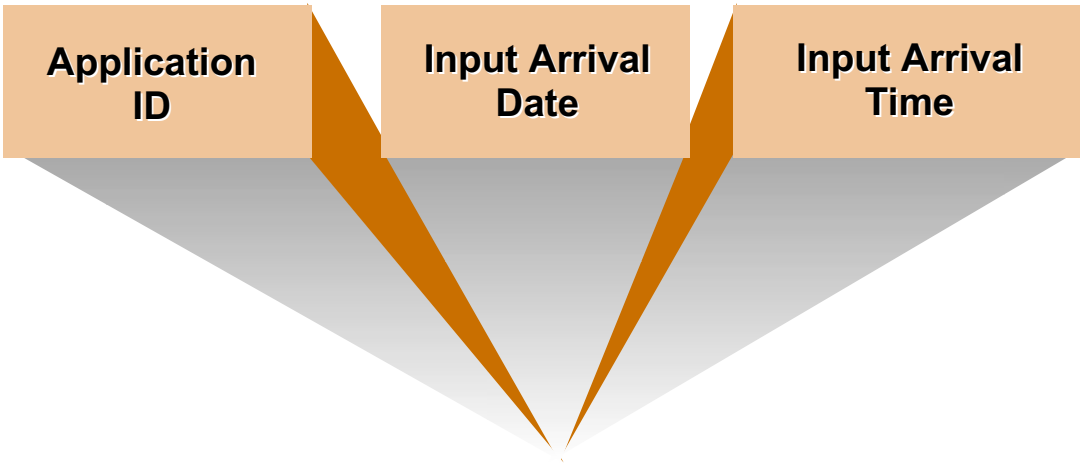
Purpose — Introduce the Gendays command and show how to verify run days specified in rule-based run cycles.

Details —See student notes.

Additional Information —

Transition Statement —

Fields Used for the Ocurrence Key



Tivoli software

Figure 4-21. Fields used for the Occurrence key TM402.0

Notes:

Each instance of an application in the long-term plan (LTP) and current plan (CP) is called an occurrence.

Each occurrence in the plans is a unique record in the plan files.

The fields used to form the key for occurrences are:

- The Application ID
- The Input Arrival date
- The Input Arrival time

Instructor Notes:

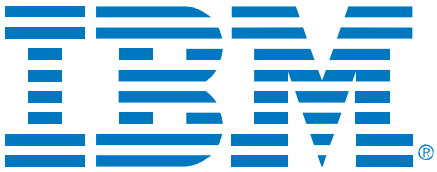
Purpose —To identify the fields that form the occurrence key.

Details —See student notes.

Additional Information —

Transition Statement —

4.4 Defining Operations



Topic 4.4

Defining Operations

Tivoli software

Figure 4-22. Defining Operations TM402.0

Notes:

Instructor Notes:

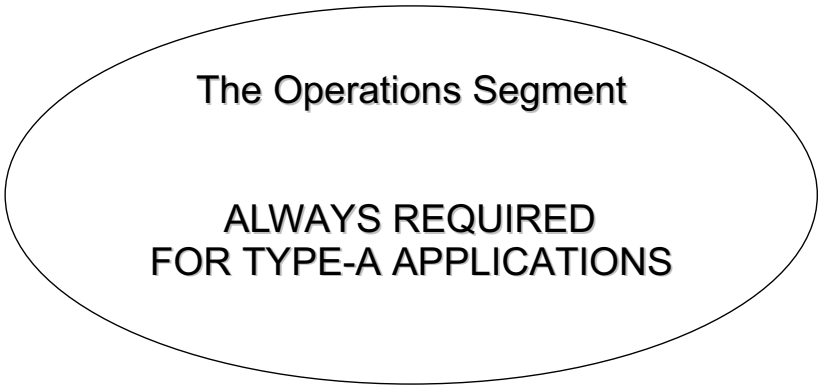
Purpose —To teach how to create operations.

Details —The operations segment holds the operations in type A applications. It is not used in type G applications.

Additional Information —

Transition Statement —This topic shows how to define operations.

The Operations Segment



Tivoli software

Figure 4-23. The Operations Segment TM402.0

Notes:

The operations segment holds the operations in type A applications. It is not used in type G applications.

Instructor Notes:

Purpose —Introduce the operations segment.

Details —See student notes.

Additional Information —

Transition Statement —You can display the layout of operations and their dependencies using either host graphics software or the Job Scheduling Console. The next two visuals demonstrate both methods.

Application Graphic - GDDM

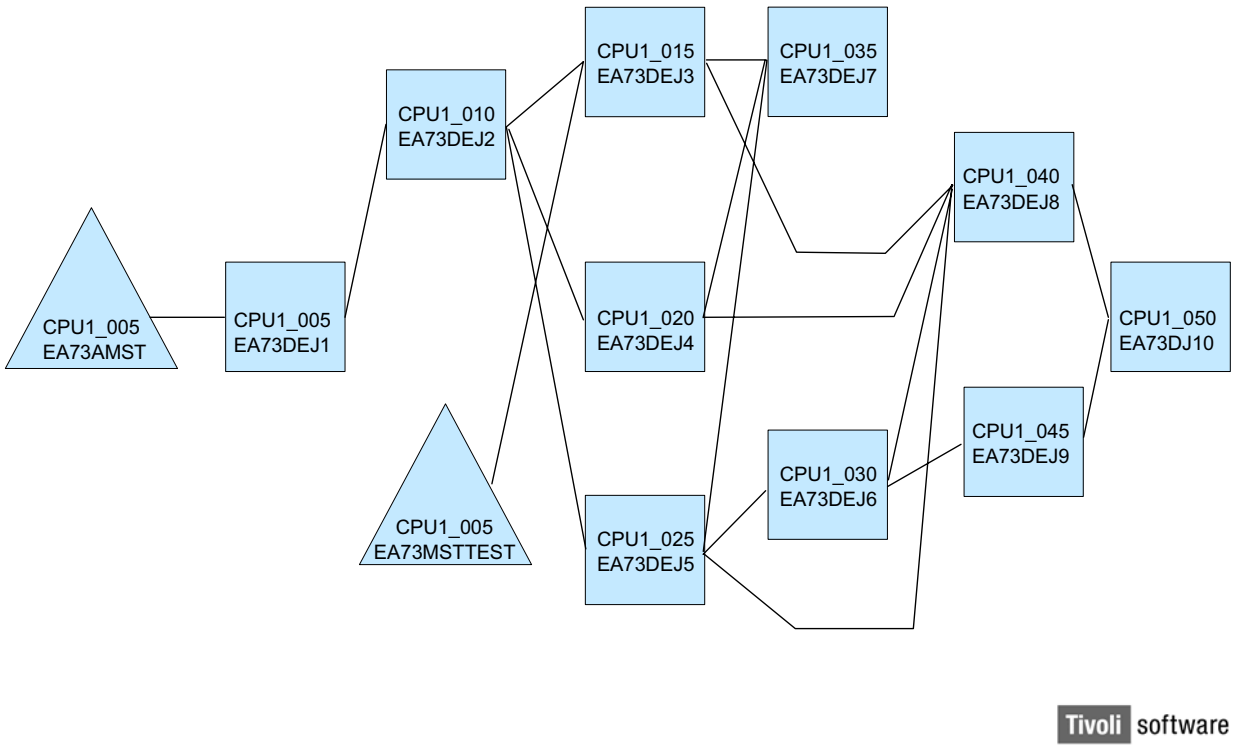


Figure 4-24. Application Graphic - GDDM TM402.0

Notes:

- TWS for z/OS uses the z/OS graphics software GDDM to display the layout of operations and their dependencies.
- GDDM supports three graphic shapes:
  - Boxes (the default)
  - Circles
  - Triangles.
- On most operations list panels, you can issue the GRAPH command to display the list of operations graphically.
- The requirements for enabling this display are:
  - GDDM must be installed on the z/OS host
  - Host Graphics must be enabled on the user's workstation.
- Once the graphic is displayed, you can customize your view for shapes, color and line type by issuing the ATTR command then specifying the appropriate attributes.
- The graphic is read from left to right.

- In this example, boxes represent computer operations within the application and triangles represent external dependencies. Accordingly, you can say that in this example, there are two external-predecessor dependencies to the application.
- The first line of text in each of the shapes is an operation ID.
- The second text line in the boxes is the job name associated with the operation.
- The second text line in the triangles is the application name in which the operation resides.
- In the application database, an application description graphic is a great tool for displaying an application and **all** of its immediate predecessors.

Instructor Notes:

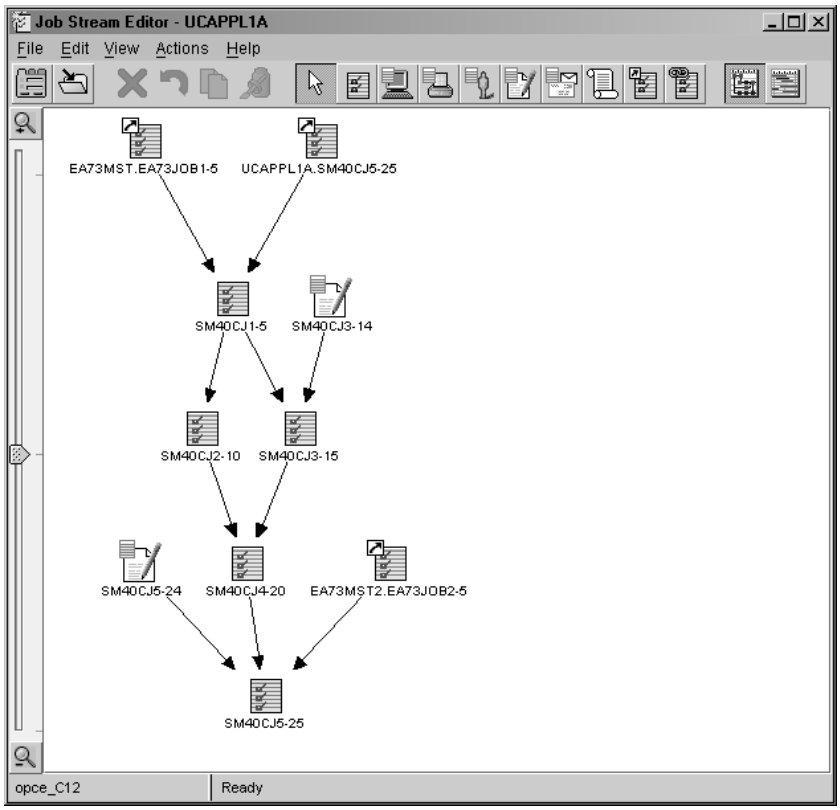
Purpose — Show an operation-dependencies view using GDDM.

Details —See student notes.

Additional Information —

Transition Statement —Let's look at a similar view using the JSC.

Application Graphic - JSC



Tivoli software

Figure 4-25. Application Graphic - JSC TM402.0

Notes:

This is an example of the display you see when you use the Job Scheduling Console to display an application in the TWS for z/OS application database.

Each object represents an operation.

The operations are displayed in a top to bottom fashion, with the first operation at the top

The arrows represent the dependency links between the operations.

Instructor Notes:

Purpose — Show an operation-dependencies view using the JSC.

Details —See student notes.

Additional Information —

Transition Statement —

Setting Up Operations

- Workstation name (R)
- Unique sequence number (R)
- Estimated duration (R)
- Jobname
- Dependencies between operations

Tivoli software

Figure 4-26. Setting Up Operations TM402.0

Notes:

- To define operations launch the OPERATIONS dialog panel by entering the OPER command on the General Information panel.
- Each operation is a task or job at a workstation, therefore a workstation must be specified.
- Every operation must have a unique number within the application; this sequence number is used to define the dependency links within the application.
- For planning purposes, every operation must be given an estimated duration (run time).
- All operations except those at WTO and nonreporting workstations must have an assigned job name.
- In applications with more than one operation, dependencies between the operations must be specified. Operation numbers are used to define the internal dependency links. On the OPERATIONS panel, you can specify up to eight internal dependencies.
- Define each operation with a workstation, an operation number, the estimated duration, and the job name.

```
GX     001    00:02:00     JOBJ1
CPU1   002    00:05:00     JOBJ1
CPU1   003    00:03:00     JOBJ2
```

- Then add the internal predecessor specifications. Use operation numbers to identify the predecessors.

```
GX     001    00:02:00    JOBJ1
CPU1   002    00:05:00    JOBJ1    001
CPU1   003    00:03:00    JOBJ2    002
```

Instructor Notes:

Purpose —Identify the minimum requirements for creating an operation.

Details —See student notes.

Additional Information —

Transition Statement —

Operations - Operations Text

```
------------------------------- OPERATIONS ---------------------- Row 1 of 1
Command ===>                                                Scroll ===> CSR

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details, J - Edit JCL
Enter the PRED command above to include predecessors in this list, or,
enter the GRAPH command to view the list graphically.

Application            : UDAPPL1A         My Application

Row  Oper      Duration  Job name  Operation text
cmd  ws   no.  HH.MM.SS
'''' ____ 000  ________  ________  ________________________
****************************** Bottom of data ********************************
```

Figure 4-27. Operations - Operations Text TM402.0

Notes:

TWS for z/OS has two panels on which to define operations in an application description.

- A text operations panel
- A predecessor operations panel

This is the text panel. On this panel, you specify up to 24 characters of descriptive text for each operation. This text is included in write to operator (WTO) messages issued for the operation.

To display this panel, you enter the **TEXT** command on the predecessor operations panel.

Instructor Notes:

Purpose — Identify the columns on operations coding panel.

Details —See student notes.

Additional Information —

Transition Statement —

Operations - Internal Predecessors

```
--------------------------------- OPERATIONS ---------------------- Row 1 of 1
Command ===>                                                Scroll ===> CSR

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details, J - Edit JCL
Enter the TEXT command above to include operation text in this list, or,
enter the GRAPH command to view the list graphically.

Application            : UDAPPL1A         My Application

Row  Oper      Duration  Job name  Internal predecessors           Morepreds
cmd  ws   no.  HH.MM.SS                                           -IntExt-
'''' ____ 000  ________  ________  ___ ___ ___ ___ ___ ___ ___ ___
******************************** Bottom of data ********************************
```

Figure 4-28. Operations - Internal Predecessors TM402.0

Notes:

This is the predecessor panel. You use this panel to define the internal dependencies between operations in the application description.

On this panel, you can specify up to eight internal predecessors for each operation.

To display this panel, you enter the **PRED** command on the text operations panel.

To define more than eight internal predecessors for an operation, you must enter the S - SELECT OPERATION DETAILS row command next to the operation.

Instructor Notes:

Purpose — Identify the columns on operations coding panel and show how to define internal predecessors.

Details —See student notes.

Additional Information —

Transition Statement —How would you define more than eight internal predecessors, and other details for an operation?

Specifying Operations Details

- External dependencies
- Extra internal dependencies
- WS resources and servers
- Automatic options
  - Example:
    - Time dependency
    - Highest acceptable return code
    - Centralized Script Management for a FTA workstation operation
- Feedback
- Times and dates for time dependency
- Operator instructions
- Restart and Cleanup options for the operation
- Extended Operation information

Tivoli software

Figure 4-29. Specifying Operations Details TM402.0

Notes:

- To define more than eight internal predecessors and additional details for an operation, you enter the OPERATION DETAILS row command next to the operation.
- External dependencies are linked at the operation level. The workstation name, operation sequence number, and name of the external application plus the job name must be entered. However, if you do not know all the names you can fill in the information you do have, using wild cards, and a list of matching jobs will be displayed from which you can select the correct one.
- If an operation has more than eight internal predecessors, you specify them here without the application name.
- WS Resources and Servers - the default is 1 PS and 0 WS Resource. If the operation is a job which needs n tape units; you can use this option to specify that detail. However, it is recommended that you use special resources instead.
- Special Resources - are symbolic names for any physical or logical resources needed for an operation. They are often used for data sets, or for tapes and cartridges. Typically

they are used to prevent allocation conflicts between jobs that share files or data sets. These resources can be SHARED between operations, or held EXCLUSIVELY by a single operation. Job tracking will not allow other operations wanting a Special Resource to start while another which holds it exclusively is running.

- Automatic Options - are used to define or specify more finite details such as:
  - Highest acceptable return code
  - Time dependency
  - Rerouting and restarting in case of workstation failure
  - Suppress the job if it is late getting started.
- Time - specifies a start time for time-dependent operations, and/or a deadline time for the operation.
- Operator Instructions - used to write online instructions for setting-up, starting, restarting, or rerunning the operation.
- Feedback instructions entered here will override the initialization defaults.
- Restart and Cleanup Options - specify the options for job restarts and data set cleanup for the operation.
- Centralized Script - This option applies to fault-tolerant workstations only. By default, scripts for jobs that run on fault-tolerant, workstations are not centralized.
- Extended Operation information - This is an optional 54 character field which allows you to specify extended jobname information that may be needed in the distributed environment.

Instructor Notes:

Purpose —Introduce the operation details options.

Details —See student notes.

Additional Information —

Transition Statement —

Operations Details Options

```
------------------------------ OPERATION DETAILS -----------------------------
Option ===>

Select one of the following:

1 PREDECESSORS         - List of predecessors
2 WS RES AND SERVERS   - Work station resources and servers
3 SPECIAL RESOURCES    - List of special resources
4 AUTOMATIC OPTIONS    - Job, WTO, and print options
5 FEEDBACK             - Feedback options
6 TIME                 - Time specifications
7 OP INSTRUCTIONS      - Operator instructions
8 JCL EDIT             - Edit JCL
9 CLEANUP OPTIONS      - Cleanup Options
10 EXTENDED INFO       - Operation Extended Info


Application             : UDAPPL1A         My Application
Operation               : UCCP 010
Duration                : 00.01.00
Jobname                 : SM40CJ1
Number of int preds     : 0
Number of ext preds     : 0
```

Figure 4-30. Operation Details Options TM402.0

Notes:

This panel shows the operation details options.

Instructor Notes:

Purpose —Show the operations details panel.

Details —See student notes.

Additional Information —

Transition Statement —

Operation Automatic Options

```
------------------------ JOB, WTO, AND PRINT OPTIONS ----------------------
Command ===>
Enter/Change data below:
Application             : UDAPPL1A         My Application
Operation               : UCCP 010
Job name                : SM40CJ1
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    ===> N            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       ===> _
```

Figure 4-31. Operation Automatic Options TM402.0

Notes:

JOB CLASS: You can specify the JES input class of the job that the operation represents. The job class that you specify here is only for documentation purposes.

This option does not apply to fault-tolerant workstations.

ERROR TRACKING: The option specifies whether TWS for z/OS should mark the operation Ended-in-error (E), if the job or started task defined in the operation fails.

HIGHEST RETURNCODE : This field specifies the highest acceptable return code from any step in the job or started task. If a return code for a step in the job or started task exceeds this value, the status of the operation is set to a E (ended-in-error).

For fault-tolerant workstations, the valid value is either zero or blank.

EXTERNAL MONITOR: This option specifies if the operation is monitored by an external product (for example, Tivoli Business Systems Manager). If you specify Y for this option, then monitoring is enabled.

For fault-tolerant workstations, the valid value is either zero or blank.

Centralized Script: This option applies to fault-tolerant workstations only. By default, scripts for jobs that run on fault-tolerant, workstations are not centralized. They are stored locally on the agent and a job definition is added to the SCRPTLIB data set allocated to the TWS for z/OS controller. In contrast, centralized scripts are stored in one or more data sets in the JOBLIB allocated to the controller. It provides features such as automatic job tailoring, job setup, automatic recovery, and TWS for z/OS user exit 01 functions.

CRITICAL: If you specify Y in the CRITICAL field, TWS for z/OS automatically sends a request to the z/OS Workload Manager (WLM) to promote a job or started task in the high-performance service class, appropriately defined for batch jobs in the WLM environment, whenever the conditions of the specified assist policy are reached.

POLICY: This option specifies the assist policy that TWS for z/OS uses to determine if to request a service class promotion for the job in the operation. The valid values for this field are:

- **L** Long duration -- If the job runs beyond its estimated duration
- **D** Deadline -- If the job is not finished when its deadline time is reached
- **S** Latest start time -- If the job is submitted after the latest start time
- **C** Conditional -- Determines whether to use Deadline or Latest start time policy.

This option does not apply to fault-tolerant workstations.

SUBMIT: Specifies whether TWS for z/OS should start the operation when all predecessors have been satisfied and all required resources are available.

HOLD/RELEASE: Use this option to control held jobs that are not submitted by TWS for z/OS. If you place jobs that are not submitted by TWS for z/OS in HOLD status (for example, by specifying TYPRUN=HOLD on the job card), and HOLD/RELEASE is set to Y, TWS for z/OS releases them according to its schedule when all dependencies are satisfied and when the requested resources are available.

If HOLD/RELEASE is set to N, TWS for z/OS releases a held job immediately without reference to its schedule.

You cannot set this option to N for fault-tolerant workstations.

TIME-DEPENDENT: In this field, you specify Y to tell TWS for z/OS not start the operation until a specified operation input arrival time is reached. If TWS for z/OS cannot start the operation at the specified input arrival time, the operation is considered late.

SUPPRESS IF LATE: Specify Y to stop this time-dependent operation being started if it is late. If you specify N, TWS for z/OS ignores the fact that the operation is late and tries to start it as soon as possible. The default is N.

DEADLINE WTO: If you specify Y for this option, TWS for z/OS issues the operator message EQQW776I if the operation is started after its deadline time. The WTO is issued only for z/OS operations that have status S (started).

The user-defined text that you specified on the Operations panel, using the TEXT command, is included in the WTO.

This option is applicable to z/OS workstation destinations only.

RESTARTABLE: This option specifies whether to restart the operation after TWS for z/OS reroutes it to an alternate computer workstation as a result of a failure or off-line condition of the primary computer workstation. This option applies to the operation only while it has a status S (started).

This option does not apply to fault-tolerant workstations.

REROUTABLE: This option specifies what action TWS for z/OS should take for this particular operation if the computer workstation that it is scheduled to run on is inactive and an alternate workstation has been specified. This option applies to the operation only when it is in status R (ready) or W (waiting). Once the operation is in status S (started), the RESTARTABLE option determines the action.

This option does not apply to fault-tolerant workstations.

Instructor Notes:

Purpose —Explain the attribute that can be assigned to an operation.

Details —See student notes.

Additional Information —

Transition Statement —

Defining a Job Setup Operation

- Must immediately precede CPU WS
- Jobname must match one at the CPU WS

Tivoli software

Figure 4-32. Defining a Job Setup Operation TM402.0

Notes:

- A job setup operation must have a computer operation with the same job name as its immediate successor.
- JCL preparation for the job setup operation should be performed on the workstation Ready List dialog.
- If the JCL is not available when the user makes the edit request, TWS for z/OS copies the JCL from the production JCL library into its own JCL repository. TWS for z/OS never changes JCL in the production library.
- When the operator has completed the preparation, TWS for z/OS will change the status of the operation to C (complete) so the successor computer operation becomes ready for processing.

Instructor Notes:

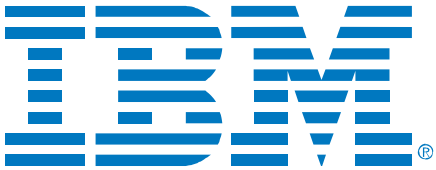
Purpose —Explain the restrictions in coding JCL prep operations.

Details —It is operationally best for JCL Prep operations not to be dependent on anything else. This means they appear on the JCL Prep WS Ready List as soon as the Current Plan is created, and thus can be prepped at any time.

Additional Information —Similar restrictions apply to Manual and Printer workstations.

Transition Statement —

4.5 Job Descriptions



Topic 4.5

Job Descriptions

Tivoli software

Figure 4-33. The Job Description TM402.0

Notes:

If you want only one main processor operation (job) in an application, you can create it with the Job Description dialog, which compresses most of the function of the Application Description panel into one panel by making some assumptions about the application that you are creating. An application that is created using the Job Description dialog, or that has the restrictions enforced by that panel, is called a *job description*.

Instructor Notes:

Purpose —Introduce and describe job descriptions.

Details —If you want only one main processor operation (job) in an application, you can create it with the Job Description dialog, which compresses most of the function of the Application Description panel into one panel by making some assumptions about the application that you are creating. An application that is created using the Job Description dialog, or that has the restrictions enforced by that panel, is called a job description.

Additional Information —

Transition Statement —

Characteristics of a Job Description

- It contains no more than three operations
  - One computer or WTO operation
  - The other two operations can be either manual completion or JCL preparation operations
- The job name of the computer operation is the same as the job Description ID
- If created through the AD dialog, its status is Active

Figure 4-34. Characteristics of a Job Description TM402.0

Notes:

- It contains no more than three operations
  - One computer or WTO operation
  - The other two operations can be either manual completion or JCL preparation operations
- The job name of the computer operation is the same as the job description ID.

Instructor Notes:

Purpose —Describe the restrictions and advantages of job descriptions.

Details —A single panel allows entry of all necessary data, including operation details and run cycle information, together with such information as Special Resources. Further panels are available to enter optional details.

Additional Information —All applications that meet the definition of a job description will be listed in the Job Description dialog by whichever method they are created.

If a Job Description is modified so it contains more than three operations, it will no longer be displayed in the JD dialog and the standard Application Dialog must be used.

Transition Statement —The next visual shows how easy it is to define a job description.

Creating a Job Description

```
------------------------------ CREATING A JOB --------------------------------
Command ===>

Edit data below:
Enter the RUN command above to select run cycles or enter the DETAILS
command to specify job details.

JOBNAME - TEXT      ===> _________          - ____________________________
OWNER: ID - TEXT    ===> TEAMC_____________ - TEAMC_____________________
CALENDAR ID         ===> ___________________    AUTHORITY GROUP ID ===> _________
VALID FROM - to     ===> 020308__ - 711231     DURATION           ===> 00.01.00
RUN TIME FROM - TO  ===> ______ - ______       TIME DEPENDENT     ===> N
WORK STATION        ===> UNCP                  PRIORITY           ===> _
JCL PREPARATION     ===> N JCL WS===> _____    HIGHEST RETURN CODE===> 0___
MANUAL INTERACTION  ===> ____________________  MANUAL WS===> _____
RUN CYCLES          ===> _________ - _____   _________ - _____   _________ - _____
                         _________ - _____   _________ - _____   _________ - _____
PREDECESSORS        ===> _________  _________  _________  _________  _________
SPECIAL          ===> ____________________________________________________ _______ _ _
RESOURCES              ____________________________________________________ _______ _ _
                       ____________________________________________________ _______ _ _
GROUP DEFINITION  ===> ___________________
```

Figure 4-35. Creating a Job Description TM402.0

Notes:

The job description dialog panel.

Instructor Notes:

Purpose —Describe the fields on the job description panel.

Details —See student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- List the segments that make up an Application
- List the types of Applications
- Create Applications with or without Run Cycles

Tivoli software

Figure 4-36. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 5. Dynamic Feedback

What This Unit is About

This unit covers the use and activation of the IBM TWS for z/OS dynamic feedback function.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe the Feedback function and its purpose
- List the parameters used for Feedback and Smoothing
- List the recommended parameter values

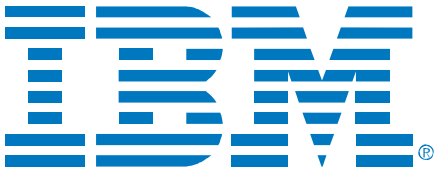
How You Will Check Your Progress

Accountability:

- Checkpoint questions

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 5: Dynamic Feedback

Tivoli software

Figure 5-1. Dynamic Feedback TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- Describe the Feedback function and its purpose
- List the parameters used for Feedback and Smoothing
- List the recommended parameter values

Tivoli software

Figure 5-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

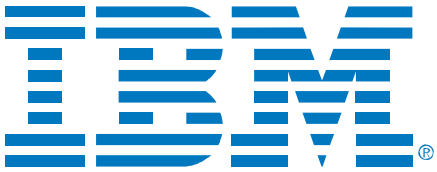
Purpose —

Details —

Additional Information —

Transition Statement —

5.1 Introducing Dynamic Feedback



Topic 5.1

Introducing Dynamic Feedback

Tivoli software

Figure 5-3. Introducing Dynamic Feedback TM402.0

Notes:

Instructor Notes:

Purpose —To introduce the topic.

Details —On initial setup of the application specification in the Application Database, the user enters the estimated run time of the jobs. Initially these times are likely to be quite inaccurate. Accurate timings are necessary for efficient scheduling.

TWS for z/OS has a facility to automatically adjust duration timings by referring to actual run times. This is known as Dynamic Feedback.

Transition Statement —

What Is Dynamic Feedback?

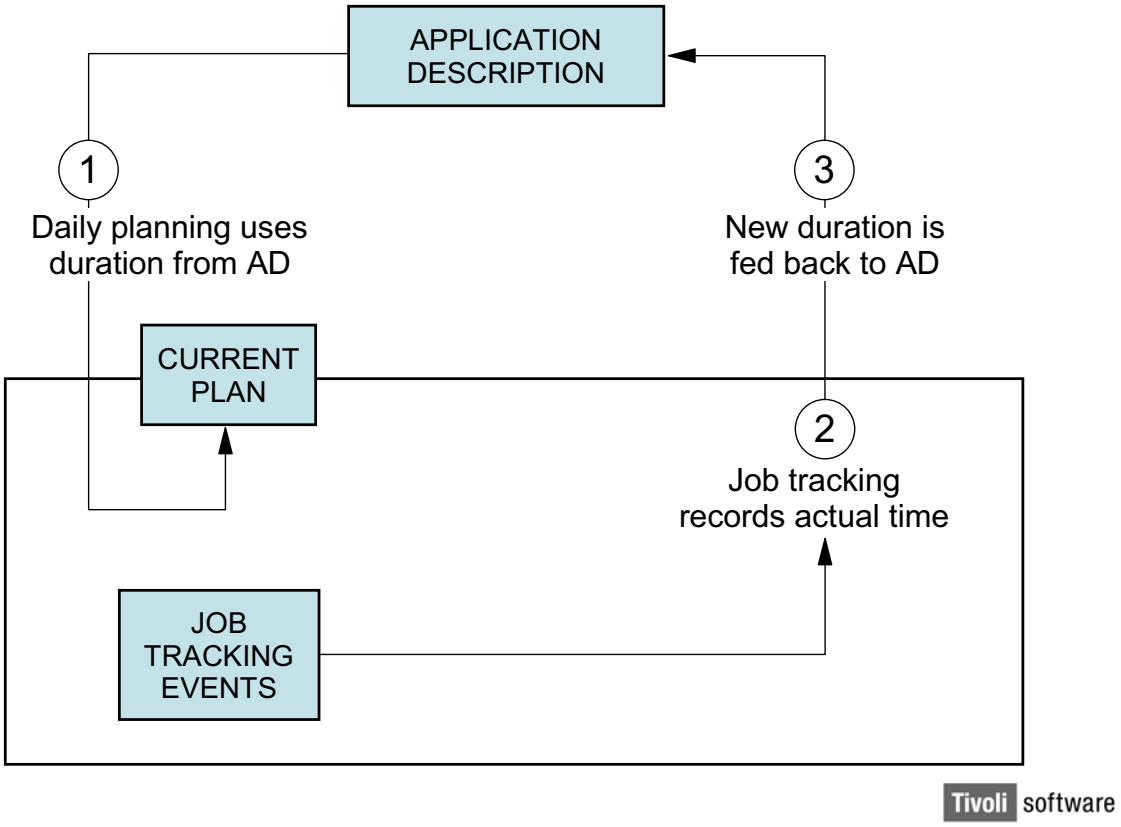


Figure 5-4. What is Dynamic Feedback? TM402.0

Notes:

- TWS for z/OS uses the estimated durations of operations to calculate the start times for operations in the current plan. The planned start-times generated will only be as accurate as the operations' durations. These times can be seen in online views and the daily-planning reports.
- TWS for z/OS can automatically adjust the durations of operations in the application database by capturing the actual run times of the jobs associated with the operations. This is known as *Dynamic Feedback*.
- Dynamic Feedback ensures that TWS for z/OS scheduling remains accurate by continually updating the operation run times in the Application Description (AD) database.
- Guided by user-specified criteria, the dynamic feedback function will use valid, actual run times, as recorded by Job Tracking, recalculate the estimated durations for operations, and update the application database with the new durations.
- When updates are necessary, they are done during the daily planning batch process.

- If the actual run time for any operation in the current plan falls outside the user-specified, criteria window TWS for z/OS will not update the duration; instead it will generate a report called a Missed Feedback Report. The report lists all operations with run times outside the limits. It also shows the estimated and actual run times for each of the errant operations.

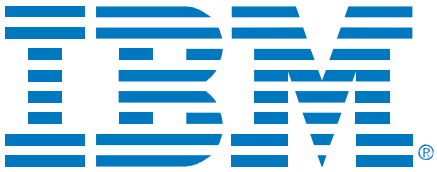
Instructor Notes:

Purpose —Introduce dynamic feedback.

Details —Dynamic Feedback ensures that TWS for z/OS scheduling remains accurate by updating the duration of operations in the Application Description database.

Transition Statement —

5.2 Factors that Influence Dynamic Feedback



Topic 5.2

Factors that Influence Dynamic Feedback

Tivoli software

Figure 5-5. Factors that Influence Dynamic Feedback TM402.0

Notes:

Instructor Notes:

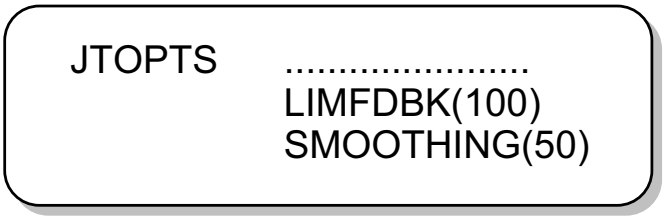
Purpose —

Details —

Additional Information —

Transition Statement —

Dynamic Feedback Parameters



- LIMFDBK
  - Sets range criteria for acceptable run times
- SMOOTHING
  - Uses a percentage of the difference between ART and OD

Tivoli software

Figure 5-6. Dynamic Feedback Parameters TM402.0

Notes:

- Two parameters, the smoothing factor (*SMOOTHING*) and the limit for feedback (*LIMFDBK*), control how measured durations are used.
- These parameters are keywords in the JTOPTS initialization statement of the controller.
- The default values are 100 for LIMFDBK and 50 for SMOOTHING.
- Any value you specify in the OPERATION DETAILS FEEDBACK for an operation overrides the installation default specified in JTOPTS.

Instructor Notes:

Purpose —Describe the TWS for z/OS parameters that influence dynamic feedback.

Details —See student notes.

Transition Statement —

Select a Feedback Limit

LF value	Result
100	No new estimated duration will be stored in the application description database
110	The new estimated duration will be stored if the measured duration is approximately between 90% and 110% of the old estimated duration
200	The new estimated duration will be stored if the measured duration is between half and double the old estimated duration
500	The new estimated duration will be stored if the measured duration is between one-fifth and five times the old estimated duration
999	The new estimated duration will be stored if the measured duration is between one-tenth and 10 times the old estimated duration

Tivoli software

Figure 5-7. Select a Feedback Limit TM402.0

Notes:

- The limit for feedback is a number, 100 through 999.
- The LIMFDBK value establishes the limits within which measured run times are considered valid.
- A value of 100 tells TWS for z/OS to disregard all run times and never update the application database.
- A value of 999 allows the greatest latitude in run times. It accepts any run time between one-tenth to ten times the estimated duration.
- A measured run time that falls outside the limits is ignored and the application description database is not updated.

Instructor Notes:

Purpose —Describe the values used for the feedback limit parameter

Details —See student notes.

Transition Statement —

Select a Smoothing Factor

Factor	Result
0	There will be no feedback
10	The new estimated duration will be the old estimated duration, plus one-tenth the difference between the measured and old estimated duration
50	The new estimated duration will be the old estimated duration, plus one-half the difference between the measured and old estimated duration
100	The measured duration replaces the old estimated duration
999	The new estimated duration will be the old estimated duration, plus 10 times the difference between the measured and old estimated duration

Tivoli software

Figure 5-8. Select a Smoothing Factor TM402.0

Notes:

- Having excluded exceptional run times by means of the Limit for Feedback will not stop differences in scheduled run times. Run Times are likely to seesaw back and forth.
- TWS for z/OS can smooth out wild swings in durations by taking the actual run time, comparing it with the previous estimated run time and calculating a new time based on the difference, and a parameter specified by the user.
- You use the Smoothing Factor parameter to dampen extreme changes in run times.
- The smoothing factor is a number, 0 to 999.
- The SMOOTHING value determines how much a measured duration will change existing values in the application description database.
- A value of 0 tells TWS for z/OS to disregard all run times and never update the application database.
- Values between 1 and 99 tells TWS for z/OS to replace the estimated duration with the sum of the old duration and a percentage of the difference between the actual run time and the old duration.

- A value of 100 tells TWS for z/OS to replace the estimated duration with the actual run time.
- A value of 999 tells TWS for z/OS to replace the estimated duration with the sum of the old duration and ten times the difference between the actual run time and the old duration.

Instructor Notes:

Purpose —Describe the values used for the smoothing parameter.

Details —See student notes.

Transition Statement —

Formula and Terminology

ND = (OD + (ART - ED) x SF/100)

ND = New Duration

OD = Old Duration

ED = Estimated Duration - same as OD

ART = Actual Run-Time

SF = Smoothing Factor

Figure 5-9. Formula and Terminology TM402.0

Notes:

The new estimated duration is calculated as follows:

- ND = OD + ((ART- ED) * SF/100) where:
  - **ND** is the new estimated duration to be stored in the AD database.
  - **OD** is the old estimated duration stored in the database.
  - **ED** is the old estimated duration that was used by daily planning to calculate the operations' planned start times.
  - **ART** is the actual run time.
  - **SF** is the smoothing factor.

Instructor Notes:

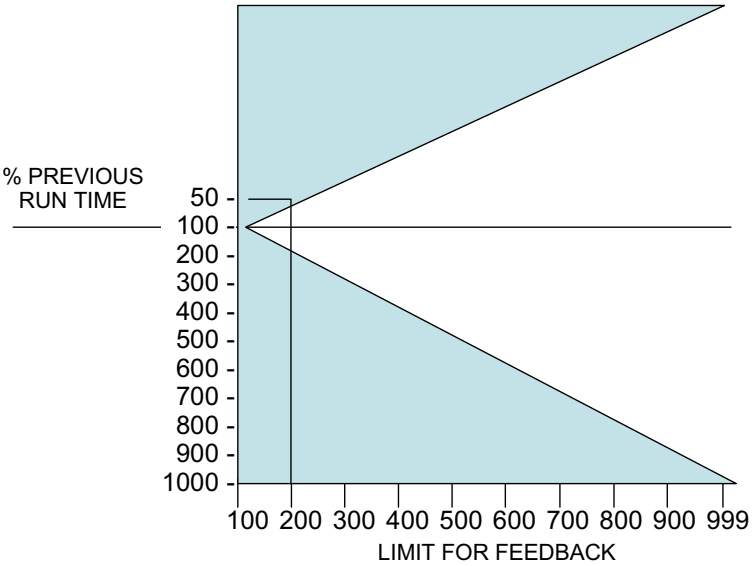
Purpose —Introduce and explain the how TWS for z/OS calculates the new duration for an operation.

Details —See student notes.

Transition Statement —Do you want the estimated to be updates in all circumstances?

Recommended Window for Feedback Limit

- Use to define the limit outside of which feedback is not to be used
  - Value in range 100 - 999
  - 100 = No update (this is the default)



- Recommended LIMFDBK = 200
  - For example, disregard the feedback run times greater than twice or less than half of previous run time

Tivoli software

Figure 5-10. Recommended Window for Feedback Limit TM402.0

Notes:

- The recommended value for your installation-wide feedback limit is 200.
- This value will allow TWS for z/OS to accept any run time between one-half to twice the estimated duration of operations in the current plan.
- In this example, the center line is the previous run time and the feedback value is 200.
- The X axis shows the Feedback Limit parameters.
- Run Times in the shaded area will be disregarded.

Instructor Notes:

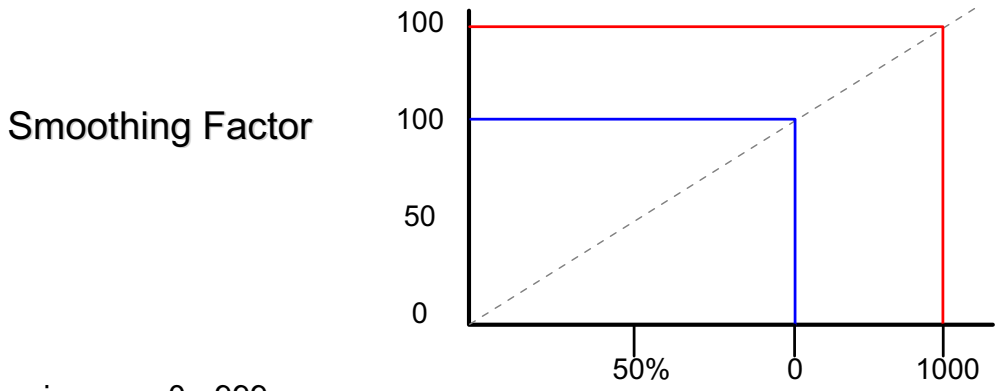
Purpose —Identify recommended value for the feedback limit parameter.

Details —See student notes.

Transition Statement —Feedback is used in conjunction with smoothing.

Recommended Value for Smoothing Factor

- New Duration = Old Duration + (Actual Duration - Old Duration) x sf/100



- Value in range 0 - 999
- 0 = No update
- 100 = Use actual duration
- 999 = New Duration = OD + 10 x (ND - OD)
- Recommended smoothing factor = 50

For example, ND will be OD plus 50% of the difference between old and current duration

Tivoli software

Figure 5-11. Recommended Value for Smoothing Factor TM402.0

Notes:

- The recommended value for installation-wide smoothing factor is 50.
- This is the default and it tells TWS for z/OS to replace the estimated duration with the sum of the old duration and fifty percent of the difference between the actual run time and the old duration.

Instructor Notes:

Purpose —Identify recommended value for the smoothing parameter.

Details —See student notes.

Transition Statement —

Capturing Operation Duration

Using TWS for z/OS to get actual durations

1. Use workstation operation default times.
2. Set SMOOTHING(100) to feed back actual durations.
3. Set LIMFDBK(999) to accept all values between 1/10 x OD and OD x 10.
4. Let all applications run at least once.
5. Modify smoothing factor and limit for feedback as required to get to recommended values.

Tivoli software

Figure 5-12. Capturing Operation Durations TM402.0

Notes:

- It is a good idea to make use of Dynamic Feedback from the very start.
- Use best-guess estimates in the initial setup then specify:
  - A Feedback Limit of 999
  - A Smoothing Factor of 100
- After all your applications have been processed, you can change the feedback limits to your own preferred settings.
- TWS for z/OS will continue to take into account the changes in run times as applications grow, and hardware becomes more efficient.
- It will keep your schedules accurate without any more effort on your part.

Instructor Notes:

Purpose —Summarize feedback.

Details —At this stage I would show the quick and dirty way to find how accurate the estimated run times are. Look at Workstation Communication Status (4.4 from primary option menu). Look at the CPU workstation. The estimate and actual run times should not be wildly different. If they are it means the estimates are not accurate and the steps on the foil could be carried out.

Also show in the AD dialog where the parameters can be entered. Then give the good news that the chosen defaults can be defined in the initialization parameters.

Any operations that fall outside the limit for feedback will be reported on in the current plan reports.

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- Describe the Feedback function and its purpose
- List the parameters used for Feedback and Smoothing
- List the recommended parameter values

Tivoli software

Figure 5-13. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 6. The TWS for z/OS Plans

What This Unit is About

This unit covers the relationship between the TWS for z/OS plans, and how to create and maintain the plans.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe the functions of TWS for z/OS plans
- Describe the relationship between the databases and the plans
- Describe the relationship between the long-term plan and the current plan

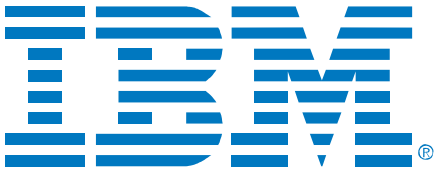
How You Will Check Your Progress

Accountability:

- Checkpoint questions

References

SH19-4543 *Tivoli Workload Scheduler for z/OS Planning and Scheduling the Workload Version 8.1*



Unit 6: TWS for z/OS Plans

Tivoli software

Figure 6-1. TWS for z/OS Plans TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- Describe the functions of TWS for z/OS plans
- Describe the relationship between the databases and the plans
- Describe the relationship between the long-term plan and the current plan

Tivoli software

Figure 6-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

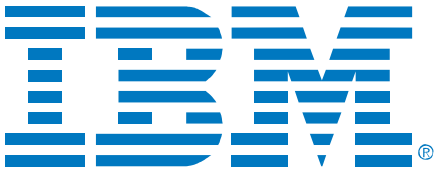
Purpose —

Details —

Additional Information —

Transition Statement —

6.1 Databases and Plans Relationship



Topic 6.1

Databases and Plans Relationship

Tivoli software

Figure 6-3. Databases and Plans Relationship TM402.0

Notes:

Instructor Notes:

Purpose —This unit covers the long term and current planning functions, and how they are used to produce the plans.

Details —

Additional Information —

Transition Statement —

Databases and Plans

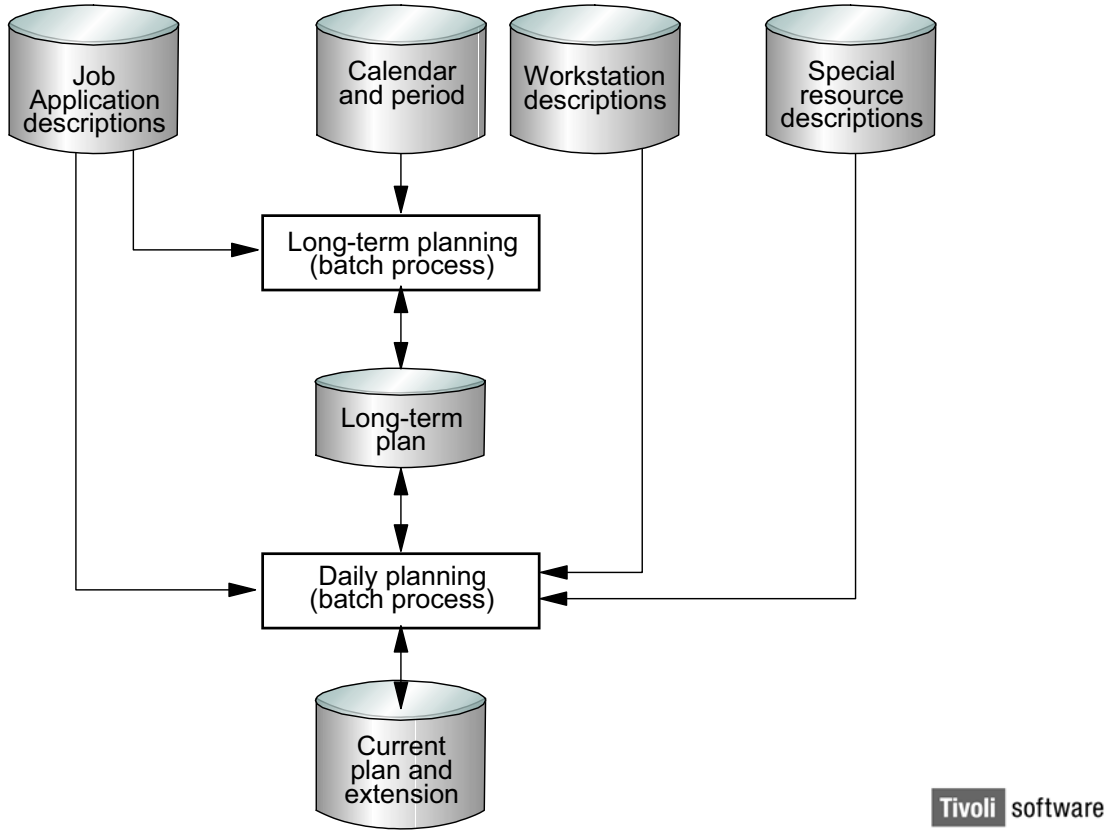


Figure 6-4. Databases and Plans TM402.0

Notes:

- TWS for z/OS maintains two plans, the long-term plan and the current plan.
- The long-term plan (LTP) is created from the application, calendar, and period databases.
- The current plan (CP) is created from the LTP, application (AD), special resource (RD), and the workstation (WS) databases.
- The CP holds occurrences created by the daily planning batch function. The online CP file can be one of two alternating data sets. The ddnames for these data sets are EQQCP1DS and EQQCP2DS.
- You can use dialog option 6.6 to display the BROWSING GENERAL CURRENT PLAN INFORMATION panel and see which current plan file is in use.

Instructor Notes:

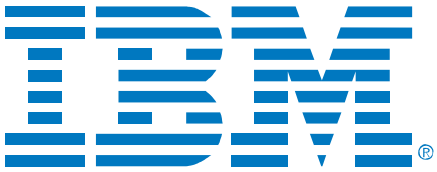
Purpose —To show the relationship between the databases and the long-term and current plans.

Details —See the student notes.

Additional Information —

Transition Statement —

6.2 The Long-Term Plan



Topic 6.2

The Long-Term Plan

Tivoli software

Figure 6-5. The Long-Term Plan TM402.0

Notes:

Instructor Notes:

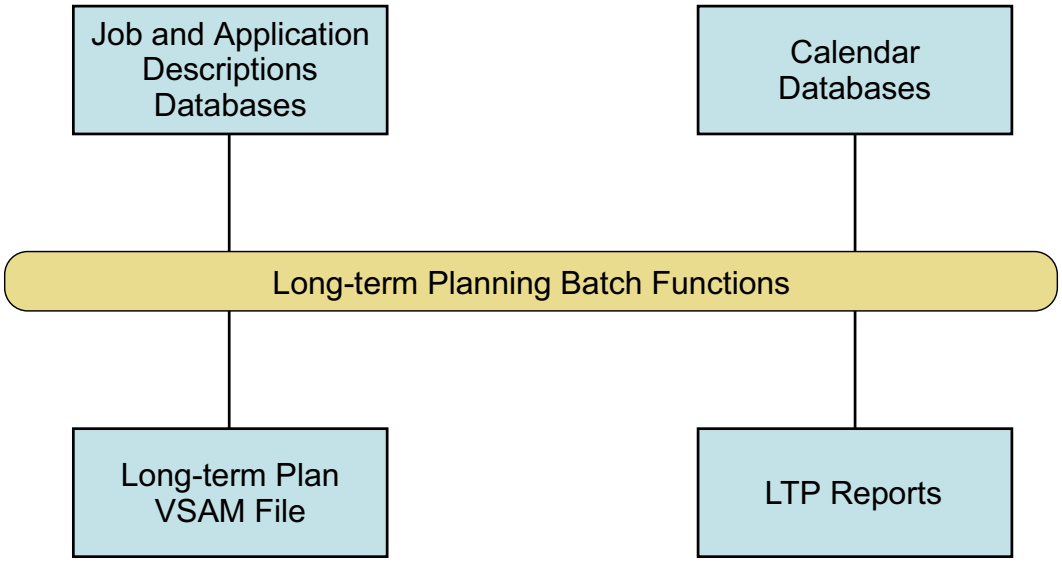
Purpose —Explain the LTP and its functions.

Details —

Additional Information —

Transition Statement —

Long-Term Planning Overview



Tivoli software

Figure 6-6. Long-Term Planning Overview TM402.0

Notes:

- The long-term planning process uses information from the databases to create a forecast of work scheduled to run.
- The Application and Job Description databases are scanned for all active application and job descriptions with valid run-cycle information.
- The relevant run cycles, and rules are used to calculate the run dates and generate occurrences.
- The calendar associated with the application is checked and occurrences are rescheduled according to the Free Day Rule.
- TWS for z/OS saves the generated occurrences in the long-term plan file.
- Batch jobs that maintain the LTP are:
  - LTC - Long-Term Plan Create -- used to create a LTP the first time TWS for z/OS is initialized, and after a LTP Refresh.
  - LTX - Long-Term Plan Extend --used to extend an existing LTP. If changes are detected in the databases, it also modifies existing occurrences to reflect the changes. This process is typically included in the TWS for z/OS batch schedule.

- LTMOA - Long-Term Plan Modify -- used to modify occurrences in the LTP to reflect changes made in the databases. The end date of the plan is not changed. LTP occurrences that have been modified through the dialog, and occurrences with input arrival times earlier than the end time of the current plan are not modified.

Instructor Notes:

Purpose —Recap the LTP.

Details —See the student notes.

Additional Information —

Transition Statement —Let us look at how occurrences are created in the LTP.

How the LTP Is Generated

The Long-term Planning Algorithm

- Determines range of LTP
- Determines free days for each application in turn
- Finds calendar period
- Generates occurrences
- Reschedules using free day rule
  - (If an occurrence is generated on a free day)
- Bypasses free days when calculating deadline
  - For example, daily application with deadline next day, Friday's run deadline would be Monday (if weekend free)

Tivoli software

Figure 6-7. How the LTP Is Generated TM402.0

Notes:

- TWS for z/OS examines every application and job description for run-cycle information. If an application is a member of a group, TWS for z/OS extracts the run-cycle and calendar information from the group definition.
- When the planning process finds a valid run cycle, it checks for generated dates within the range of the long-term plan.
- TWS for z/OS creates an occurrence in the long-term plan for each valid run cycle in the application or job description.
- The planning process checks the defined calendar to determine if the *free-day* rule should be applied.
- TWS for z/OS detects and reports a duplicate occurrence, but does not store it in the long-term plan.
- When the planning process creates an occurrence, the operations are examined for external predecessors. If an operation has one or more external predecessors, the planning process tries to resolve the dependency in the long-term plan. The

dependency links the closest occurrence, that is, the occurrence with an equal or the nearest, earlier input arrival time. If a predecessor application occurrence does not exist in the long-term plan, no dependency is created. In this case, the planning process issues a message to indicate a potential problem.

Instructor Notes:

Purpose —To explain the process of generating occurrences in the LTP.

Details —See the student notes.

Additional Information —

Transition Statement —

Producing the Long-Term Plan

```
----------------------- MAINTAINING THE LONG TERM PLAN --------------------
Option ===>

Select one of the following:

1 ONLINE          - Occurrence utility:
                      Browse, Create, Delete, List, and Modify
                      Occurrences and Dependencies in the long term plan.
                      Job setup for occurrences in the long term plan.

2 BATCH           - Plan utilities:
                      Modify, Create and Extend the long term plan
                      from run-cycle information.
                      Make a Trial long term plan.
                      Print long term plan.

3 ADD             - Create an occurrence in the long term plan

4 STATUS          - Display status of the long term plan

5 SET DEFAULTS    - Set defaults to be used for browsing long term plan
```

Figure 6-8. Producing the Long-Term Plan TM402.0

Notes:

- The long-term plan is the reservoir of planned work that is used by daily planning to create the current plan.
- The functions you use to produce the long-term plan are batch functions.
- On the MAINTAINING THE LONG TERM PLAN panel, select the BATCH to:
  - Create a new long-term plan
  - Modify the existing long-term plan
  - Extend the existing long-term plan
  - Create Trial long-term plans
  - Print the long-term plan

Instructor Notes:

Purpose —Introduce and explain the LTP dialog options.

Details —See the student notes.

Additional Information —

Transition Statement —

Long-Term Plan Batch-Functions

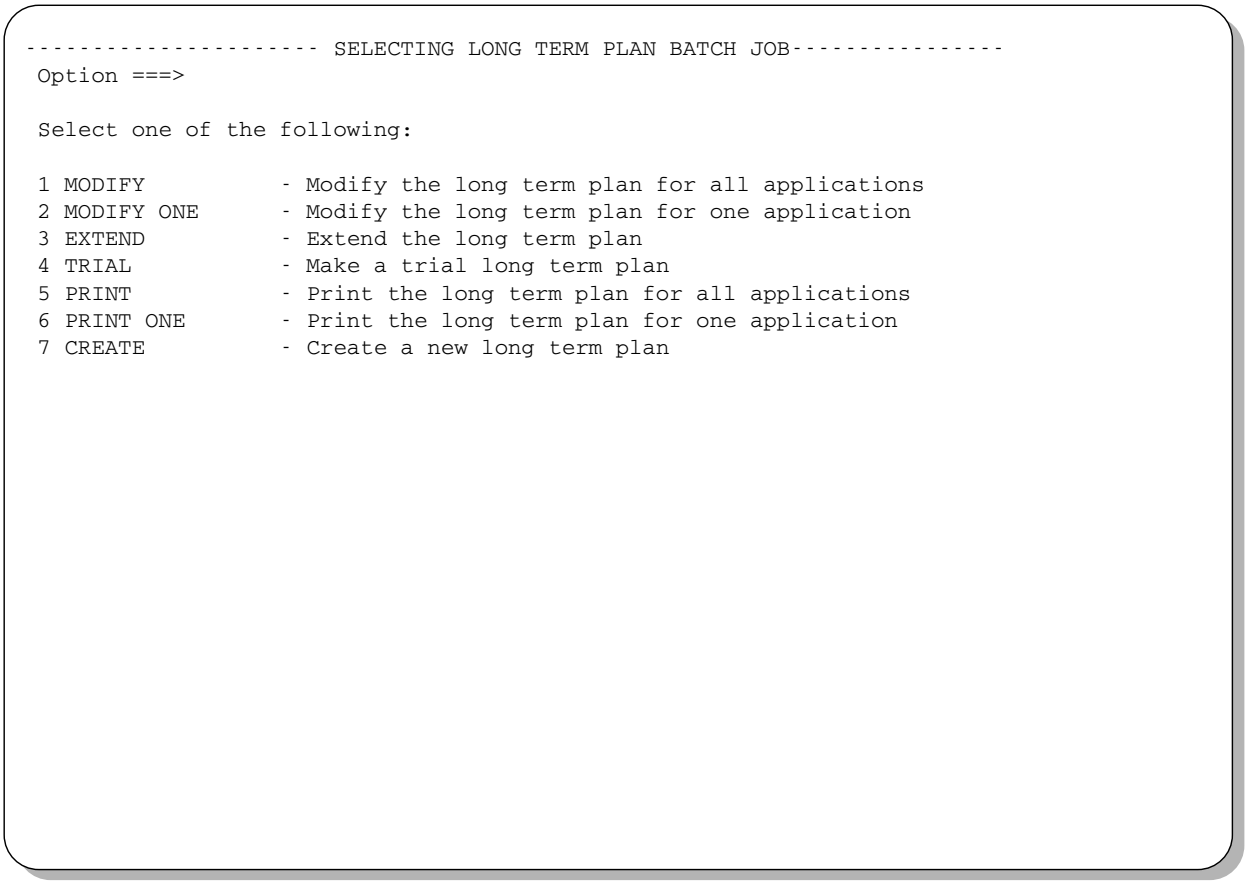


Figure 6-9. Long-Term Plan Batch Functions TM402.0

Notes:

- MODIFY - Long-Term plan modify all -- This function modifies the contents of the LTP to reflects changes made in the TWS for z/OS databases. It modifies only occurrences that have not been modified through the LTP dialog, and occurrences with input arrival times later than the end time of the current plan; it does not change the end-date of the existing LTP. It also resolves external dependency links.
- MODIFY ONE - Long-Term plan modify one -- With one exception, this function does the same things as the MODIFY but for a single named application. It does not resolve external dependency links.
- EXTEND - Long-Term plan extend -- Along with all the functions of the MODIFY, the EXTEND changes the LTP end-date and adds more days based on user specification. Users typically automate the long-term plan extension by scheduling the batch job in a TWS for z/OS application.
- TRIAL - Make a trial long-term plan -- This function allows users to create a test long-term plan on paper without affecting the existing, production long-term plan. The

user specifies start and end dates and the plan is generated with the results saved in a sequential file of the user's choice.

- PRINT and PRINT ONE -- These names of these functions are self-explanatory.
- CREATE - Create a new LTP --Use this function the first time TWS for is initialized and after a LTP Refresh only. You cannot use the create function when a current plan exists.

Instructor Notes:

Purpose —Provide a detailed explanation of the LTP batch functions.

Details —See the student notes.

Additional Information —

Transition Statement —

Creating and Extending the Long-Term Plan

```
------------------------- CREATING THE LONG TERM PLAN -------------------------
Command ===>

Enter/Change data below:

Long term plan:

 START              ===>  020505__   Date in format YYMMDD

 END                ===>  020511__   Date in format YYMMDD




----------------- EXTENDING THE LONG TERM PLAN ---------------------
Command ===>

Enter/Change data below:

Current end           : 020511

NEW END DATE       ===> _________    New long term plan end date
                                     in format YYMMDD

EXTENSION LENGTH   ===> _____        DDDD     Extend plan by
```

Figure 6-10. Creating and Extending the Long-Term Plan TM402.0

Notes:

- The 24-hour day is the minimum time for which you can create a long-term plan.
- When creating a long-term plan, specify start and end-dates for the plan.
- Invoke the long-term plan extend batch function to perpetuate an existing long-term plan.
- The long-term plan extend function requires either an extension length in days, or a specific end-date. Use EXTENSION LENGTH when creating a batch job to automate the long-term plan extension.

Instructor Notes:

Purpose —Show how to specify the creation and extension lengths for the LTP.

Details —See the student notes.

Additional Information —

Transition Statement —

Generate JCL for Batch Function

```
 ----------------------- GENERATING JCL FOR A BATCH JOB -----------------------
Command ===>

Enter/change data below and press ENTER to submit/edit the JCL.

JCL to be generated for: Create a new long term plan

SYSOUT CLASS       ===> _           (Used only if output to system printer)
LOCAL PRINTER NAME ===> _________   (Used only if output on local printer)
                                    (Used only if CLASS is blank)
DATASET NAME       ===> INGC100.OPCE.LTCRE.LIST_____________________
                                    (Used only if CLASS and LOCAL PRINTER
                                    are both blank). If blank default name
                                    used is INGC100.OPCE.LTCRE.LIST

SUBMIT/EDIT JOB    ===> E           S to submit JOB, E to edit

Job statement    :
 ===> //OPCBATCH  JOB (9999),'OPC BATCH JOB',REGION=0M,_____________________
 ===> //   CLASS=A,MSGCLASS=Q,MSGLEVEL=(1,1),NOTIFY=&SYSUID_________________
 ===> ____________________________________________________________________
 ===> ____________________________________________________________________
```

Figure 6-11. Generate JCL for Batch Function TM402.0

Notes:

- TWS for z/OS generates a listing whenever you run one of its batch functions. The listing shows the results of the function. For example, you get a full listing of the long-term plan file contents when you create a long-term plan.
- The listing can be sent to one of these output destinations:
  - A system printer -- SYSOUT CLASS
  - A local printer -- LOCAL PRINTER NAME
  - A data set -- DATASET NAME
- When invoked from the dialog panel, users typical send the output to a data set. If the DATASET NAME field is blank, TWS for z/OS creates an output data set with the user's TSO ID as the high-level qualifier, the controller subsystem name as the next qualifier, the function being run, and LIST as the low-level qualifier.
- The first time you use the dialog to invoke a TWS for z/OS batch function, you should type **E**, EDIT, in the SUBMIT/EDIT JOB field. This allows you to review and save the JCL before you submit it.

- You must also supply a job card for your JCL. You will have to enter it the first time only. When you terminate your session normally, ISPF saves the job card in your ISPF profile data set. EQQAPROF is the member that holds the job card and other TWS for z/OS user options.
- The batch job is generated when you press Enter on this panel.

Instructor Notes:

Purpose —Describe the input fields on the dialog panel used to generate batch-job JCL for the TWS for z/OS batch functions.

Details —See the student notes.

Additional Information —

Transition Statement —What do we see in the LTP?

Contents of the Long-Term Plan

```
LONG TERM PLAN OCCURRENCES:

    Application id   Owner id         Input arrival  Deadline       P C Pre Suc
                                     date     time  date     time
   SM4P#DLYA#OPC00  INSTRUCTOR       020505   05.00 020506   06.00 5 N   0   1
   OPCMAINT         TEAM#            020506   06.00 020507   06.00 5 N   0   0
   SM4P#DLYA#OPC00  INSTRUCTOR       020506   05.00 020507   06.00 5 N   1   2
   SM4P#DLYW#OPC01  INSTRUCTOR       020506   06.00 020507   06.00 5 N   2   0
```

Figure 6-12. Contents of the Long-Term Plan TM402.0

Notes:

- The LTP File contains:
  - A list of application occurrences in chronological order (input arrival date and time).
  - The deadline date and time of each occurrence. TWS for z/OS calculates the date from the deadline day value specified on the application run cycle.
  - Resolved external dependencies - **Pre**decessors and **Suc**cessors.
  - The status of each occurrence - **C**ompleted -- **Y**es or **N**o.
- The LTP does not store all the Application Description (AD) information in its own file; if you browse an occurrence in the LTP, you are looking at the AD database.

Instructor Notes:

Purpose —Discuss the contents of the LTP.

Details —The LTP File contains the basic timetable - a list of application occurrences by date and input arrival time. (Remember the input arrival time is the time taken from the run cycle, and is one of the fields that forms the occurrence key).

The list also includes external dependencies.

The LTP does not store all the Application Description (AD) information in its own file; if you browse an occurrence in the LTP, you are looking at the AD database.

The list also records whether the occurrence has been marked as completed.

Additional Information —

Transition Statement —Let's see how the LTP is extended.

Long-Term Plan Extension

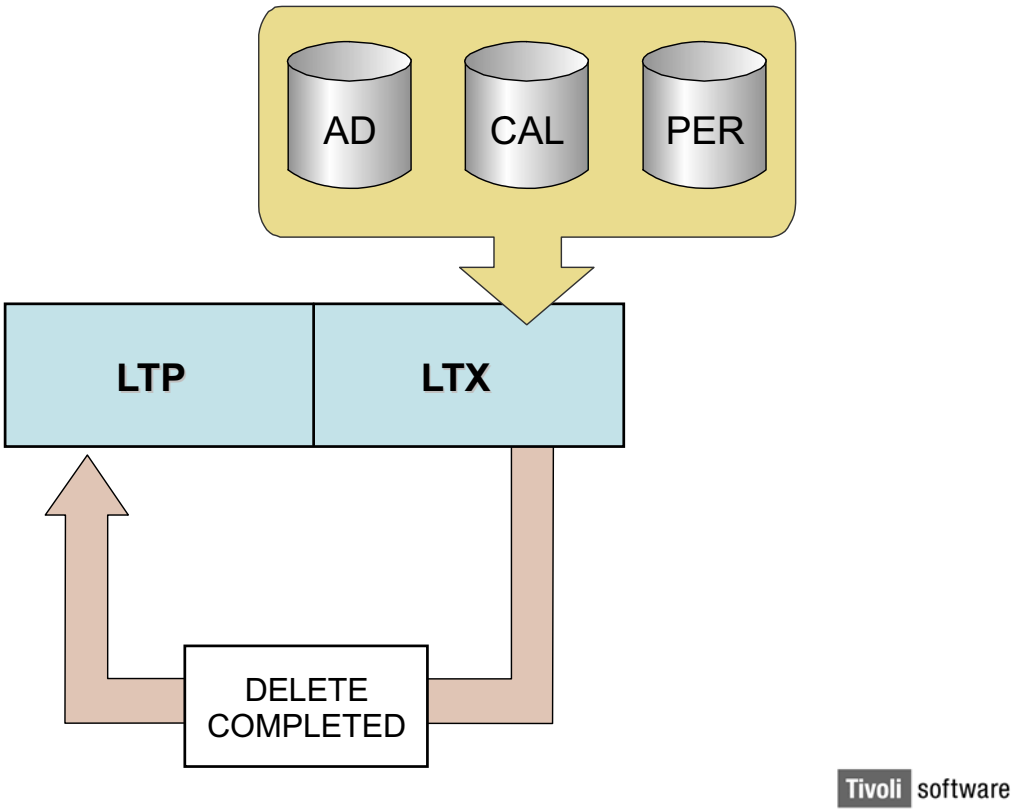


Figure 6-13. Long-Term Plan Extension TM402.0

Notes:

- The LTP extension batch process:
  - Searches the AD database, and uses valid run cycles to schedule application occurrences
  - Checks the Calendar database to ascertain free days, and reschedules according to the free day rule
  - Resolves external dependencies
  - **Deletes completed occurrences**
- When you create a *new current plan (NCP)* part of the process is to update the long-term plan with any occurrences that were completed in the previous current plan.
- When you extend or modify the long-term plan in batch, completed occurrences are removed.
- The LTP batch function deletes completed occurrences in scheduled-day lots. That is, all the day's occurrences or none are removed for a particular day.

- If you have an occurrence in your long-term plan that is not completed, all occurrences scheduled for that day and all following days are retained until that occurrence is completed.
- To avoid excessive growth of your long-term plan file, you should check for uncompleted occurrences on a regular basis.

Instructor Notes:

Purpose —Discuss the LTP extension batch process.

Details —See the student notes.

Additional Information —

Transition Statement —The LTP create, extend, and modify all batch functions create external dependency links based on AD definitions.

LTP Dependencies

Dependencies in the long-term plan

- Determines range of LTP
- Resolved by scanning backwards in time until first predecessor occurrence is found
- Start point for the scan is operation input arrival time
  - (if Time-Dependent job) or the input arrival time of the occurrence

Tivoli software

Figure 6-14. LTP Dependencies TM402.0

Notes:

- LTP dependencies refer to external dependencies only.
- Only the long-term plan **create**, **extend**, and **modify all** batch functions resolve external dependencies.
- When resolving dependencies, TWS for z/OS uses the long-term plan that already exists. Occurrences that fall within the current plan are also considered in the dependency resolution process.
- After it generates its list of occurrences, the LTP batch job scans backwards to find the most appropriate predecessor application. If an input arrival time is specified for the dependent operation, the operation IA is used; otherwise, TWS for z/OS uses the application occurrence IA from the run cycle.
- In all cases, a dependency will be resolved if the successor's input arrival is later than, or equal to, the predecessor's input arrival.
- If the AD has several external dependencies at different points they will all be resolved.

Instructor Notes:

Purpose —Describe how external dependencies are generated.

Details —See the student notes.

Additional Information —

Transition Statement —

Application Dependencies

```
---------------------------- BROWSING DEPENDENCIES ---------------- Row 1 of 3
Command ===>                                                  Scroll ===> CSR

View data below:

Application           : SM4P#DLYA#OPC00  LTPEXTEND DAILY BY 1 DAY
Input arrival         : 020506 05.00
Deadline              : 020507 06.00

Dep     Application id   Input arrival   Complete  Manually Deleted
Type                     date     time             created
P       SM4P#DLYA#OPC00  020505   05.00  N         N        N
S       SM4P#DLYW#OPC01  020506   06.00  N         N        N
S       SM4P#DLYA#OPC00  020507   05.00  N         N        N
****************************** Bottom of data ********************************
```

Figure 6-15. Application Dependencies TM402.0

Notes:

- This example shows the predecessor occurrence has the same input arrival time as the successor.
- It also shows that TWS for z/OS notes and tracks and any dependency-related dialog actions in the LTP.

Instructor Notes:

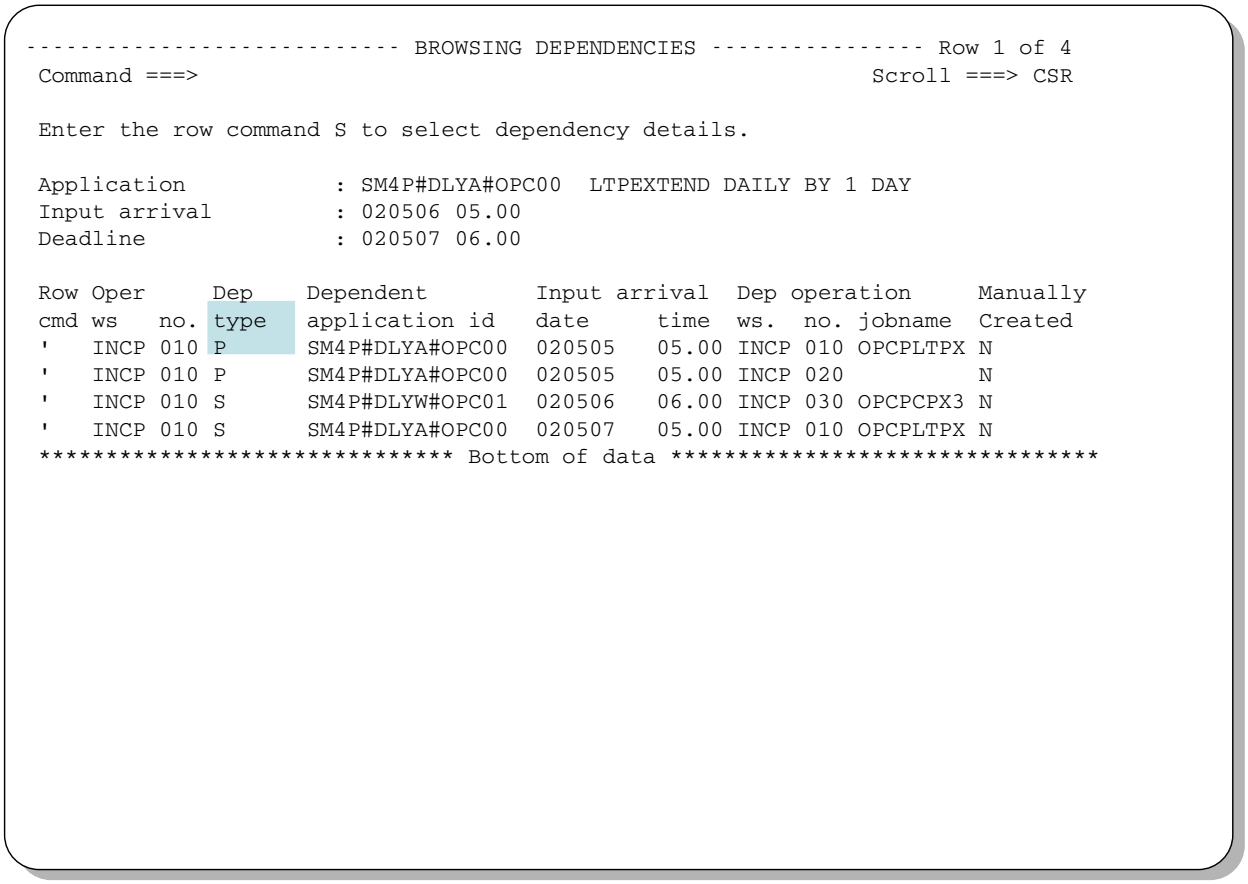
Purpose —Show an example of the LTP that you use to browse external dependencies.

Details —See the student notes.

Additional Information —

Transition Statement —

Operation Dependencies



```
---------------------------- BROWSING DEPENDENCIES ---------------- Row 1 of 4
 Command ===>                                                Scroll ===> CSR

 Enter the row command S to select dependency details.

 Application           : SM4P#DLYA#OPC00  LTPEXTEND DAILY BY 1 DAY
 Input arrival         : 020506 05.00
 Deadline              : 020507 06.00

 Row Oper     Dep    Dependent         Input arrival  Dep operation     Manually
 cmd ws   no. type   application id    date     time  ws.  no. jobname  Created
 '   INCP 010 P      SM4P#DLYA#OPC00   020505   05.00 INCP 010 OPCPLTPX N
 '   INCP 010 P      SM4P#DLYA#OPC00   020505   05.00 INCP 020          N
 '   INCP 010 S      SM4P#DLYW#OPC01   020506   06.00 INCP 030 OPCPCPX3 N
 '   INCP 010 S      SM4P#DLYA#OPC00   020507   05.00 INCP 010 OPCPLTPX N
 ****************************** Bottom of data ********************************
```

Figure 6-16. Operation Dependencies TM402.0

Notes:

An example of the LTP operation-level dependencies dialog.

Instructor Notes:

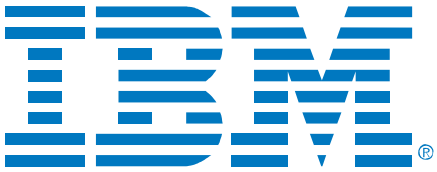
Purpose —Explain the operation-level dependencies LTP dialog.

Details —See the student notes.

Additional Information —

Transition Statement —

6.3 The Current Plan



Topic 6.3

The Current Plan

Tivoli software

Figure 6-17. The Current Plan TM402.0

Notes:

Instructor Notes:

Purpose —Clarify the CP and its relationship to the LTP.

Details —

Additional Information —

Transition Statement —Let's see the connection between the LTP and the CP.

Daily Planning: Requirements and Results

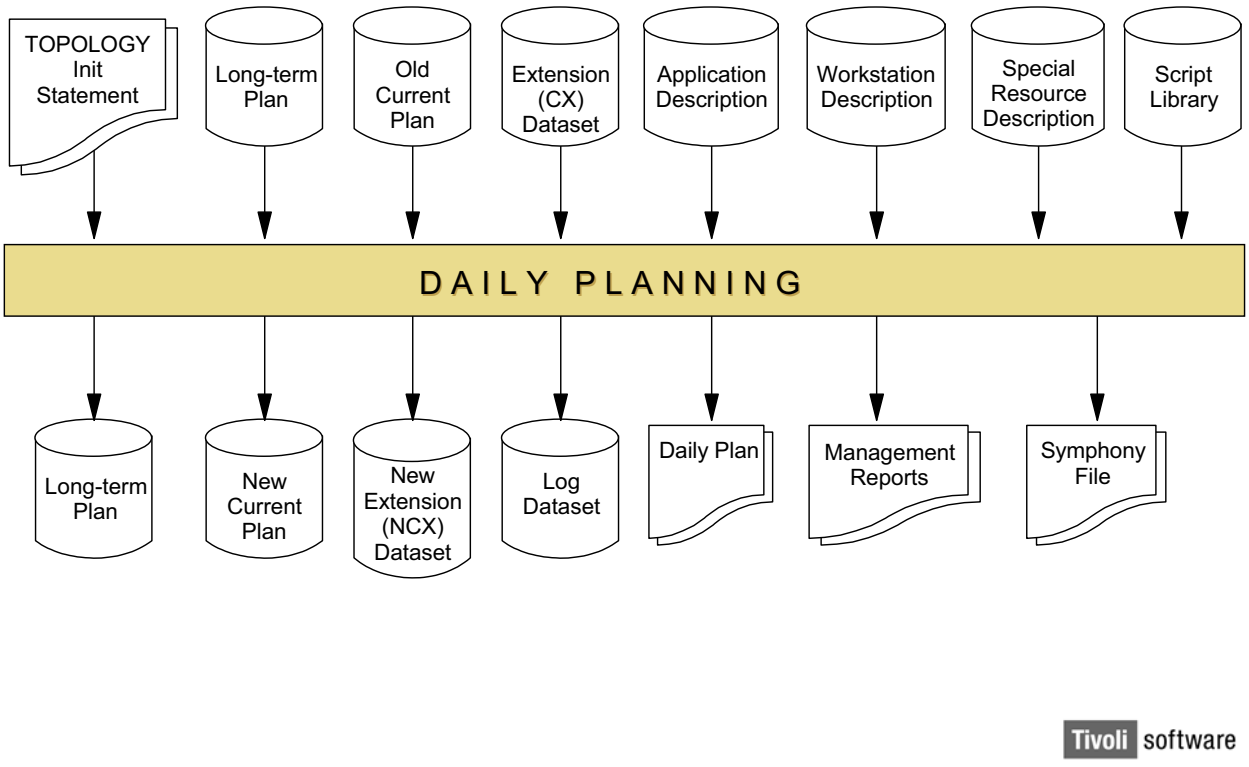


Figure 6-18. Daily Planning - Requirements and Results TM402.0

Notes:

- The process of producing the current plan is called *daily planning*.
- Daily planning requires input from:
  - The long-term plan which contains a list of application occurrences to run each day. The long-term plan details the input arrival and deadline times as well as external dependencies for every occurrence.
  - Old current plan if one exists. Uncompleted applications are carried forward into the new current plan, and completed occurrences are marked in the LTP.
  - Application, workstation, and special resource databases. The application description database, which contains the detail of applications at operation level. The workstation description database, which shows the open intervals, parallel servers, and fixed resources available at each workstation. The resource description database, which has details of special resources.
  - Tivoli Workload Scheduler (TWS) topology statements and the distributed environment script (job description) library, if the end-to-end feature is installed. The

TOPOLOGY initialization statement is used for the Symphony file. The script library is used in the creation of the Symphony file.

- Daily planning collects and processes data from the input data sets to update relevant data sets and produce current plan reports. The results are:
  - A new current plan which contains:
    - Uncompleted operations from the previous current plan
    - New occurrences
    - Potential predecessor records for each occurrence.
  - Updates to the long-term plan for occurrences that are marked complete or have been deleted in the previous current plan
  - Creation of the distributed environment Symphony file for the distributed agents, if the end-to-end feature is installed
  - Daily planning reports management reports.

Instructor Notes:

Purpose —Describe at a high-level, the process used to create the current plan.

Details —See the student notes.

Additional Information —

Transition Statement —

Daily Planning Batch Functions

```
-------------------------- PRODUCING OPC DAILY PLANS --------------------------
Option ===>

Select one of the following :

1 REPLAN           -  Replan current planning period
2 EXTEND           -  Extend the current planning period
3 TRIAL            -  Produce a trial plan
4 PRINT CURRENT    -  Print statistics for current planning period
5 SYMPHONY RENEW   -  Create Symphony file starting from Current Plan
```

Figure 6-19. Daily Planning Batch Functions TM402.0

Notes:

REPLAN -- This removes completed occurrences from the current plan, updates the long-term plan with the occurrence data, and recalculates the input arrival times of the occurrences remaining in the current plan. It does not change the end-time or add new occurrences from the long-term plan.

EXTEND -- Creates a new current plan if none exists, or extends the existing plan from its end time for a user-specified length of time. This function:

- Updates the LTP with completed occurrences
- Copies from the LTP occurrences with input arrivals that fall within the range of the new current plan
- Carries forward all uncompleted occurrences from the previous current plan.

TRIAL -- Create a trial current plan within the range of the long-term plan without affecting the existing in-production current plan.

PRINT CURRENT -- Print statistics for the current planning period.

SYMPHONY RENEW -- You can use this option to recover from error situations, when you add or change information in the current plan, or after you have initiated current plan recovery actions.

Instructor Notes:

Purpose —Identify the daily planning batch functions.

Details —See the student notes.

Additional Information —

Transition Statement —

The Daily Planning Process

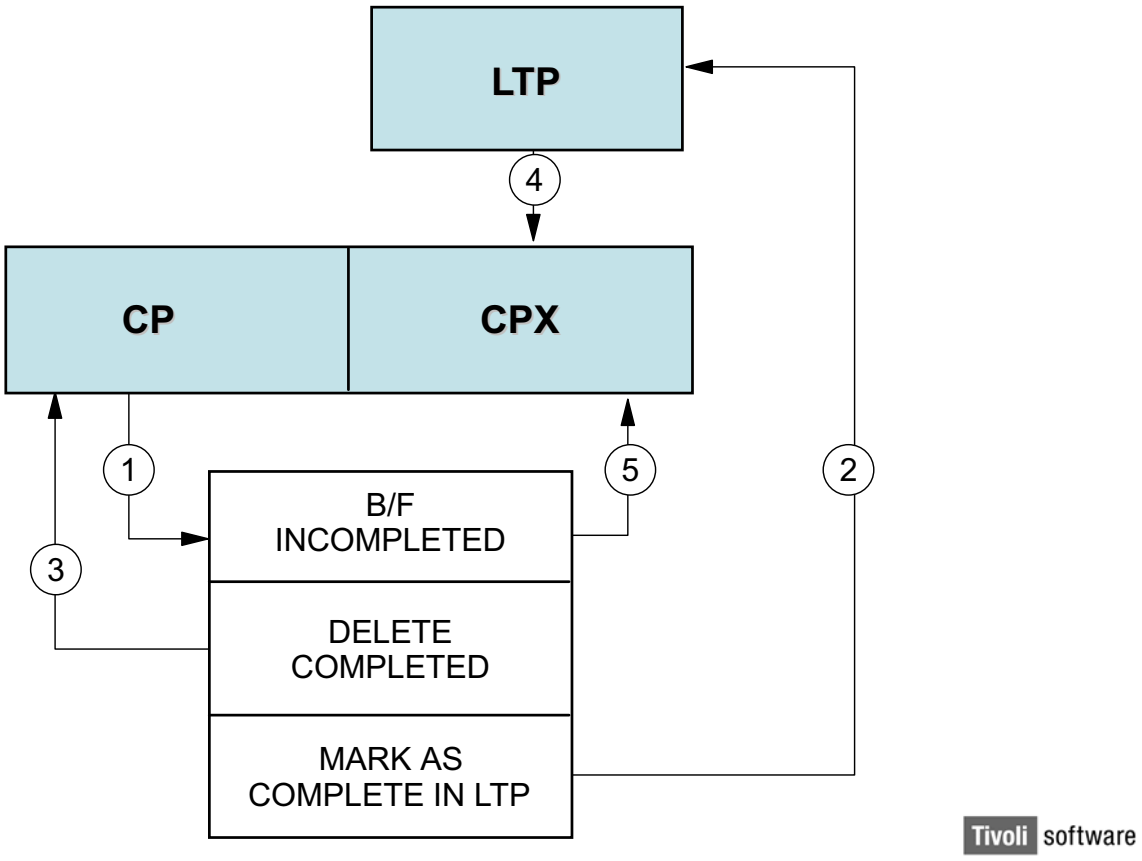


Figure 6-20. The Daily Planning Process TM402.0

Notes:

- When the Current Plan is extended the Daily Planning process brings forward the occurrences from the old current plan - still running, complete, in error, and so forth.
- Daily planning marks these brought forward occurrences locked in LTP until it has checked the old completed ones. They are deleted from the CP, and marked as complete in the LTP.
- Daily Planning examines the LTP for all occurrences scheduled to commence during the life of the CP, and brings them into the CP.
- The new Current Plan includes only new occurrences, and old uncompleted occurrences.
- If the end-to-end feature is active the daily planning process continues with these actions:
  - In the new current plan, the daily planning job flags the jobs and job streams that will be added to the Symphony file.
  - The job and job stream information are added to the new Symphony file.

- TWS for z/OS sends the Symphony file to the Tivoli Workload Scheduler (TWS) domain manager.
- TWS applies all outstanding events of the old Symphony file to the new Symphony file.
- In TWS, the new Symphony file is distributed to other fault-tolerant agents.

Instructor Notes:

Purpose —Describe the daily planning process used to create the CP.

Details —See the student notes.

Additional Information —

Transition Statement —If the end-to-end feature is installed, the daily planning process also creates the symphony file for the distributed environment.

The Symphony File

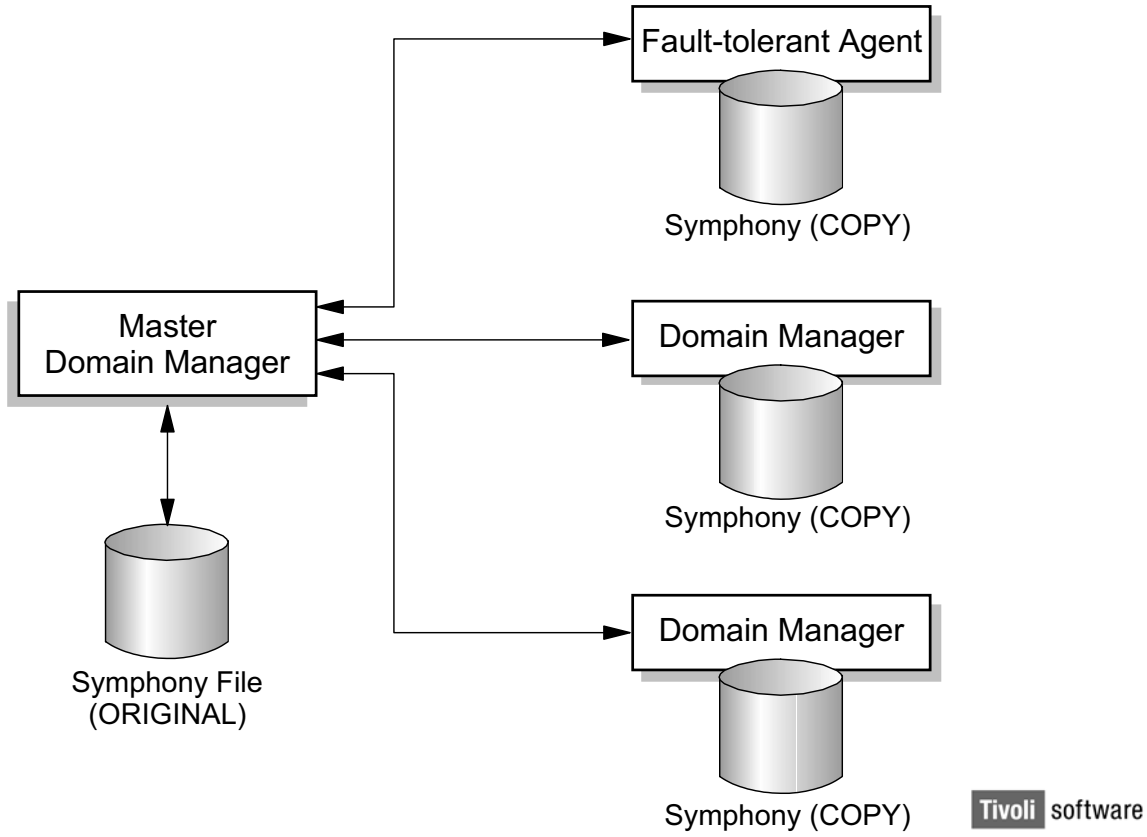


Figure 6-21. The Symphony File TM402.0

Notes:

- Before the start of each new day, the master domain manager creates a production control file, named ***Symphony***. The master domain manager sends a copy of the new Symphony file to each of its automatically linked agents and subordinate domain managers. The domain managers, in turn, send copies to their automatically linked agents and any subordinate domain managers.
- Scheduling messages like job starts and completions are passed from the agents to their domain managers, who in turn pass the information up the tree until it gets to the master domain manager. To keep the Symphony copies in sync, the master domain manager broadcasts the updating messages throughout the hierarchical tree.

Instructor Notes:

Purpose —Introduce the concepts used to propagate the work planned for the distributed environment.

Details —See the student notes.

Additional Information —

Transition Statement —TWS for z/OS is able to generate the symphony file because the topology of the distributed environments is identified in topology statements and parameters.

Distributed System Topology

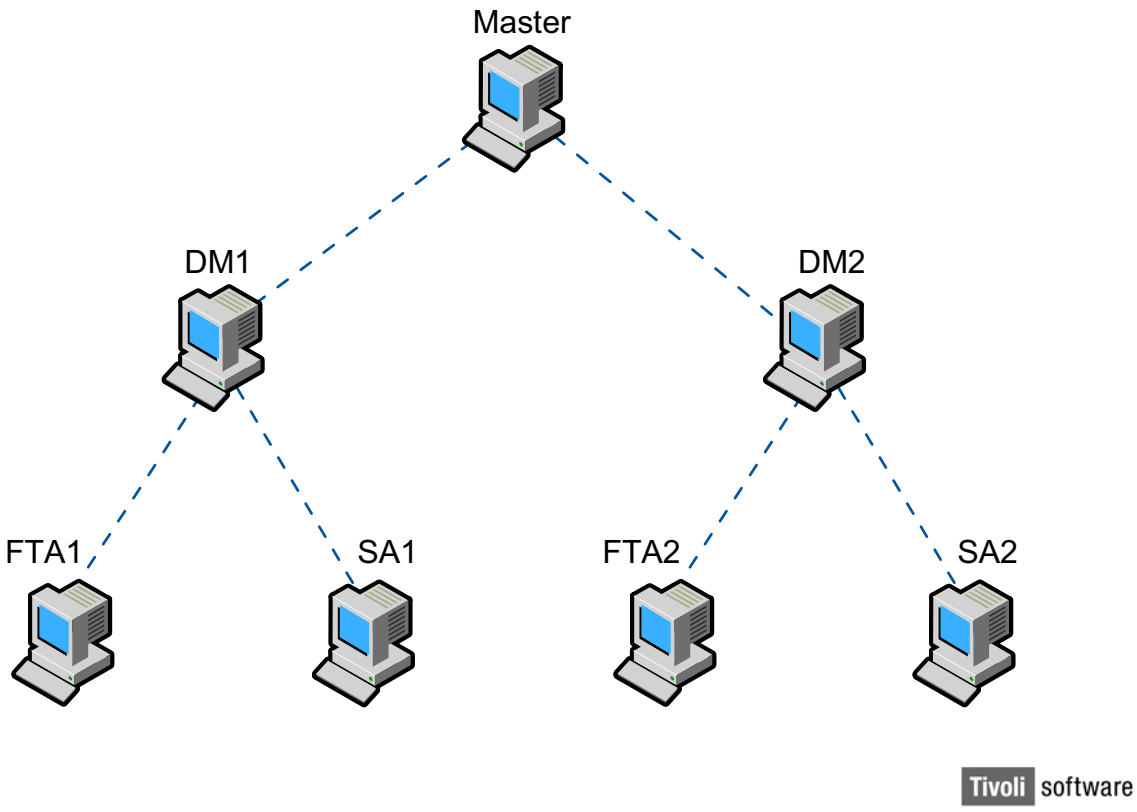


Figure 6-22. Distributed Systems Topology TM402.0

Notes:

- A TWS network contains of at least one domain, the ***master domain***, in which the master domain manager is the management hub. Additional domains can be used to divide a widely distributed network into smaller, locally managed groups.
- In a single domain configuration, the master domain manager maintains communications with all of the workstations in the TWS network.
- In a multidomain configuration, the master domain manager communicates with the workstations in its domain, and subordinate domain managers.
- The subordinate domain managers, in turn, communicate with the workstations in their domains and subordinate domain managers.
- Using multiple domains reduces the amount of network traffic by reducing the communications between the master domain manager and other computers. Multiple domains also provide fault-tolerance by limiting the problems caused by losing a domain manager to a single domain. To limit the effects further, backup domain managers can be designated to take over if their domain managers fail.

Instructor Notes:

Purpose —Describe a distributed environment domain.

Details —See the student notes.

Additional Information —

Transition Statement —

Topology Parameters

EQQPARM (TPLGINFO)

CPUREC

CPUNAME(cpu_name)
CPUOS(AIX | HPUX | POSIX | UNIX | WNT | OTHER)
CPUNODE(node_name)
CPUTCPIP(port_number | 31111)
CPUTYPE(SAGENT | XAGENT | FTA)

DOMREC

DOMAIN(domain)
DOMMNGR(domain_mamager_name)
DOMPARENT(parent_domain)

Tivoli software

Figure 6-23. Topology Parameters TM402.0

Notes:

- You define the topology of the TWS distributed environment using TWS for z/OS parameters. One of these parameters identifies a PDS member in which you use these statements:
  - **CPUREC** to specify the configuration options for fault-tolerant workstations
  - **DOMREC** to define domains for distributed agents networks
- CPUREC requires at least four parameters:
  - **CPUNAME** - a four character alphanumeric name assigned to the workstation
  - **CPUOS** - one of the accepted values representing the operating system of the workstation
  - **CPUNODE** - the node name or IP address of the workstation. A fully qualified domain name of up to 52 characters is acceptable
  - **CPUTYPE** - the type of distributed agent. The accepted values are:
    - **FTA** for fault-tolerant agents
    - **SAGENT** for standard agents
    - **XAGENT** for extended agents

- A domain definition must be created for each TWS domain. Although TWS for z/OS acts as the master domain manager for TWS domains, no DOMREC statement is required. In the DOMREC statement of the distributed agent that connects to TWS for z/OS, **MASTERDM** must be specified for the DOMPARENT parameter. This points to TWS for z/OS as the master domain manager. All other domains are now ineligible to use the name MASTERDM. The distributed agent to which TWS for z/OS connects must be a fault-tolerant agent.
- DOMREC requires all three of its parameters:
  - **DOMAIN** - a 16 character name of the domain
  - **DOMMNGR** - the TWS name of the domain manager which must be a FTA. The domain manager name for the TWS for z/OS host is OPCMASTER
  - **DOMPARENT** - the name of the parent domain. In the DOMREC of the FTA connected to TWS for z/OS, specify MASTERDM.
- See the *TWS for z/OS Customization and Tuning Guide* for additional information about the parameters.

Instructor Notes:

Purpose —Introduce TWS for z/OS parameters used to describe the TWS domain.

Details —See the student notes.

Additional Information —

Transition Statement —

Creating or Extending the Current Plan

```
------------------------ EXTENDING CURRENT PLAN PERIOD ------------------------
Command ===>

Enter/change data below and press ENTER

Current plan end date :

START DATE          ===> 020505__    YYMMDD  If no current plan exists
    TIME            ===> 11.14       HH.MM
END DATE            ===> _________   YYMMDD  Specific END date
    TIME            ===> _____       HH.MM     Specific END time
EXTENSION LENGTH    ===> 02400       HHHMM     Extend plan by
          TYPE      ===> W           A - includes all days
                                     W - includes only work days
Report selection :                   Y if report wanted, otherwise N
 WS SUMMARY         ===> Y           Summary for all work stations
 OPERATING PLAN     ===> Y           Daily operation plan
 WS PLANS           ===> Y           Plans for all work stations
 INPUT ARRIVAL      ===> Y           List of input arrival operations
 NON REPORTING      ===> Y           Plans for non reporting work station
 CURRENT PERIOD     ===> Y           Print current period results
 PLANNED RESOURCE   ===> Y           Planned resource utilization
 ACTUAL  RESOURCE   ===> Y           Actual  resource utilization
```

Figure 6-24. Creating or Extending the Current Plan TM402.0

Notes:

- The Current-plan end date field is blank when no current plan exists.
- If no current plan exists, TWS for z/OS by default uses the current date and time for the start of the current plan.
- When creating or extending the CP, you can specify either a specific end date and time or an extension length in hours and minutes.
- You can tell TWS for z/OS to extend the CP to end on work days only or on any day. **W** in the Type field specifies end on work days only. **A** in the Type field specifies all days are acceptable end days.

Instructor Notes:

Purpose —Describe the fields on the CP extend dialog panel.

Details —See the student notes.

Additional Information —

Transition Statement —

Daily Planning Reports

```
------------------------ EXTENDING CURRENT PLAN PERIOD ------------------------
Command ===>

Enter/change data below and press ENTER

Current plan end date :

START DATE          ===> 020505__    YYMMDD  If no current plan exists
      TIME          ===> 11.14       HH.MM
END DATE            ===> _________   YYMMDD  Specific END date
    TIME            ===> _____       HH.MM     Specific END time
EXTENSION LENGTH    ===> 02400       HHHMM     Extend plan by
          TYPE      ===> W           A - includes all days
                                     W - includes only work days
Report selection :                   Y if report wanted, otherwise N
 WS SUMMARY         ===> Y           Summary for all work stations
 OPERATING PLAN     ===> Y           Daily operation plan
 WS PLANS           ===> Y           Plans for all work stations
 INPUT ARRIVAL      ===> Y           List of input arrival operations
 NON REPORTING      ===> Y           Plans for non reporting work station
 CURRENT PERIOD     ===> Y           Print current period results
 PLANNED RESOURCE   ===> Y           Planned resource utilization
 ACTUAL  RESOURCE   ===> Y           Actual  resource utilization
```

Figure 6-25. Daily Planning Reports TM402.0

Notes:

- When creating, extending, or replanning the current plan, you can produce reports on the results.
- Daily planning provides two printed reports:
  - Plans
  - Management reports
- You can include all or some of these in a plan report:
  - Histograms that show the planned use of all workstations in the current plan
  - A list of all operations and occurrences to be processed
  - Planned utilization of special resources
  - Workstation plans which list all operations to be performed at each workstation, in order of planned start time
  - A list all operations to be performed at each workstation, in order of input arrival time.

- Management reports can be created for the current period and for the previous period. This is a list of the information that you can select to include in a management report:
  - Current period results which show this information about the old current plan:
    - Completed applications
    - Operations ended in error.
  - Previous period results which show this information about the 24-hour period preceding the old current plan:
    - Summary of completed applications
    - Completed applications
    - Operations ended in error
    - Special resource actual utilization
    - Error statistics on completed applications
    - Workstation histograms -- actual utilization
    - Missed feedback report

Instructor Notes:

Purpose —Identify and describe available daily planning reports.

Details —See the student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- Describe the functions of TWS for z/OS plans
- Describe the relationship between the databases and the plans
- Describe the relationship between the long-term plan and the current plan

Tivoli software

Figure 6-26. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 7. Setting Up and Using the Job Scheduling Console

What This Unit is About

This unit describes the Job Scheduling Console (JSC) installation requirements and tasks, and basic logon and use.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List the environmental requirements for installing the JSC
- Install the JSC on a workstation
- Log on and use the JSC.

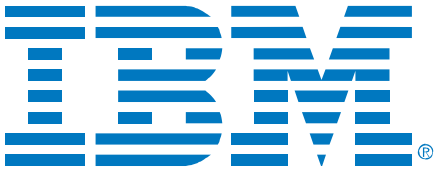
How You Will Check Your Progress

Accountability:

- Machine Exercises

References

SC32-1263	*IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*
SC32-1257	*IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide Version 1.3*



Unit 7: Setting Up and Using the Job Scheduling Console

Tivoli software

Figure 7-1. Unit 9: Setting Up and Using the Job Scheduling Console TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- List the components required for implementing the Job Scheduling Console (JSC)
- Install the Job Scheduling Console
- Use the Job Scheduling Console

Tivoli software

Figure 7-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

Purpose —

Details — The objectives of this unit are to highlight the tasks and requirements for the successful installation of the Tivoli Job Scheduling Console.

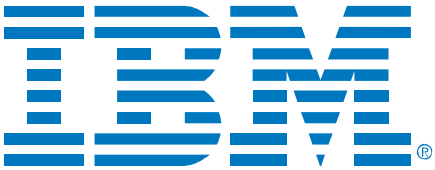
Additional Information —

Transition Statement —

7.1 Introduction and Overview

Instructor Topic Introduction

You may need Tivoli Management Framework and some workstation skills in the areas of Microsoft Windows NT and/or UNIX to implement the Job Scheduling Console and its supporting components.



Topic 7.1

Introduction and Overview

Tivoli software

Figure 7-3. Introduction and Overview TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

What Is the Job Scheduling Console?

- A Graphic User Interface to:
  - Tivoli Workload Scheduler for Distributed environments
  - TWS for z/OS Scheduling functions
- It has two main functions
  - Scheduling work in both environments
  - Monitoring and controlling scheduled work

Tivoli software

Figure 7-4. What Is the Job Scheduling Console TM402.0

Notes:

The Tivoli Job Scheduling Console (JSC) is a Java-based interactive graphical user interface for creating, modifying, and deleting objects in the TWS and TWS for z/OS databases.

The JSC allows users to monitor and control objects scheduled in the TWS for z/OS current plan, and work with TWS for z/OS and TWS simultaneously from the same graphical console.

Instructor Notes:

Purpose —Introduce the JSC.

Details —The Tivoli Job Scheduling Console (JSC) is an interactive graphical user interface for creating, modifying, and deleting objects in the TWS and TWS for z/OS databases. The JSC also allows you to monitor and control objects scheduled in the TWS for z/OS current plan, and work with TWS for z/OS and TWS simultaneously from the same graphical console.

Additional Information —

Transition Statement —

Supported Platforms for the JSC

- Windows
- IBM AIX
- HP-UX
- Linux
- Sun Solaris

Figure 7-5. Supported Platforms for the JSC TM402.0

Notes:

Currently, the only operating platforms on which the JSC can be installed are:

- Microsoft Windows:
  - NT 4.0 - service pack 6A (SP6A)
  - Professional
  - Server, and Advanced Server 2000 - service pack 2 (SP2)
  - Datacenter
- IBM AIX - Versions 5.1 and 5.2
- HP-UX - Versions 11.0, 11i, and 11.22
- Sun Solaris - Versions 7, 8, and 9
- Linux
  - Red Hat 7.2
  - SueSe Enterprise Server 8

Instructor Notes:

Purpose —Outline the software requirements for installation of the JSC on a workstation.

Details —Keep this simple because the audience may have a little difficulty with the various platforms or may not be familiar with Tivoli Framework.

Currently the only operating platforms on which you can install the JSC and its related components are:

- Microsoft Windows:
  - NT, Professional, Server, and Advanced Server 2000
- IBM AIX
- HP-UX
- Sun Solaris
- Linux

The JSC must be installed in a Tivoli Managed Environment (TME). The TME is built on a foundation known as Tivoli Management Framework (TMF).

To operate on the TWS for z/OS databases and plans through the JSC, the JSC Connector must be installed in the Tivoli Framework environment. The JSC Connector requires the Job Scheduling Services (JSS). The JSS must also be installed in the Tivoli Framework environment. All these pieces can be installed on the same workstation if necessary.

The workstation on which the JSC is to be installed must have active TCP/IP communications support and the Java runtime environment installed.

Additional Information — Two CDs are supplied with the JSC software. One has the JSC, the Connector, and the JSS code; and the other has the Java runtime environment.

Transition Statement —Refer to the *JSC User's Guide* for more information about the currently supported versions of the operating platforms, and the required levels of the component software.

JSC Operating Environment Requirements

- Tivoli Management Framework
- Tivoli Job Scheduling Services
- IBM Tivoli Workload Scheduler Connector
- Java Run-time Environment
- TCP/IP Network Communication
- Supported Operating Platform

Tivoli software

Figure 7-6. JSC Operating Environment Requirements TM402.0

Notes:

- There are a number of prerequisite components that must be installed and operational before you can install and use the JSC. Primarily the environment in which components must reside is called a Tivoli Managed Environment (TME). The TME is built on a software foundation known as Tivoli Management Framework (TMF).
- The JSC uses the functions and facilities of two other components which must be installed in a TME. Those two components are:
  - The IBM Workload Scheduler Connector
  - The Job Scheduling Services (JSS).
- Typically Framework, the JSS and the connector are installed on a server; but if necessary they can be installed on the workstation on which you are installing the JSC.
- TCP/IP is required for communication between the JSC and the connector.
- The Java run-time environment is required because the JSC is a Java-based graphical interface

Note: Two CDs are supplied with the JSC installation package, one holds the JSC and Connector software; the other has the Java run-time libraries.

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

z/OS System Software Requirements

- TWS for z/OS 8.1 or later

OR

- OPC Version 2.1 or later

- TCP/IP installed and operational

Tivoli software

Figure 7-7. z/OS System Software Requirements TM402.0

Notes:

On the z/OS host, TWS for z/OS 8.1 or later must be installed and TCP/IP must be installed and active.

Instructor Notes:

Purpose —Describe the software requirements on the z/OS system.

Details —By this time you should have satisfied the z/OS system software requirements for installing and using the JSC. TWS for z/OS or OPC version 2.1 and TCP/IP must be installed.

Additional Information —

Transition Statement —

An Overview of the Components

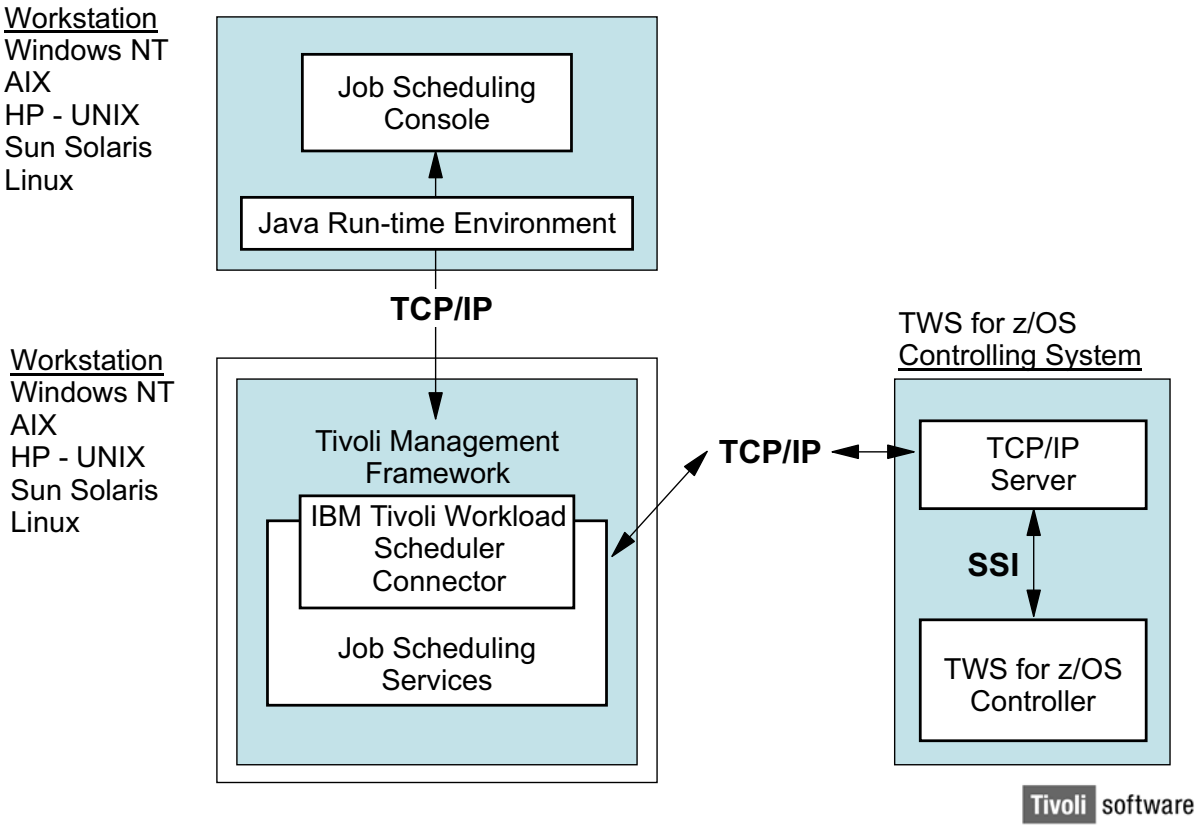


Figure 7-8. An Overview of the Components TM402.0

Notes:

The Job Scheduling Console is a Java-based graphical user interface therefore the Java Run-time Environment must be installed on the JSC workstation. Install the JSC on any workstation that has a TCP/IP connection to the IBM Tivoli Workload Scheduler Connector.

The Connector manages the traffic between the Job Scheduling Console and the TCP/IP server on the TWS for z/OS controlling system. The Connector translates instructions entered via the console into TWS for z/OS commands or queries. The Connector must be installed on either a Tivoli Managed Region (TMR) server or managed node.

On the Connector system Tivoli Management Framework and the Job Scheduling Services software must be installed before the JSC Connector can be installed. The Job Scheduling Services component provides a common Framework interface to job-scheduling applications.

The TCP/IP Server routes requests from the JSC user to the TWS for z/OS controller via the z/OS Subsystem Interface (SSI).

TCP/IP is the protocol used to connect the JSC to the JSC Connector which in turn connects to the TCP/IP server.

Instructor Notes:

Purpose —Describe the components that support the JSC.

Details —This visual shows a simple representation of the Job Scheduling Console (JSC) and its supporting components.

The **Job Scheduling Console** is Java-based so the Java Run time Environment must be installed on the workstation on which the JSC is installed. You can install the JSC on any workstation that has a TCP/IP connection to the Connector.

The JSC **Connector** manages the traffic between the Job Scheduling Console and the TCP/IP server on the TWS for z/OS controlling system. The Connector acts like a protocol converter that translates the instructions entered through the console into TWS for z/OS commands. The Connector must be installed on either a Tivoli Managed Region (TMR) server or managed node.

On the Connector system Tivoli Management Framework and the **Job Scheduling Services** software must be installed before the JSC Connector can be installed. The Job Scheduling Services component provides a common Framework interface to job scheduling applications.

The **TCP/IP** Server routes requests from the JSC user to the TWS for z/OS controller via the z/OS Subsystem Interface (**SSI**).

TCP/IP is the protocol used to connect the JSC to the JSC Connector which in turn connects to the TCP/IP server.

Additional Information —

Transition Statement —

Implementation Tasks

- On the TWS for z/OS System:
  - Implement the TCP/IP Server
- On a TMR Server:
  - Install Tivoli Management Framework
    - For Windows, AIX, HP-UX, and Sun Solaris - Version 3.7.1 or Version 4.1
    - For Linux - Version 3.7B or Version 4.1
  - Install the Tivoli Management Environment Desktop
  - Install Tivoli Job Scheduling Services
  - Install IBM Tivoli Workload Scheduler for z/OS Connector
- On a Workstation:
  - Install the Tivoli Job Scheduling Console

Tivoli software

Figure 7-9. Implementation Tasks TM402.0

Notes:

On the TWS for z/OS controlling system, implement the TCP/IP server. This task is covered in detail in the TWS for z/OS Implementation course (TM41).

On either a TMR server or a Tivoli Managed node, ensure that the correct level of TMF is installed. For Microsoft Windows, AIX, HP-UX and Sun Solaris platforms, you require Tivoli Managed Framework version 3.7.1, or version 4.1. For Linux platforms you require TMF version 3.7B, or version 4.1.

This machine will be used as your connector machine to the TWS for z/OS controlling system.

Installation of the Tivoli Desktop is not required but it makes installation of the other components easier. The Tivoli Desktop simplifies installation of the JSS component and the Connector component because it makes the process menu-driven.

Job Scheduling Services version 1.2 is a perquisite for installing version 1.3 of the TWS for z/OS Connector, and version 1.3 of the Job Scheduling Console.

On the JSC workstation, install the JSC.

Instructor Notes:

Purpose —Itemize and discuss the tasks that must be performed to install the JSC.

Details —On the Connector workstation which must be either a TMR server or a Tivoli Manage node, ensure that the correct level of TMF is installed.

Windows, AIX, HP-UX and Sun Solaris platforms require TMF version 3.7.1. Linux requires TMF version 3.7B.

Installation of the Tivoli Desktop is not required but it makes installation of the other components easier. I recommend using the Tivoli Desktop to install the JSS then the Connector because it makes the installation process menu-driven.

On the JSC workstation, install the JSC. On both the Connector and JSC workstations, you must have TCP/IP network communications active. If necessary, all the workstation software can be installed on the same workstation.

On the TWS for z/OS controlling system, implement the TCP/IP server.

Additional Information —

Transition Statement —

Components Installation Matrix

Currently installed in the TMR	Installation actions
– No Tivoli Management Framework No Job – Scheduling Services No IBM Tivoli – Workload Scheduler Connector	Run the installation program to install ✓ Tivoli Management Framework version 4.1 ✓ Job Scheduling Services version 1.2 ✓ IBM Tivoli Workload Scheduler for z/OS Connector version 1.3.
– Tivoli Management Framework version 3.7.1 or 4.1 – No Job Scheduling Services – No IBM Tivoli Workload Scheduler Connector	Run the installation program to install ✓ Job Scheduling Services version 1.2 ✓ IBM Tivoli Workload Scheduler for z/OS Connector version 1.3.
– Tivoli Management Framework version 4.1 – Job Scheduling Services version 1.2 – IBMTivoli Workload Scheduler Connector version 1.2	Run the installation program to: ✓ Upgrade IBM Tivoli Workload Scheduler for z/OS Connector version 1.2 to version 1.3
– Tivoli Management Framework version 3.7.1 – Job Scheduling Services version 1.1 – IBM Tivoli Workload Scheduler Connector version 1.2	Run the installation program to: ✓ Upgrade Job Scheduling Services version 1.1 to version 1.2 ✓ Upgrade to IBM Tivoli Workload Scheduler for z/OS Connector version 1.2 to version 1.3

Tivoli software

Figure 7-10. Component Installation Matrix TM402.0

Notes:

This table summarizes the sequence of actions that you must take when installing the components required for installing the JSC for the TWS for z/OS 8.2 End-to-end environment.

Instructor Notes:

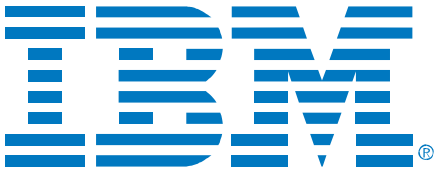
Purpose —

Details —

Additional Information —

Transition Statement —

7.2 The Connector



Topic 7.2

The Connector

Tivoli software

Figure 7-11. The Connector TM402.0

Notes:

Instructor Notes:

Purpose —Describe the connector, its function, and its requirements.

Details —Connectors are installed on a Tivoli Management Server and on managed nodes that have access to the TWS for z/OS controlling system. You need a Connector instance for each Controller. You can run multiple JSC connector instances on the same TMR server or Tivoli managed node. Each JSC connector connects to a single TWS for z/OS TCP/IP server.

Additional Information —

Transition Statement —

IBM Tivoli Workload Scheduler Connector Instances

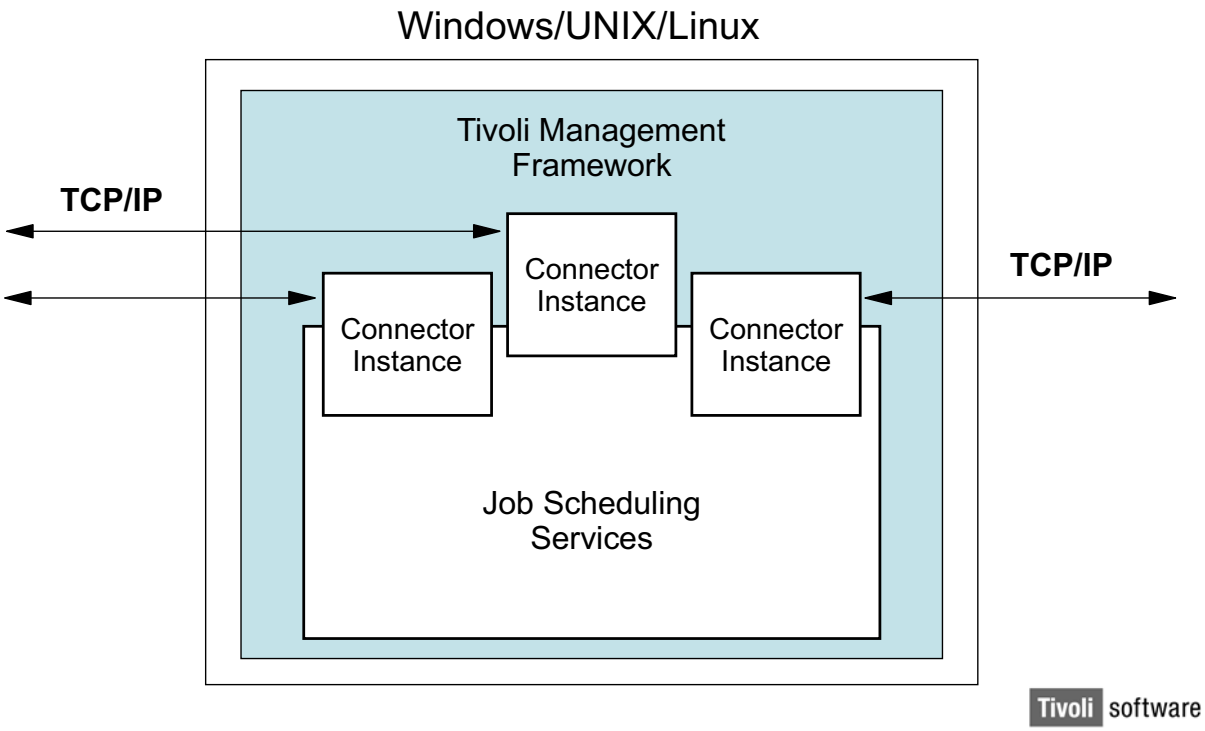


Figure 7-12. The IBM Tivoli Workload Scheduler Connector TM402.0

Notes:

- Before the you can use the Job Scheduling Console (JSC) to manage TWS for z/OS resources, and plans, you must create and activate a Connector instance for the TWS for z/OS controller.
- The Connector performs the function of a translator for requests and responses between the JSC and the TWS for z/OS controller.
- You must create a connector instance for each TWS for z/OS controller.
- A connector instance communicates with a single TCP/IP server on the controlling TWS for z/OS system. The TCP/IP server forwards requests and responses to and from the TWS for z/OS controller on behalf of the connector.
- Each connector instance can communicate with only one TCP/IP server; but you can create multiple connector instances for communication with a single TCP/IP server.
- Connectors are installed on Tivoli Managed Region (TMR) Servers and on Tivoli managed nodes that have access to the TWS for z/OS controlling system.
- You can start multiple connector instances on the same TMR server or Tivoli managed node.

Instructor Notes:

Purpose —Describe the connector, its function, and its requirements.

Details —Connectors are installed on a Tivoli Management Server and on managed nodes that have access to the TWS for z/OS controlling system. You need a Connector instance for each Controller. You can run multiple JSC connector instances on the same TMR server or Tivoli managed node. Each JSC connector connects to a single TWS for z/OS TCP/IP server.

Additional Information —Before the you can use the Job Scheduling Console (JSC) to operate on TWS for z/OS resources, a Connector instance must be created and started.

The Connector performs the function of a translator for requests and responses between the JSC and the TWS for z/OS controller.

You must create a connector instance for each TWS for z/OS controller.

A connector instance communicates with a single TCP/IP server on the controlling TWS for z/OS system. The TCP/IP server forwards requests and responses to and from the TWS for z/OS controller on behalf of the connector.

Each connector instance can communicate with only one TCP/IP server; but you can create multiple connector instances for communication with a single TCP/IP server.

Connectors are installed on Tivoli Managed Region (TMR) Servers and on Tivoli managed nodes that have access to the TWS for z/OS controlling system.

You can start multiple connector instances on the same TMR server or Tivoli managed node.

Transition Statement —

Customizing the Connector

- Create Connector Instances
  - One for each TWS for z/OS controller
- Set Authorized Roles
  - User authorization to manage instances

Figure 7-13. Customizing the Connector TM402.0

Notes:

Once installed the IBM Tivoli Workload Scheduler Connector must be customized before it can be used. As an administrator, you must:

- Create Connector Instances for the TWS for z/OS controllers
- Determine who can manage the instances

  The levels of user authorization is set in the Tivoli Framework environment.

Instructor Notes:

Purpose —Explain how the connector supports the JSC.

Details —Once installed the JSC Connector must be customized before it can be used.

Listed in this visual are the areas to be considered:

- Creating Connector Instances for the TWS for z/OS controllers
- Determine who can manage the instances - the levels of user authorization.

Additional Information —

Transition Statement —

Managing Connector Instances

```
wopcconn
   -create -h node -e engine_name  -a address -p port
   -start -e engine_name | -o object_id
   -stop -e engine_name  | -o object_id
   -restart -e engine_name  | -o object_id
   -remove -e engine_name  | -o object_id
   -view -e engine_name  | -o object_id
```

Tivoli software

Figure 7-14. Managing Connector Instances TM402.0

Notes:

To create, remove, and manage IBM Workload Scheduler Connector instances, use the utility wopcconn. This program is downloaded when the Connector is installed.

Use the wopcconn utility to create, start, stop, restart, remove, view, and change the settings of instances.

The bold switches (-a for example) are used to pass values to the utility.

The italicized names after the switches represent what values are expected:

- ***node*** is the name or the object ID (OID) of the managed node on which you are creating the instance. The name of the Tivoli Managed Region Server is the default.
- ***engine_name*** is the name of the new or existing instance.
- ***object_id*** is the object ID of the instance.
- ***address*** is the IP address of the controlling TWS for z/OS system to which you want to connect.
- ***port*** is the port number of the Workload Scheduler for z/OS TCP/IP server to which the Connector must communicate.

Instructor Notes:

Purpose —Explain the wopcconn command and its switches.

Details —To create, remove, and manage JSC Connector instances, you use the Connector Instance utility ***wopcconn***. This program is downloaded when you install the Connector.

Before the JSC can be used to operate on TWS for z/OS resources, a Connector instance must be created and started. Use the wopcconn utility to create, start, stop, restart, remove, view, and change the settings of instances.

The **bold** switches (**-a** for example) are used to pass values to the utility.

The ***italicized*** names after the switches represent what values are expected:

- ***node*** is the name or the object ID (OID) of the managed node on which you are creating the instance. The name of the **Tivoli server** is the default.
- ***engine_name*** is the name of the new or existing instance.
- ***object_id*** is the object ID of the instance.
- ***address*** is the IP address of the controlling TWS for z/OS system to which you want to connect.
- ***port*** is the port number of the Workload Scheduler for z/OS TCP/IP server to which the connector must connect.

Additional Information —

Transition Statement —

Creating a Connector Instance

wopcconn

-create
-e opsaopcc
-a 10.31.227.89
-p 425

Tivoli software

Figure 7-15. Creating a Connector Instance TM402.0

Notes:

In this example:

The connector instance utility wopcconn is used in create mode.

- **opsaopcc** is the engine_name - the name of the connector instance
- **10.31.227.89** is the address - the IP address of the z/OS system on which the TWS for z/OS controller and TCP/IP server are installed
- **425** is the port - the port number of the TCP/IP server with which the IBM Workload Scheduler connector will communicate.

The command is issued from the command line prompt on the workstation.

You can also use **wopcconn** interactively by entering the command without any arguments. In the interactive mode you are prompted for the required values.

Instructor Notes:

Purpose —Demonstrate the wopcconn command

Details —In this example:

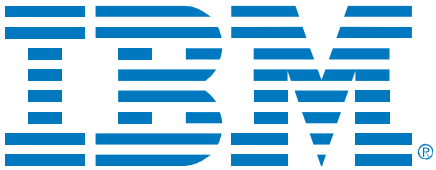
- The connector instance utility wopcconn is used in create mode
- opsaopcc is the *engine_name*, the name of the instance
- 10.31.227.89 is the *address*, the IP address of the z/OS system on which the TWS for z/OS controller is installed
- 425 is the port, the port number of the TCP/IP server to which JSC connector will connect

The command is issued from a command line. You can invoke wopcconn interactively also by entering the command without any arguments.

Additional Information —

Transition Statement —

7.3 The Job Scheduling Console



Topic 7.3

The Job Scheduling Console

Tivoli software

Figure 7-16. The Job Scheduling Console TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Installing the Job Scheduling Console

- On Windows :
  - Insert the JSC CD-ROM into the CD-ROM drive
  - Click Start and select Run
  - In the Open field, enter:drv:\Install, where: drv: is the name of the CD-ROM drive.
- On AIX and HP-UX :
  - Ensure that the Java Runtime Environment is installed
  - Add to your system path the ../jre/bin subdirectory of the directory where you install the JSC
  - Mount the CD-ROM drive
  - Enter: sh install.bin
- On Solaris and Linux :
  - The install file is the directory into which you placed the downloaded zip file - sh install.bin

Tivoli software

Figure 7-17. Installing the JSC TM402.0

Notes:

Installation of the Job Scheduling Console is menu-driven. On Windows NT, double-click the executable file install.exe which is on the Tivoli JSC CD-ROM. You are guided through the installation process. Be aware that when selecting languages to install, your selected language will be used in the JSC dialog only if it matches the regional settings of your workstation.

Instructor Notes:

Purpose —List the installation procedure by operating platform.

Details —Installation of the Job Scheduling Console is menu-driven. For example on Windows NT just double-click the executable file install.exe which is a deliverable on the Tivoli JSC CD-ROM. You are guided through the installation process. Be aware that when selecting languages to install, your selected language will be used in the JSC dialog only if it matches the regional settings of your workstation.

Additional Information —

Transition Statement —

Logging On to the JSC

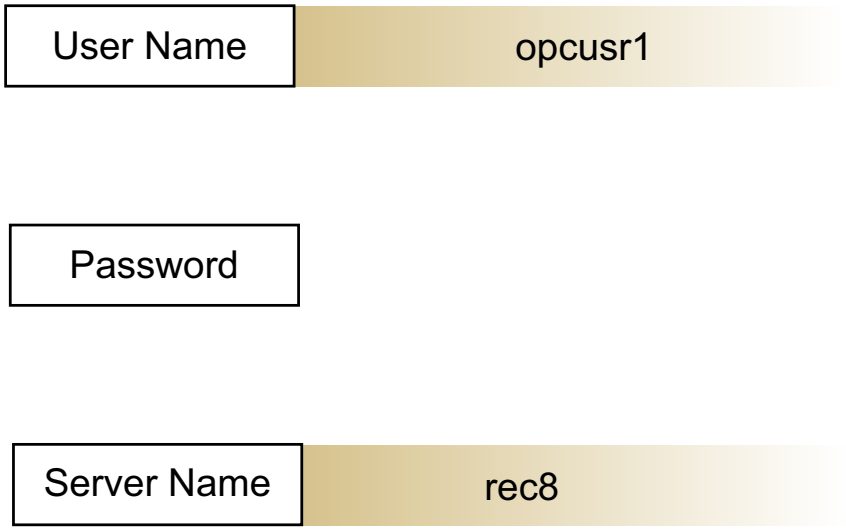


Figure 7-18. Logging on to the JSC TM402.0

Notes:

- From the desktop on the workstation, double-click the JS Console icon to start the Job Scheduling Console, or select JSC from the JSC folder.
- When prompted for the Host name, enter the name of the Tivoli managed node on which the IBM Tivoli Workload Scheduler Connector is installed.
- You then log in with a Tivoli Administrator ID and password. The ID and password must have been previously defined on the host (TMR server), and the ID should also be mapped to a RACF user ID on the controlling TWS for z/OS system.

Instructor Notes:

Purpose —Describe how to logon to the JSC.

Details —See the student notes.

Additional Information —

Transition Statement —

Job Scheduling Console Security

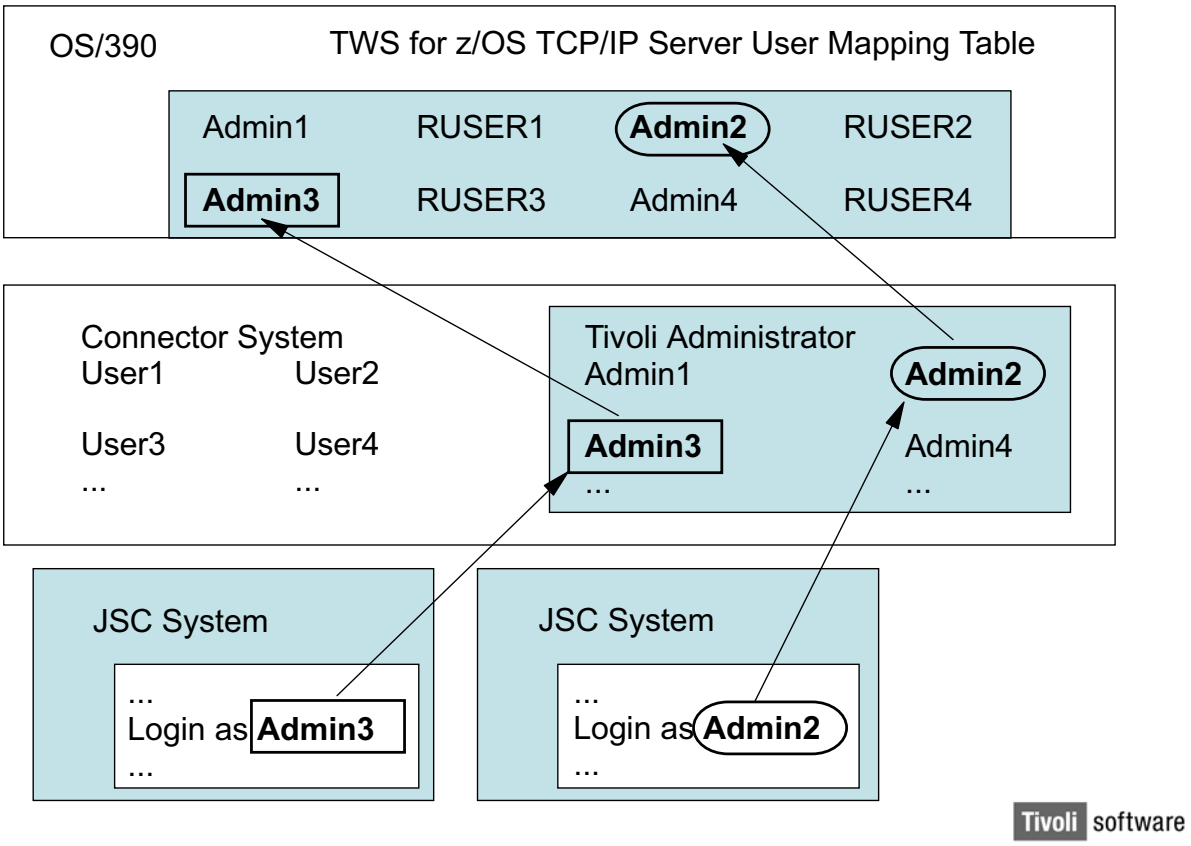


Figure 7-19. Job Scheduling Console Security TM402.0

Notes:

To log on to the Job Scheduling Console the user must be an authorized Tivoli administrator; that is, someone who has an ID which has been defined as a Tivoli administrator. At a minimum, the ID must have a TMR role user.

You must define the JSC users' IDs on the Connector workstation.

For users who use the connector instance only to communicate with TWS for z/OS, set their TMR role to user.

For users who create, start, stop, remove, and change connector instances must have the TMR roles of administrator, senior, or super.

JSC OS

When requests are forwarded through the connector instance to the TCP/IP server on the TWS for z/OS host, the TCP/IP server maps the Tivoli administrator user Id to a RACF user ID and forwards the request to the TWS for z/OS controller. TWS for z/OS then checks to see if the RACF user is authorized for the request.

Instructor Notes:

Purpose —Discuss JSC security at a high level.

Details —Typically security is handled by security administrators. Use this visual simply to give students a high-level view of how a user gets access to TWS for z/OS resources from the JSC.

To log on to the Job Scheduling Console the user needs to provide the user ID of a Tivoli administrator and the corresponding password.

On the system with the JSC connector running, the Tivoli administrator must be defined.

For a user who just wants to use the connector instance to communicate with TWS for z/OS, the Tivoli administrator needs to have the TMR role of user. A Tivoli administrator who needs to create, start, stop, remove, and change JSC OS connector instances must have the TMR roles of administrator, senior, or super.

TWS for z/OS dialog requests are forwarded through the JSC connector to the TCP/IP server on the TWS for z/OS host. The server maps the Tivoli administrator user ID to a RACF user ID.

Security checking for TWS for z/OS is done with the RACF user ID.

Additional Information —

Transition Statement —

Job Scheduling Console Terminology (1 of 2)

TWS for z/OS	TWS
Application description	Job stream
Application group	Job stream template
Current plan	Plan
External dependency	External job
In-effect date for run cycles	Valid from
Input arrival time	(Earliest) start time
Negative run cycle	Exclusionary run cycle
Occurrence	Job stream instance
Controller	Engine
Operation	Job
Operation number	Job identifier

Figure 7-20. Job Scheduling Terminology (1 of 2) TM402.0

Notes:

Job stream - A sequence of jobs, including the resources and workstations that support them, and scheduling information. (Application/Job Description)

Job stream template - A grouping of job streams that provides scheduling information, such as a calendar, free-day rules, and run cycles that are inherited by all the job streams that have been created using the template. (Group Definition)

Plan - A detailed plan of batch-run activity that covers a period of at least one minute and not more than 21 days. The plan encompasses all job and job stream instances and the resources and workstations on which the jobs will run. (Current Plan)

External job - A job from one job stream that is a predecessor for a job in another job stream. (External dependency)

Valid from - The first date that a run cycle is valid. (In Effect)

(Earliest) start time - The planned start time for a job or job stream. (Input arrival time)

Exclusionary run cycle - Specifies when a job stream must not run. (Negative run cycle)

Job stream instance - A job stream that is scheduled for a specific run date in the plan. (Occurrence)

Engine - The component that runs on the controlling system, and contains the tasks that manage the plans and databases. (Controller)

Job - A unit of work that is part of a job stream and is processed at a workstation. (Operation)

Job identifier - The number that identifies a job (Operation number).

Instructor Notes:

Purpose —Introduce JSC terminology.

Details —See the student notes.

Additional Information —

Transition Statement —

Job Scheduling Console Terminology (2 of 2)

TWS for z/OS	TWS
Operations in the current plan	Job instances
Out-of-effect date for run cycles	Valid to
Run cycle with offsets	Offset-based run cycle
Run cycle with rules	Rule-based run cycle
Special resources	Logical resources
Status: Complete	Successful
Status: Delete	Canceled
Status: Started	Running
Task	Job

Tivoli software

Figure 7-21. Job Scheduling Terminology (2 of 2) TM402.0

Notes:

Job instances - A job scheduled for a specific run date in the plan. (Operations in the Current Plan)

Valid to -The last date that a run cycle is valid. (Out of Effect)

Offset-based run cycle - Includes a user-defined period and an offset, such as the third day in a 90-day period.

Rule-based run cycle - Includes a rule, such as the first Friday of March or the second workday of the week.

Logical resources - Any type of limited resource that is needed to run a job. (Special resources)

Successful - The job or job stream has been completed. (Completed)

Canceled - The job or job stream has been deleted from the plan. (Deleted)

Running - The job has started (jobs only). (Started)

Job - A job performed at a computer workstation

Instructor Notes:

Purpose —Introduce JSC terminology.

Details —See the student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- List the components required for implementing the Job Scheduling Console (JSC)
- Implement the Job Scheduling Console
- Use the Job Scheduling Console

Tivoli software

Figure 7-22. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 8. Using the Restart and Cleanup Function

What This Unit is About

This unit introduces the IBM TWS for z/OS restart and cleanup function, and describes how to use the function.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List the tasks for which you can use the restart and cleanup function
- Use the dialogs to specify restart and cleanup options for selected operations
- Use the restart and cleanup dialog to:
  - Perform job and step-level restarts
  - Override data set cleanup actions

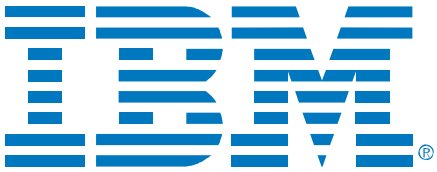
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 8: Restart and Cleanup

Tivoli software

Figure 8-1. Restart and Cleanup TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- List the tasks that can be performed by the Restart and Cleanup function
- Specify appropriate restart and cleanup options for selected operations
- Use the restart and cleanup dialog to:
  - Perform job or step-level operation restarts
  - Override data set cleanup actions

Tivoli software

Figure 8-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

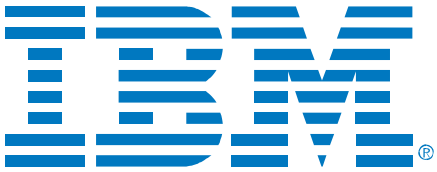
Purpose —

Details —

Additional Information —

Transition Statement —

8.1 Introducing the Restart and Cleanup Function



Topic 8.1

Introducing the Restart and Cleanup Function

Tivoli software

Figure 8-3. Introducing the Restart and Cleanup Function TM402.0

Notes:

Instructor Notes:

Purpose —Introduce the restart and cleanup function.

Details —

Additional Information —

Transition Statement —

What is Restart and Cleanup?

- A TWS for z/OS function which supports:
  - Restarting operations at job-level or step-level
  - Data set cleanup for restarted operations
  - For OS/390 and z/OS jobs only

Tivoli software

Figure 8-4. What is Restart and Cleanup? TM402.0

Notes:

- Restart and cleanup is a TWS for z/OS function that performs basically two tasks:
  - Restarting an operation at the job-level or the step-level
  - Cleaning up the associated data sets
- Restart and cleanup can be invoked for jobs in the z/OS environment only.

Instructor Notes:

Purpose —Define the restart and cleanup function.

Details —See the student notes.

Additional Information —

Transition Statement —

Restart and Cleanup: An Overview

- Uses Data Store
- Browsing the job log
  - Enterprise-wide
- Data set clean up
  - Catalogs data sets
  - Uncatalog data sets
  - Delete data sets
  - Generation Data Group (GDG) name resolution
- Interface with Removable Media Manager (RMM)
- Restart at both, the step and job-level
  - Automatic identification of the best restart step
- JCL modifications for restarts
  - In stream JCL
  - Procedures

Tivoli software

Figure 8-5. Restart and Cleanup: An Overview TM402.0

Notes:

The restart and cleanup function is used for step-level restarts of z/OS jobs or started tasks even when there are no catalog modifications. Restart and cleanup is also used to catalog, uncatalog, or delete data sets for job restarts; and allows users to browse JES job logs, and job logs from other operating platforms that do not have an SDSF-like facility. , and to request .

The data store facility which collects structured (steps and data sets) and, optionally, unstructured (SYSOUT) information for all submitted jobs, is an integral part of restart and cleanup.

Restart and cleanup actions can be initiated automatically or manually by a user through the TWS for z/OS dialog. If catalog modifications are required, TWS for z/OS will reset the catalog to its pre job-run state. This function is applicable to both generation data group (GDG) data sets and data sets allocated within the JCL.

The restart and cleanup function manages resolution of generation data group (GDG) names, JCL containing nested INCLUDEs or PROCs, and IF-THEN-ELSE statements.

TWS for z/OS interfaces with Removable Media Manager (RMM) which assists with the management of data sets.

For some restarts, JCL modifications may be necessary. The restart and cleanup function is capable of handling these modifications and even identifying the "best restart step" in a job or procedure.

Instructor Notes:

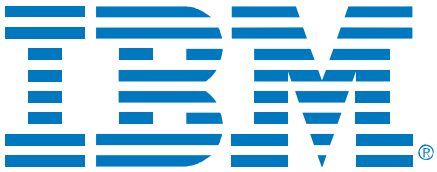
Purpose —Provide a more detailed description of the functions performed by TWS for z/OS restart and cleanup.

Details —See the student notes.

Additional Information —

Transition Statement —Because data store is crucial to restart and cleanup, let us examine what it does.

8.2 Restart and Cleanup Options



Topic 8.2

Restart and Cleanup Options

Tivoli software

Figure 8-6. Restart and Cleanup Options TM402.0

Notes:

Instructor Notes:

Purpose —This topic introduces and describes the dialog options for selecting restart and cleanup for an operation.

Details —Restart and cleanup is enabled by TWS for z/OS initialization parameters; but it must be selected for each operation.

Additional Information —

Transition Statement —Let us see how to select restart and cleanup for an operation.

Cleanup Options for Operation Restart

Operation Details - Option 9

```
-------------------- RESTART AND CLEANUP OPERATION DETAILS --------------------
Command ===>
Enter/Change data below:
Application             : DEMO#MULTI#RUN   DEMO MULTIPLE DAILY RUNS
Operation               : UNCP 015
Job name                : DEMOPROC


Clean Up Type       ===> M    Clean up Type (A/I/M/N)

Expanded JCL        ===> Y    Use expanded JCL for Restart (Y/N)

User Sysout         ===> N    Log user sysout too (Y/N)
```

Figure 8-7. Cleanup Options for Operation Restart TM402.0

Notes:

- You can specify the cleanup action to be taken on computer workstations for operations running on z/OS. Also, you can specify if TWS for z/OS will use the JCL extracted from the JESJCL sysout. If necessary, you can specify if user sysout support is needed.
- Even if the restart and cleanup function is activated in your system, it is by default not used for operations. To use the function for job you must specify your choices on the RESTART AND CLEANUP OPERATIONS DETAILS panel. You specify your choices for:
  - Clean Up Type:
    - **A** -- Automatic. When the operation is ready to be submitted and the controller selects it for submissions, the controller automatically finds the cleanup actions to be taken and also inserts them as the first step in the JCL of the restarted job.
    - **I** -- Immediate. Data set cleanup is immediately performed if the operation ends in error.
    - **M** -- Manual. Data set cleanup actions are deferred for the operation. They are performed when initiated manually from the panel.

— **N** None. No data set cleanup actions are performed.

- Which JCL to use for restarts:
  - — **Y** -- Use the fully expanded JCL.
  - — **N** -- Use the JCL contained in the libraries of TWS for z/OS.
- Storing of users sysout:
  - — **Y** -- Data store logs user sysout.
  - — **N** -- Data store does not log user sysout.

Instructor Notes:

Purpose —Describe the restart and cleanup operation details.

Details —See the student notes.

Additional Information —

Transition Statement —The next visual describes the types of data set cleanup actions.

Cleanup Actions

- Delete data sets created in a job
- Uncatalog data sets cataloged in a job
- Catalog data sets uncataloged in a job

Tivoli software

Figure 8-8. Cleanup Actions TM402.0

Notes:

- When you perform a cleanup, you take previously allocated data sets and either catalog, uncatalog, or delete them.
- These actions can be together with restarts or completed separately. For example, you can specify the step restart and cleanup of an operation, which are completed at job run time. In other situations, you can run the data set cleanup separately, before the job reruns. In the second case, the step restart only begins after the cleanup completes.

Instructor Notes:

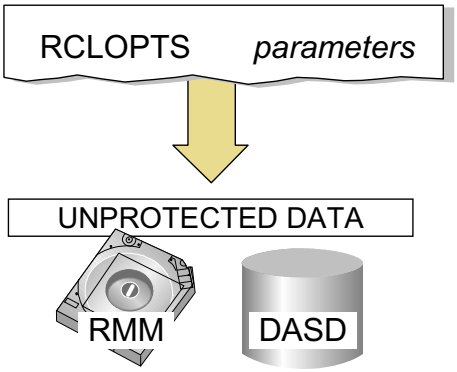
Purpose —List and describe the types of cleanup actions that can by performed by the restart and cleanup function.

Details —See the student notes.

Additional Information —

Transition Statement —I am sure that every installation has data sets that should be excluded from cleanup actions. The next visual shows how data sets are selected for cleanup actions.

Which Data Set Are Selected?



Disposition or Step Completion	Eligibility for Cleanup Action
DISP = NEW and within restart range	Eligible for Cleanup Action
DISP = OLD	NOT Eligible for Cleanup Action
DISP = SHR	NOT Eligible for Cleanup Action
DISP = MOD	NOT Eligible for Cleanup Action
JOB STEP is FLUSHED	NOT Eligible for Cleanup Action
SMS Managed	Valid Action = DELETE ONLY
Data Set has been MIGRATED	HDELETE macro invoked by TWS for z/OS

Tivoli software

Figure 8-9. Which Data Sets are Selected? TM402.0

Notes:

- When implementing the TWS for z/OS Restart and Cleanup function, you can protect data sets from TWS for z/OS deletion by using one or more of these keyword parameters in the RCLOPTS initialization statement of the controller:
  - **DDPROT(DD List)** - Defines a list of DD names that identify protected data sets.
  - **DSNPROT(DSN List)** - Defines a the list of the protected data set names.
  - **DDPRMEM(member name)** - Specifies the name of a member of the TWS for z/OS parameter library containing a list of protected DD names. DDPRMEM is mutually exclusive with DDPROT. You can refresh the list dynamically via the z/OS Modify command.
  - **DSNPRMEM (member name)** - Specifies the name of a member of the TWS for z/OS parameter library that contains the list of protected Data set names. DSNPRMEM is mutually exclusive with DSNPROT. You can refresh the list dynamically via the z/OS Modify command.
- All other data sets, including tape volumes defined to Removable Media Manager (RMM) are eligible for cleanup.

- To activate cleanup of tape volumes defined to RMM, specify RMM(YES) in the RCLOPTS initialization statement of the controller.
- This table shows which data sets are then eligible for cleanup actions.

Instructor Notes:

Purpose —Describe how to be make data sets eligible or ineligible for cleanup actions.

Details —See the student notes.

Additional Information —

Transition Statement —Once selected for cleanup, you also must define when the cleanup action should be initiated.

When Are Cleanup Actions Initiated?

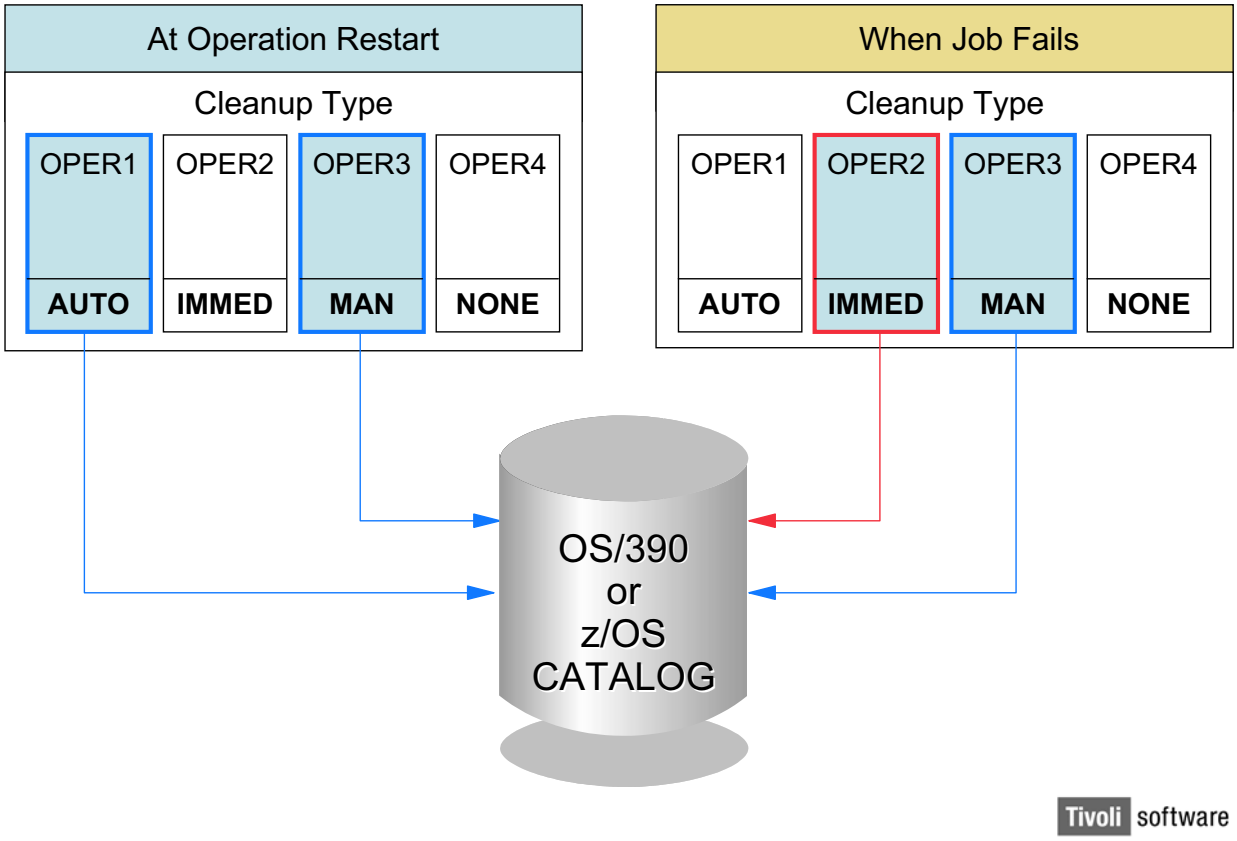


Figure 8-10. When are Cleanup Actions Initiated? TM402.0

Notes:

- AUTO (Automatic) -- Cleanup actions are automatically initiated at operation restart only.
- IMMED (Immediate) -- Cleanup actions are automatically initiated by TWS for z/OS as soon as the job ends in error.
- MAN (Manual) -- Cleanup actions are initiated by a dialog user and can be done either at operation restart or when the job fails.
- To view the result of a cleanup action, select option 4 (DISPLAY CLEANUP) on the Operation Restart and Cleanup panel. The View Cleanup Results panel is displayed; it provides the following status information:
  - Whether the data set cleanup was completed or ended-in-error
  - The job name and job ID that performed the cleanup actions
  - Whether the scheduler removed the data set from the catalog, un-cataloged the data set, or cataloged the data set.

- Whenever the operation is started from the panels, the cleanup actions are shown to the user for confirmation, only if the AUTOMATIC CHECK OPC panel option is set to YES.

Instructor Notes:

Purpose —To identify and describe how to control the start of data set cleanup.

Details —See the student notes.

Additional Information —

Transition Statement —By default, TWS for z/OS initiates cleanup actions without showing you the affected data set. The next visual shows how to request notification of data sets targeted for cleanup actions.

Request Notification of Cleanup Actions

```
Option 0.7 - CLEANUP CHECK

------------------ SETTING CHECK FOR AUTOMATIC CLEANUP TYPE -------------------
Command ===>


CLEANUP CHECK    ===> Y    Specify Y to  check  the list of the cleanup
                           actions returned  by OPC  when you request a
                           RESTART function with cleanup type automatic.

                           Specify N to bypass the check.

                           Default is N .
```

Figure 8-11. Request Notification of Cleanup Actions TM402.0

Notes:

Select the TWS for z/OS primary panel option **0.7** to request notification of cleanup actions.

Instructor Notes:

Purpose —Show how to request notification of cleanup actions.

Details —See the student notes.

Additional Information —

Transition Statement —

Exceptions to IMMEDIATE Cleanup Actions

- These error codes:

At or Before Submission	After Submission
OSEQ	MCP
OSUB	OSSQ
OSUF	OSSS
OSUP	OFSQ
OJCV	OFSS
JCLI	OFSC

Tivoli software

Figure 8-12. Exceptions to IMMEDIATE Cleanup Action TM402.0

Notes:

- If an operation ends with one of these errors, TWS for z/OS does not perform any immediate cleanup actions.
- The error codes in the After Submission column indicate that workload restart has failed, or that the operation has been manually set to error. TWS for z/OS automatically changes cleanup action from immediate to manual.

Instructor Notes:

Purpose —Identify and describe exceptions to immediate cleanup actions.

Details —See the student notes.

Additional Information —

Transition Statement —

Overriding Cleanup Actions

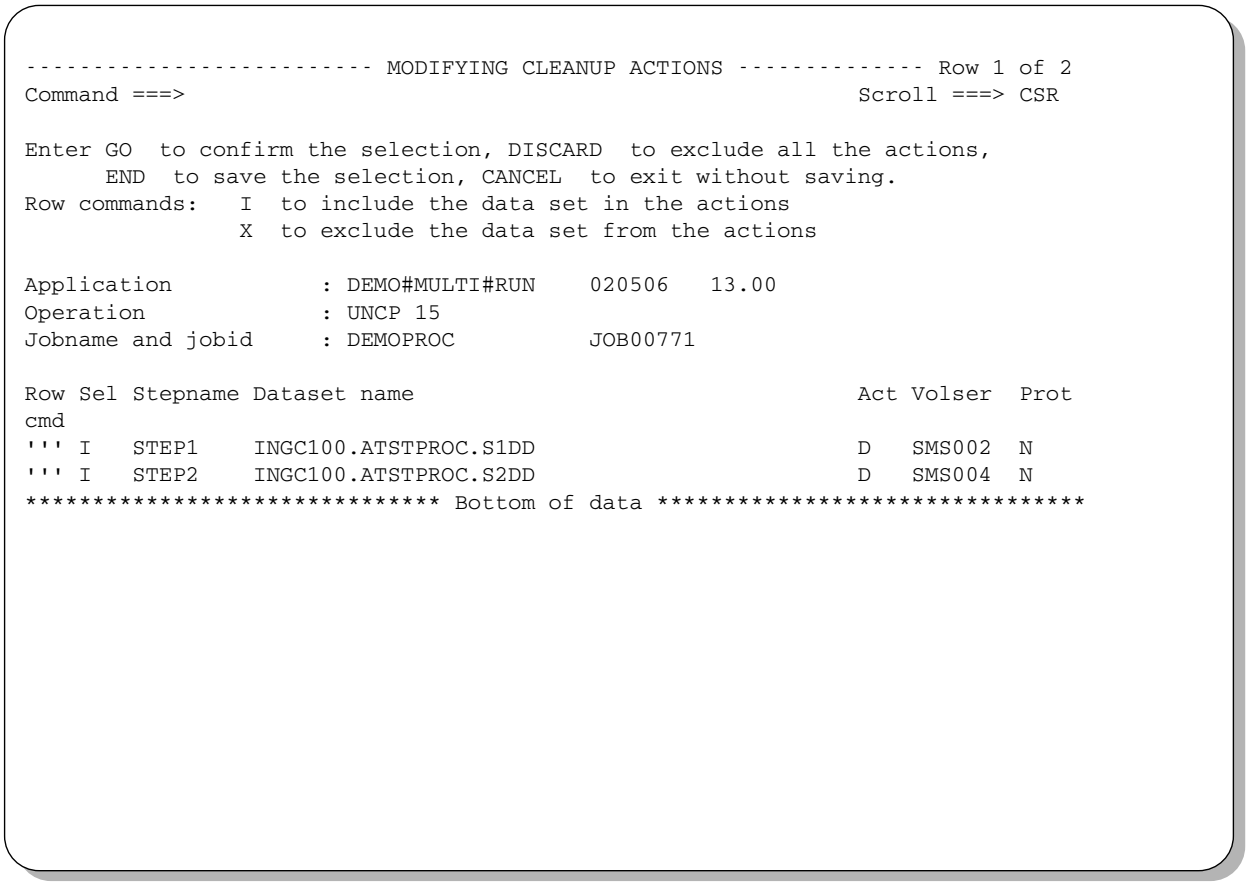


Figure 8-13. Overriding Cleanup Actions TM402.0

Notes:

- You can modify or browse cleanup information from a number of places in the TWS for z/OS dialog. In most cases, you will probably be using the Handling Operations Ended in Error panel to browse, modify, execute, or discard cleanup actions.
- When you select a job restart, the Modifying Cleanup Actions panel is displayed.
- You can override the cleanup for selected data sets on this panel.

Instructor Notes:

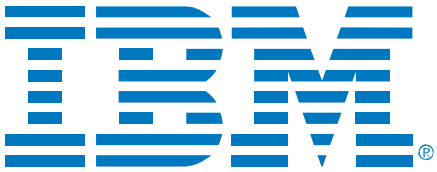
Purpose —Introduce and describe how to override suggested cleanup actions.

Details —See the student notes.

Additional Information —

Transition Statement —

8.3 Using the Restart and Cleanup Dialog



Topic 8.3

Using the Restart and Cleanup Dialog

Tivoli software

Figure 8-14. Using the Restart and Cleanup Dialog TM402.0

Notes:

Instructor Notes:

Purpose —Describe how to us the restart and cleanup dialog in the CP.

Details —

Additional Information —

Transition Statement —

Operation Restart and Cleanup

```
----------------------- OPERATION RESTART AND CLEANUP -----------------------
Option ===>

Application           : DEMO#MULTI#RUN    020506    13.00
Operation             : UNCP 15
Jobname and jobid     : DEMOPROC          JOB00772

Clean Up Result       : Completed


Edit JCL        ===> Y             Edit JCL before Restart
Expanded JCL    ===> N             Use Expanded JCL


Select one of the following:

1 STEP RESTART                   - Request a Step Restart
2 JOB RESTART                    - Request a Job  Restart
3 START CLEANUP                  - Request Cleanup
4 DISPLAY CLEANUP                - Display Cleanup result
```

Figure 8-15. Operation Restart and Cleanup TM402.0

Notes:

- The start cleanup options can be requested only for an operation with the cleanup type set to automatic, immediate, or manual.
- When you request restart and cleanup, either to rerun a completed operation or to restart an ended-in-error operation, you see the OPERATION RESTART AND CLEANUP panel.
- On this panel, you can perform the cleanup independent of the restart and view the results or you can choose to initiate the cleanup by selecting job or step restart.

Instructor Notes:

Purpose —Introduce navigation of the restart and cleanup dialog in the CP.

Details —See the student notes.

Additional Information —

Transition Statement —The next visual introduces the step restart dialog.

Step Restart Selection List

```
------------------------ STEP RESTART SELECTION LIST ------------ Row 1 of 15
Command ===>                                                  Scroll ===> CSR

Primary commands:  GO  - to confirm the selection, END - to save it,
                   CANCEL - to exit without saving
User Selection     S - First step to be restarted
                   E - Last step to be restarted
                   X - Step must be excluded
                   I - Step is included if inside the restart range.

Application           : DEMO#MULTI#RUN    020506   13.00
Operation             : UNCP 15
Jobname and jobid     : DEMOPROC          JOB00772

Best Restart Step     :         DELDS2   0002
Current selected Step :         DELDS2   0002

Usr Act Rest Step StepName ProcStep PgmName  Step      Step       Compl.
Sel Sel      No                             Type      Status     Code
X   X   N    0001 EQQCLEAN EQQCLEAN EQQCLEAN Clean up  Executed   0000
S   S   B    0002          DELDS2   EQQDELDS Normal    Executed   0000
'   I   Y    0003          STEP1    IEFBR14  Normal    Executed   0000
'   I   N    0004          RC04#1   OPCUTIL  Normal    Executed   0004
```

Figure 8-16. Step Restart Selection List TM402.0

Notes:

- If you select step restart, the Step Restart Selection panel is displayed.
- After selecting the restart step, enter the GO command, and the Modifying the Cleanup Action panel is displayed.

Instructor Notes:

Purpose —Introduce the step-restart selection dialog.

Details —See the student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- List the tasks that can be performed by the Restart and Cleanup function
- Specify appropriate restart and cleanup options for selected operations
- Use the restart and cleanup dialog to:
  - Perform job or step-level operation restarts
  - Override data set cleanup actions

Tivoli software

Figure 8-17. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 9. Special Resources

What This Unit is About

This unit introduces the concept, function, and features of IBM Tivoli Workload Scheduler for z/OS special resources.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Describe how special resources can be used
- Define special resources in the database
- Use the special resource monitor.

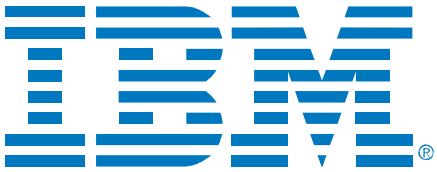
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 9: Special Resources

Tivoli software

Figure 9-1. Special Resources TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- Describe how special resources can be used
- Define special resources in the database
- Use the special resource monitor

Figure 9-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

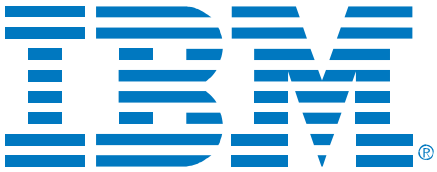
Purpose —

Details —

Additional Information —

Transition Statement —

9.1 Introducing Special Resources



Topic 9.1

Introducing Special Resources

Tivoli software

Figure 9-3. Introducing Special Resources TM402.0

Notes:

Instructor Notes:

Purpose —This topic introduces the concept of TWS for z/OS special resources and discusses the uses of special resources.

Details —

Additional Information —

Transition Statement —

The Special Resources Concept

Special Resource Group:
Tools

Special Resource Umbrella
is available

A Special Resource
is a logical construct

Special Resource Sunglasses
is needed to run application
HOLIDAYS

Tivoli software

Figure 9-4. The Special Resources Concept TM402.0

Notes:

- It is recommended that you use TWS for z/OS special resources instead of workstation fixed resources to manage limited resources in your scheduling environment.
- You can use special resources as logical representations of any type of limited resource.
- You also use special resources to model logical constructs. For example:
  - Certain jobs, although they are not interdependent, cannot be run at the same time.
  - A backup job must not be started when a specified subsystem is active.
- Special resources are logical representations of real-world scheduling requirements.

Instructor Notes:

Purpose —Introduce key concepts of special resources.

Details —As done with the rest of the TWS for z/OS environment, you can uses special resources to model workload requirements.

Additional Information —

Transition Statement —

Special Resources

- Represent real resources:
  - Hardware
  - Datasets
  - Whatever
- Can restrict operations running together
- Give accurate schedules
- Availability dynamically adjusted

Figure 9-5. Special Resources TM402.0

Notes:

- You use special resources to describe special scheduling requirements for your environment.
- You can use these logical entities to represent any type of limited resource such as:
  - tape drives
  - communication lines
  - data sets
  - databases
  - JES initiators
- You can also use special resources to serialize the running of work.
- You can define special resources in the special resource data base or dynamically create them.
- TWS for z/OS daily planning functions generates utilization reports for special resources associated with operations in the current plan.

- You can use the special resource monitor dialog to monitor and modify special resources in the current plan
- Special resources can be used with TWS for z/OS operations and TWS jobs; however, you lose the fault-tolerance of the distributed agent if you use special resources with its jobs. Only the controller determines the availability of a resource and consequently lets the distributed agent start the operation.

Instructor Notes:

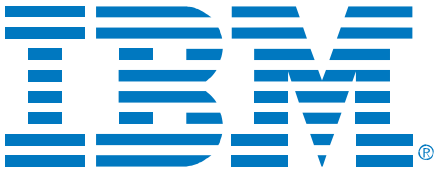
Purpose —Get more specific in identifying the uses of special resources.

Details —You can use special resources to represent whatever you want. When creating the current plan, the daily planning process takes into account special resource requirements for operations, and the availability of the required special resources.

Additional Information —

Transition Statement —Let us learn how to define Special Resources.

9.2 Creating Special Resources



Topic 9.2

Creating Special Resources

Tivoli software

Figure 9-6. Creating Special Resources TM402.0

Notes:

Instructor Notes:

Purpose —Describe how to create a special resource using the dialogs.

Details —

Additional Information —

Transition Statement —

The Special Resources Database

- Name resource
- Availability
- Amount
- Planning/control
- Action on error
- Hiperbatch
- Time intervals
- Workstations

Figure 9-7. The Special Resource Database TM402.0

Notes:

- You create resources using the Special Resources panel.
- The Special Resource panel updates the resource database, which has these details of each resource:
  - **Name** -- Up to 44 characters.
  - **Availability** -- Yes (Y) or no (N).
  - **Quantity** -- 1 to 999999.
  - **Used for** -- How TWS for z/OS is to use the resource. For:
    - P (planning, when the current plan is extended)
    - C (control, when an operation starts)
    - B (both)
    - N (neither)
  - **On-error action** -- What happens if an operation using this resource ends in error (and does not have an overriding keep-on-error specification in the operation definition):

  - F (free its full allocation of this resource, both those allocated exclusive and those allocated shared)
  - FS (free its full shared allocation of this resource)
  - FX (free its full exclusive allocation of this resource)
  - K (keep its full allocation of this resource)
  - Blank (use the default specified in the ONERROR keyword of the RESOPTS statement. Refer to *Customization and Tuning*.)
- **Connected workstations** -- A list of the workstations where operations can allocate the resource.

- The quantity, availability, and list of workstations, can vary with time. You can create time *intervals* to control each special resource.

Instructor Notes:

Purpose —Describe the fields in the creating a special resource dialog in the SR database.

Details —See the student notes.

Additional Information —

Transition Statement —

Creating a Special Resource

```
------------------------ CREATING A SPECIAL RESOURCE ------------------------
 Option ===>

 Select one of the following:

 1 INTERVALS  - Specify intervals
 2 WS         - Modify default connected work stations

 SPECIAL RESOURCE   ===> TAPES2_____________________________________
 TEXT               ===> MODEL 2 TAPE DRIVES_____________________
 SPECRES GROUP ID   ===> ________
 Hiperbatch         ===> N       DLF object Y or N
 USED FOR           ===> B       Planning and control C , P , B or N
 ON ERROR           ===> F_      On error action F , FS , FX , K or blank



 Defaults
   QUANTITY         ===> 15____  Number available 1-999999
   AVAILABLE        ===> Y       Available Y or N
```

Figure 9-8. Creating a Special Resource TM402.0

Notes:

- This is the panel you use to create special resources.
- You use the INTERVALS option to create time intervals in which you can specify quantity and availability attributes for the resource.
- You use the WS option to specify the names of workstations that can allocate (connect to) the special resource. If you do not use this option, all workstations will allocate the resource.
- The default values specify the allocated availability and quantity of the resource for non-defined time intervals.

Instructor Notes:

Purpose —Explain the fields on the SR creation panel.

Details —See the student notes.

Additional Information —

Transition Statement —

Specifying Availability Intervals

```
---------------- MODIFYING INTERVALS FOR A SPECIAL RESOURCE ----- Row 1 of 10
Command ===>                                                  Scroll ===> CSR

Enter any of the row commands below:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete, or,
S - Work stations

Special resource : TAPES2
Text             : MODEL 2 TAPE DRIVES

Row Day of         From  To    Qty    A
cmd week or Date   Time  Time
''' MONDAY_________ 06.00 18.00 5_____ Y
''' MONDAY_________ 18.00 24.00 10____ Y
''' TUESDAY________ 06.00 18.00 5_____ Y
''' TUESDAY________ 18.00 24.00 10____ Y
''' WEDNESDAY______ 06.00 18.00 5_____ Y
''' WEDNESDAY______ 18.00 24.00 10____ Y
''' THURSDAY_______ 06.00 18.00 5_____ Y
''' THURSDAY_______ 18.00 24.00 10____ Y
''' FRIDAY_________ 06.00 18.00 5_____ Y
''' FRIDAY_________ 18.00 24.00 10____ Y
****************************** Bottom of data ********************************
```

Figure 9-9. Specifying Availability Intervals TM402.0

Notes:

- You use this panel to specify availability and quantity for time intervals you specify.
- TWS for z/OS automatically allocates the resource based on your specifications.
- The values specified on this panel override the defaults.
- If necessary, you can enter the Work Stations row command **S** to an interval to display the panel on which you can specify the workstations that can connect to the special resource.

Instructor Notes:

Purpose —Describe how to use define availability intervals.

Details —See the student notes.

Additional Information —

Transition Statement —From this panel, you can select intervals during which to control access to the SR.

Specifying Work Station Connections

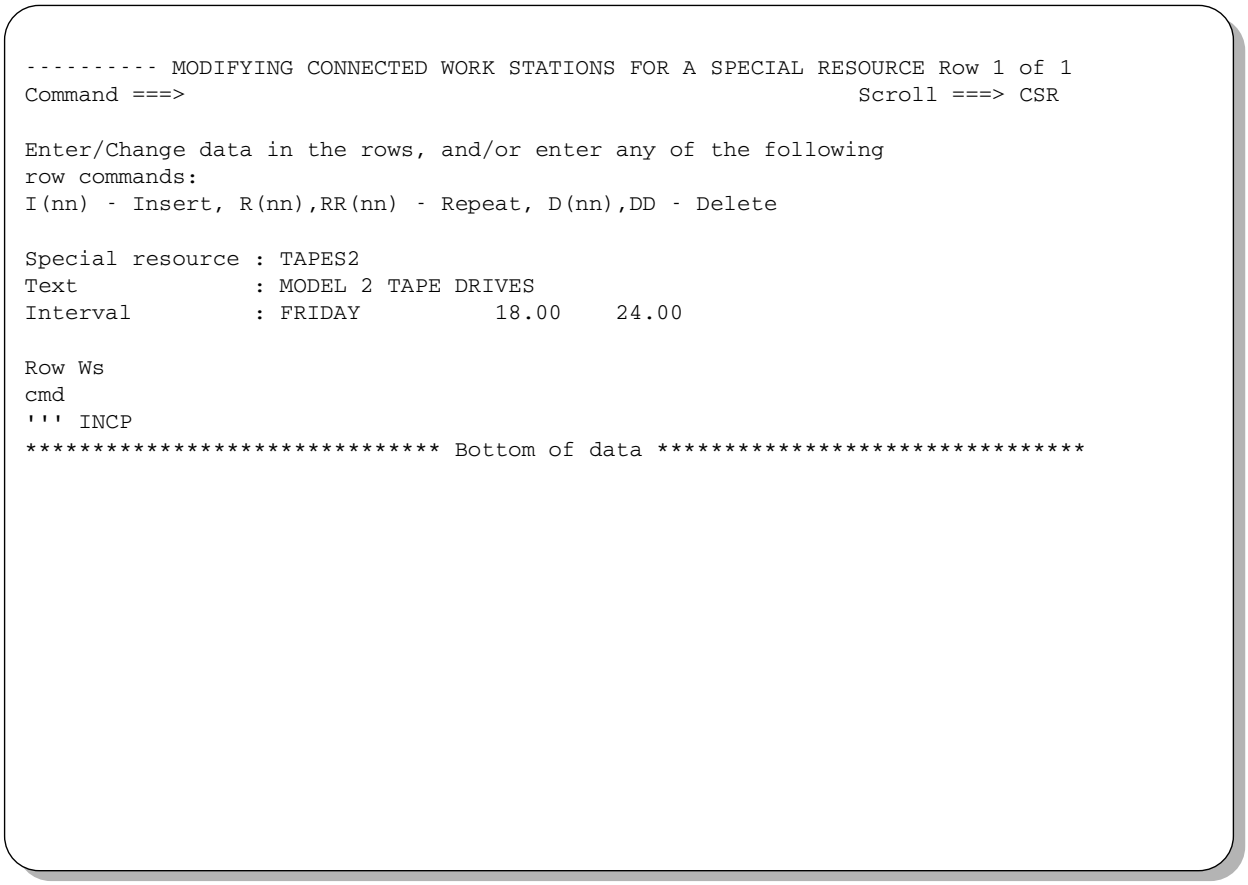


Figure 9-10. Specifying Work Station Connections TM402.0

Notes:

On this panel, you specify the names of workstations that you authorize to allocate the special resource.

Instructor Notes:

Purpose —Show how to restrict workstation access to the SR.

Details —See the student notes.

Additional Information —

Transition Statement —You can also create special resources dynamically.

Creating Resources Dynamically

- BATCHOPT .....
  DYNAMICADD(YES)
- RESOPTS .....
  DYNAMICADD(YES/EVENT/OPER)
- SRSTAT CREATE(YES)

Tivoli software

Figure 9-11. Creating Resources Dynamically TM402.0

Notes:

If one of these situations arise, TWS for z/OS can dynamically create the referenced special resource in the current plan:

1. Daily planning discovers an operation needs a resource that is not defined in the special resource database.
2. An operation or event refers to a resource that is in the database but is not in the current plan.
3. An operation or event refers to a resource that is neither in the database nor in the current plan.

In the first situation if YES is specified for the DYNAMICADD parameter in the BATCHOPT statement , TWS for z/OS creates a special resource during planning. The special resource database is not updated. The special resource is added to the current plan instead.

In situation number 2 TWS for z/OS creates a special resource in the current plan if it is defined in the database but was not added during daily planning. -- Resources are only added during daily planning if there is an operation in the plan that references the resource.

-- When you refer to the resource during the life of the plan, TWS for z/OS adds the resource from the database.

The third situation requires you to set the keyword value for the DYNAMICADD parameter in the RESOPTS statement to YES or OPER, if you add an occurrence to the current plan through the dialog and one of the operations references a special resource.

If you use SRSTAT, a TSO-authorized TWS for z/OS command for managing special resources, TWS for z/OS looks at the CREATE keyword of SRSTAT (it must be YES) and the DYNAMICADD keyword of RESOPTS (it must be EVENT or YES).

Instructor Notes:

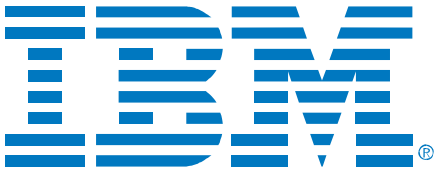
Purpose —Introduce the TWS for z/OS parameters that are used to allow dynamic creation of special resources.

Details —See the student notes.

Additional Information —

Transition Statement —Dynamic creation facilitates using SRs that were not previously defined.

9.3 Using Special Resources



Topic 9.3

Using Special Resources

Tivoli software

Figure 9-12. Using Special Resources TM402.0

Notes:

Instructor Notes:

Purpose —Show how to define special resource requirements and monitor the use of special resources in the CP.

Details —

Additional Information —

Transition Statement —To use the SRs you must state requirements in the operation in the AD database.

Specifying Operations Resource Requirements

- Resource name
- Amount required
- Exclusive or shared use
- Keep on error

Tivoli software

Figure 9-13. Specifying Operations Resource Requirements TM402.0

Notes:

You use operation details option 3 to define special resources for an operation.

Resource name: The name of the resource, up to 44 characters. If you are unsure of the name, you can use the global search characters % and * and TWS for z/OS will display a list of matching names from which you can select the resources that you need.

Amount required: The number of resources that the operation allocates. If you leave this field blank, the operation is allocated the whole quantity currently available.

Exclusive or shared use: A special resource unit allocated to an operation as shared can be used by other sharing operations at the same time, but is unavailable to operations that need exclusive allocation. Resource units allocated as exclusive can be used only by that operation. TWS for z/OS does not start another operation that requires the allocated units until the first operation ends.

Action on error: This attribute specifies whether the special resources remains allocated if the operation fails. If left blank, TWS for z/OS takes the default for the resource.

Instructor Notes:

Purpose —Explain fields on the operations SR panel.

Details —See the student notes.

Additional Information —

Transition Statement —

Special Resource Operation Detail Example

```
------------------------------ SPECIAL RESOURCES ------------------ Row 1 of 1
Command ===>                                                Scroll ===> CSR

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete

Operation              : UNCP 015

Row  Special                                      Qty    Shr Keep On
cmd  Resource                                            Ex  Error
'''' TAPES2_____________________________________ 3_____ X   N
****************************** Bottom of data ********************************
```

Figure 9-14. Special Resource Operation Detail Example TM402.0

Notes:

This is an example of how you assign a special resource to an operation.

Instructor Notes:

Purpose —Demonstrate the Special Resource Operation Detail panel.

Details —See the student notes.

Additional Information —

Transition Statement —Because the availability special resources can affect the timely completion of work, let us examine a SR report that is generated by the daily planning process.

Special Resource Utilization Report Example

```
RESOURCE NAME    : TAPES2                          DESCRIPTION: MODEL 2 TAPE DRIVES
DYNAMICALLY ADDED: NO
INTERVAL START|INTERVAL   END|          |         |    PLANNED|ALLOCATION FAILURES BY|NUMBER OF
DATE      TIME|DATE      TIME|AVAILABLE|QUANTITY|UTILIZATION| CONNECT |    QUANTITY|CONTENTIONS
================================================================================================
02/05/03 05.00|02/05/03 06.00|     YES |      15|          0 |       0 |           0|           0
02/05/03 06.00|02/05/03 18.00|     YES |       5|          0 |       0 |           0|           0
02/05/03 18.00|02/05/03 24.00|     YES |      10|          0 |       0 |           0|           0
02/05/04 00.00|02/05/04 23.59|     YES |      15|          3 |       0 |           0|           0
02/05/05 00.00|02/05/05 23.59|     YES |      15|          0 |       0 |           0|           0
02/05/06 00.00|02/05/06 06.00|     YES |      15|          0 |       0 |           0|           0
02/05/06 06.00|02/05/06 06.06|     YES |       5|          0 |       0 |           0|           0

                            >>>>>>>  END OF PLANNED RESOURCE UTILIZATION REPORT   <<<<<<<
```

Figure 9-15. Special Resource Utilization Report Example TM402.0

Notes:

This is an example of the planned special resource utilization report that is generated by the daily planning batch function.

Instructor Notes:

Purpose —Describe the columns in a Special Resource Utilization Report.

Details —This example shows a planned SR utilization report. Refer to the Planning and Scheduling the section of workload publication.

Additional Information —

Transition Statement —In the CP, you have the ability to monitor the availability and use of special resources. Let us take a look at the special resource monitor.

The Special Resource Monitor

- Monitors and manages resources in current plan
- Can change amount and availability
  - Manually
  - By other programs
  - By RODM
- Lists operations waiting for resource
- Lists operations using the resource

Tivoli software

Figure 9-16. The Special Resource Monitor TM402.0

Notes:

- The special resource monitor is a tool that you use to monitor and manage special resources in the current plan.
- You use the special resource monitor to look at and modify special resources in the current plan.
- In the special resource monitor, you see the list of resources that are needed by operations in the current plan.
- Why do you need to monitor special resources when operations automatically hold and release them according to their descriptions in the current plan?

  Assume that a special resource has been created to represent the tape drives reserved for TWS for z/OS scheduled jobs. The current plan is built assuming that a certain number of tape drives will be available. If one of the tape drives is taken off-line and you do not use RODM or otherwise automatically notify TWS for z/OS; it no longer has a true picture of the real world, and continues to allocate the off-line tape drive to any job

that needs a tape drive. The job will wait, because z/OS knows that the tape drive is off-line.

To prevent TWS for z/OS from starting a job that will only wait and use operating system resources, you use the Special Resource Monitor panel to reduce the number of tape drives by one. You specify a deviation of -1 (minus one) to show that the quantity is reduced. When the tape drive is online again, you use the special resource monitor panel to reset the deviation.

Instructor Notes:

Purpose —Introduce and explain the special resource monitor.

Details —See the student notes.

Additional Information —

Transition Statement —Let us examine the final piece of the puzzle.

Special Resources and Hiperbatch

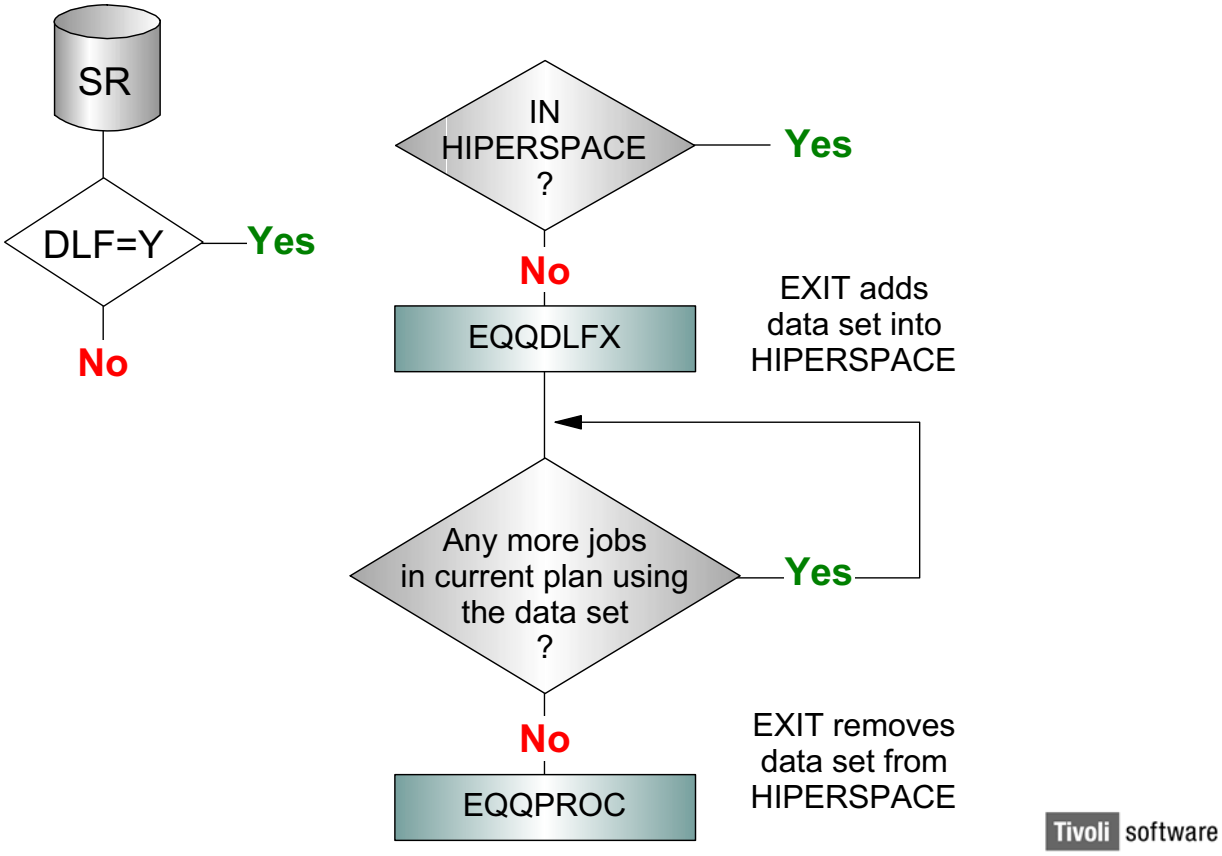


Figure 9-17. Special Resources and Hiperbatch TM402.0

Notes:

- *Hiperbatch* is a z/OS performance enhancement that works with the Data Lookaside Facility (DLF) to allow batch jobs and started tasks to share access to a data set, or data object.
- Within TWS for z/OS, a data set that is eligible for Hiperbatch is treated as a resource. Using the Resources panel, you can define data sets with the DLF attribute.
- TWS for z/OS issues enqueues on the job and data set name to notify the DLF exit that the job to be scheduled will use Hiperbatch.
- When the job ends, TWS for z/OS checks if the same data set will be used by the immediate successor operation or by any other ready operation. If so TWS for z/OS does not purge the data object. Otherwise, TWS for z/OS initiates purge processing of the data object (that is, TWS for z/OS removes it from Hiperspace).

Instructor Notes:

Purpose —Describe Hiperbatch and its relationship to special resources.

Details —See the student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- Describe how special resources can be used
- Define special resources in the database
- Use the special resource monitor

Tivoli software

Figure 9-18. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 10. Unplanned Work

What This Unit is About

This unit introduces TSO authorized IBM TWS for z/OS commands, the event triggered tracking, and data set triggering functions that you can use to handle unplanned work.

What You Should Be Able to Do

After completing this unit, you should be able to:

- Use the OPSTAT and SRSTAT commands to handle unplanned work and control work flow
- Use Event Triggered Tracking to handle unplanned work
- Create Data Set Triggering table entries
- Use Data Set Triggering to enhance Event Triggered Tracking

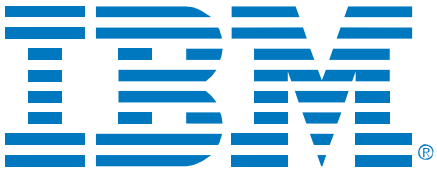
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 10: Unplanned Work

Tivoli software

Figure 10-1. Unplanned Work TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- Use the OPSTAT and SRSTAT commands to handle unplanned work and control workflow
- Use Event Triggered Tracking to handle unplanned work
- Create Data Set Triggering table entries
- Use Data Set Triggering to enhance Event Triggered Tracking

Figure 10-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unplanned Work

Figure 10-3. Unplanned Work TM402.0

Notes:

- Whatever the label -- unplanned, on-request, ad-hoc, or on-demand, this type of work forms a large part of the work handled by many scheduling departments. TWS for z/OS is well equipped the handle unplanned work.
- This unit covers some of the functions and commands that you can use to handle unplanned work. It covers:
  - TSO-authorized, TWS for z/OS commands OPSTAT and SRSTAT
  - Event Triggered Tracking
  - Data set triggering

Instructor Notes:

Purpose —To introduce the unit.

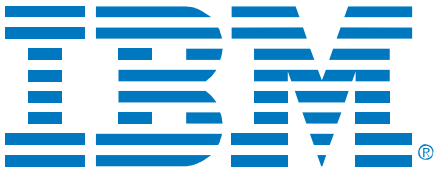
Details —Whatever the label -- unplanned, on-request, ad-hoc, or on-demand, this type of work forms a large part of the work handled by many scheduling departments. TWS for z/OS is well equipped the handle unplanned work.

- This unit covers some of the functions and commands that you can use to handle unplanned work. It covers:
  - TSO-authorized, TWS for z/OS commands OPSTAT and SRSTAT
  - Event Triggered Tracking
  - Data set triggering

Additional Information —

Transition Statement —We will start with the OPSTAT and SRSTAT commands.

10.1 OPSTAT and SRSTAT Commands



Topic 10.1

OPSTAT and SRSTAT Commands

Tivoli software

Figure 10-4. OPSTAT and SRSTAT Commands TM402.0

Notes:

Instructor Notes:

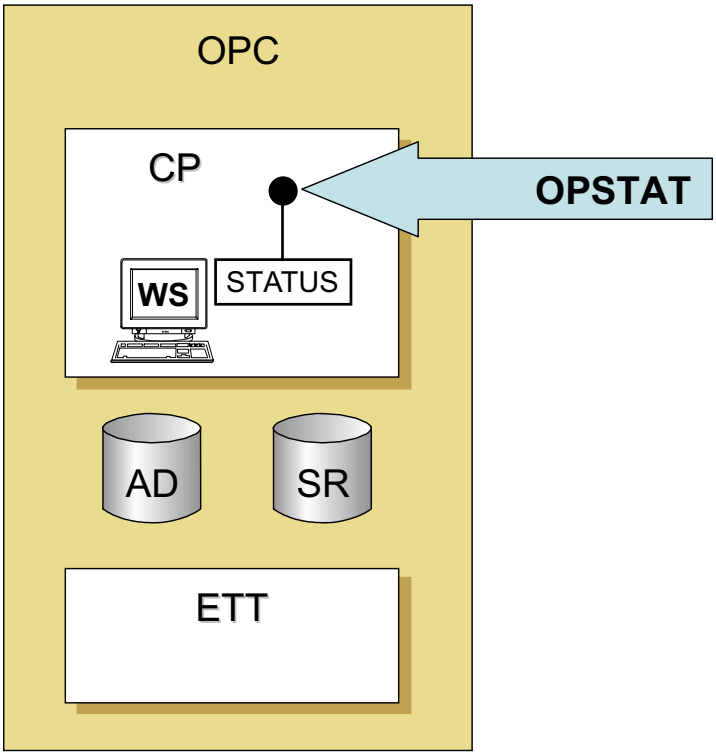
Purpose —To introduce the OPSTAT and SRSTAT commands.

Details —TWS for z/OS has features which allow events that occur outside of its environment to affect work under its control. Two of these features can be found in the use of the OPSTAT and SRSTAT TSO-authorized commands.

Additional Information —

Transition Statement —First we look at the OPSTAT command.

The OPSTAT Command



Tivoli software

Figure 10-5. The OPSTAT Command TM402.0

Notes:

- You can use the OPSTAT command (Operation Status) to change the status of an operation at any workstation, except non-reporting workstations.
- You can use the status to:
  - **C** - completed (probably the most common use)
  - **E** ended in error
  - **I** - interrupted
  - **S** - started
  - **X** - reset the status to its previous logical status
  - **Q** - extended status Q set
  - **T** - extended status S set
- Events generated by OPSTAT are matched against operations on the ready list. Events received for operations in waiting (W) or complete (C) status are ignored. You cannot change the status of a waiting C because it has incomplete predecessors.
- Jobs and started tasks that are running are always allowed to finish.

- You can issue the command directly from TSO, or from a batch job.
- The OPSTAT command can be used to place the starting of an operation into the hands of a user who needs to control it. By submitting a job with the OPSTAT command, the user can set the status of a manual predecessor operation to complete, and subsequently release successor operations.
- You can use the OPSTAT command from both inside and outside of TWS for z/OS.

Instructor Notes:

Purpose —Explain OPSTAT.

Details —The illustration is used again in this unit, and works well in explaining the concept. I start by reminding students of the Current Plan exercise where their computer operation was dependent on a JCL (job setup) workstation operation; and the computer operation would not start until the job setup was complete.

You can use OPSTAT to release non-waiting operations in the CP from outside TWS for z/OS.

Additional Information —Box represents TWS for z/OS controlled area. Within is (from bottom up) ETT, the AD (application database) and SR (special resource database). The OPSTAT command comes in from outside and changes the status of an operation at a workstation in the CP (current plan).

Transition Statement —Let's look at the command syntax.

Using the OPSTAT Command

- User needs update authority to the Ready List
- The WSNAME parameter must be specified to identify the workstation
- The command is directed to the tracker not the controller
- Specify enough parameters to avoid ambiguity
- Minimum recommend parameters:
  - ADID(*application_description_name*)
  - JOBNAME(*jobname*)
  - STATUS(*new_status*)
  - SUBSYS(**MSTR**)
  - TRACE(**1**)
  - WSNAME(*workstation_name*)

Tivoli software

Figure 10-6. Using the OPSTAT Command TM402.0

Notes:

- You can invoke OPSTAT as a TSO command or by using a batch job step that executes program EQQEVPGM.
- The ID of the user who issues the command must be authorized to update the Ready List.
- You should use the minimum recommended parameters and add others if necessary to ensure that the correct operation is updated.
- If you specify **MSTR**, the OPSTAT command is directed to all tracker subsystems on the z/OS system where the OPSTAT command is issued. The advantage of this is your job is not affected by subsystem name changes.
- You should also use the full parameter name not abbreviations to avoid the command from failing due to ambiguity.

Instructor Notes:

Purpose — Explain the command syntax and describe the minimum, recommended parameters for the command.

Details —See the student notes. and use the Planning and Scheduling the Workload publication as a reference source.

Additional Information —

Transition Statement —The next visual shows an example of the command as used in a batch job.

OPSTAT Example

```
//*
//* THIS JCL IS USED TO DEMONSTRATE USE OF THE OPSTAT COMMAND
//*
//STEP1    EXEC PGM=EQQEVPGM,REGION=512K
//EQQMLIB  DD DSN=OPC.EQQMLIB,DISP=SHR
//EQQMLOG  DD SYSOUT=*
//SYSIN    DD *
  OPSTAT WSNAME(GAUT) STATUS(C) JOBNAME(EA73NJOB) ADID(MYAPPL)
       SUBSYS(MSTR) TRACE(1)


/*
```

Figure 10-7. OPSTAT Example TM402.0

Notes:

- This example shows the required DD statements the program you execute when using a batch job.
- The command can span multiple lines without continuation characters. See *Planning and Scheduling the Workload* for more information about the parameters.

Instructor Notes:

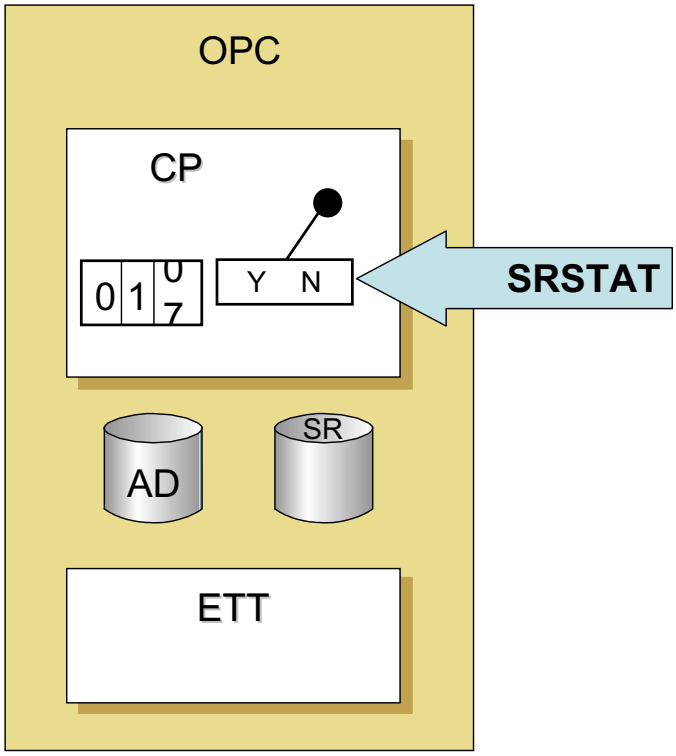
Purpose —Demonstrate the command in a batch job.

Details —See the student notes.

Additional Information —

Transition Statement —We now examine the SRSTAT command.

The SRSTAT Command



Tivoli software

Figure 10-8. The SRSTAT Command TM402.0

Notes:

- The SRSTAT command (special resource status) can be used to change the overriding (global) availability, amount, and deviation of special resources.
- The SRSTAT command can be issued direct from TSO, or within a batch job. The batch job can be in a TWS for z/OS application or one submitted outside the control of TWS for z/OS.
- You can include the command in a job step which creates a dataset, which is represented by a special resource, so it is made available for other jobs to use without them having to wait until the job is completed.

Instructor Notes:

Purpose —Introduce the SRSTAT command.

Details — The SRSTAT command can:3

- Make the SR available or not available
- Alter the global quantity of the SR
- Alter the deviation quantity of the SR.

Previously SRSTAT made changes to the SR database, now all changes are made to the SR monitor.

The expected way of changing the amount of resources is to use deviation. If you manually update the deviation field then you over type the existing deviation and put in a new value that is added to, or subtracted from the global, interval or default value. If you change deviation through SRSTAT the figure currently in the deviation field is modified by the amount stated in SRSTAT; it is not replaced by it. The intention is that you do not need to be aware of the current value in the field, and several users can alter the value.

Additional Information —Box represents TWS for z/OS controlled area. Within is (from bottom up) ETT, the AD (application database) and SR (special resource database). In the current plan (CP) is the special resource monitor. The SRSTAT command comes in from outside and changes the availability and amount of the special resource.

Transition Statement —The next visual shows an example of the command as used in a batch job.

Using the SRSTAT Command

- User needs update authority to the named resource
- The command is directed to the tracker not the controller
- Minimum recommend parameters:
  - AVAIL(*resource_availability*)
  - SUBSYS(**MSTR**)
  - TRACE(**1**)

Tivoli software

Figure 10-9. Using the SRSTAT Command TM402.0

Notes:

- The issuer of the command must be authorized to update the named resource.
- If you specify **MSTR**, the SRSTAT command is directed to all tracker subsystems on the z/OS system where the command is issued. The advantage of this is your job is not affected by subsystem name changes.
- You should also use the full parameter name not abbreviations to avoid the command from failing due to ambiguity.

Instructor Notes:

Purpose — Explain the command syntax and describe the minimum, recommended parameters for the command.

Details —See the student notes. and use the Planning and Scheduling the Workload publication as a reference source.

Additional Information —

Transition Statement —

SRSTAT Example

```
//*
//* THIS JCL IS USED TO DEMONSTRATE USE OF THE SRSTAT COMMAND
//*
//STEP1    EXEC PGM=EQQEVPGM,REGION=512K
//EQQMLIB  DD DSN=OPC.EQQMLIB,DISP=SHR
//EQQMLOG  DD SYSOUT=*
//SYSIN    DD *
  SRSTAT 'SPECIAL.RESOURCE.TEAM1' SUBSYS(MSTR) AVAIL(YES)


/*
```

Figure 10-10. SRSTAT Example TM402.0

Notes:

- This example shows the required DD statements the program you execute when using a batch job.
- The command can span multiple lines without continuation characters. Refer to *Planning and Scheduling the Workload* for more information about the parameters.

Instructor Notes:

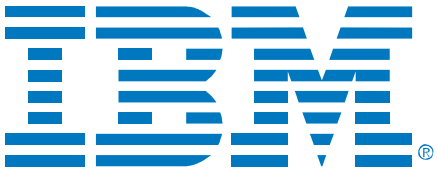
Purpose —Demonstrate the command in a batch job.

Details —See the student notes.

Additional Information —

Transition Statement —

10.2 Event-Triggered Tracking



Topic 10.2

Event-Triggered Tracking

Tivoli software

Figure 10-11. Event-Triggered Tracking TM402.0

Notes:

Instructor Notes:

Purpose —Introduce the topic

Details —Event-Triggered Tracking (ETT) is another TWS for z/OS feature that literally uses events that occur outside its environment to trigger the addition of work into the current plan.

In many installations some of the work to be run on any given day cannot be scheduled in the normal way. Some may be controlled by users, and possibly submitted by them. Some batch work, by its very nature, cannot be planned into a timed schedule very well. It may depend on some other work being done outside TWS for z/OS, such as the receipt or preparation of a dataset or other file. It may not be able to run until it is submitted from another system which is not part of the TWS for z/OS complex. The reasons may simply be historical, for example, the Finance department has always run its own updates.

Whatever the reasons for some of the work needing to be handled outside TWS for z/OS, ETT can control it.

The unplanned work is set up as an Application Description in the TWS for z/OS database, and a triggering event will add it to the Current Plan.

Additional Information —

Transition Statement —Let us take a look at ETT.

What Is Event-Triggered Tracking?

- Can be used to track work submitted outside of the TWS for z/OS environment
- Uses job-start or special resource events as triggers
- Adds application occurrences to the current plan
  - Real applications
  - Dummy applications

Figure 10-12. What Is Event-Triggered Tracking? TM402.0

Notes:

- The ETT function helps to improve the handling of unplanned work.
- Some batch work, by its very nature, cannot be planned into a timed schedule. It may depend on some other work being done outside of TWS for z/OS, such as:
  - The receipt or preparation of a dataset or other file.
  - It is submitted from another system which is not part of the z/OS complex.
  - The Finance department has always run its own updates.
- Whatever the reasons for some of the work needing to be handled outside of TWS for z/OS, ETT can control it.
- The unplanned work is set up as an Application Description in the TWS for z/OS database, and a triggering event will add it to the Current Plan.
- The event can be a job-start event or a special resource becoming available.

Instructor Notes:

Purpose —Introduce concept of ETT and define it.

Details —See the student notes.

Additional Information —

Transition Statement —There are two types of ETT triggers.

Event Types

- Special Resource Events generated by:
  - The SRSTAT command
  - Data Set Triggering
- Job-start Events generated by jobs started outside of the current plan

Figure 10-13. Event Types TM402.0

Notes:

- The two types of events that can act as triggers are:
  - **R** - *Resource events*
  - **J** - *Job events*
- TWS for z/OS generates a resource event when the status of a named special resources is set to available. This can be caused when a user issues the SRSTAT command or TWS for z/OS issues its own SRSTAT command internally to satisfy a data set trigger.
- TWS for z/OS generates a job event when a named job is submitted outside of its control.
- For both event types, the triggers must be specified in the event triggered tracking criteria table.

Instructor Notes:

Purpose —Identify the types of ETT triggers and how they are generated.

Details —See the student notes.

Additional Information —

Transition Statement —ETT triggers and the work that they add to the CP must be defined in the TWS for z/OS ETT database.

Specifying ETT Criteria

```
----------------------- MODIFYING ETT TRACKING CRITERIA ---------- Row 1 of 16
Command ===>                                                 Scroll ===> CSR

Change data in the rows, and/or enter any of the following row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat,  D(nn),DD - Delete

Row  Name of triggering event                    Id of associated  E  J  D  A
cmd                                              application       T  R  R  S
'''' L3027*_____________________________________ ETTDEMO1_______  J  Y  N  N
'''' L3027SRT___________________________________ EA736SRT_______  J  N  N  N
'''' L30271*____________________________________ DUMMY#ETT#ADD___  J  Y  N  N
'''' JOHNB.SPECIAL.RESOURCE_____________________ ETTDEMO1_______  R  N  N  Y
'''' SYS2.OPC.OPCD.G*___________________________ DUMMY#ETT#ADD#2A  R  N  Y  N
```

Figure 10-14. Specifying ETT Criteria TM402.0

Notes:

This example shows jobname and special resource triggers in the ETT table.

Instructor Notes:

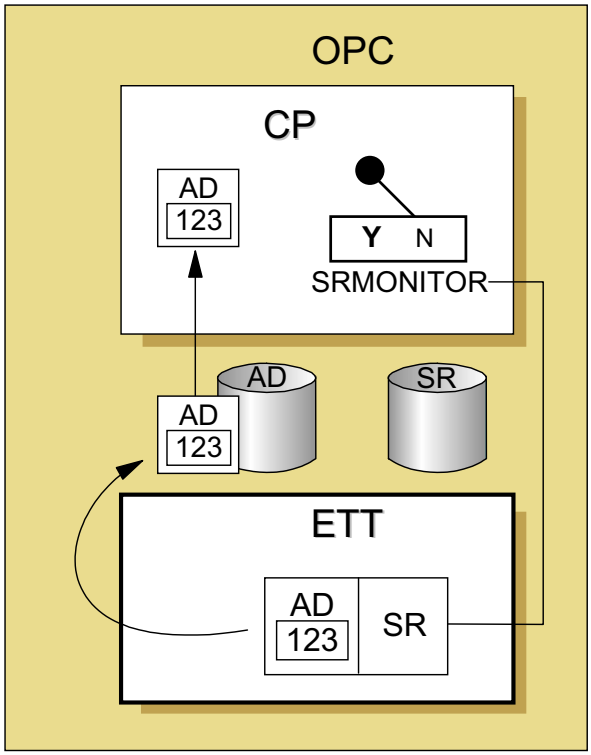
Purpose —Demonstrate defining ETT triggering criteria in the ETT database.

Details —If possible, perform an online demonstration.

Additional Information —

Transition Statement —Let us examine how the different triggers work.

Special Resource Event Triggers



Tivoli software

Figure 10-15. Special Resource Event Trigger TM402.0

Notes:

- When the status of a Special Resource is changed to available by the SRSTAT command ETT will check the ETT database. If it finds the SRSTAT name it will look for the associated Application Description in the AD database, and add it into the current plan.
- A resource event could, for example, set the status of 'IMS.MAIN.DBASE' to available. This could be the trigger that sets off one or more TWS for z/OS applications which do the backups, consolidations, orders and so forth. This SR status can be set from outside TWS for z/OS if necessary.
- Some example uses for ETT:
  - Start IMS housekeeping when IMS is closed.
  - Start work which needs datasets/files created by TSO or VM users.
  - Start work which depends on files being transmitted from non-TWS for z/OS systems - even from other countries.
  - Start work which depends on jobs submitted by application owners from TSO.

- Many events which take place outside TWS for z/OS can be used as triggers to add applications to the current plan. Any application that you want TWS for z/OS to add to the current plan must exist as an active application in the TWS for z/OS application database.

Instructor Notes:

Purpose —Describe how a special resource trigger works.

Details —Changing the status of the SR manually in the SR database or in the monitor does not generate a triggering event. The triggering event is generated only when the SR is made available by the SRSTAT command.

What ETT is really looking at is the event log, not the monitor or database. So what this means is if the same SRSTAT command (or the same named job if using jobname trigger) keeps being issued then multiple occurrences of the AD will be added. The AS field can be set to Y which will disregard SRSTAT if the current status is already available.

Additional Information —Box represents TWS for z/OS controlled area. When SRSTAT makes SR available ETT checks if it has a resource event with that name, if it does it adds the associated AD from the database into the CP.

Transition Statement —Let's now consider ETT by job name.

Jobname Event Trigger

- Triggering Criteria:
  - Explicit jobname
  - Generic jobname using * and/or %
- Jobname of the first operation in the added application occurrence can be replaced with the jobname of the triggering job

Figure 10-16. Jobname Event Trigger TM402.0

Notes:

- Using a jobname as the ETT trigger can cause an application to be added into the current plan.
- ETT compares jobnames with those specified as triggering events in the ETT database. If one matches it will load the associated application from the AD database into the current plan.
- The trigger name can be generic, for example, a jobname starting PAYR* could add application MASTERPAYROLL.
- You can choose to replace the first jobname in the added application with the jobname of the triggering job event to enable tracking of jobs submitted outside of TWS for z/OS.

Instructor Notes:

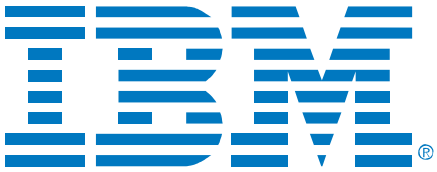
Purpose —Describe how a jobname trigger works.

Details —See the student notes.

Additional Information —

Transition Statement —In many installations, users want to add work to the CP after receiving a file. The next topic shows how to use another TWS for z/OS function to enhance the already powerful ETT function.

10.3 Data Set Triggering



Topic 10.3

Data Set Triggering

Tivoli software

Figure 10-17. Data Set Triggering TM402.0

Notes:

Instructor Notes:

Purpose —To introduce the concept of data set triggering.

Details —You can use data set triggering to start operations already in the CP or to add application to the CP. Data set triggering causes TWS for z/OS to issue the SRSTAT command whenever a data set is closed after read/update processing.

Additional Information —

Transition Statement —Why data set triggering?

Why Data Set Triggering?

- Automatically create a special resource availability event when a dataset is closed after update or read
- Usually used in conjunction with Event Triggered Tracking (ETT)

Tivoli software

Figure 10-18. Why Data Set Triggering? TM402.0

Notes:

- Data set triggering is a TWS for z/OS facility that you use to create special resource availability events when specific data sets are closed after read or update processing.
- Data set triggering can be used to further enhance the ETT function by using the special resource event as an ETT trigger.

Instructor Notes:

Purpose —Introduce data set triggering, define its function and explain the advantages derived from using it.

Details —See the student notes.

Additional Information —

Transition Statement —To take advantage of this function, you must create a list of the data sets that you wish to use as triggers.

Creating a Data Set Triggering Table

EQQLSENT

- STRING = *string/LASTENTRY*
- POS = *numeric position*
- USERID = *user ID filter criteria*
- JOBNAME = *jobname filter criteria*
- AINDIC = (*Y/N*)

Tivoli software

Figure 10-19. Creating a Data Set Triggering Table TM402.0

Notes:

- You specify the names of the data sets on which to trigger by creating entries in a data set triggering table.
- You create entries in a PDS member by invoking the EQQLSENT macro with parameters that specify the selection criteria for each data set to be placed in the data set triggering table
  - **STRING**: A required keyword specifying the character string to be searched for. The string can be 1 to 44 characters long.
  - **POS**: A required keyword specifying the numeric position within the data set name where the specified string begins.
  - **USERID**: An optional keyword specifying a character string, which is the user ID associated with the job, started task, or TSO user that requested the activity against the data set that resulted in the data set close. The string can be 1 to 8 characters long and can be generic.
  - **JOBNAME**: An optional keyword specifying a character string which is the name of the job, started task that requested the activity against the data set that resulted in

the data set close. The string can be 1 to 8 characters long and can contain wild card characters.

- **AINDIC**: An optional keyword specifying if the special resource should be set to available (Y) or unavailable (N). The default is to make the resource available.

- The last entry in the table must be defined as the last entry:

```
EQQLSENT STRING=LASTENTRY
```

- The entries are then assembled and loaded into a table named EQQDSLST.

Instructor Notes:

Purpose —Identify the data set table parameters and explain how to create entries in a data set triggering table.

Details —See the student notes.

Additional Information —

Transition Statement —On the next visual, you will see some example entries.

Data Set Triggering Table: Example Entries

```
EQQLSENT STRING=INGC100.DSTRI,POS=1,JOBNAME=EA73DTX1,USERID=INGC100
 EQQLSENT STRING=SYS2.OPC.OPCX.G,POS=1,USERID=INGC100
 EQQLSENT STRING=CASHWEST,POS=20,JOBNAME=PCP1D055

 EQQLSENT STRING=HAZPULL.ONLINE.TRANS,POS=20,JOBNAME=PCPID100

 EQQLSENT STRING=LASTENTRY
```

Figure 10-20. Data Set Triggering Table: Example Entries TM402.0

Notes:

- These are data set triggering table example entries.

Instructor Notes:

Purpose —Demonstrate how to create data set triggering table entries.

Details —Refer to the *TWS for z/OS Installation Guide*.

Additional Information —

Transition Statement —After creating and assembling the data set triggering table, you must load it.

Loading the Data Set Triggering Table

- F *subsystem_name*,NEWDSLST
  where *subsystem_name* is the Tracker subsystem name
- Example:

```
F  OPTE, NEWDSLST
```

Tivoli software

Figure 10-21. Loading the Data Set Triggering Table TM402.0

Notes:

- Use the z/OS operator modify command to dynamically load the assembled data set triggering table.
- The format of the command is:

  MODIFY *tracker_subsystem_name*,**NEWDSLST**

- Check the tracker message log to verify the table is successfully loaded.

Instructor Notes:

Purpose —Explain how to load/reload the data set triggering table.

Details —See the student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- Use the OPSTAT and SRSTAT commands to handle unplanned work and control workflow
- Use Event Triggered Tracking to handle unplanned work
- Create Data Set Triggering table entries
- Use Data Set Triggering to enhance Event Triggered Tracking

Tivoli software

Figure 10-22. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 11. Automated Job Tailoring

What This Unit is About

This unit introduces automated job tailoring, which enables jobs to be automatically edited using IBM TWS for z/OS JCL variables and directives.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List the main goal of automated job tailoring
- Use JCL automation directives to perform presubmission job tailoring
- Create and use IBM TWS for z/OS JCL variables

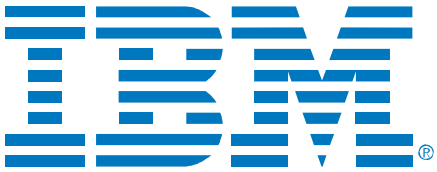
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 11: Automated Job Tailoring

Tivoli software

Figure 11-1. Automated Job Tailoring TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- List the main goal of automated job tailoring
- Use JCL automation directives to perform pre-submisson job tailoring
- Create and use TWS for z/OS JCL variables

Tivoli software

Figure 11-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

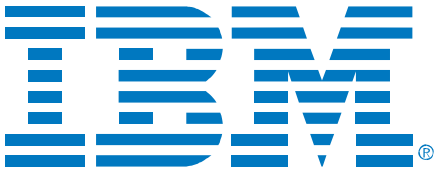
Purpose —

Details —

Additional Information —

Transition Statement —

11.1 What Is Automated Job Tailoring?



Topic: 1.1

What Is Automated Job Tailoring?

Tivoli software

Figure 11-3. What Is Automated Job Tailoring? TM402.0

Notes:

Instructor Notes:

Purpose —Identify and introduce TWS for z/OS automated job tailoring.

Details —This unit describes TWS for z/OS automated job tailoring, which enables jobs to be automatically edited using information that is known only at job setup or submit. To accomplish the task, TWS for z/OS uses special job statements called directives.

For jobs to be submitted on a z/OS system, these job statements will be z/OS JCL, but TWS for z/OS JCL tailoring directives can be included in jobs to be submitted on other supported operating systems. The directives have the same format for all supported operating systems.

Additional Information —

Transition Statement —What does automated job tailoring give you?

Automated Job Tailoring

Automatic presubmission editing which involves the use of some or all of these:

- Special in-line statements called directives
- Dynamic inclusion and exclusion of job statements
- TWS for z/OS-supplied or user-defined variables

Figure 11-4. Automated Job Tailoring TM402.0

Notes:

- TWS for z/OS provides automatic job tailoring facilities which enable the automatic editing of jobs using information that is known only at job setup or submit time.
- TWS for z/OS automated job tailoring uses:
  - Dynamic inclusion/exclusion of inline job statements
  - Dynamic inclusion of JCL from the TWS for z/OS job library or a user exit
  - Dynamic creation of temporary variables
  - Date arithmetic using OPC supplied variables
  - Variable substitution of user-defined or OPC supplied variables.
- Automated job tailoring can reduce your dependency on time-consuming and error-prone manual editing of jobs.
- User-defined variable details are held in tables stored in the database. The database is accessed from option (9 - JCLVAR) on the database panel.

Instructor Notes:

Purpose —List the TWS for z/OS job tailoring functions.

Details —See the student notes.

Additional Information —

Transition Statement —Let us see how job tailoring is accomplished.

JCL Automation Directives

- Directives are used to control:
  - Variable substitution
  - Inclusion of job statements
  - Exclusion of job statements
- Directives currently used are:
  - BEGIN
  - END
  - FETCH
  - SCAN
  - SEARCH
  - SETFORM
  - SETVAR
  - TABLE

Figure 11-5. JCL Automation Directives TM402.0

Notes:

- TWS for z/OS uses special comment statements, called *directives*, to manage the inclusion and exclusion of lines and to control aspects of variable substitution.
- Based on the directives used in a job, TWS for z/OS excludes lines in the job by skipping them at job setup or at job submit.
- Using directives, you can also include job statements from a member in the EQQJBLIB library or supply them through a user-defined JCL-embedded exit.
- These are the directives that are currently supported:
  - BEGIN and END: marks the start and end of:
    - JCL variable substitution
    - lines of job statements included or excluded
  - FETCH: names a PDS member containing job statements to be retrieved for inclusion.
  - SCAN: indicates the start of JCL substitution.

- SEARCH: defines the variable tables that are searched when attempting to resolve a variable.
- SETFORM: defines the format of supplied, dynamic-format date/time variables
- SETVAR: creates temporary variables using arithmetic expressions together with supplied date variables.
- TABLE: defines a variable table that will be searched before the variable tables in any existing concatenation when resolving JCL variables.

Instructor Notes:

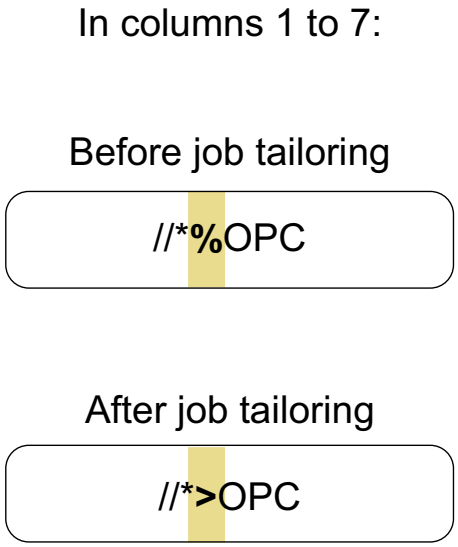
Purpose —Introduce JCL tailoring directives.

Details —See the student notes.

Additional Information —

Transition Statement —

Recognizing Job Tailoring Statements



Tivoli software

Figure 11-6. Recognizing Job Tailoring Statements TM402.0

Notes:

- The general syntax of a directives is:
  - Each directive must begin on a new 80-byte line.
  - All directives begin with **//*%OPC** in columns 1 to 7 followed by at least one space.
  - Directive parameters can be coded in any order.
  - Directive parameters can occur more than once in the same directive.
  - Directive parameters are separated by commas with no embedded blanks between parameters on the same line.
  - If more than one parameter value is specified, parentheses are required. For example, this is correct:

    ```
    NAME=TABLE1
    But this is incorrect:
    NAME=TABLE1,TABLE2
    It should be defined:
    NAME=(TABLE1,TABLE2)
    ```
  - A directive specification cannot exceed 71 characters. It can be continued on a new line if the directive is split by a comma after a complete parameter or sub parameter.

- Positions 72 to 80 are ignored.
- Each continuation line must begin with //*%OPC in columns 1 to 7 followed by a least one space.
- If the directive is executed successfully, the //*%OPC is changed to **//*>OPC.**

Instructor Notes:

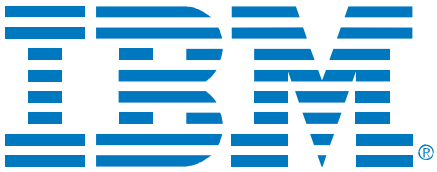
Purpose —Describe the syntax used in job tailoring statements.

Details —See the student notes.

Additional Information —

Transition Statement —

11.2 Introducing JCL Directives



Topic 1.2

Introducing JCL Directives

Tivoli software

Figure 11-7. Introducing JCL Directives TM402.0

Notes:

Instructor Notes:

Purpose —Cover some of the more basic JCL directives.

Details —In this topic, cover just the basic directives at a very high level.

Additional Information —

Transition Statement —The first set of directives that we examine can be used to dynamically include and exclude job statements.

Dynamic Inclusion and Exclusion

- Two methods available:

FETCH

BEGIN
job statement
END

Tivoli software

Figure 11-8. Dynamic Inclusion and Exclusions TM402.0

Notes:

- There are two sets of directives that you can use to include and exclude job statements when using TWS for z/OS automated job tailoring.
  - FETCH : This directive lets you include in your job, lines that TWS for z/OS fetches from a partitioned data set member or supplied by a JCL-embedded exit.
  - BEGIN and END : These directives are used in pairs. You use them with the ACTION keyword and a value to include or exclude the job statements within the lines marked by the directives.

Instructor Notes:

Purpose —Identify the two ways that can be used to include/exclude job statements.

Details —See the student notes.

Additional Information —

Transition Statement —The next visual discusses the Fetch directive.

The Fetch Directive

FETCH

- Conditional fetching of job statements from:
  - Job library in EQQJBLIB concatenation
  - Through a user exit

Example:

```
//*%OPC FETCH MEMBER=AUDTFILE,PHASE=SUBMIT,
//*%OPC    COMP=(&OADID..EQ.(SM4P#DLYW#OPC01))
```

Tivoli software

Figure 11-9. The FETCH Directive TM402.0

Notes:

- **Member**=*member_name* defines the PDS member that is retrieved from a data set in the TWS for z/OS job library concatenation, EQQJBLIB. The lines in the member are included immediately after the FETCH directive.
- If the JCL is being fetched via a JCL-embedded exit, the keyword EXIT is used and the exit is named.
- **PHASE=SUBMIT** specifies the FETCH should take effect during the submit phase of the operation. The alternative is at setup time which involves manual intervention.
- The FETCH directive is normally used with the keyword ***COMP***.
- **COMP**=(*expression1* .EQ. (*expression2*)) is a keyword that you use to test expressions to allow conditional invocation of directives. This example is comparing the name of the application occurrence with the name of a specified application.
  - If the application occurrence name is SM4P#DLYW#OPC01, then TWS for z/OS performs the fetch.
    Refer to *Planning and Scheduling the Workload* for more information about the COMP keyword.

Instructor Notes:

Purpose —Describe the fetch directive and how it is used.

Details —See the student notes.

Additional Information —

Transition Statement —Let us now look at the BEGIN and END pair.

Begin and End Directives

BEGIN
job statement
END

- Mark sections of in-line statements
- Conditional inclusion/exclusion marked job statements

Example:

```
//*%OPC BEGIN ACTION=INCLUDE,PHASE=SUBMIT,
//*%OPC       COMP=(&OWW..NE.(&CWW.))
//EQQTROUT DD DSN=%GDGDS.TRKLOG(+1),
//            DISP=(NEW,CATLG),SPACE=(CYL,(2,2),RLSE),
//            UNIT=SYSDA,
//            DCB=(LRECL=32000,RECFM=VBS)
//*%OPC END ACTION=INCLUDE
/*
```

Tivoli software

Figure 11-10. BEGIN and END Directives TM402.0

Notes:

- **BEGIN ACTION=INCLUDE** marks the start of a set of job statements you want TWS for z/OS to include in the job.
- **END ACTION=INCLUDE** marks the end of the set of job statements you want TWS for z/OS to include in the job.
- Unlike the FETCH directive, the job statements are not retrieved, they are already in the job.
- The BEGIN and END pair are typically used with the COMP keyword.
- BEGIN and END directives that specify ACTION=INCLUDE or ACTION=EXCLUDE cannot be nested and cannot overlap.

Instructor Notes:

Purpose —Describe the directives and how they are used.

Details —See the student notes.

Additional Information —

Transition Statement —Along with the job-statement, editing directives, date-manipulation directives help to enhance the flexibility of TWS for z/OS automated job tailoring. The next visual introduces two date-manipulation directives.

Date Manipulation Directives

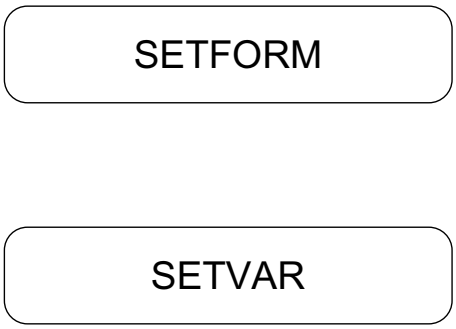


Figure 11-11. Date Manipulation Directives TM402.0

Notes:

- There are a fairly extensive number of TWS for z/OS-supplied, date-related variables. With the supplied date-related variables and two date manipulation directives, you can almost eliminate the need for manual intervention to set up dates or date-related job requirements.
- The two date manipulation directives are:
  - SETFORM which defines the format of dynamic-format supplied variables.
  - SETVAR which creates a temporary variable using an arithmetic expression together with supplied date variables.

Instructor Notes:

Purpose —Introduce the directives.

Details —See the student notes.

Additional Information —

Transition Statement —Let's look at the SETFORM directive.

The SETFORM Directive

SETFORM

- Dynamically formats dates to a user-specified form

Example:

```
//* This example demonstrates using the SETFORM directive
//* to dynamaically change the printed format of dates and times.
//*
//*%OPC SETFORM OCDATE=(MM:DD:YY)
//* The IA date for this occurrence is: &OCDATE.
//*

RESULT
//* The IA date for this occurrence is: 05:07:02
```

Tivoli software

Figure 11-12. The SETFORM Directive TM402.0

Notes:

- **SETFORM OCDATE=(MM:DD:YY)**
  - **OCDATE** is a dynamic-format supplied variable which when substituted provides the occurrence input arrival date.
  - **(MM:DD:YY)** is the user-specified format, using date-related keywords, in which the occurrence input arrival date is displayed.
- The set form directive format expression can contain a combination of time-related keywords, date-related keywords, and delimiters.
- The date-related keywords are:
  - **CC** Represents the century. This is used in combination with YY to define the format of a full year, such as 2002.
  - **YY** Represents the last two figures in the year.
  - **MM** Represents the month.
  - **DDD** Represents day-in-year. This is substituted before DD
  - **DD** Represents the day in the month.
- The time-related keywords are:
  - **HH** Represents the hour.

  - **MM** Represents the minutes.
- Any other characters in the format expression are regarded as delimiters.

Instructor Notes:

Purpose —Describe the SETFORM directive and how it is used.

Details —See the student notes.

Additional Information —

Transition Statement —Now, let's look at the SETVAR directive and how it is used.

The SETVAR Directive

SETVAR

- Dynamically creates a temporary a date-related variable
- The date-related variable contains the result of an arithmetic date-calulation.

Example:

```
//* This example demonstrates date arithmetic
//*
//* The IA date for this occurrence is: &OYMD2.
//* Then on checks to be printed set the date to 2 work days
//* from the IA date.
//*%OPC SETVAR TVAR=(OYMD2+2WD)
//* The date on the checks will be: &TVAR.

RESULT
//* The IA date for this occurrence is: 02/05/07
//* Then on checks to be printed set the date to 2 work days
//* from the IA date.
//*>OPC SETVAR TVAR=(OYMD2+2WD)
//* The date on the checks will be: 02/05/09
```

Tivoli software

Figure 11-13. The SETVAR Directive TM402.0

Notes:

- **SETVAR TVAR=(OYMD2+2WD)**
- **TVAR** is the name of the temporary variable you are creating in the job. The name must begin with a **T**. You can use temporary variables date-related values only. Temporary variables disappear when the job ends.
- **OYMD2** is a supplied date variable which resolves the occurrence input arrival date in the format, YY/MM/DD.
- **2WD** specifies two workdays.
- The expression **OYMD2+2WD** tells TWS for z/OS to add 2 work days to the occurrence input arrival date.
- Two arithmetic operators are supported in the SETVAR directive. They are the addition (+) and subtraction (-).
- Date arithmetic can be done on:
  - WD—work days, as defined for the calendar used by the occurrence.
  - CD—calendar days.
  - WK—weeks. Weeks are converted to days before the arithmetic is performed.
  - MO—months.

- YR—years.

Instructor Notes:

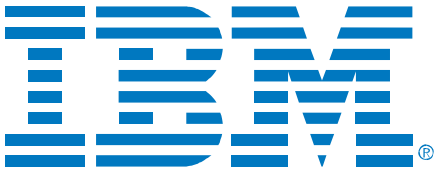
Purpose —Describe the SETVAR directive and how it is used.

Details —See the student notes.

Additional Information —

Transition Statement —

11.3 Introducing JCL Variables



Topic 11.3

Introducing JCL Variables

Tivoli software

Figure 11-14. Introducing JCL Variables TM402.0

Notes:

Instructor Notes:

Purpose —Topic introduces JCL variables and keeps discussion at a basic level.

Details —Refer to the *Planning and Scheduling the Workload Manual.*

Additional Information —

Transition Statement —

JCL Variables

- Can be used in jobs running on any operating platform
- Automatic substitution performed before job-submission
- Can be:
  - User-created
  - TWS for z/OS-supplied (built-in)
- User-created variables are stored in tables in the Variables database
- Supports:
  - Interdependence of values - Dependent variables
  - Validation of user-supplied values - Promptable variables

Tivoli software

Figure 11-15. JCL Variables? TM402.0

Notes:

- Variables can be used in job statements for any operating platform directly connected to the controller. This includes z/OS systems and supported tracker agents.
- TWS for z/OS does not support variable substitution for jobs running on fault-tolerant agents in the distributed environment.
- For jobs running on z/OS systems, you can use JCL variables in:
  - Executable job statements
  - Comment statements
  - Instream data
  - Instream procedures
- You can test and verify variable substitution at job setup by using a user-tailored program interface (PIF) program.
- Normally substitution takes place at jobsubmit time as defined by the user. This is the default.
- User-defined variables are created and saved in the TWS for z/OS variables database.

- For jobs that require manual setup because they use information that is generally available just before job-submission, you can create promptable and or dependent variables. These types of variables can reduce manual data- entry errors.

Instructor Notes:

Purpose —Describe TWS for z/OS JCL variables, where, and how they are used.

Details —Keep it simple. See the student notes.

Additional Information —

Transition Statement —

Using Variables in a Job

- Specify the SCAN directive before the first variable in the job
- A JCL variable can begin with:
  - & - ampersand
  - % - percent
  - ? - question mark
- A JCL variable can be terminated with:
  - A period
  - A blank
  - One of 13 other non-alphabetic characters

Tivoli software

Figure 11-16. Using Variables in a Job TM402.0

Notes:

- Use the SCAN directive before specifying any variables in the job.
- You can prevent variable substitution in a job by using the NOSCAN value for the ACTION keyword in a BEGIN and END directives pair.
- You prefix variables with one of there symbols:
  - **&** (ampersand)
  - **%** (percent sign)
  - **?** (question mark)
- The symbol determines how TWS for z/OS carries out the variable substitution.
- You denote the end of a variable with a period, a blank or one of 13 other symbols

Instructor Notes:

Purpose —Identify the characters that can be used to signify a variable name when it is used in a job.

Details —See the student notes. and use the Planning and Scheduling the Workload manual as a reference source.

Additional Information —

Transition Statement —Each of the three JCL variable prefix characters perform in a specific manner. The next three visuals will shed more light on the subject.

Types of Variables

- Simple
  - One to one substitution
- Compound
  - Variables concatenated to each other
  - Forms new variables

Figure 11-17. Types of Variables TM402.0

Notes:

- Simple -- These are variables you use without concatenation.
- Compound -- These are two or more simple variables that are concatenated to dynamically create new variables.

Instructor Notes:

Purpose —Introduce the different types of variables and how to use them.

Details —See the student notes.

Additional Information —

Transition Statement —The next visual shows how you can dynamically create variable names when you use percent (%) sign compound variables.

Using Compound Variables

Percent (%) variables are substituted right to left

```
//STEPLIB  DD  DSN=MY.%DATA%SET
```

on first pass

```
//STEPLIB  DD  DSN=MY.%DATALIB
```

on second pass

```
//STEPLIB  DD  DSN=MY.MAINFILE
```

Example Variable Table:

Variable Name	Default Value
SET	LIB
DATALIB	MAINFILE

Tivoli software

Figure 11-18. Using Compound Variables TM402.0

Notes:

- This example shows two simple variables defined in the variable table.
- The variables used in the job statement are prefixed with the percent sign symbol. This means that the two variables that are concatenated to form the compound variable will be resolved starting with the rightmost variable.
  - The % symbol indicates a right to left substitution sequence
- Since two variables form the compound variable, TWS for z/OS will resolve them in two passes.
- %SET is resolved in the first pass, and the value is placed in the column where the variable was specified. This has now created a new variable %DATALIB which TWS for z/OS resolves on the second pass.

Instructor Notes:

Purpose —Describe how concatenated variables are substituted.

Details —See the student notes.

Additional Information —

Transition Statement —Now, let us look at variables prefixed with the question mark (?).

Using Tabular Variables

- Question mark (?) variables are Tabular
- User can specify a column into which the default value is placed
- Specification can be placed in:
  - The variable definition in the database OR
  - In the job when the variable is used
- Usage:
  - ?VARNAME - uses column specified in database
  - ?nnVARNAME - uses column number specified by nn

Tivoli software

Figure 11-19. Using Tabular Variables TM402.0

Notes:

- Question mark variables are *tabular*; that is, you can specify in which column on the line the variable value should begin when the variable is substituted.
- The position at which the value is placed can be defined in the TWS for z/OS database when the variable is defined, or can be specified in the job where the variable is used.
- You will probably use these variables primarily within in-stream data, although they can be used anywhere within your job.

Instructor Notes:

Purpose —Introduce tabular variables.

Details —See the student notes. and demonstrate on the board.

Additional Information —

Transition Statement —Occasionally, required processing information must be specified by an operator before the job is submitted by TWS for z/OS. You can use variable validation to verify the operator's input.

Promptable Variables - Validation Criteria

- Values supplied by users prior to job-submission
- User can view a customized prompt which requests input of a valid value
- The user's input can be validated based on one of these criteria:
  - ALPHA
  - NUM
  - ENUM
  - HEX
  - BIT
  - PICT
  - NAME
  - DSNAME
  - RANGE
  - LIST

Tivoli software

Figure 11-20. Promptable Variables - Validation Criteria TM402.0

Notes:

- In cases where values are known only just before job-submission, you may want to use promptable variables to validate operator entries.
- TWS can check input entered by the operator at job setup or by a substitution exit.
- Input can be validated based on criteria you specify.
- Refer to *Planning and Scheduling the Workload* for details on the criteria.

Instructor Notes:

Purpose —To introduce promptable variables and describe the validation criteria.

Details —See the student notes.

Additional Information —

Transition Statement —Where can you specify the names of variable tables that TWS for z/OS can search to resolve variables used in a job?

User-Created Variable Tables

- Variable tables can be specified in:
  - Period definitons
  - Run cycles in applicatons
  - The Modify Current Plan dialog
  - The Modify Long-term Plan dialog
  - Jobs
  - TWS for z/OS initialization statement

Figure 11-21. User-created Variable Tables TM402.0

Notes:

- You can control where TWS for z/OS looks to find a value for the variables that you use in a job.
- You can specify the name of your variable table in:
  - Period definitions
  - Run cycles in applications
  - The Modify Current Plan dialog
  - The Modify Long-Term Plan dialog
  - Jobs using the TABLE and SEARCH directives
  - TWS for z/OS initialization statement (global variable table).

Instructor Notes:

Purpose —Identify the places where variable table names can be specified.

Details —See the student notes.

Additional Information —

Transition Statement —Because variable tables can be specified in all these places, there is an order of precedence that TWS for z/OS uses when searching for a variable.

Scanning Order of Precedence for Variable Resolution

Variable tables are scanned in this order until the end or the variable is found:

- Job-specific tables
  1. Name in the TABLE directive
  2. Names in the SEARCH directive
- Application-related tables
  3. Name on the MCP dialog
  4. Name on the modify LTP dialog
  5. Name on the run cycle
  6. Name associated with the period definition
- Global table
  7. Value specified in the JTOPTS GTABLE initialization statement

Figure 11-22. Scanning Order of Precedence for Variable Resolution TM402.0

Notes:

- Variable tables can be referred to in a number of places. It may well be that a job has references to more than one table, in directives in the job, attached to the run cycle used to schedule the application, and attached to the period used by the run cycle.
- This visual shows the order that the tables will be examined for the variable. If the variable is not in the first table, or if that table is not found, then the next table in order of precedence will be examined.
- If a value for the variable cannot be found then substitution does not take place, and the job fails without **any** of the variables substituted.
- The error code for failed variable substitution is **OJCV**.

Instructor Notes:

Purpose —Define the search order.

Details —See the student notes.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- List the main goal of automated job tailoring
- Use JCL automation directives to perform pre-submisson job tailoring
- Create and use TWS for z/OS JCL variables

Tivoli software

Figure 11-23. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit 12. Automatic Recovery

What This Unit is About

This unit introduces the IBM TWS for z/OS automatic recovery function and how to code the control statements.

What You Should Be Able to Do

After completing this unit, you should be able to:

- List the types of automatic recovery actions
- Code automatic recovery statements

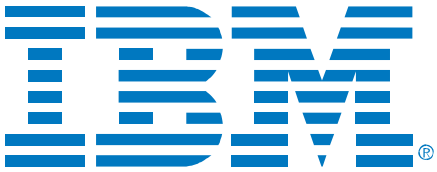
How You Will Check Your Progress

Accountability:

- Machine exercises

References

SC32-1263 *IBM Tivoli Workload Scheduler for z/OS Managing the Workload Version 8.2*



Unit 12: Automatic Recovery

Tivoli software

Figure 12-1. Automatic Recovery TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Unit Objectives

After completing this unit, you should be able to:

- List the types of automatic recovery actions
- Code automatic recovery statements

Tivoli software

Figure 12-2. Unit Objectives TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

TWS for z/OS Auto Recovery Function

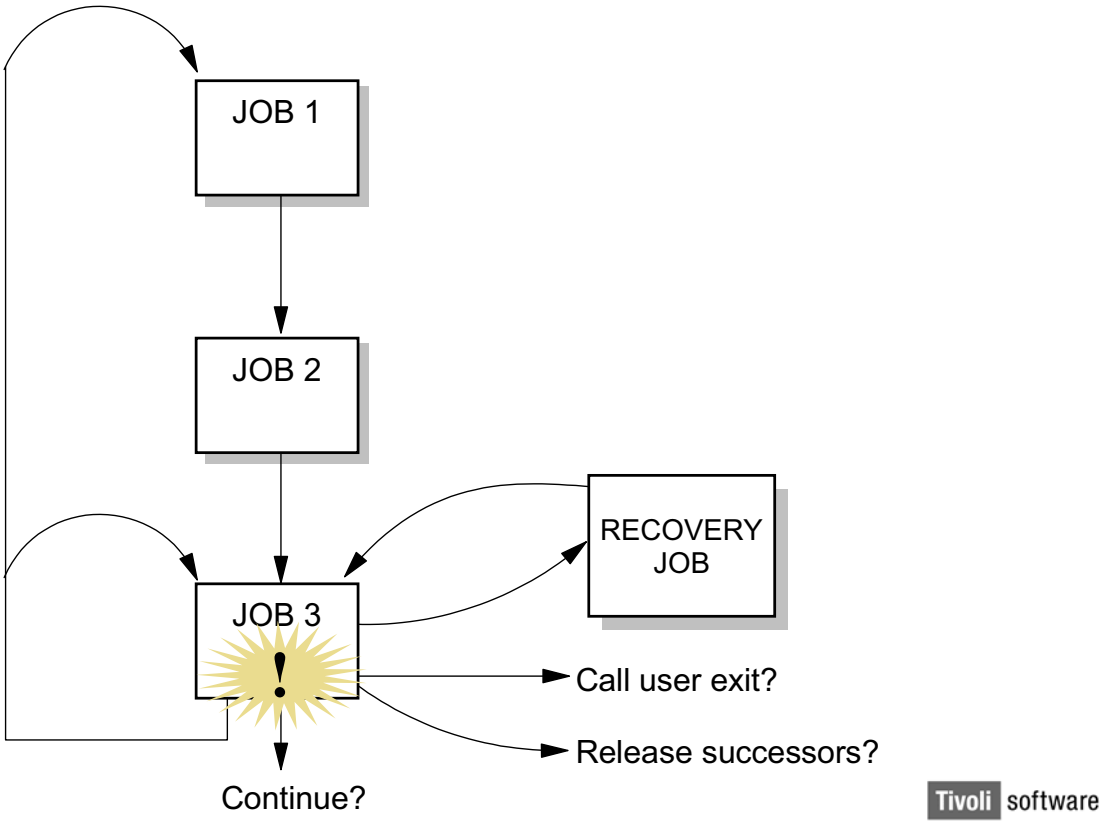


Figure 12-3. TWS for z/OS Auto Recovery Function TM402.0

Notes:

- Tivoli Workload Scheduler for z/OS supports the automatic recovery of failed jobs or started tasks.
- You specify the recovery criteria as part of the JCL for the operation in the form of special control statements.
- If activated, the automatic recovery function in TWS for z/OS will attempt to take the appropriate actions when a job fails.
- This feature of TWS for z/OS is provided to allow some automation of recovery without manual intervention.
- If a failing job operation has a cleanup action defined and it contains automatic recovery statements, cleanup action is performed before automatic recovery. For automatic recovery, you can use only the immediate or none cleanup types.
- The Automatic Recovery function can be switched on or off via the Service Functions panel.

Instructor Notes:

Purpose —Introduces AR concept.

Details —This foil is to talk through, describing (briefly) some of the possibilities. See the student notes.

Additional Information —

Transition Statement —

Automatic Job Recovery

- Start/Stop via Dialog
- Recovery Actions
- Selection Criteria
- Control Statements in JCL

Figure 12-4. Automatic Job Recovery TM402.0

Notes:

- You can set up automatic recovery procedures for many predictable failures by using OPC RECOVER control statements in your JCL.
- Automatic Recovery actions are defined in statements.
- The basic contents of the recovery statements in the JCL are:
  - Recovery actions to be performed
  - Criteria controlling which actions are to be taken.

Instructor Notes:

Purpose —Explain AR works through statements that contain the failure criteria and recovery actions.

Details —I take the points in reverse order, first state that AR works by means of statements in the JCL. These contain the failures and what is to be done if they occur.

Additional Information —

Transition Statement —Here are the possible actions that you can specify in your recovery statement.

Recovery Actions

- Restart at failed operation
- Restart at another operation *
- Add another application into CP
- Release a successor dependency
- Restart at failed or another step *
- Leave in error

* *With or without JCL changes*

Tivoli software

Figure 12-5. Recovery Actions TM402.0

Notes:

- In the auto recovery statements in your JCL, you specify recovery actions for the possible failures.
- The JCL can contain multiple recovery actions for different or recursive failures.
- Each recovery statement can be used only once.
- Recovery actions can include JCL and job execution modifications prior to restarting. Some of these actions are:
  - Step deletions
  - Inclusion of procedures
  - Restarting at a step other than the first step
- The recovery parameters are:
  - **RESTART** Specifies whether the occurrence should be restarted.
  - **RESJOB** Specifies the name of the job or started task from which the occurrence should be rerun.

- **ADDAPPL** Specifies an application or a list of applications that should be added as occurrences in the current plan.
- **RELSUCC** Specifies the application ID of a successor occurrence, or a list of IDs.
- **ALTWS** Specifies the name of an alternate workstation that the operation should be run on.
- **ALTJOB** Specifies an alternate job or started-task name to use when the job is restarted. This is used in a MAS complex to allow the restart of a job that has not yet ended as far as JES is concerned.

Instructor Notes:

Purpose —List the possible recovery actions.

Details —See the student notes.

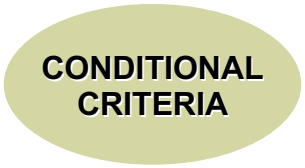
Additional Information —

Transition Statement —Let us see what goes into a recovery statement.

Recovery Statements (1 of 2)

Can be Conditional

May contain:



- ERRSTEP (Step in error)
- JOBCODE (Error code for job)
- STEPCODE (Error code for step)
- TIME (Time selection parameters)

Tivoli software

Figure 12-6. Recovery Statements (1 of 2) TM402.0

Notes:

Conditional Criteria

- The selection parameters in the recovery statement are: (all are optional)
  - ERRSTEP - restricts recovery actions only to the steps listed
  - JOBCODE - restricts recovery to be valid only for the completion codes and/or return codes specified
  - STEPCODE - only the step return codes listed will be valid
  - TIME - restricts recovery actions to specific times
- You can put several combinations of selection criteria in the OPC Recover statement. For example: To take different actions at different times of day.

Instructor Notes:

Purpose —Describe the selection parameters that can be used in recovery statements.

Details —I talk these through, there is not too much to say on them.

Additional Information —

Transition Statement —Having defined the possible failure, next we can code what we want to do about it.

Recovery Statements (2 of 2)

- DELSTEP (Step to delete)
- RESSTEP (Step at which to restart)
- ADDPROC (Call a procedure)
- CALLEXIT (Program name)

- ADDAPPL (Add an application into CP)
- RESTART (YES or NO)
- RESJOB (Restart at jobname x)
- RELSUCC (Release successor applicatio

Tivoli software

Figure 12-7. Recovery Statements (2 of 2) TM402.0

Notes:

JCL Rebuild Options

- In the OPC Recover statement, you can include one or more of these optional parameters to rebuild the JCL before resubmission:
  - DELSTEP -Specifies a step or a list or range of steps that should be deleted from the inline JCL before rerunning the failed operation.
  - RESSTEP - Specifies the name of the job or started-task step at which the operation should be restarted.
  - ADDPROC -Specifies the name, or a list of names, of JCL procedure library members to be added to the inline JCL before rerunning the failed operation.
  - CALLEXIT -Specifies the name of a program exit module that should be called before the restart.

Recovery Action Options

- You can include one or more of these actions in your recovery statement:
  - ADDAPPL -Specifies an application or a list of applications that should be added as occurrences in the current plan.

- RESTART (Y or N) - Specifies whether the occurrence should be restarted.
- RESJOB -Specifies the name of the job or started task from which the occurrence should be rerun.
- RELSUCC - Specifies one or more occurrence application IDs to be released from the dependency. TWS for z/OS will then start the successor occurrences if all other requirements are satisfied.
- ALTWS - Specifies the name of an alternate workstation on which operation is to be run. This overrides the alternate workstation defined in the WS description.
- ALTJOB - Used to specify an alternate job or started-task name to use when the job is restarted. This is used in a MAS (Multi Access Spool) complex to allow restart of a job that has not yet ended, as far as JES is concerned.

Instructor Notes:

Purpose —Describe the selection parameters that can be used in recovery statements

Details —Again, talk through these and keep it simple.

Additional Information —

Transition Statement —

Unit Summary

Having completed this unit, you should be able to:

- List the types of automatic recovery actions
- Code automatic recovery statements

Tivoli software

Figure 12-8. Unit Summary TM402.0

Notes:

Instructor Notes:

Purpose —

Details —

Additional Information —

Transition Statement —

Appendix A. Status, Error, and Reason Codes

This appendix lists status, error, and operation reason codes.

The scheduler assigns a status code to every occurrence and every operation in the current plan. An error code is also assigned for any operation that ends in error. You can see the status of operations by using the Workstation Communication panel or the Query Current Plan panel.

The codes assigned by the scheduler are not just for documentation purposes. They report the real status of the operation and are used by several scheduler functions to make important decisions about the running of the operation.

Occurrence Status Codes

C	Complete
D	Deleted
E	An operation in the occurrence has ended-in-error
P	A pending predecessor exists for the occurrence
S	Started
U	Undecided (the status is not known)
W	No operations in the occurrence have started

Operation Status and Extended Status Codes

When the scheduler displays the status of an operation, it uses the format xy, where x is the status code and y, if present, is the extended status code.

Operation Status Codes

A	Arriving—the operation is ready for processing; no predecessors were defined
R	Ready for processing; all predecessors are complete
S	Started
C	Complete
D	Deleted
I	The operation is interrupted
*	Ready—at least one predecessor is defined on a non-reporting workstation; all predecessors are complete
E	The operation has ended-in-error
W	The operation is waiting for a predecessor to complete
U	Undecided—the operation status is not known.

Extended Status Codes

Together with the normal status codes, the scheduler maintains extended status codes that provide additional information about the status of operations. The extended status code is not always present.

The following extended status codes are valid, depending on the type and status of the operation:

Valid for all operations that have a status of arriving (A) or ready (* or R):	
X	Waiting for resource
H	A panel user has used the HOLD command on the operation
N	A panel user has used the NOP command on the operation

Valid for all operations that have a status of arriving (A), ready (* or R), started (S), or error (E):	
M	The status of the operation has been manually set by a panel user from the ready list.

Valid only for computer workstation operations that have a status of arrived (A) or ready (* or R)	
T	Waiting until a particular time
L	The operation is a late time-dependent operation with the suppress-if-late attribute
R	The operation has ended in error but was automatically reset (the completion code is defined in the installation options to be automatically reset)
E	An error occurred during job submission or release
D	Close down in progress
Blank	The scheduler is in the process of submitting this job. The scheduler is waiting for the availability of a parallel server or a critical resource, or the operation is not to be submitted automatically

Valid only for computer workstation operations that have a status of started (S):	
Q	The job has been added to the JES job queue. For fault-tolerant workstations, it is waiting for submission
S	The job or started task is executing
M	The status of the job or started task has been manually set to S
U	Submit in progress
Blank	The job has been successfully submitted but has not yet been reported as added to the JES job queue.

Valid only for computer workstation operations that have a status of ready (R) or error (E):	
A	The job is waiting for a manual cleanup action to be initiated or discarded by a panel user (the cleanup type is manual).
B	The job is waiting for a cleanup action to be started (the cleanup type is automatic or immediate).
C	A restart and cleanup process is in progress (data set cleanup or step restart, or both). The job is waiting for the process to be completed.

Error Codes

The scheduler assigns error codes to certain operations and to job and started task steps. These codes are used by the automatic job recovery function to decide a recovery action.

CAN	The job or started task was canceled by the operator or by a TSO user before execution.
CCUN	The completion code is unknown. The job or started task has ended, but no completion code is available.
JCCE	An error during JCC (job completion checker) processing prevented the JCC from determining an error code for the operation.
JCL	A JCL error was recognized after the job or started task began to execute, or a JCL error was recognized after syntax checking in the internal reader.
JCLI	A JCL error occurred immediately; that is, the error was detected before the job or started task began.
MCP	The operation was manually set to error in the MCP panel
OFxx	The system that the operation is defined on has gone offline. xx is the status and extended status of the failing operation. Operations that were running (status SS) have a step-code error status of OFFL.
OJCV	An error occurred during JCL-variable substitution when the job or started task was submitted, or the scheduler detected an error in the RECOVER statement during automatic recovery. Browse the JCL for the operation or the EQQMLOG data set to find more information about the failure.

OSEQ	A job or started task began to execute before all its predecessors have completed. For fault-tolerant workstations, the OSEQ code can indicate that a dependency on another operation or a special resource was added after the job started, but before the event reached the controller.
OSUB	A failure occurred when the scheduler attempted to submit a job or start a started task. In the case of a started task, it could be that the started task is a subsystem that is not started by JES, or the scheduler subsystem EQQSTC ddname is not allocated to a JES-defined procedure library. The operation should be marked as ended-in-error
OSUF	A failure occurred when the scheduler attempted to retrieve the JCL for a job or started task. Possible causes are: the job name in the application description does not match the one on the job card, or the job is missing JCL.
OSUP	A time operation is late, and the SUPPRESSACTION parameter of the JTOPTS initialization statement specified that the operation should be marked as ended-in-error.
OSxx	The system on which the operation is defined has failed. The WSFAILURE parameter on the JTOPTS initialization statement specifies that started operations should be marked as ended-in-error. xx is the status and extended status of the failing operation. Operations that were running (status SS) have a step-code error of OSYS.
PCAN	A print operation was canceled by the operator.
CLNA	A failure occurred when the scheduler attempted to complete the JCL tailoring during the restart and cleanup process.
CLNC	A failure occurred when the scheduler attempted to execute the data set cleanup during the restart and cleanup process.
CLNO	A failure occurred when the scheduler attempted to retrieve the historical job log data during the restart and cleanup process.
nnnn	Step return code
Sxxx	System abend code.
Uxxx	User abend code in hexadecimal notation. For example user abend 2750 is represented in the scheduler as UABE
xxxx	User-defined error code

Job Log Retrieval Status Codes

When the job log retrieval function is used, the scheduler maintains status information to report on the retrieval of the log. The following status codes are possible:

C	Completed—the controller has received the log.
E	Error. There was an error retrieving the log.
I	Initiated. The controller has sent a retrieval request to the tracker, but the tracker has not yet processed the request.
S	Started. The controller has sent a retrieval request to the tracker, and the tracker has started to retrieve the log
blank	The controller has not sent any retrieval request to the tracker.

Operation Reason Codes

If your ready list layout includes the RSNC field, you can see these operation reason codes.

Note that the codes are listed in hierarchical order. For example, if job submission failed, and job submission is deactivated, code D is obtained—not code F.

D	Job submission deactivated
C	Workstation is closed
P	All parallel servers in use
A	Automatic reset error condition
F	Job submission failed
J	No automatic job submission
L	Job is late
T	Start time not reached
1	Not enough free WS resource 1
2	Not enough free WS resource 2
H	Closedown in progress.
S	Waiting for special resource