Workload Automation
Version 8.6

Overview

SC32-1256-12

Workload Automation
Version 8.6

Overview

SC32-1256-12

Note
Before using this information and the product it supports, read the information in "Notices" on page 85.

This edition applies to version 8, release 6 of IBM Tivoli Workload Automation (program numbers 5698-A17, 5698-WSH, and 5698-WSE) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC32-1256-11.

Contents

Figures

About this publication

IBM® Tivoli® Workload Automation Overview describes the suite of Tivoli Workload Scheduler and its enterprise workload management functions. This publication provides introductory information about the following products. It does not provide detailed technical explanations about how they work.

- Tivoli Workload Scheduler
- Tivoli Workload Scheduler for Applications
- Tivoli Workload Scheduler for z/OS®
- Dynamic Workload Console

This publication describes:

- The structure of the product
- Where it fits in single-host and multiple-host systems
- Major functions
- How it works with other products

What is new in this release

For information about the new or changed functions in this release, see Chapter 1, "Summary of enhancements," on page 1.

For information about the APARs that this release addresses, see the Tivoli Workload Scheduler Download Document at http://www.ibm.com/support/docview.wss?rs=672&uid=swg24027501.

What is new in this publication

This section describes what has changed in this publication since Tivoli Workload Scheduler version 8.5.1 Fixpack 1.

The following changes were made:

- Chapter 1, "Summary of enhancements," on page 1 lists the product enhancements available in the latest version.
- The interoperability tables that document the compatibility among different Tivoli Workload Scheduler, Tivoli Workload Scheduler for z/OS, and Dynamic Workload Console versions have moved to the Tivoli Workload Scheduler Release notes.

Who should read this publication

This publication is intended for:

- Data processing (DP) operations managers and their technical advisors who are evaluating the product or planning their scheduling service
- Individuals who require general information for evaluating, installing, or using the product.

Publications

Full details of Tivoli Workload Automation publications can be found in *Tivoli Workload Automation: Publications*, . This document also contains information on the conventions used in the publications.

A glossary of terms used in the product can be found in *Tivoli Workload Automation: Glossary*, .

Both of these are in the Information Center as separate publications.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully. With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

For full information with respect to the Dynamic Workload Console, see the Accessibility Appendix in the *Tivoli Workload Scheduler: User's Guide and Reference*, SC32-1274.

Tivoli technical training

For Tivoli technical training information, refer to the following IBM Tivoli Education website:

http://www.ibm.com/software/tivoli/education

Support information

If you have a problem with your IBM software, you want to resolve it quickly. IBM provides the following ways for you to obtain the support you need:

Online
Go to the IBM Software Support site at http://www.ibm.com/software/support/probsub.html and follow the instructions.

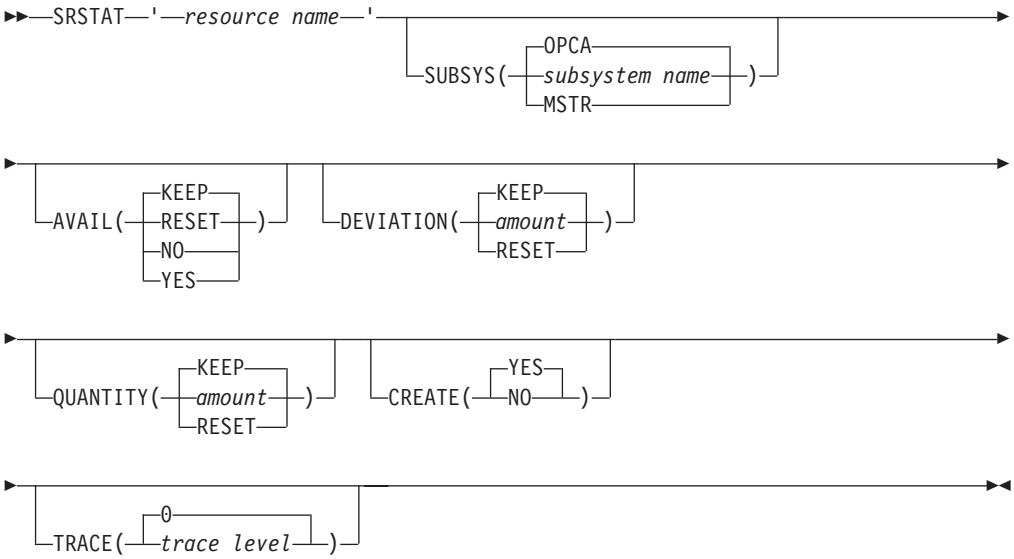
IBM Support Assistant
The IBM Support Assistant (ISA) is a free local software serviceability workbench that helps you resolve questions and problems with IBM software products. The ISA provides quick access to support-related information and serviceability tools for problem determination. To install the ISA software, go to http://www.ibm.com/software/support/isa.

Troubleshooting Guide
For more information about resolving problems, see the problem determination information for this product.

For more information about these three ways of resolving problems, see the appendix on support information in *Tivoli Workload Scheduler: Troubleshooting Guide*, SC32-1275.

How to read the syntax diagrams

Throughout this publication, syntax is described in diagrams like the one shown here, which describes the SRSTAT TSO command:



The symbols have these meanings:

The statement begins here.

The statement is continued on the next line.

The statement is continued from a previous line.

The statement ends here.

2 right arrow heads at beginning of line
The statement begins here.

1 right arrow head at end of line
The statement is continued on the next line.

1 right arrow head at beginning of line
The statement is continued from a previous line.

One right arrow head, one left arrow head at end of line
The statement ends here.

Read the syntax diagrams from left to right and from top to bottom, following the path of the line.

These are the conventions used in the diagrams:

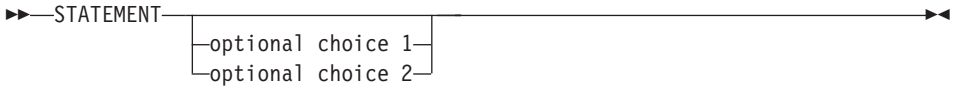
- Required items appear on the horizontal line (main path):

- Optional items appear below the main path:

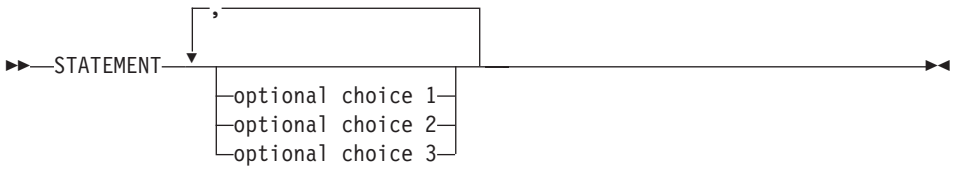
- An arrow returning to the left above the item indicates an item that you can repeat. If a separator is required between items, it is shown on the repeat arrow.

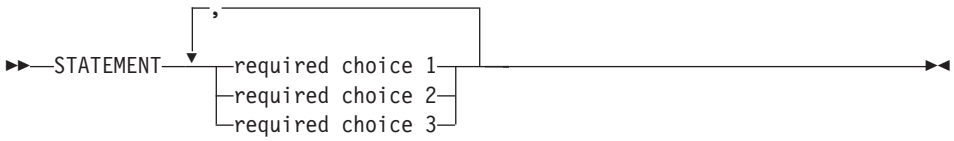
- If you can choose from two or more items, they appear vertically in a stack.
  - If you must choose one of the items, one item of the stack appears on the main path:

  - If choosing one of the items is optional, the entire stack appears below the main path:



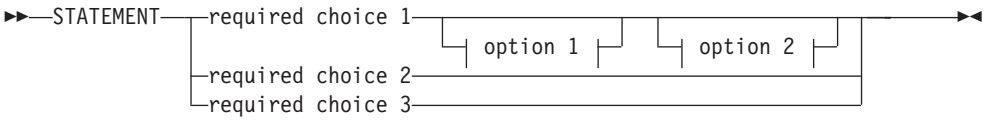
  - A repeat arrow above a stack indicates that you can make more than one choice from the stacked items:



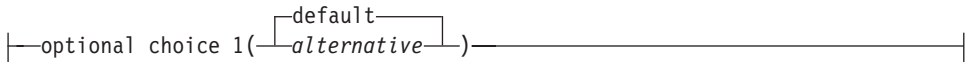


- Parameters that are above the main line are default parameters:

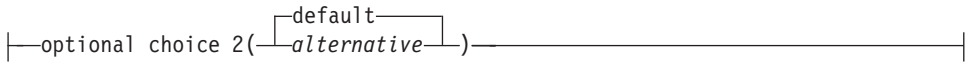
- Keywords appear in uppercase (for example, `STATEMENT`).
- Parentheses and commas must be entered as part of the command syntax as shown.
- For complex commands, the item attributes might not fit on one horizontal line. If that line cannot be split, the attributes appear at the bottom of the syntax diagram:



option 1



option 2



How to read syntax diagrams

Chapter 1. Summary of enhancements

The enhancements provided in Tivoli Workload Automation Version 8.6 add power to both schedulers and extend the scope of their functionality.

Important improvements such as cross dependencies, added dynamicity to the Tivoli Workload Scheduler for z/OS agent and the new Tivoli Workload Scheduler distributed - Agent for z/OS are proof that the declared intention to achieve ever increasing integration between the distributed and the host scheduler worlds is a reality.

The new workstation types, dynamic agents, pools, dynamic pools, and dynamic domain managers are dramatically changing the makeup of your scheduling environment.

The added capability to develop your own custom plug-ins (job types with advanced options) with the Tivoli Workload Scheduler Integration Workbench provide wider margins for your workload design.

An ever more performing Dynamic Workload Console provides an easy and accessible interface to run your work.

This chapter includes:

Tivoli Workload Scheduler for z/OS enhancements

This section describes the enhancements added with Tivoli Workload Scheduler for z/OS version 8.6:

The following features are no longer supported:

- Job Scheduling Console
- EJB job types

Dynamic capabilities added to Tivoli Workload Scheduler for z/OS agents

Dynamic capabilities have been added to Tivoli Workload Scheduler for z/OS agents so that now the scheduler automatically assigns your submitted workload to the workstations that best meet both the hardware and software requirements needed to run it. In this case, you install the Tivoli Workload Scheduler for z/OS agents on the distributed systems adding the dynamic capabilities, and connect them to the dynamic domain manager. Refer to *Tivoli Workload Scheduler: Planning and Installation Guide* for a detailed explanation on how to install a dynamic domain manager for a z/OS controller.

See also *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities*.

Defining and scheduling new and existing jobs with dynamic capabilities

While the standard Tivoli Workload Scheduler for z/OS job is a generic script or command, you can define and schedule jobs to perform specific tasks, such as database, file transfer, Java, and web services operations. To define and schedule these job types with advanced options, you do not need to have specific skills on the applications where the job runs.

The job types with advanced options include both those supplied with the product and the additional types implemented through the custom plug-ins.

You define job types with advanced options connecting to a z/OS engine with the Dynamic Workload Console, with the exception of the IBM i job type which you can create only with the JOBREC statement. From the Dynamic Workload Console, you open the Workload Designer and select the job type you want to create. When the job definition is saved, it is stored in the JCL library and is available for scheduling from Tivoli Workload Scheduler for z/OS.

The job types with advanced options run on Tivoli Workload Scheduler for z/OS agents in both the static configuration (Tivoli Workload Scheduler for z/OS agent connected directly to the z/OS controller) and in the dynamic configuration (Tivoli Workload Scheduler for z/OS agent connected to the dynamic domain manager).

For more information about the procedure for defining job types with advanced options, see the section about creating job types with advanced options in *Tivoli Workload Scheduler: Dynamic Workload Console User's Guide*. For more information about each job type, see the Dynamic Workload Console online help. For information about how to create jobs using the JOBREC statement, see the section about JOBREC in *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities*.

In addition to configuring job types with advanced options using the Dynamic Workload Console, you can use the related configuration files. For more information, see the section about configuring to schedule job types with advanced options in *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities*.

You can also create custom plug-ins to implement your own job types with advanced options for applications not supported by Tivoli Workload Scheduler for z/OS. For more information about how to create custom plug-ins, see *Tivoli Workload Automation: Developer's Guide: Extending Tivoli Workload Automation*.

The agents with dynamic capabilities can run the jobs you created for the existing Tivoli Workload Scheduler for z/OS workstation types. To run these jobs on the dynamic agents, change the specification of the workstation where you want the job to run. The major advantage is that you can use the workflows you previously created without additional effort.

Support for cross dependencies among jobs running on different scheduling engines

This feature enables you to integrate workload running on different engines, which can be a mix of Tivoli Workload Scheduler for z/OS engines (Controller) and Tivoli Workload Scheduler engines (Master Domain Manager and Backup Master Domain Manager).

A cross dependency is, from a logical point of view, a dependency of a local job on a job instance that is scheduled to run on a remote engine plan. To implement a cross dependency, you need to define the following objects:

Remote engine workstation
A new type of workstation that represents locally a remote Tivoli Workload Scheduler engine, either distributed or z/OS. This type of workstation uses a connection based on HTTP or HTTPS protocol to allow the local environment to communicate with the remote environment.

Remote job
A job scheduled to run on a remote Tivoli Workload Scheduler engine.

Shadow job
A job defined locally, on a remote engine workstation, which is used to map a remote job. The shadow job definition contains all the information necessary to correctly match, in the remote engine plan, the remote job instance.

Bind The process to associate a shadow job with a remote job instance scheduled in the remote Tivoli Workload Scheduler engine plan.

To define a cross dependency, you:

1. Create a shadow job that runs on a remote engine workstation defined in your local engine. The workstation points to the remote engine where the remote job is scheduled to run.
2. Define a normal dependency that makes your local job dependent on the shadow job.

See *Chapter 20. Defining and managing cross dependencies* in *Tivoli Workload Scheduler for z/OS: Managing the Workload* for reference.

Enhancements to ISPF panels

A new panel style can be applied to the AD application to enable you to list and browse a single AD, and also for the CP operation, to list and browse a single operation in the plan. The new panel style gives you a quick, at-a-glance scrollable view of the AD application descriptions and CP operations. A further enhancement related to color-coded fields enables you to quickly distinguish the status of

applications and operations. The color code can also be applied to distinguish applications from groups of applications. See *Chapter 7. Creating applications and groups -> Listing applications* in *Tivoli Workload Scheduler for z/OS: Managing the Workload*.

Send generated reports by email

Reports generated by running batch programs can optionally be sent to other users by email. Simply customize the file, `SENDREPORT.MAC` available in Tivoli Workload Scheduler for z/OS, to include the email address, the subject, a message, the name of the report, and the name of the output file generated. See *Appendix B. Batch programs -> Sending generated reports by email* in *Tivoli Workload Scheduler for z/OS: Managing the Workload*.

Automatic job log retrieval

Job logs are automatically retrieved when a job ends in error. In previous releases, job logs had to be retrieved manually when they were required. New parameters in the RCLOPTS and HTTPOPTS initialization statements can be configured to automatically retrieve the job log when a job ends in error. The automatic job log retrieval can be customized also for z-centric jobs that end in error. You can also configure to automatically retrieve the restart information for a restart and cleanup action. The new parameters are:

- `JOBLOGRETRIEVAL` and `RESTARTINFORETRIEVAL` in the `RCLOPTS` statement
- `JOBLOGRETRIEVAL` in the `HTTPOPTS` statement.

See *Tivoli Workload Scheduler for z/OS: Customization and Tuning*

Enhancements to Variable substitution

The variable substitution mechanism used in Tivoli Workload Scheduler for z/OS native job types works the same way when scheduling jobs on Tivoli Workload Scheduler for z/OS agents; however, the mechanism changes when you define job types with advanced options. With version 8.6, the support for new job types with advanced options that perform specific operations, such as Java jobs that run a Java class, file transfer jobs that transfer files to and from a server, and web service operations, to name a few. These new types of jobs are created and edited using the Dynamic Workload Console. The following parameters have been added to the `HTTPOPTS` statement that enables you to define a list of variable tables that must be searched:

- `VARTABLES`
- `VARFAIL`
- `VARSUB`

See *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

See also *Chapter 22. Job tailoring->Variable substitution for job types with advanced options* in *Tivoli Workload Scheduler for z/OS: Managing the Workload*.

Installing the Tivoli Workload Scheduler for z/OS connector on WebSphere Application Server for z/OS

You can now install the Tivoli Workload Scheduler for z/OS connector on IBM WebSphere Application Server for z/OS to maintain your business in a z/OS environment and simultaneously manage your workload using modern applications like Web Services.

See details in *Chapter 8. Installing and uninstalling on WebSphere Application Server for z/OS* of *Tivoli Workload Scheduler for z/OS: Planning and Installation Guide*.

Support for extended format VSAM data sets

Tivoli Workload Scheduler for z/OS Version 8.6 supports an extended format VSAM data set that can be allocated for JS data sets that exceed 4 GB. See the extended data sets included in the EQQPCS01 sample.

See also *Chapter 4. Installing->Step 9. Allocating data sets->Allocating the VSAM data sets* in *Tivoli Workload Scheduler for z/OS: Planning and Installation Guide*.

Keeping external dependencies on completed operations in the extended plan

The new `KEEPCOMPDEPS` parameter of the `BATCHOPT` initialization statement determines the permanence of external dependencies on a completed operation that belongs to occurrences that are still active when a daily plan job is submitted and either extended or replanned. This feature, if specified, facilitates a rerun operation because the external dependencies remain defined on the operation when the plan is extended and therefore there is no need to redefine them.

See the `BATCHOPT` statement in *Tivoli Workload Scheduler for z/OS: Customization and Tuning* for a description of `KEEPCOMPDEPS`. See also *Chapter 11. Producing the current plan->Extending the current plan* and *Chapter 26. Updating the current plan->Running work on request->Rerunning an occurrence in the current plan from a specific operation* in *Tivoli Workload Scheduler for z/OS: Managing the Workload*.

Enhancements for RACF user fields

For the `AD.UFVAL` and `CP.UFVAL` subresources that you specify in the `AUTHDEF` inizialization statement to protect user fields, support has been provided to protect user fields longer than 54 characters. Define a new RACF profile or use an existing profile and specify, for example, `MAXLNTH=80` to ensure that user field names and user field values up to 80 characters are supported.

See *Chapter 3. Implementing security->Functions and data that you can protect* in *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

Tivoli Workload Scheduler enhancements

This section describes the enhancements added with Tivoli Workload Scheduler version 8.6:

- "Using the Federated Repository security mechanism for authentication enhancements" on page 10
- "Upgrading when there are corrupt registry files" on page 10
- "Keeping you constantly and quickly informed" on page 10

The following features are no longer supported:

- Job Scheduling Console
- EJB job types
- Tivoli Data Warehouse integration

New Dynamic Domain Managers

This new domain manager type includes the dynamic workload broker component of previous versions. It enables you to more easily schedule and control your static and dynamic workload both in the distributed and end-to-end environments. A dynamic domain manager lets you run your dynamic schedule even if the master domain manager and the backup master domain manager are unavailable. It also improves fault-tolerant and dynamic agents scalability because the workload of the agents in the domain is directly controlled by the dynamic domain manager to which they are directly connected. If you want to ensure that your workload runs even if the connection to the dynamic domain manager is unavailable, install a backup dynamic domain manager.

A dynamic domain manager or backup dynamic domain manager include the following:

- Fault-tolerant agent
- Broker server
- Dynamic agent
- Plan Connector

When you install a dynamic domain manager or backup dynamic domain manager, the following workstation types are created in the Tivoli Workload Scheduler database:

Broker
: For the broker server

Agent For the dynamic agent

More information is available in *Chapter 4. Installing->Installing a dynamic domain manager or backup dynamic domain manager* in *Tivoli Workload Scheduler: Planning and Installation Guide*.

Defining and scheduling new and existing jobs with dynamic capabilities

You can use new workstation types with dynamic capabilities, named dynamic agents, pools, and dynamic pools, to run the jobs you created for the existing Tivoli Workload Scheduler workstation types. To run these jobs on the new workstation types, change the specification of the workstation where you want the job to run. The major advantage is that you can use the workflows you previously created without additional effort.

Also, while the existing Tivoli Workload Scheduler job is a generic script or command, you can now define and schedule jobs to perform specific tasks, such as

database, file transfer, Java, and web services operations. To define and schedule these job types with advanced options, you do not need to have specific skills on the applications where the job runs. In addition, you can have your developing team write the code for new custom job types with advanced options on the Tivoli Workload Scheduler Integration Workbench following a guided and documented procedure. See the Integration Workbench online help or the new *Tivoli Workload Automation: Developer's Guide: Extending Tivoli Workload Automation*.

To run these job types, you can use only the **dynamic agent**, a new workstation type, which you create simply by running the related installation process. The dynamic agent is automatically created and registered at installation time. You can also organize the dynamic agents in groups, called pools or dynamic pools.

A **pool** is a workstation that groups a set of dynamic agents with similar hardware or software characteristics to submit jobs to. Tivoli Workload Scheduler balances the jobs among the dynamic agents within the pool and automatically reassigns jobs to available workstations if a workstation is no longer available. To create a pool of dynamic agents in your environment, define a workstation of type pool hosted by the dynamic workload broker workstation, then select the dynamic agents you want to add to the pool.

A **dynamic pool** is a workstation that groups a set of dynamic agents, which is dynamically defined based on the resource requirements you specify. For example, if you require a workstation with low CPU usage and Windows installed to run your job, you specify these requirements using the Dynamic Workload Console or the **composer** command. When you save the set of requirements, a new workstation is automatically created in the Tivoli Workload Scheduler database. This workstation maps all the dynamic agents in your environment that meet the requirements you specified. The resulting pool is dynamically updated whenever a new suitable dynamic agent becomes available. Jobs scheduled on this workstation automatically inherit the requirements defined for the workstation.

If you want to use the dynamic capability when scheduling job types with advanced options, you schedule them on pools and dynamic pools, which dynamically assign the job to the best available resource. If you are interested only in defining job types with advanced options, without using the dynamic scheduling capability, you schedule these jobs on a specific dynamic agent, on which the job runs statically.

For more information, see the section about adding dynamic capabilities to your environment in *Tivoli Workload Scheduler: Scheduling Workload Dynamically*. For information about how to create pools and dynamic pools using the Dynamic Workload Console, see the section on creating a pool of agents in *Tivoli Workload Scheduler: Dynamic Workload Console User's Guide*. For more information about how to create job definitions, pools and dynamic pools using the **composer** command, see *Tivoli Workload Scheduler: User's Guide and Reference*.

Support for cross dependencies among jobs running on different scheduling engines

This feature enables you to integrate workload running on different engines, which can be a mix of Tivoli Workload Scheduler engines (Master Domain Manager and Backup Master Domain Manager) and Tivoli Workload Scheduler for z/OS engines (Controller).

A cross dependency is, from a logical point of view, a dependency of a local job on a job instance that is scheduled to run on a remote engine plan. To implement a cross dependency, you need to define the following objects:

Remote engine workstation
A new type of workstation that represents locally a remote Tivoli Workload Scheduler engine, either distributed or z/OS. This type of workstation uses a connection based on HTTP or HTTPS protocol to allow the local environment to communicate with the remote environment.

Remote job
A job scheduled to run on the remote Tivoli Workload Scheduler or Tivoli Workload Scheduler for z/OS engine.

Shadow job
A job defined locally, on a remote engine workstation, which is used to map a remote job. The shadow job definition contains all the information necessary to correctly match, in the remote engine plan, the remote job instance.

Bind The process to associate a shadow job with a remote job instance scheduled in the remote engine plan.

To define a cross dependency, you:

1. Create a shadow job that runs on a remote engine workstation defined in your local engine. The workstation points to the remote engine where the remote job is scheduled to run.
2. Define a normal dependency that makes your local job dependent on the shadow job.

See *Chapter 16. Defining and managing cross dependencies* in *Tivoli Workload Scheduler: User's Guide and Reference* for reference.

The Tivoli Workload Scheduler distributed - Agent for z/OS

The agent for z/OS provides the capability of running workload that is defined and submitted in Tivoli Workload Scheduler on a z/OS system. You run the definition and planning tasks on Tivoli Workload Scheduler, while the execution is demanded to the JES2 subsystem of z/OS.

The agent is installed on z/OS and connects via HTTP with the dynamic workload broker component of a master domain manager or of a dynamic domain manager. Upon connection, it is automatically defined as a workstation of type `agent` on Tivoli Workload Scheduler.

The agent acts as a proxy between Tivoli Workload Scheduler and JES2: it passes workload to JES2 and returns status events to Tivoli Workload Scheduler. You can use either the Dynamic Workload Console or the composer and conman command lines to define and submit jobs of type JCL from Tivoli Workload Scheduler and to track their execution. Jobs of type JCL include the JCL statements that will be run by JES2. The job execution events are also tracked in the message log (MLOG) of z/OS.

The agent is sold as a separate product named `Tivoli Workload Scheduler distributed - Agent for z/OS` and is targeted for Tivoli Workload Scheduler customers who also have a z/OS system. The agent for z/OS is not a replacement for Tivoli Workload Scheduler for z/OS as it lacks the sophisticated scheduling features of this scheduler.

New command to run batch reports from the command line interface

The new `reportcli` command enables you to schedule batch reports to run on a timely basis. See *Chapter 12. Getting reports and statistics->Running batch reports from the command line interface* in *Tivoli Workload Scheduler: User's Guide and Reference* for details.

Checking prerequisites

If you are preparing to install or upgrade on UNIX and Linux operating systems, Tivoli Workload Scheduler automatically runs a prerequisite check on your system. Having an environment that meets the Tivoli Workload Scheduler system requirements ensures that an installation succeeds without any delays or complications.

The prerequisite check verifies that:

- The operating system is supported for the product.
- The necessary engine software patch levels are installed.
- There is enough permanent and temporary disk space.
- There is enough memory and virtual memory swap space.
- The necessary kernel parameters are correctly configured.

See the chapter about checking prerequisites in *Tivoli Workload Scheduler: Planning and Installation Guide*.

Use of *twsinst* extended to Windows operating systems

You can use the `twsinst` script to install a Tivoli Workload Scheduler fault-tolerant or dynamic agent. You can use the script as an alternative to the silent installation wizard. In addition, if you are installing the dynamic agent, you can use the script to add to the agent the Java runtime necessary to run job types with advanced options. Previously, it was available on UNIX only. See *Chapter 4. Installing->Installing agents using twsinst* in *Tivoli Workload Scheduler: Planning and Installation Guide* for more information.

Creating and upgrading Tivoli Workload Scheduler database tables before installing or upgrading the product

Using this procedure, the database administrator, can create or upgrade the database tables before the IT administrator installs or upgrades the product with a user different from the database administrator user. In this way, only the database administrator manages all the confidential information related to the database such as the database administrator user ID and password. The IT administrator does not need to know this information when installing or upgrading the product.

The IT administrator can perform:

- The installation, specifying as database administrator user name the user to be granted access, by the administrator of the DB2 server, to the Tivoli Workload Scheduler database.
- The upgrade, by using another user that has the same permissions as the user that installed the product.

See the chapter about Creating and upgrading Tivoli Workload Scheduler database tables in *Tivoli Workload Scheduler: Planning and Installation Guide*.

Using the Federated Repository security mechanism for authentication enhancements

Versions of Tivoli Workload Scheduler from 8.6 onwards are configured for authentication (through the embedded WebSphere Application Server) in VMM (Virtual Member Manager) mode. This creates a Federated User Registry, which supports the contemporaneous use of more than one user registry. The user registry choices and LDAP server options remain much the same as prior to version 8.6.

See *Chapter 5. Upgrading->Upgrading a master domain manager or backup master domain manager instance->Upgrading overview->Updating authentication* in *Tivoli Workload Scheduler: Planning and Installation Guide* for more information.

Upgrading when there are corrupt registry files

It is now possible to upgrade a stand-alone fault-tolerant agent that has corrupt registry files without having to reinstall the product. Tivoli Workload Scheduler version 8.6 has a recovery option you can run to recreate the necessary files. You can also use this option when upgrading nodes in clusters, where the node on which you want to perform the upgrade is not available or is in an inconsistent state. The recovery option re-creates the registry files and the Software Distribution information without having to reinstall the complete product.

See *Chapter 5. Upgrading->Upgrading when there are corrupt registry files* in *Tivoli Workload Scheduler: Planning and Installation Guide* for more information.

Keeping you constantly and quickly informed

Keep constantly and quickly informed about product news, updates, technotes, APARs, and fixes using the "News and Updates" feature. To use this feature you must be connected to the Internet. See the chapter about the Launchpad in *Tivoli Workload Scheduler: Planning and Installation Guide*.

Dynamic Workload Console enhancements

This section describes the enhancements added with Dynamic Workload Console version 8.6:

Multiple engine query

A new task to monitor Jobs and Job streams running on multiple engines is available. It allows you to use a unique query that collects into one view monitoring data gathered from different scheduling environments, even mixed environments (z/OS and distributed). See *Chapter 8. Monitoring your Objects in the Plan->Monitoring your Workload->Creating a task to Monitor Job Streams on Multiple Engines* in *Tivoli Workload Scheduler: Dynamic Workload Console User's Guide* for details.

New look and feel for the dashboard

Dashboard performance has been improved together with its new look and feel that is integrated with Tivoli Integrated Portal. The dashboard can now also be accessed from your mobile device. See *Chapter 8. Monitoring your Objects in the Plan->Monitoring the progress of your plan* in *Tivoli Workload Scheduler: Dynamic Workload Console User's Guide* for more information.

Support for dynamic scheduling

New workstation types (Agent, Pool, Dynamic Pool) can be defined and monitored through the Dynamic Workload Console to support the enhanced dynamic scheduling capability.

Support for the new job types with advanced options

New job types with advanced options can be defined from the Workload Designer to perform specific tasks, such as database, file transfer, Java, and web service operations, as well as any custom job types created by your developing team with the Tivoli Workload Scheduler Integration Workbench.

Support for Cross Dependencies

A new type of job (Shadow job) and a new type of workstation (Remote Engine workstation) can be defined and monitored through the Dynamic Workload Console to allow the creation of dependencies between two different scheduling environments, including mixed z/OS and Distributed environments.

Enhanced management of user settings

Dynamic Workload Console settings, like user preferences, configured tasks and engine connections, can now be imported and exported to and from the currently-selected settings repository, which can be a File registry (by default) or a Database registry. User settings can be saved in an xml file that can be easily modified, and then imported into the same or another instance of Dynamic Workload Console.

See *Chapter 3. Getting Started->Managing user settings* in *Tivoli Workload Scheduler: Dynamic Workload Console User's Guide* for further information.

Additional usability improvements

The following usability improvements have been implemented:

- From the Monitor Task result tables, you can right-click on an item to open a context menu containing all the actions available for the selected object. If you do not want to specify an engine each time you run a task, you can save a preferred engine that is proposed as the default one for all the tasks associated to an "Ask when needed engine". The engine persistence is valid for the current session only.
- If there are jobs in the plan that have been rerun, now in the Monitor Jobs view you can show only the last occurrence of the rerun chain for each job.

Support for mobile device access

You can now use your mobile device to open the dashboard, see the jobs in plan, click them to view their details and job log, and also send this information using email. Support is provided for the following:

- IOS based devices - IPhone, IPod Touch, IPad

- Android-based devices version 2.2 or later

See *Chapter 6. Accessing the Console from Your Mobile Device* in *Tivoli Workload Scheduler: Dynamic Workload Console User's Guide* for reference.

Tivoli Workload Automation documentation enhancements

A set of Developer Guides have been added that are targeted to software developers and document methods and procedures for developing interfaces and plug-ins. They can be found in the IBM Tivoli Workload Automation V8.6 Information Center and they are:

Driving Tivoli Workload Automation
Describes two application programming interfaces which you can use to drive Tivoli Workload Automation products from your own applications:

- The Java application programming interface for creating your own GUI or command-line interface to perform all the functions of the command-line programs composer, conman, and planman and the Dynamic Workload Console for both Tivoli Workload Scheduler and Tivoli Workload Scheduler for z/OS.
- The web services interface to create your own web client application to perform a subset of Tivoli Workload Scheduler and Tivoli Workload Scheduler for z/OS functions to manage jobs and job streams in the plan.

You can use the Tivoli Workload Automation Software Development Kit Integration Workbench provided with the product to develop and implement these application programming interfaces.

Extending Tivoli Workload Automation
Explains how to extend Tivoli Workload Automation by creating plug-ins that add functionality relevant to your business activities in two main areas:

- Event-driven workload automation
- Job types with advanced options

In addition, it documents how to create Java jobs that implement a Java project of your creation on the target workstation.

Driving Tivoli Workload Scheduler for z/OS
Describes how to use the programming interfaces for Tivoli Workload Scheduler for z/OS to help you plan, schedule, and monitor work in the production department of your computer installation.

It comprises all the documentation previously released in *Tivoli Workload Scheduler for z/OS: Programming interfaces*.

Chapter 2. Overview of Tivoli Workload Automation

Tivoli Workload Automation is the state-of-the-art production workload manager, designed to help you meet your present and future data processing challenges. Its scope encompasses your entire enterprise information system, including heterogeneous environments.

Pressures on today's data processing (DP) environment are making it increasingly difficult to maintain the same level of service to customers. Many installations find that their batch window is shrinking. More critical jobs must be finished before the morning online work begins. Conversely, requirements for the integrated availability of online services during the traditional batch window put pressure on the resources available for processing the production workload. Increasing by 7 days a week, 24 hours a day is not only a DP objective but a requirement.

Users and owners of DP services are also making more use of batch services than ever before. The batch workload tends to increase each year at a rate slightly below the increase in the online workload. Combine this with the increase in data use by batch jobs, and the end result is a significant increase in the volume of work.

Furthermore, there is a shortage of people with the required skills to operate and manage increasingly complex DP environments. The complex interrelationships between production activities, between manual and machine tasks, have become unmanageable without a workload management tool.

Tivoli Workload Automation simplifies systems management across heterogeneous environments by integrating systems management functions. There are three main components to the portfolio:

- Tivoli Workload Scheduler for z/OS

  The scheduler in z/OS environments
- Tivoli Workload Scheduler

  The scheduler in distributed environments
- Dynamic Workload Console

  A Web-based, graphical user interface for both Tivoli Workload Scheduler for z/OS and Tivoli Workload Scheduler

The Job Scheduling Console has been replaced by Dynamic Workload Console as the Tivoli Workload Automation graphical user interface.

The state-of-the-art solution

The portfolio provides leading-edge solutions to problems in production workload management. It can automate, plan, and control the processing of your enterprise's entire production workload, not just the batch subset. The portfolio works as an “automatic driver” for your production workload to maximize the throughput of work, and optimize your resources, but also allows you to intervene manually as required.

When the portfolio interfaces with other system management products, it forms part of an integrated automation and systems management platform for your DP operation.

Comprehensive workload planning

The portfolio forms operating plans based on user descriptions of the operations department and its production workload. These plans provide the basis for your service level agreements and give you a picture of the production workload at any precise time.

Good planning is the cornerstone of any successful management technique. Effective planning also helps you maximize return on your investments in information technology.

Centralized systems management

The portfolio automates, monitors, and controls the flow of work through your enterprise's entire DP operation on both local and remote systems. From a single point of control, the portfolio analyzes the status of the production work and drives the processing of the workload according to installation business policies. It supports a multiple-end-user environment, enabling distributed processing and control across sites and departments within your enterprise.

Systems management integration

Solutions to today's systems management problems require an integration of application programs and processes. The portfolio offers you integration with the following:

- Agents for controlling the workload on non-z/OS platforms
- Other systems management applications and architecture environments.

The portfolio interfaces directly with some of the z/OS products as well as with a number of other IBM products to provide a comprehensive, automated processing facility and an integrated approach for the control of complex production workloads.

NetView®. NetView is the IBM platform for network management and automation. You can use the interface for Tivoli Workload Scheduler for z/OS with NetView to pass information about the work that is being processed. The portfolio lets you communicate with NetView in conjunction with the production workload processing. Tivoli Workload Scheduler for z/OS can also pass information to NetView for alert handling in response to situations that occur while processing the production workload. NetView can automatically trigger Tivoli Workload Scheduler for z/OS to perform actions in response to these situations using a variety of methods. Tivoli Workload Scheduler/NetView is a NetView application that gives network managers the ability to monitor and diagnose Tivoli Workload Scheduler networks from a NetView management node. It includes a set of submaps and symbols to view Tivoli Workload Scheduler networks topographically and determine the status of job scheduling activity and critical Tivoli Workload Scheduler processes on each workstation.

Workload Manager (WLM). WLM controls the amount of system resources available to each work unit in host environments. Tivoli Workload Scheduler for z/OS works in concert with WLM to detect critical jobs and move them to a higher-performance service class. In addition with WLM, critical jobs receive more system resources and complete more quickly.

Resource Object Data Manager (RODM). RODM provides a central location for storing, retrieving, and managing the operational resource information needed for

network and systems management. You can map a special resource to a RODM object. This lets you schedule the production workload considering actual resource availability, dynamically updated.

Tivoli Decision Support for z/OS (Decision Support). Decision Support helps you effectively manage the performance of your system by collecting performance data in a DATABASE 2 (DB2®) database and presenting the data in a variety of formats for use in systems management. Decision Support uses data from Tivoli Workload Scheduler for z/OS to produce summary and management reports about the production workload, both planned and actual results.

Output Manager for z/OS. Helps customers increase productivity and reduce the costs of printing by providing a means for storing and handling reports in a z/OS environment. When a dialog user requests to view a job log or to automatically rebuild the JCL for a step-level restart, Tivoli Workload Scheduler for z/OS interfaces with Output Manager. This interface removes the requirement to duplicate job log information, saving both CPU cycles and direct access storage device (DASD) space.

Tivoli Information Management for z/OS. Supports the administration of the systems management process of an enterprise's hardware, software, and related resources. An interface with Tivoli Information Management for z/OS is provided for reporting problems detected while processing the production workload.

Resource Access Control Facility (RACF®). RACF is the IBM product for data security. You can use RACF as the primary tool to protect your Tivoli Workload Scheduler for z/OS services and data at the level required by your enterprise. With RACF 2.1 and later, you can use a Tivoli Workload Scheduler for z/OS reserved resource class to protect your resources.

IBM Tivoli Monitoring (ITM). You can use it to monitor your hardware, operating systems, applications, databases. It provides proactive monitoring and automated fault management, and includes a specific module for Business Integration. You can also manage configuration and collect monitoring information for reporting, performance analysis, trend predictions and enterprise wide business impact assessment.

IBM Tivoli Service Request Manager (TSRM). It is an incident management system. TSRM can function as a service desk for both internal IT assets and internal corporate, non-IT enterprise assets, such as facilities or fleet. TSRM helps to improve IT performance by providing automation of processes, better visibility of service support functions, commitments, and measurements.

Tivoli System Automation for z/OS (SA z/OS). SA z/OS initiates automation procedures that perform operator functions to manage z/OS components, data sets, and subsystems. SA z/OS includes an automation feature for Tivoli Workload Scheduler for z/OS. You can define an automation workstation in Tivoli Workload Scheduler for z/OS to handle system automation operations with a specific set of options.

Data Facility Hierarchical Storage Manager (DFHSM). Tivoli Workload Scheduler for z/OS catalog management functions invoke DFHSM to recall migrated data sets during data set cleanup for a failed or rerun job.

CICS® and IMS™ (Computer Information Control System and Information Management System). Tivoli Workload Scheduler for z/OS lets you schedule the

starting and stopping of started tasks. Because Tivoli Workload Scheduler for z/OS tracks the status of started tasks, you can serialize work, such as backups of your transaction databases, according to the status of your CICS or IMS subsystems.

Tivoli Business Systems Manager. Tivoli Business Systems Manager provides monitoring and event management of resources, applications, and subsystems with the objective of providing continuous availability for the enterprise. Using Tivoli Business Systems Manager with the portfolio provides the ability to manage strategic applications from a unique business systems perspective. Tivoli Business Systems Manager monitors batch-related applications and operations represented by the portfolio and seamlessly integrates these objects with all other business objects monitored by Tivoli Business Systems Manager.

Tivoli Enterprise Console®. The Tivoli Enterprise Console is a powerful, rules-based event management application that integrates network, systems, database, and application management. It offers a centralized, global view of your computing enterprise while ensuring the high availability of your application and computing resources. Tivoli Enterprise Console acts as a central collection point for alarms and events from a variety of sources, including those from Tivoli applications. Tivoli Workload Scheduler runs a Tivoli Enterprise Console adapter that reads events from the Tivoli Workload Scheduler log file.

Besides these IBM products, there are many products from other software vendors that work with or process data from the portfolio.

For white papers about using IBM products, refer to the following link: http://www-03.ibm.com/support/techdocs/atsmastr.nsf/Web/WP-ByProduct?OpenDocument&Start=1&Count=1000&Expand=18.

An integration scenario

This section shows how you can monitor late *critical jobs* and perform *incident management*, by integrating Tivoli Workload Scheduler for z/OS with the following products:

- IBM Tivoli Monitoring (ITM)
- Maximo Tivoli Service Request Manager (TSRM)
- Tivoli System Automation (SA)

Tivoli Workload Scheduler for z/OS schedules jobs according to the defined current plan. ITM is configured with a *situation* that sends an email notification to Maximo TSRM, when a critical job is late.

The integrate components work as follows:

1. ITM detects a critical job tied to a WLM scheduling environment.
2. ITM sends an email causing the automatic opening of a service request.
3. The Maximo TSRM operator captures the service request and runs a launch in context of the Dynamic Workload Console, to perform incident analysis.
4. The analysis confirms that a critical job is waiting for a WLM scheduling environment.
5. A System Automation job is submitted through Dynamic Workload Console, to make available the WLM scheduling environment.
6. As soon as the WLM scheduling environment is available, the scheduler submits again the critical job.
7. The service request is closed.

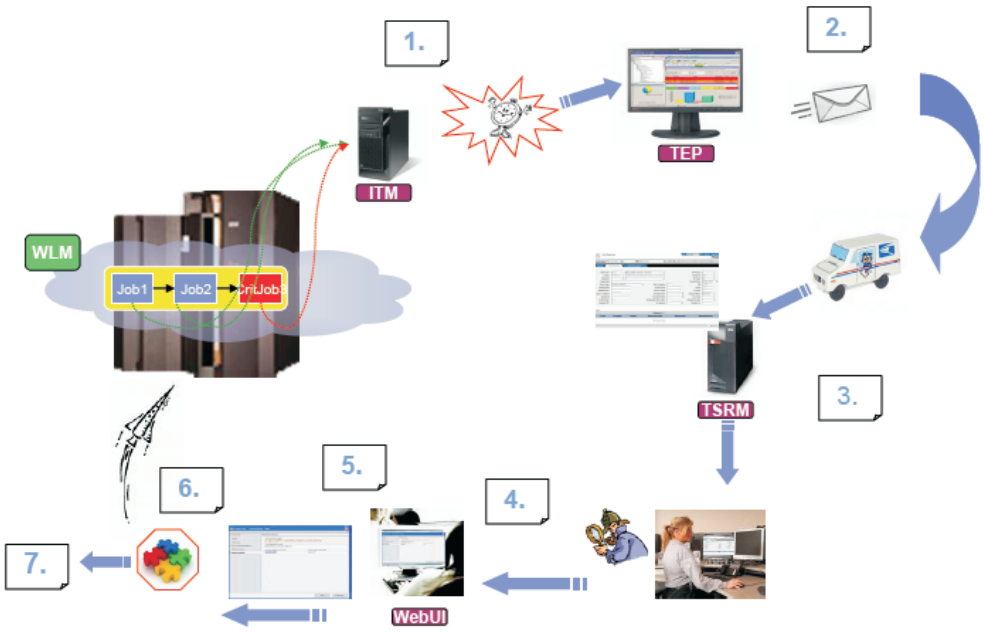


Figure 1. Integration scenario for Tivoli Workload Scheduler for z/OS.

Automation

By automating management of your production workload with the portfolio, you can minimize human errors in production workload processing and free your staff for more productive work. The portfolio helps you plan, drive, and control the processing of your production workload. These are important steps toward automation and unattended operations. Whether you are running one or more systems at a single site, or at several distributed sites, the portfolio helps you automate your production workload by:

- Coordinating all shifts and production work across installations of all sizes, from a single point of control
- Automating complex and repetitive operator tasks
- Dynamically modifying your production workload schedule in response to changes in the production environment (such as urgent jobs, changed priorities, or hardware failures) and then managing the workload accordingly
- Resolving workload dependencies
- Managing utilization of shared resources
- Tracking each unit of work
- Detecting unsuccessful processing
- Displaying status information and instructions to guide operations personnel in their work
- Interfacing with other key IBM products to provide an integrated automation platform

The portfolio lets you centralize and integrate control of your production workload and reduces the number of tasks that your staff need to perform.

Workload monitoring

Besides providing a single point of control for the production workload across your systems, the portfolio:

- Monitors the production workload in real time, providing operations staff with the latest information on the status of the workload so that they can react quickly when problems occur.
- Provides security interfaces that ensure the protection of your services and data.
- Enables manual intervention in the processing of work.
- Reports the current status of your production workload processing.
- Provides reports that can serve as the basis for documenting your service level agreements with users. Your customers can see when and how their work is to be processed.

Automatic workload recovery

The portfolio enables processing production workload to continue even when system or connection failures occur. If one system fails, the portfolio can restart the processing on another system. When the controlling system is running in a z/OS system complex (sysplex), a hot standby function can automatically transfer control of the production workload to another system in the sysplex. Because the portfolio continues to manage the production workload during failures, you can maintain the integrity of your processing schedules and continue to service your customers.

In Tivoli Workload Scheduler, a switchmgr function provides the possibility to replace a failing master domain manager or domain manager workstation with an appropriately configured backup fault-tolerant agent or domain manager .

Productivity

The portfolio represents real productivity gains by ensuring fast and accurate performance through automation. Many of today's automation solutions quote unrealistic productivity benefits. Some of the tasks automated should never be performed, or certainly not as often as they are by automation. Because of this, it is difficult to correlate real productivity benefits to your enterprise.

The tasks the portfolio performs not only have to be performed, but have to be performed correctly, every time, and as quickly as possible. Many of these tasks, traditionally performed by DP professionals, are tedious and as a result prone to error. With the portfolio, your DP staff can use their time more efficiently.

Business solutions

The portfolio provides business solutions by:

- Driving production according to your business objectives
- Automating the production workload to enhance company productivity
- Providing you with information about current and future workloads
- Managing a high number of activities efficiently.

User productivity

Your DP staff and end users can realize significant productivity gains through the portfolio's:

- Fast-path implementation.

- Immediate response to dialog requests for workload status inquiries. Users are provided with detailed real-time information about production workload processing so that they can detect and promptly correct errors.
- Automation of operator tasks such as error recovery and data set cleanup.

Growth incentive

As you implement automation and control you can manage greater production workload volumes. The portfolio brings growth within your DP operation by providing:

- Ways of absorbing the increasing batch workload without increasing operations personnel
- An open interface for submitting and tracking the workload on a variety of operating systems
- Interfaces with other systems management application programs
- An open interface for, and communicating with, programs on other platforms
- Management of current and future production workload volumes
- Simulation facilities to forecast future workloads

How Tivoli Workload Automation benefits your staff

In a typical enterprise, many people contribute to the implementation and operation of Tivoli Workload Automation:

- Scheduling manager
- Operations manager
- Shift supervisor
- Application programmer
- Console operators
- Workstation operators, such as print operators, job setup staff, and login receptionists
- End users
- Service desk

This section describes how the portfolio can directly benefit your DP staff.

Role of the scheduling manager as the focal point

Tivoli Workload Automation makes it possible for the scheduling manager to maintain current and future production processing across your enterprise. The portfolio benefits the scheduling manager in the following ways:

- Automatically scheduling all production workload activities.
- Automatically resolving the complexity of production workload dependencies and driving the work in the most efficient way.
- Supporting the simulation of future workloads on the system. The scheduling manager can evaluate, in advance, the effect of changes in production workload volumes or processing resources.
- Giving a real-time view of the status of work as it flows through the system so that the scheduling manager can quickly:
  - Respond to customer queries about the status of their work
  - Identify problems in the workload processing.
- Providing facilities for manual intervention.

- Managing many workload problems automatically. The production-workload-restart facilities, hot standby, automatic recovery of jobs and started tasks, and data set cleanup provide the scheduling manager with comprehensive error-management and disaster-management facilities.
- Providing a log of changes to the production workload data through the audit-trail facility. This assists the scheduling manager in resolving problems caused by user errors.
- Managing hard-to-plan work.

Role of the operations manager

The reporting, planning, and control functions can help the operations manager do the following:

- Improve the efficiency of the operation
- Improve control of service levels and quality
- Set service level agreements for end-user applications and for services provided
- Improve relationships with end-user departments
- Increase the return on your IT investment
- Develop staff potential.

A powerful tool for the shift supervisor

The portfolio is important for the shift supervisor, especially in multisystem complexes, where local and remote systems are controlled from a central site. The portfolio can help the shift supervisor do the following:

- Monitor and control the production workload through multisystem complexes
- Control the use of mountable devices
- Separate information about work status from system and other information
- Provide end users with status information directly
- Manage the workload if a system failure occurs
- Make changes to the current plan in response to unplanned events, such as equipment failures, personnel absences, and rush jobs.

Role of the application programmer

The user-authority checking functionality enables application development groups to use all the planning and control functions in parallel with, but in isolation from, production systems and services.

The portfolio can be a valuable tool for application development staff when they are doing the following:

- Packaging new applications for the production environment
- Testing new JCL in final packaged form
- Testing new applications and modifying existing ones

Console operators

The portfolio can free console operators from the following time-consuming tasks:

- Starting and stopping started tasks
- Preparing JCL before job submission
- Submitting jobs
- Verifying the sequence of work
- Reporting job status
- Performing data set cleanup in recovery and rerun situations
- Responding to workload failure
- Preparing the JCL for step-level restarts.

Workstation operators

The portfolio helps workstation operators do their work by providing the following:

- Complete and timely status information
- Up-to-date ready lists that prioritize the work flow
- Online assistance in operator instructions.

End users and the service desk

Your end users often need to be informed about the status of workload processing. They can use the Dynamic Workload Console to check the status of the processing of their job streams themselves from a personal workstation. End users can make queries using the Dynamic Workload Console without having to be familiar with the portfolio, ISPF, or TSO, and without having to be logged on to a local system.

The help desk can use the Dynamic Workload Console in the same way to answer queries from end users about the progress of their workload processing.

Summary

Tivoli Workload Automation communicates with other key IBM products to provide a comprehensive, automated processing facility and an integrated solution for the control of all production workloads. Here are the benefits that the portfolio offers you:

Increased automation
Increases efficiency and uses DP resources more effectively, resulting in improved service levels for your customers.

Improved systems management integration
Provides a unified solution to your systems management problems.

More effective control of DP operations
Lets you implement change and manage growth more efficiently.

Increased availability
Is made possible by automatic workload recovery.

Opportunities for growth
Are made possible by your ability to manage greater workload volumes.

Investment protection
Is made easier by building on your current investment in z/OS and allowing existing customers to build on their existing investment in workload management.

Improved customer satisfaction
Is achieved thanks to higher levels of service and availability, fewer errors, and faster response to problems.

Greater productivity
Results because repetitive, error-prone tasks are automated and operations personnel can use their time more efficiently.

Integration of multiple operating environments
Provides a single controlling point for the cooperating systems that comprise your DP operation.

The portfolio is more than just a batch scheduling tool: it is a production management system with the capability to schedule *all* the work running on any system.

Chapter 3. Tivoli Workload Automation and ITUP

This chapter explains where Tivoli Workload Automation is placed within the IBM Tivoli Unified Process (ITUP).

The IBM Tivoli Unified Process provides detailed documentation of IT Service Management processes based on industry best practices, to help users to significantly improve their organization's efficiency and effectiveness. ITUP helps users to easily understand processes, the relationships between processes, and the roles and tools involved in an efficient process implementation.

The processes described in ITUP are strongly aligned with the Information Technology Infrastructure Library (ITIL) which is based on best practices observed within the IT industry. ITIL provides high-level guidance of what to implement, but not how to implement. ITUP contains detailed process diagrams and descriptions to help users understand processes and their relationships, making ITIL recommendations easier to follow.

ITUP is based on the IBM Process Reference Model for IT™ (PRM-IT), which was jointly developed by IBM Global Services and Tivoli. PRM-IT provides detailed process guidance for all activities that fall under the office of the CIO, including, but not limited to, IT Service Management.

The ITUP processes

ITUP describes a comprehensive set of processes within an IT organization. Each process is defined by:

Tool Mentors that describe best practice use of IBM tools in a process context
Tool mentors help users identify which IBM products and solutions can be used to perform specific process activities and details their appropriate use. By following this guidance users can reduce time, effort, and errors, and get the maximum value out of their investments.

Role definitions, responsibilities, and resources
ITUP describes the roles and responsibilities of all actors in the process model. Users can identify their roles and understand the activities they need to perform and the tools available to help them.

Work products and other information
ITUP describes all work products, often referred to as artifacts, produced as output or required as input by processes and activities. Other information such as key terms and concepts are also defined.

Scenarios describing common problems and best practice solutions
Scenarios help users understand how real world problems can be addressed with process improvement and integration, proper tool use, and defined roles and responsibilities.

Service execution and workload management

Among the processes supported by Tivoli Workload Automation, service execution is one of the key IT processes described by ITUP. Its mission is to deliver operational services to the IT infrastructure and to the enterprise.

The main service execution activities supported by Tivoli Workload Automation are:

- Deliver service
- Manage delivery resources
- Manage workload

Workload management has the target to maximize the utilization of task execution resources and to minimize the total time that is required to deliver the output of task processing. This activity operates at both a macro-level and micro-level to prepare work schedules and to pre-process work items where necessary so that the delivery resources can be matched to the demands of the flow of work in an optimal fashion.

The objectives of workload scheduling focus on:

- Managing the execution of activities according to business calendars, time constraints, and resource availability.
- Managing activities that have interactions between each other and have dependencies with each other and external entities and events.
- Enabling integration with business application environments like SAP R3 and PeopleSoft, and managing the running of activities in those environments.
- Managing the life cycle for defining the activities to run and their running.
- Enabling monitoring and control on the running of these activities and collecting results and historical running data.

Managing workload with Tivoli Workload Automation

Tivoli Workload Automation is a portfolio of products provided by IBM to automate all workload management tasks. The scheduling features of Tivoli Workload Automation help you plan every phase of your workload production. During the processing period, the production control programs manage the production environment and automate most operator activities. The schedulers prepare jobs for running, resolve interdependencies, and launch and track jobs. Because jobs start running as soon as their dependencies are satisfied, idle time is minimized, and throughput improves significantly. Jobs never run out of sequence, and, if a job fails, the schedulers can handle the recovery process with little or no operator intervention.

Workload management is based on a database that contains the definitions of the scheduling objects. There are two versions of the scheduling objects database depending on the placement of the main workload controller: it can be based on a mainframe computer (in this case, z/OS) or on a distributed platform. Some of the scheduling objects can exist in both of the databases, some apply only to the distributed platform, and others might apply only to the mainframe platform.

The minimum set of object definitions that are required to produce a workload consists of a workstation, a job, and a job stream. Other required scheduling objects might be predefined and exist by default.

A workstation is a definition that represents a computer system or another entity that is capable of running specific tasks, and that has the ability to report the status of task execution to the scheduler. With the Tivoli Workload Automation interfaces, you can identify the physical resources associated with the workstations.

A job is the representation of a task (an executable file, program, or command) that is scheduled and launched by the scheduler. The job is run by a workstation and, after running, has a status that indicates if the run was successful or not. A job definition can specify information on what to do whenever its run was not successful. Jobs not included in a job stream do not have any attribute for running, and are only the description of a task with a definition on how to perform it in a form that is known to the specified workstation.

A job stream represents a container for related jobs and organizes them, in terms of run time, sequencing, concurrency limitations, repetitions, assigning priority or resources, and so on. Job streams are the macro elements of the workload that you manage.

The scheduling plan is the to-do list that tells Tivoli Workload Scheduler or Tivoli Workload Scheduler for z/OS what jobs to run, and what dependencies must be satisfied before each job is launched. Tivoli Workload Scheduler or Tivoli Workload Scheduler for z/OS builds the plan using the elements that are stored in the scheduling database.

The running of a plan requires tracking to identify possible problems that can impact the effective delivery of the work products. It is possible to perform the tracking from a Web-based Java interface, the Dynamic Workload Console, on either of the platforms (z/OS and distributed). As an alternate interface to the Dynamic Workload Console on the z/OS platform you can also use the ISPF panel interface, and on the distributed platforms you can use the command-line interface.

See Chapter 5, “A business scenario,” on page 29 which describes a possible implementation of workload management based on Tivoli Workload Automation.

Chapter 4. Who performs workload management

The primary roles most directly responsible for workload management are:

The IT administrator
Is the general IT administrator of all the hardware and software used by the company. He is in charge of installing, supporting, and maintaining servers or other computer systems, and planning for and responding to service outages and other problems.

He installs and maintains the job scheduling tool.

The Tivoli Workload Scheduler IT administrator
A good deal of his time is focused on keeping job scheduling running smoothly. He rarely does any actual scheduling himself, but instead acts as the person in the background who supports those who do. The Tivoli Workload Scheduler IT administrator:

- Defines and maintains the security for the job scheduling tool.
- On certain occasions does a minimal amount of tuning and customization of the job scheduling tool.
- Guarantees that the job scheduling tool environments are up and running all of the time, and if something goes wrong he needs to quickly resolve the problem.
- Monitors the health status of the job scheduling tool infrastructure. Uses tools that alert him (usually via email or pager) and create alerts or automatically open a trouble-ticket to alert the responsible person when there is a problem.
- Occasionally spends his time helping to fix job scheduling problems that the job schedulers cannot understand.
- Generates and uses reports.
- Occasionally documents major problems and work-arounds on the community website.
- Interacts mainly with fellow team members, programmers, and job schedulers.
- Sometimes makes suggestions to management about capacity planning and IT software purchases.

The job scheduler
Is the primary actor in workload management and needs to easily create and maintain a plan containing the company workload. He is responsible for modeling the company workload, and for designing, fixing, and maintaining schedules. His main responsibilities are to:

- Manage workload complexity and dependencies.
- Optimize schedule efficiency, flexibility, resiliency.
- Analyze and fix modeling issues.
- Look proactively for the schedule's integrity.

The scheduling operator
Is responsible for performing all operational processes and procedures, ensuring the business continuity of the workflow. His main responsibilities are to:

- Monitor critical events and perform first analysis of problems.

- Manage and coordinate the resolution of issues.
- Ensure that operations continue.

He is usually not dedicated to monitoring job scheduling alone.

The Scheduling and Operations manager

He:

- Does not use job scheduling tools himself; but is interested in the operational data from the tools, such as reports on long and late running jobs and service level agreement status.
- Makes sure his team has the knowledge and tools they need to schedule and manage jobs efficiently.
- Is always looking for ways to reduce cost in his organization by making his team more efficient.
- Believes that process is the key to IT management and also thinks that his team's job scheduling process can be improved. He is familiar with ITUP of which his IT organization has implemented the basic aspects of change management. Consequently, his team follows this process.

Chapter 5. A business scenario

The purpose of the following scenario is to show how the choice of the correct workload scheduling product, together with process improvement and integration, and well-defined roles and responsibilities, can improve the business of a manufacturing enterprise.

The company

Fine Cola is a medium-sized enterprise that produces and distributes soft drinks to retailers across the country. It owns a production plant and several strategically located distribution centers. The primary customers of Fine Cola are foodstore chains and the quantity and size of their orders is usually regular and stable. Order quantities, however, peek in the warmer season and during holidays. Moreover, in the mid term, Fine Cola wants to increase its business by gaining market in other countries. Fine Cola's sales people are always keen to place new orders and increase the customer portfolio. These characteristics determine Fine Cola's production and distribution processes. Production and distribution can be broken down into ongoing subprocesses or phases which are constantly interlocked with each other. They are:

Inventory
Underlays the entire production process. The raw materials database is sized on the production levels supplemented by minimum safety levels. The production levels are in turn based on the order quantity for the specific period.

Ordering
Raw material quantity levels must be available to production according to the preset production levels. Orders must be planned and issued in advance to take into account delivery times by third-party suppliers.

Production
General production levels are planned for well in advance based on customer orders. Production is regularly increased by an additional five percent to provide the capability to honor unplanned-for orders.

Supply
From the production plant the soft drinks are transported to the distribution centers according to the customer delivery schedules.

Delivery
The last phase of the process. Fine Cola sodas are delivered from the distribution centers to the customer shelves.

Inventory, ordering, and production take place in the production plant. Supply takes place from the production plant to the distribution centers. Delivery takes place from the distribution centers to the end destinations.

These phases are tightly bound to each other. While each soda placed on the shelf might be regarded as the outcome of a specific sequence that starts with inventory and terminates with delivery, all phases are actually constantly interwoven. In fact, the same data is shared in one way or another by all or most phases, and applications are designed to carry on the daily operations and set up future ones.

Fine Cola uses the following databases for running the above-mentioned subprocesses:

Customer Orders
Contains all orders for the upcoming period from Fine Cola's customer base. Provides input to:
- Inventory

Raw Materials
Contains the quantities in stock of the raw materials required to produce Fine Cola's sodas. From here, orders are dispatched to suppliers when stock levels reach a pre-set minimum. Receives input from:
- Production Volumes

Production Volumes
Contains the quantities of sodas that are to be produced daily according to order volumes. Provides input to:
- Inventory
- Raw Materials

Receives input from:
- Inventory

Inventory
Contains the quantities in stock of the finished product. Is monitored to verify that the quantities in stock are sufficient to honor the orders of a specific time interval. Provides input to and receives input from:
- Production Volumes
- To Supply

To Supply
Contains the quantities of sodas that must be sent periodically from the manufacturing plant to the distribution centers to satisfy foodstore orders for the upcoming period. Provides input to:
- Inventory
- To Deliver

To Deliver
Contains the quantities that are to be delivered from each distribution center to the foodstores in its area. Provides input to:
- Customer Orders

Receives input from:
- To Supply

The company workload is both application oriented, such as accounting, payroll, supplier and utility payments, purchasing, ordering, fulfillment, and system-oriented, such as data backup, migration, export, transfer or load operations. Typically, the workload processes multiple data items such as accounts, orders, transactions, database records, at the same time.

These core applications are highly relevant for the profitability of the company and also directly influence customer satisfaction.

To create added value and exceed customers expectations, the company must strengthen integration with business applications and provide complete scheduling capabilities and tighter integration with enterprise applications.

The challenge

Currently the databases are not automatically integrated with each other and need continual human intervention to be updated. This affects Fine Cola's operations because:

- The process as a whole is onerous and prone to error.
- The interfaces between phases are slow and not very efficient.

The company realizes it needs to better integrate with the distribution centers because processing is extremely low during the regular office hours in the warmer seasons and during holidays. Users experience applications freezing, often taking considerable time before being available for them to use again. This lack of integration is causing problems for the organization in terms of lost productivity, while applications come back online. This is a problem because the interruption of important processing is not acceptable when the company wants to expand the business. The response time for service level agreements (SLAs) must continue to be met if a resource goes down, a workstation breaks, or there is urgency for maintenance, and even more during peak periods even if the resources are geographically distributed. On the other hand the company does not want to buy new IT resources (hardware, software, applications) because this would not be used during the other periods of the year.

Fine Cola realizes that their main weakness lies in their processing. They need to implement a solution that:

- Integrates the data behind their processing workflow from inventory to distribution. This makes it possible to automatically trigger the daily operations without much need for human intervention. It also gives Fine Cola complete control over the entire business process, reducing human intervention only to exception handling.
- Integrates external data coming from third parties, such as selected customers and raw material suppliers, into their process flow. Such data is provided to Fine Cola in several formats and from different applications and should be integrated into Fine Cola's databases in a seamless manner.
- Enables daily backups of their data as well as subsequent reorganization of the DB2 database with as little impact as possible on their processes. Processing of data collected online during the previous day is the next step.
- Optimizes capacity across the IT infrastructure and runs a high workload, much more than before, using shared resources, even if the resources are geographically distributed.
- Ensures 24x7x365 availability of critical business services. Disaster recovery plans are no longer sufficient because the business requires recovery within a couple of hours, not days. Recovering from last night tapes and recapturing lost transactions after a system or application failure is no longer a viable option for the company in a highly competitive market.
- Has very low probability of failure leading to maximum system reliability.

The main company goal at this time is to obtain an integrated workload solution that can entirely choreograph its business application tasks. This means solutions that optimize capacity across the IT infrastructure and run a tremendous workload, much more than before, using less resources. For example, if the company has a problem and a primary server does not process the workload, the company wants to automate the quick redistribution of system resources to process workloads and scale up or down for flawless execution. In this way the company reduces costs because it speeds recovery time, no matter what the source. The goal is to have a

view of the best available resources across this dynamically shifting cross-enterprise pool.

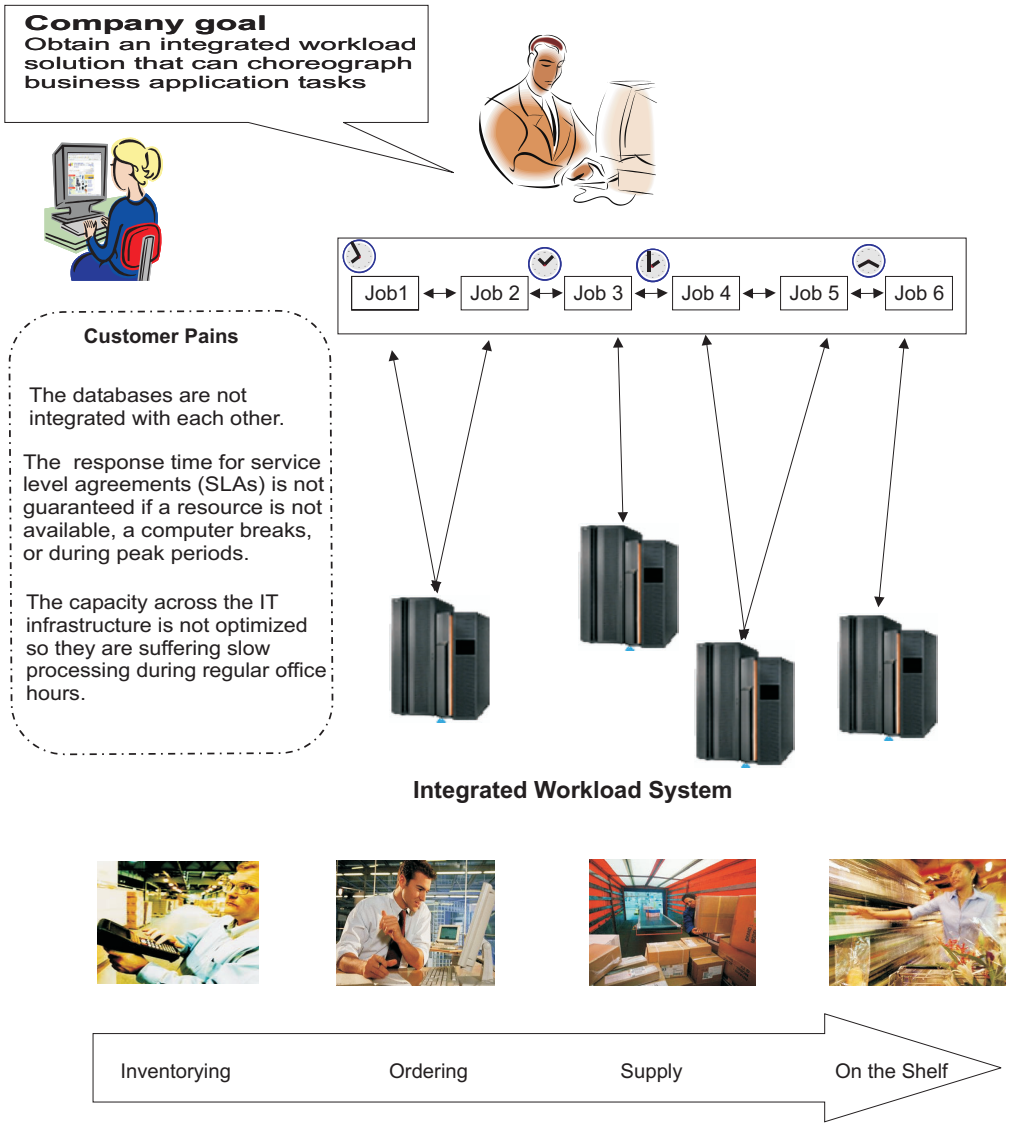


Figure 2. The Fine Cola company integrated workload solution

The solution

Fine Cola decides that one important step toward improving their process execution is to adopt a solution based on automatic and dynamic workload scheduling. The solution is based on a choice that strengthens integration with business applications to run the following tasks:

- Read data from one database to update other databases.
- Read data from external applications, process it, and add it to the appropriate databases.
- Provide the information necessary for the operation of every phase.
- Trigger some of the phases when predetermined thresholds are reached.
- Back up their data without interrupting production.

- From a capacity management perspective, understands the size of an application and what resources it requires, models that against the existing resources and is able to predict and forecast the capacity that the new application needs as it is defined in the enterprise.
- From an availability management perspective, use the resources available in the environment to support the application and understand out how to work to effectively schedule, monitor, and manage that application as it is submitted. Then if the resources are not available, interact with the change management and provisioning processes to dynamically allocate the necessary resources.
- Have a business management process monitoring all the various policies and driving a consistent view of the policies for the application.

After analyzing the workload management products available on the market, Fine Cola has chosen to use IBM Tivoli Workload Scheduler and specifically the dynamic domain manager to:

- Optimize and automate the tasks to process their applications and dynamically adapt their processing in response to changes in the environment.
- Plan, choreograph, and schedule required changes to applications to minimize the impact of changes on critical production workloads, and ensure that workload processes are updated to reflect changes throughout asset life cycles.
- Minimize the total amount of time that is required to deliver the output of the task resolution processes.
- Handle dependencies between tasks, data, and external applications so that the entire workload can be managed homogeneously in the same process flow.
- Create a policy-based view of workflow automation, not just workload automation, but cross-enterprise workflow, and direct that workflow across the enterprise while planning, scheduling, managing, and monitoring all these things. Dynamically tuning the cross-enterprise capacity to support this dynamic view of workloads.
- Automatically transfer entire workloads across multiple platforms, and update policies across multiple platforms.
- Balance between the ability to provide sophisticated planning, testing, choreographing, monitoring, and adaptation of workload processes with fault tolerance and redundancy for high availability of the scheduling infrastructure, while minimizing server and network resource requirements.
- Perfectly integrate with each other.

Tivoli Workload Scheduler operates at both a macro-level and micro-level to prepare work schedules and to preprocess work items where necessary so that the delivery resources can be matched to the demands of the flow of work in an optimal fashion.

The dynamic domain manager dynamically routes workload to the best available resources based on application requirements and business policies. Moreover it optimizes the IT computing resource use according to SLAs.

Fine Cola's applications are mapped to what in Tivoli Workload Scheduler terminology are units of work called jobs. Some of these jobs are statically allocated to dedicated resources to run (static job definition), others are dynamically allocated to physical or virtual resources according to the job importance, requirements, scheduling policies, and based on the environment resource characteristics, relationships, availability, load, and performance (dynamic job definition). They drive the resource allocation to meet the job SLA and the resource optimization.

Jobs that run as a unit (such as a weekly backup application), along with times, priorities, and other dependencies that determine the exact order of the jobs are grouped into job streams.

Fine Cola's job streams are collections of jobs that are grouped for organizational purposes. The jobs of any particular job stream are related because they:

- Operate toward the completion of related tasks. For example, the jobs of `Jobstream100` run tasks designed to convert incoming customer orders into operational data.
- Might be dependent on each other. Some jobs might have to wait for the completion of predecessor jobs before they can start running. The jobs are usually laid out in a sequence where the outcome of a predecessor is fed to a successor job.
- Share the same programs, applications, and databases.
- Share the same time-frames within the planning period.

Using Tivoli Workload Scheduler, Fine Cola's business process is structured in the following way:

1. At the start of each day, `Jobstream100`:
   a. Extracts the new incoming orders from the Customer Orders database.
   b. Checks an external application where a number of selected customers can place unforeseen orders. If there are orders, they are extracted and merged with the other data.
   c. Copies the consolidated orders into a separate database view.
   d. Sorts them by due delivery date and by quantity and makes a report.
2. As soon as the report is available, `Jobstream200` extracts the numbers from the report and compares them with relevant data in the Inventory database. The goal is to determine the production volume required in the next production cycle to satisfy the orders.
3. `Jobstream300` extracts the production volume data and updates the Production Volumes database with the quantities of each type of soda that is to be manufactured in the next cycle.
4. `Jobstream400` reads the data in the Production Volumes database and:
   a. Calculates the quantities of raw materials required to run the upcoming production cycle.
   b. Flags these quantities as `allocated to next cycle` in the Raw Materials database.
   c. Checks the quantities to see if they have reached the minimum stock levels and triggers orders to Fine Cola's raw material suppliers if necessary.
5. `Jobstream500` reads the report with upcoming due orders from the Customer Orders database and:
   a. Produces the transportation schedules and destinations.
   b. Updates the To Supply database.
   c. Sends the delivery schedules to the distribution centers.
6. `Jobstream600` reads the distribution center databases and:
   a. Extracts the orders that have been filled.
   b. Updates the Customer Orders database so that invoices can be prepared and sent.
7. `Jobstream700` makes a backup of every database.

Fine Cola sets up a long term plan that encompasses the entire workload, spanning job streams that run on a daily basis and job streams that have other reoccurrences. From the long term plan, a current plan is extracted at the beginning of every time unit. The time period of the current plan can be chosen to vary from some hours to several days. Fine Cola has chosen to set their current plan on a daily basis. At the start of every day a new daily plan is built by their workload scheduling software: data is taken from the long term plan and from the daily plan of the previous day to include any jobs that might not have completed.

The company must also ensure that during peek periods the jobs in the critical path are run in the required time frame. To ensure this they converted some jobs from static definition to dynamic definition to manage the extra orders using the dynamic domain manager. With the dynamic domain manager, the company can:

- Manage the automatic discovery of available resources in the scheduling environment with their characteristics and relationships.
- Assign to the job the appropriate resources for running based on the job requirements and on the administration polices.
- Optimize the use of the resources by assigning to the job the required resources based on the SLA.
- Manage and control the resource consumption and load.
- Dispatch jobs to target resources that meet the requirements to run the job.

The Tivoli Workload Scheduler relational database contains the information related to the jobs, the job streams, the workstations where they run, and the time specifications that rule their operation. It also contains data used by the dynamic domain manager, such as information about the current IT environment, the resource real time performance, and load data. It also stores the job definitions and keeps track of resources assigned to each job.

In this way, Fine Cola's scheduling analyst can create and change any of these objects at any time and Fine Cola's IT administrator can dynamically assign the best set of resources to match allocation requests based on the defined policies, without any impact on the business.

The IT administrator can also ensure the correct concurrent or exclusive use of the resources across the different jobs according to resource characteristics. If the resource request cannot be immediately carried out, he can use dynamic scheduling to automatically queue the resource until changes in the resource utilization or in the environment lead to its fulfillment.

The workload scheduling plan can be changed as quickly and dynamically as the business and operational needs require. The scheduling analyst makes full use of the trial and forecast planning options available in the scheduler to adjust and optimize workload scheduling and, as a consequence, Fine Cola's line of operations.

To respond to any unexpected and unplanned-for demands, individual jobs can be added ad hoc to the scheduling plan at any time.

Moreover, the company can use dynamic scheduling to rapidly adapt to the increase of workload during peak periods driving the requirement for workload virtualization, that is the ability to manage and control the workload so that it can be slit, routed to appropriate resources and capacity, and dynamically moved around in logical resource pools.

If a resource is not available, the SLA defined continues to be met because the job processing is restarted from the point where the failure happens.

Typical everyday scenarios

This section describes roles and responsibilities of Fine Cola's IT staff and everyday scenarios they might face on any typical day. Fine Cola's IT staff, involved in workload scheduling are:

- The scheduling analyst. He is in charge of modeling the company workload, and for designing, fixing, and maintaining schedules. His main responsibilities are:
  - Managing Fine Cola's workload complexity and dependencies.
  - Optimizing the schedule's efficiency, flexibility, and resilience.
  - Analyzing and fixing modeling issues; look pro-actively for the schedule's integrity.
- The operations analyst. His main responsibilities are:
  - Monitoring critical events and performing first analysis of problems.
  - Managing and coordinating the resolution of issues.
  - Ensuring that operations continue.
- The IT infrastructure administrator. His main responsibilities are:
  - Fulfilling the need to assign physical or virtual resources to jobs according to the job importance, requirements, scheduling policies, and based on the environment resources characteristics, relationships, availability, load, and performance.
  - Managing the advanced reservation or provisioning of required resources.
  - Drive the resource allocation to meet the job SLA and the resource optimization data without service disruption, and possibly transparently for the end users.
  - Backing up the schedule daily with no impact on operations.
  - Ensuring high availability of the infrastructure. If a resource goes down or a workstation is not available, the SLA-defined availability must continue to be met.
  - Defining and maintaining the environment topology.

Managing the workload

Together with the IT infrastructure administrator and other staff, the scheduling analyst agrees on a change in the application workflow that should go into production in a month. The change impacts `Jobstream100` and includes:

- Defining a new job and replacing some job dependencies in the job stream.
- Defining two `Jobstream100` instances to run twice a day for a week and during the summer season. He must therefore:
  - Define a run cycle for each of the two instances. The first run cycle has the expected start time of 9 a.m. The second is scheduled to start at 5 p.m.
  - Agree with the IT infrastructure administrator the pool of resources that satisfy the job SLA in terms of RAM and microprocessors.

He then proceeds in the following way:

1. He reviews the new stream logic and sets a plan. He wants to design the changes, test them over three days, and have a first automatic test run within a week.
2. He proceeds to apply the changes to `Jobstream100`. While he does this, he realizes that the application specialist must modify the tasks (scripts) contained in some of the jobs.

3. For this reason he leaves the job stream in draft state while the work is still in progress, so that it is not included in the plan generated every day.
4. To apply the changes he operates directly using the job stream editor available with Fine Cola's scheduling product: defining a new job by renaming an existing job definition and adding a new dependency.
5. After he has finished drafting the changes, he saves the job stream with a validity date set to tomorrow and active status on the test system.
6. Before launching the plan containing the modified job stream, he generates a trial plan to verify that the dependencies are correctly resolved.
7. When the trial plan ends, he analyzes job statistics and finds that a different design of dependencies could improve total elapsed time.
8. He applies the changes, sets the new dependencies, and creates a plan extension. The job stream is rescheduled and run in a test environment correctly and timely.
9. He meets the IT infrastructure administrator to verify that all the resources involved in the running of the new plan are available on the following days.
10. The IT infrastructure administrator, after analyzing the plan of availability of the IT resources, warns him that one of the required resources will not be available the next week.
11. The IT infrastructure administrator asks the scheduling analyst to run a forecast plan.
12. He then runs a forecast plan, which contains the scheduled activities for next week, to verify that the unavailability of the resources will not cause any major problem.
13. He finds that the unavailability of the resources will cause a decrease in performance because the other resources become overloaded.
14. He notifies the IT infrastructure administrator of the potential problem.
15. The IT infrastructure administrator analyzes the availability of resources between departments and realizes that the resources belonging to another department meet the requirements to run the job definition.
16. The scheduling analyst moves the workload from static to dynamic resource allocation. He uses the dynamic domain manager to route workloads to the best available systems by matching load requirements and business policies to available resource capacities.
17. He identifies the jobs in the critical path and modifies their definitions so that they can be run dynamically.
18. He finally sets a date to run the new plan in the production environment and notifies the IT infrastructure administrator.

Figure 3 on page 38 shows how the Fine Cola company can dynamically manage its workload using the added dynamic scheduling capability of Tivoli Workload Scheduler and satisfying the SLA response time.

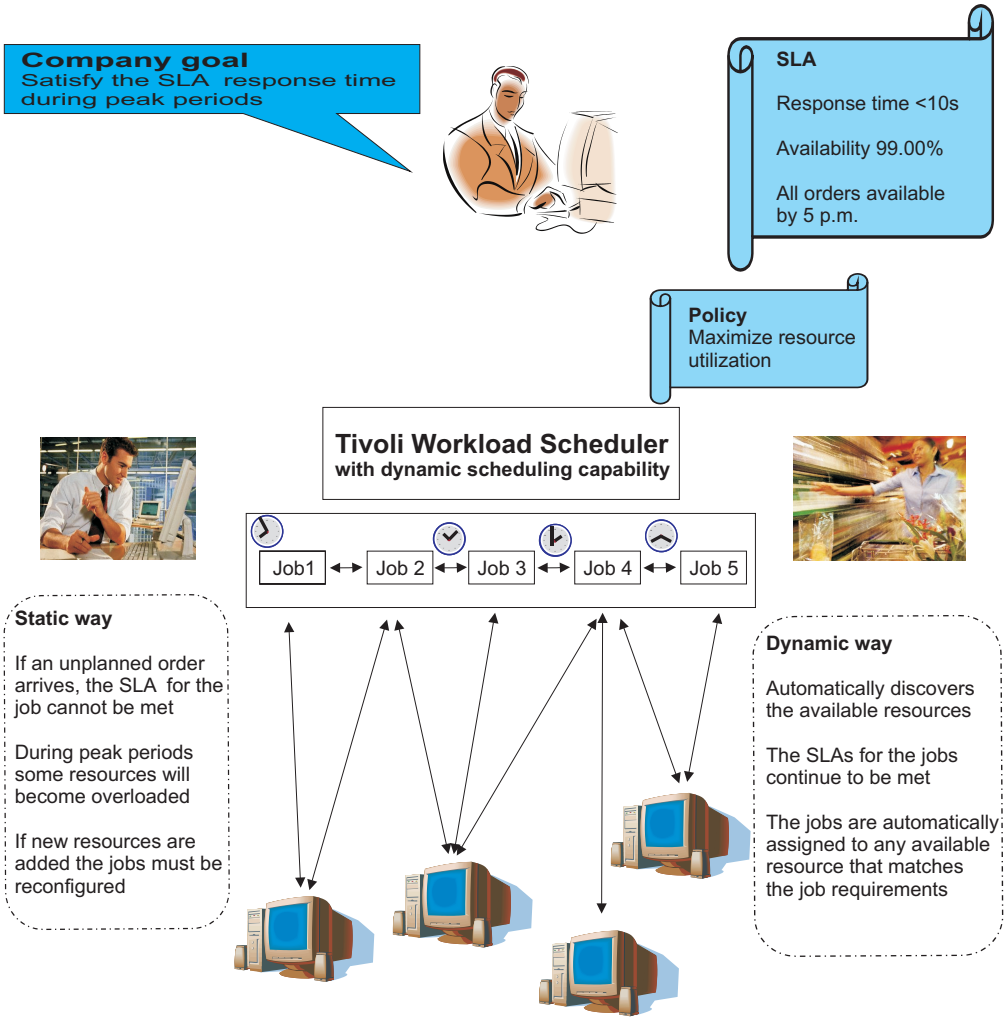


Figure 3. How to satisfy SLA response time during peak periods using the dynamic scheduling capability of Tivoli Workload Scheduler.

Monitoring the workload

While the operations analyst monitors the automated workload on a typical work day, he realizes that `Job306`, which is in the critical path of the schedule, is in the `abend` status. Because of this, `Jobstream300` does not complete in the necessary time frame, causing a negative effect on the rest of the schedule. The consequent delay in running the plan might cause problems to Fine Cola's daily operations. With the help of the Dynamic Workload Console he then:

1. Analyzes the job and error logs in the current plan and finds that the error occurred for an unmanaged exception. The error might occur again and he cannot simply restart `Job306`.
2. Identifies the application specialist who is responsible for `Job306` and opens a problem ticket containing all the information concerning the job.
3. Queries the status of depending jobs, exports the list in Comma Separated Variables (CSV) format, and attaches it to the ticket so that it can be viewed with a spreadsheet. Requests that the ticket be answered with high priority.

After an internal analysis, the application specialist finds that there is a broken execution path that must be fixed. The expected time for resolution is three hours, including a hot fix and a regression test.

One hour later, however, the operations analyst realizes that even if the application support team works overtime, the fix will not be completed before the end of the day and it will be impossible to close the daily processing today. He checks the status of the depending jobs and sets a target time to have the hot fix loaded into production during the night.

Then, sometime during the night:

1. The application team releases the hot fix and notifies the scheduling analyst who loads the new job into the production system, and notifies the operations analyst.
2. The operations analyst connects to the scheduling system from home to restart the job stream.
3. The operations analyst restarts `Job306`. The fix works and the job completes, as expected, one hour too late to complete the depending jobs before the next daily plan extension.
4. Early next morning the plan for the day is created. Because of the functionality of the latest version of Tivoli Workload Scheduler, the jobs depending on `Job306`, that could not complete in time, are now simply moved to run today, keeping their name and all their active file dependencies.
5. The operations analyst monitors the process remotely. When he arrives at work in the morning, he checks the actual completion of the daily workload. Everything completed successfully and he closes the ticket.

Managing the organization of the IT infrastructure

Two weeks before Christmas, the IT infrastructure administrator receives a notification from the scheduling analyst that an unplanned order adds so many tasks to a job stream in the critical path that its completion is delayed by a day. This causes a delay also in the completion of the plan scheduled to run the week before Christmas. The scheduling analyst advises him that he has already run the forecast plan and verified that with this addition the SLA for the job stream cannot be met and also the resources will become overloaded. To avoid this, concurrent jobs that need to use the same resource will need to wait until the requested quantity is available causing delay in the delivery of the order.

To find a solution to the potential problem and achieve the goals set for workload processing, without buying additional resources, using the dynamic domain manager, he proceeds in the following way:

1. He performs an automatic discovery of the resources available in the scheduling domain with their characteristics and relationships.
2. He finds a pool of resources in the Inventory department that meet the SLA to run the jobs. These resources have the required RAM, microprocessor, operating system, and application environments to run the new job stream and will be used at half their capacity during Christmas.

   Without the use of dynamic scheduling he could not adapt the new workload processing to match load requirements with business policies and priorities, and resource availability and capacity. The only way to solve the problem would be to buy new hardware to run the added job streams increasing the cost of IT management infrastructure without optimizing the use of the existing resources.

3. He determines, based on the policies and jobs dispatching, how many new resources are required to run the new job stream.
4. He manages the definition of business-oriented performance goals for the entire domain of servers, provides an end-to-end view of actual performance relative to those goals, and manages the server resource allocation and load to meet the performance goals.
5. He identifies the required resources and finds an agreement with the Inventory department manager, to share the required resource between the two departments.
6. He defines a new logical resource in which he outlines the machines that are shared between the departments.
7. He communicates to the Ordering department the new agreement with the resource optimization.
8. Now he can guarantee the running of jobs within the time frame according to policies, rules, and resources planned availability. In this way he can also satisfy the optimization policy to maximize resource utilization.
9. The scheduling analyst now builds a feasible production plan.

   Using dynamic scheduling he met the constraints imposed by rules and policies and achieved SLA goals, optimizing execution time, throughput, cost, and reliability.

The benefits

By adopting a workload scheduling strategy, and in particular by using Tivoli Workload Scheduler and its dynamic scheduling capabilities, Fine Cola is experiencing significant and immediate benefits, such as:

- The successful integration of all its manufacturing and distribution processes.

  Because of how Fine Cola implemented their new processing flow, every customer order is active from the time a customer service representative receives it until the loading dock ships the merchandise and finance sends an invoice. Now orders can be tracked more easily, and manufacturing, inventory, and shipping among many different locations can be coordinated simultaneously. If an unplanned order arrives, it can be easily managed in the new dynamic IT infrastructure.
- The standardization and speeding up of the manufacturing process.

  Tivoli Workload Scheduler has helped to automate many of the steps of Fine Cola's manufacturing process. This results in savings in time and increase in productivity.
- Reduce inventory

  The manufacturing process flows more smoothly, and this improves visibility of the order fulfillment process inside the company. This can lead to reduced inventory of the raw materials used, and can help better plan deliveries to customers, reducing the finished goods inventory at the warehouses and shipping docks.
- Optimize IT infrastructures

  The dynamic allocation of the IT resources maximizes the workload throughput across the enterprise reducing costs, improving performance, and aligning IT with business needs and service demands.
- Guarantees Fault Tolerance and High Availability

  Tivoli Workload Scheduler can recover from server, agent, and communication failures and it can restart from the point where the failure happened. No status

information will be lost due to failure events. Moreover if a computer breaks, its workload is automatically routed to another computer that can guarantee the SLAs.

In conclusion, this solution provides business value because it:

- Delivers service response times according to service level objectives.
- Understands dependencies on services for each line of business.
- Accommodates unpredictable use patterns with predictive logic.
- Understands service relationships to each other and to the IT infrastructure and business process layers.
- Provides network fault tolerance and high availability of the scheduling infrastructure.
- Reduces system and operational complexity and leverages IT staff skills and knowledge.
- Integrates systems quickly and easily, with minimal disruption to existing business processes.

A business scenario

Chapter 6. Tivoli Workload Scheduler

Tivoli Workload Scheduler's scheduling features help you plan every phase of production. During the plan processing period, the Tivoli Workload Scheduler production control programs manage the production environment and automate most operator activities. Tivoli Workload Scheduler prepares jobs for execution, resolves interdependencies, and launches and tracks each job. Because jobs start running as soon as their dependencies are satisfied, idle time is minimized, and throughput improves significantly. Jobs never run out of sequence, and, if a job fails, Tivoli Workload Scheduler handles the recovery process with little or no operator intervention.

Overview

The next sections provide an outline of Tivoli Workload Scheduler.

What is Tivoli Workload Scheduler

Tivoli Workload Scheduler is composed of three parts:

Tivoli Workload Scheduler engine
The scheduling engine. It runs on every computer of a Tivoli Workload Scheduler network. During installation, the engine is configured for the role that the workstation will play within the scheduling network, such as master domain manager, domain manager, or agent.

Tivoli Workload Scheduler Connector
It maps Dynamic Workload Consolecommands to the Tivoli Workload Scheduler engine through the embedded version of WebSphere® Application Server. The Tivoli Workload Scheduler connector usually runs on the master and on any of the fault-tolerant agents (FTAs) that you plan to use as backup machines for the master workstation.

The Dynamic Workload Console
Is Web-based, light, powerful, and user friendly. It can be used on any computer that has a web browser and provides access to all the current Tivoli Workload Scheduler functions. It is the strategic graphical user interface for the entire Tivoli Workload Automation portfolio.

The Tivoli Workload Scheduler network

A Tivoli Workload Scheduler network is made up of the workstations, or CPUs, on which jobs and job streams are run.

A Tivoli Workload Scheduler network contains at least one Tivoli Workload Scheduler 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.

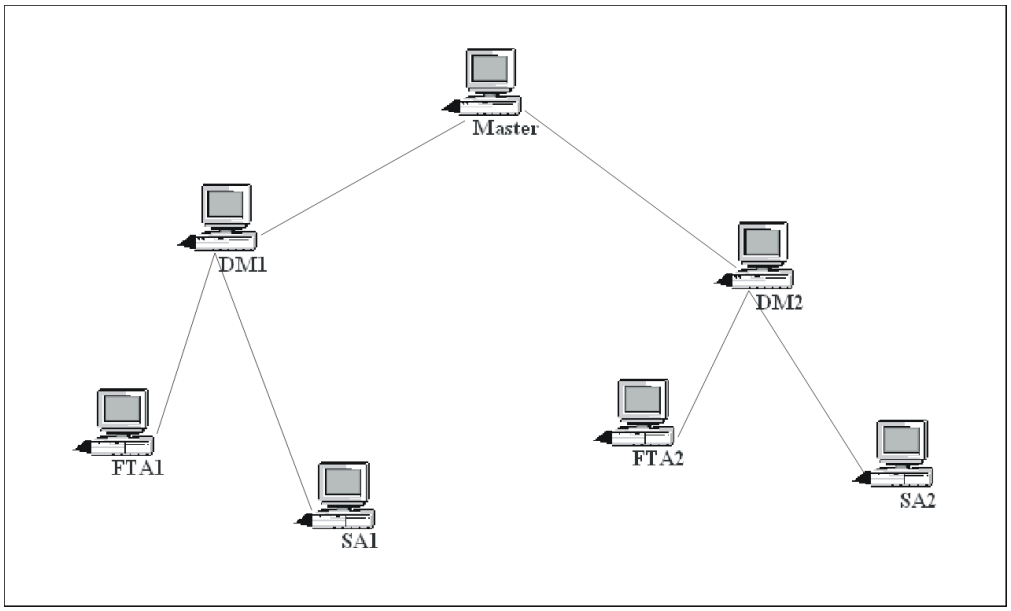


Figure 4. This Tivoli Workload Scheduler network is made up by two domains

Using multiple domains reduces the amount of network traffic by reducing the amount of communication required between the master domain manager and other computers.

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

In a multi-domain configuration, the master domain manager communicates with the workstations in its domain and with the subordinate domain managers. The subordinate domain managers, in turn, communicate with the workstations in their domains and with their subordinate domain managers. 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, you can designate backup domain managers to take over if their domain managers fail.

Every time the production plan is created or extended the master domain manager creates a production control file, named Symphony. Tivoli Workload Scheduler is then restarted in the network, and the master domain manager sends a copy of the new production control 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 subordinate domain managers.

Once the network is started, scheduling messages like job starts and completions are passed from the agents to their domain managers, through the parent domain managers to the master domain manager. The master domain manager then broadcasts the messages throughout the hierarchical tree to update the production control files of domain managers and fault tolerant agents running in Full Status mode.

Manager and agent types

Primarily, workstation definitions refer to physical workstations. However, in the case of extended and network agents, the workstations are logical definitions that must be hosted by a physical Tivoli Workload Scheduler workstation.

Tivoli Workload Scheduler workstations can be of the following types:

Master domain manager (MDM)
The domain manager in the topmost domain of a Tivoli Workload Scheduler network. It either contains or connects to the relational database that stores the scheduling object definitions. It creates or updates the production file when the plan is created or extended and distributes it in the network. It performs all logging and reporting for the network.

Backup master
A fault-tolerant agent or domain manager capable of assuming the responsibilities of the master domain manager for automatic workload recovery.

Domain manager
The management hub in a domain. All communications to and from the agents in a domain are routed through the domain manager.

Backup domain manager
A fault-tolerant agent capable of assuming the responsibilities of its domain manager.

Dynamic domain manager
An installed component in a distributed Tivoli Workload Scheduler network that is the management hub in a domain. All communication to and from the dynamic agents in the domain is routed through the dynamic domain manager.

Backup dynamic domain manager
A workstation which can act as a backup for the dynamic domain manager, when problems occur. It is effectively a dynamic domain manager, waiting to be activated. Its use is optional.

Fault-tolerant agent (FTA)
A workstation capable of resolving local dependencies and launching its jobs in the absence of a domain manager.

Dynamic agent
An agent installed with dynamic scheduling capabilities. It is assigned the execution of dynamic workload based on the state of its resources at the time of execution.

Standard agent
A workstation that launches jobs only under the direction of its domain manager. It is not fault-tolerant.

Extended agent
A logical workstation definition that helps you launch and control jobs on other systems and applications, such as PeopleSoft, Oracle Applications, SAP, and z/OS.

z-centric agent
Runs jobs scheduled from Tivoli Workload Scheduler for z/OS but is installed in the Tivoli Workload Scheduler environment. It has no fault-tolerance and communicates directly with the Tivoli Workload Scheduler for z/OS controller through the RESTful HTTP interface. In Tivoli Workload Scheduler for z/OS it has the same functionality as a computer automatic workstation even though it runs in the distributed environment.

Network Agent
A logical workstation definition for creating dependencies between jobs and job streams in separate Tivoli Workload Scheduler networks.

Topology

A key to choosing how to set up Tivoli Workload Scheduler domains for an enterprise is the concept of localized processing. The idea is to separate or localize the enterprises's scheduling needs based on a common set of characteristics.

Common characteristics are things such as geographical locations, business functions, and application groupings. Grouping related processing can limit the amount of interdependency information that needs to be communicated between domains. The benefits of localizing processing in domains are:

- Decreased network traffic. Keeping processing localized to domains eliminates the need for frequent interdomain communications.
- Provides a convenient way to tighten security and simplify administration. Security and administration can be defined at, and limited to, the domain level. Instead of network-wide or workstation-specific administration, you can have domain administration.
- Network and workstation fault tolerance can be optimized. In a multiple domain Tivoli Workload Scheduler network, you can define backups for each domain manager, so that problems in one domain do not disrupt operations in other domains.

Networking

The following questions will help in making decisions about how to set up your enterprise's Tivoli Workload Scheduler network. Some questions involve aspects of your network, and others involve the applications controlled by Tivoli Workload Scheduler. You may need to consult with other people in your organization to resolve some issues.

- How large is your Tivoli Workload Scheduler network? How many computers does it hold? How many applications and jobs does it run?

  The size of your network will help you decide whether to use a single domain or the multiple domain architecture. If you have a small number of computers, or a small number of applications to control with Tivoli Workload Scheduler, there may not be a need for multiple domains.
- How many geographic locations will be covered in your Tivoli Workload Scheduler network? How reliable and efficient is the communication between locations?

  This is one of the primary reasons for choosing a multiple domain architecture. One domain for each geographical location is a common configuration. If you choose single domain architecture, you will be more reliant on the network to maintain continuous processing.
- Do you need centralized or decentralized management of Tivoli Workload Scheduler?

  A Tivoli Workload Scheduler network, with either a single domain or multiple domains, gives you the ability to manage Tivoli Workload Scheduler from a single node, the master domain manager. If you want to manage multiple locations separately, you can consider the installation of a separate Tivoli Workload Scheduler network at each location. Note that some degree of decentralized management is possible in a stand-alone Tivoli Workload Scheduler network by mounting or sharing file systems.

- Do you have multiple physical or logical entities at a single site? Are there different buildings, and several floors in each building? Are there different departments or business functions? Are there different applications?

  These may be reasons for choosing a multi-domain configuration. For example, a domain for each building, department, business function, or each application (manufacturing, financial, engineering, etc.).
- Do you run applications, like SAP R/3, that will operate with Tivoli Workload Scheduler?

  If they are discrete and separate from other applications, you may choose to put them in a separate Tivoli Workload Scheduler domain.
- Would you like your Tivoli Workload Scheduler domains to mirror your Windows domains?

  This is not required, but may be useful.
- Do you want to isolate or differentiate a set of systems based on performance or other criteria?

  This may provide another reason to define multiple Tivoli Workload Scheduler domains to localize systems based on performance or platform type.
- How much network traffic do you have now?

  If your network traffic is manageable, the need for multiple domains is less important.
- Do your job dependencies cross system boundaries, geographical boundaries, or application boundaries? For example, does the start of Job1 on CPU3 depend on the completion of Job2 running on CPU4?

  The degree of interdependence between jobs is an important consideration when laying out your Tivoli Workload Scheduler network. If you use multiple domains, you should try to keep interdependent objects in the same domain. This will decrease network traffic and take better advantage of the domain architecture.
- What level of fault-tolerance do you require?

  An obvious disadvantage of the single domain configuration is the reliance on a single domain manager. In a multi-domain network, the loss of a single domain manager affects only the agents in its domain.

Tivoli Workload Scheduler components

Tivoli Workload Scheduler uses several manager processes to efficiently segregate and manage networking, dependency resolution, and job launching. These processes communicate among themselves through the use of message queues. Message queues are also used by the Console Manager (conman) to integrate operator commands into the batch process.

On any computer running Tivoli Workload Scheduler there are a series of active management processes. They are started as a system service, or by the `StartUp` command. The following are the main processes:

Netman
: The network management process that establishes network connections between remote `mailman` processes and local `Writer` processes.

Mailman
: The mail management process that sends and receives inter-CPU messages.

Batchman
The production control process. Working from Symphony™, the production control file, it runs jobs streams, resolves dependencies, and directs `jobman` to launch jobs.

Writer The network `writer` process that passes incoming messages to the local `mailman` process.

Jobman
The job management process that launches and tracks jobs under the direction of `batchman`.

In addition, Tivoli Workload Scheduler uses two command line interfaces:

Composer
The command-line program used to define, manage, and store scheduling objects in the Tivoli Workload Scheduler database. The composer command-line program can be installed and used on any computer connected through TCP/IP to the system where the master domain manager is installed. It does not require the installation of a Tivoli Workload Scheduler workstation as a prerequisite. It communicates through HTTP/HTTPS with the master domain manager where the relational database management system (RDBMS) is installed. The HTTP/HTTPS communication setup and the authentication check are managed by the WebSphere Application Server - Express® infrastructure. The composer uses edit files to update the scheduling database.

Conman
The console manager. It is the user interface for plan running activities by means of the command line interface. `Conman` writes information that is received by either the local `netman` or `mailman` processes.

Tivoli Workload Scheduler scheduling objects

Scheduling with Tivoli Workload Scheduler includes the capability to do the following:

- Schedule jobs across a network.
- Group jobs into job streams according, for example, to function or application.
- Set limits on the number of jobs that can run concurrently.
- Create job streams based on day of the week, on specified dates and times, or by customized calendars.
- Ensure correct processing order by identifying dependencies such as successful completion of previous jobs, availability of resources, or existence of required files.
- Set automatic recovery procedures for unsuccessful jobs.
- Forward incomplete jobs to the next production day.

Starting from version 8.3, the Tivoli Workload Scheduler scheduling objects are stored in a relational database. This results in a significant improvement, in comparison with previous versions, of how objects are defined and managed in the database. Each object can now be managed independently without having to use lists of scheduling objects like calendars, parameters, prompts and resources. The command syntax used to define and manage these objects has also become direct and powerful.

Tivoli Workload Scheduler administrators and operators work with these objects for their scheduling activity:

Workstation
Also referred to as *CPU*. Usually an individual computer on which jobs and job streams are run. Workstations are defined in the Tivoli Workload Scheduler database as a unique object. A workstation definition is required for every computer that executes jobs or job streams in the Tivoli Workload Scheduler network.

Workstation class
A group of workstations. Any number of workstations can be placed in a class. Job streams and jobs can be assigned to execute on a workstation class. This makes replication of a job or job stream across many workstations easy.

Job A script or command, run on the user's behalf, run and controlled by Tivoli Workload Scheduler.

Job stream
A list of jobs that run as a unit (such as a weekly backup application), along with run cycles, times, priorities, and other dependencies that determine the exact order in which the jobs run.

Calendar
A list of scheduling dates. Each calendar can be assigned to multiple job streams. Assigning a calendar to a job stream causes that job stream to run on the dates specified in the calendar. A calendar can be used as an inclusive or as an exclusive run cycle.

Run cycle
A cycle that specifies the days that a job stream is scheduled to run. Run cycles are defined as part of job streams and may include calendars that were previously defined. There are three types of run cycles: a Simple run cycle, a Weekly run cycle, or a Calendar run cycle (commonly called a calendar). Each type of run cycle can be inclusive or exclusive. That is, each run cycle can define the days when a job stream is included in the production cycle, or when the job stream is excluded from the production cycle.

Prompt
An object that can be used as a dependency for jobs and job streams. A prompt must be answered affirmatively for the dependent job or job stream to launch. There are two types of prompts: predefined and ad hoc. An ad hoc prompt is defined within the properties of a job or job stream and is unique to that job or job stream. A predefined prompt is defined in the Tivoli Workload Scheduler database and can be used by any job or job stream.

Resource
An object representing either physical or logical resources on your system. Once defined in the Tivoli Workload Scheduler database, resources can be used as dependencies for jobs and job streams. For example, you can define a resource named *tapes* with a unit value of two. Then, define jobs that require two available tape drives as a dependency. Jobs with this dependency cannot run concurrently because each time a job is run the *tapes* resource is in use.

Variable and variable table
A variable can be used to substitute values in scheduling objects contained in jobs and job streams; that is, in JCL, log on, prompts dependencies, file dependencies, and recovery prompts. The values are replaced in the job scripts at run time. Variables are global (that is, they can be used on any agent of the domain) and are defined in the database in groups called `variable tables`.

Parameter
A parameter can be used to substitute values in jobs and job streams just

like global variables. The difference is that a parameter is defined on the specific workstation where the related job is to run and has no global effect, but only on that specific workstation. Parameters cannot be used when scripting extended agent jobs.

User On Windows workstations, the user name specified in the `Logon` field of a job definition must have a matching user definition. The definitions provide the user passwords required by Tivoli Workload Scheduler to launch jobs.

Event rule
A scheduling event rule defines a set of actions that are to run upon the occurrence of specific event conditions. The definition of an event rule correlates events and triggers actions. When you define an event rule, you specify one or more events, a correlation rule, and the one or more actions that are triggered by those events. Moreover, you can specify validity dates, a daily time interval of activity, and a common time zone for all the time restrictions that are set.

You can control how jobs and job streams are processed with the following attributes:

Dependencies
Conditions that must be satisfied before a job or job stream can run. You can set the following types of dependencies:

- A predecessor job or job stream must have completed successfully.
- One or more specific resources must be available.
- Access to specific files must be granted.
- An affirmative response to a prompt.

Time constraints
Conditions based on time, such as:

- The time at which a job or job stream should start.
- The time after which a job or job stream cannot start.
- The repetition rate at which a job or job stream is to be run within a specified time slot.

Job priority
A priority system according to which jobs and job streams are queued for execution.

Job fence
A filter defined for workstations. Only jobs and job streams whose priority exceeds the job fence value can run on a workstation.

Limit Sets a limit to the number of jobs that can be launched concurrently on a workstation.

The production process

Tivoli Workload Scheduler production is based on a plan that runs in a *production period*. The production period is defined by the user when creating or extending the production plan. It can span from a few hours to some days (by default it is a 24 hours period). Before the start of each production period, Tivoli Workload Scheduler executes a program that creates the production plan starting from the modeling data stored in the database and from an intermediate plan called *preproduction plan*. Then another program includes the uncompleted schedules from the previous production period into the current plan and logs all the statistics of the previous production into an archive.

All of the required information for that production period is placed into a production control file named Symphony. During the production period, the production control database is continually being updated to reflect the work that needs to be done, the work in progress, and the work that has been completed. A copy of the Symphony file is sent to all subordinate domain managers and to all the fault-tolerant agents in the same domain. The subordinate domain managers distribute their copy to all the fault-tolerant agents in their domain and to all the domain managers that are subordinate to them, and so on down the line. This causes fault-tolerant agents throughout the network to continue processing even if the network connection to their domain manager is down. From the graphical interfaces or the command line interface, the operator can view and make changes in the current production by making changes in the Symphony file.

Tivoli Workload Scheduler processes monitor the production control database and make calls to the operating system to launch jobs as required. The operating system runs the job, and in return informs Tivoli Workload Scheduler if the job completed successfully or not. This information is entered into the production control database to indicate the status of the job.

Scheduling

Scheduling can be accomplished either through the Tivoli Workload Scheduler command line interface or one of the two graphical interfaces.

Scheduling includes the following tasks:

- Defining and maintaining workstations
- Defining scheduling objects
- Defining job streams
- Starting and stopping production processing
- Viewing and modifying jobs and job streams.

Defining scheduling objects

Scheduling objects are workstations, workstation classes, domains, jobs, job streams, resources, prompts, calendars, variables and variable tables, parameters, and event rules. Scheduling objects are managed with the Composer program and are stored in the Tivoli Workload Scheduler database. To create or modify an object, you can use either the Tivoli Workload Scheduler command line interface or one of the graphical interfaces.

Creating job streams

The primary processing task of Tivoli Workload Scheduler is to run job streams. A job stream is an outline of batch processing consisting of a list of jobs. Job streams can be defined using either the command line interface or one of the graphical interfaces. Using either graphical interface you can easily create and modify job streams. You can use their job stream editors to work with the jobs and the `follows` dependencies between the jobs, as well as the job stream run cycles. You can also easily specify time restrictions, resource dependencies, file dependencies, and prompt dependencies at the job stream level.

Job streams can be defined as *draft*. A draft job stream is not considered when resolving dependencies and is not added to the production plan. It becomes actual only after the *draft* keyword is removed from its definition, and the `JnextPlan` command is run to add it to the preproduction plan and so to the production plan.

Setting job recovery

When defining a job, consider that in some instances it might not complete successfully. The administrator can define a recovery option and recovery actions when defining the job. The following recovery options are available:

- Do not continue with the next job. This stops the execution of the job stream and puts it in the *stuck* state. This is the default action.
- Continue with the next job.
- Run the job again.

Optionally, a recovery prompt can be associated to the job. A recovery prompt is a local prompt to display when the job completes unsuccessfully. Processing does not continue until the prompt is answered affirmatively.

Another option is to define a recovery job that can be run in the place of the original job if it completes unsuccessfully. The recovery job must have been defined previously. Processing stops if the recovery job does not complete successfully.

Defining and managing mission-critical jobs

Job schedulers can use the Tivoli Workload Scheduler command line or the Dynamic Workload Console to flag jobs as mission-critical and specify their deadlines. A critical job and all its predecessors make up what is called a *critical network*. At planning time, Tivoli Workload Scheduler calculates the start time of the critical job and of each of its predecessors starting from the critical job deadline and estimated duration. While the plan runs, this information is dynamically kept up-to-date based on how the plan is progressing. If a predecessor, or the critical job itself, is becoming late, Tivoli Workload Scheduler automatically prioritizes its submission and promotes it to get more system resources and thus meet its deadline.

Within a critical network, Tivoli Workload Scheduler dynamically identifies the path of predecessors that is potentially most at risk; this is called the *critical path*. Tivoli Workload Scheduler calculates the level of risk that each critical job has of missing its deadline; a high risk indicates that the estimated end of the critical job is after its deadline while a potential risk indicates that some predecessors of the critical job have a warning condition, for example are late or in error.

The Dynamic Workload Console provides specialized views for tracking the progress of critical jobs and their predecessors. Job schedulers and operators can access the views from the Dashboard or by creating Monitor Critical Jobs tasks.

The initial view lists all critical jobs for the engine, showing the status: normal, potential risk, or high risk. From this view, an operator can navigate to see:

- The hot list of jobs that put the critical deadline at risk.
- The critical path.
- Details of all critical predecessors.
- Details of completed critical predecessors.
- Job logs of jobs that have already run.

Using the views, operators can monitor the progress of the critical network , find out about current and potential problems, release dependencies, and rerun jobs.

For example:

1. To flag a critical job and follow it up, the Job scheduler opens the Workload Designer on the Dynamic Workload Console, marks the specific job as critical, and sets the deadline for 5 a.m.

   When `JnextPlan` is run, the critical start dates for this job, and all the jobs that are identified as its predecessors, are calculated.
2. To track a specific critical job, the operator proceeds as follows:
   a. The operator checks the dashboards and sees that there are critical jobs scheduled on one of the engines.
   b. He clicks the link to get a list of the critical jobs.

      The specific job shows a `Potential Risk` status.
   c. He selects the job and clicks **Hot List** to see the predecessor job or jobs that are putting the critical job at risk.

      One of the predecessor jobs is listed as being in error.
   d. He selects the job and clicks **Job log**.

      The log shows that the job failed because of incorrect credentials for a related database.
   e. After discovering that the database password was changed that day, he changes the job definition in the symphony file and reruns the job.
   f. When he comes back to the dashboard, he notices that there are no longer any jobs in potential risk. Also, the critical jobs list that was opened when clicking on the potential risk link no longer shows the critical job after the job is rerun.
   g. The job is now running after being automatically promoted, getting higher priority for submission and system resources.
   h. No further problems need fixing and the critical job finally completes at 4.45 a.m.

Scheduling workload dynamically

You can choose to set Tivoli Workload Scheduler to dynamically associate your submitted workload (or part of it) to the best available resources at run time.

The Tivoli Workload Scheduler installation process includes the option to install the dynamic scheduling capability. If you select this option, you get the following functionality:

- Automatically discover scheduling environment resources
- Match job requirements to available resources
- Control and optimize use of resources
- Automatically follow resource changes
- Request additional resources when needed

You can submit Tivoli Workload Scheduler jobs, including jobs defined to run on extended agents, as well as J2EE applications (if you selected the option to schedule J2EE at installation time). To schedule workload dynamically, you:

1. Use the Dynamic Workload Console to define the agents you want to use for running workload as logical resources or groups of resources.
2. Update your Tivoli Workload Scheduler job definitions to make as destination CPU the dynamic workload broker workstation (this workstation works as a bridge between the scheduler engine and the pool of resources)
3. For every Tivoli Workload Scheduler job, add a JSDL (Job Submission Description Language) job definition where you match the job with required

resources, candidate hosts, and scheduling and optimization preferences. Use the Dynamic Workload Console to do this easily.

When a job is thus submitted, either as part of a job stream in the plan or through ad hoc submission, Tivoli Workload Scheduler checks the job requirements, the available resources and the related characteristics and submits the job to the resource that best meets the requirements.

Running production

Production consists of taking the definitions of the scheduling objects from the database, together with their time constraints and their dependencies, and building and running the production control file.

Running the plan

The production plan contains information about which jobs to run, on which fault-tolerant agent, and what dependencies must be satisfied before each job is launched. Tivoli Workload Scheduler creates the production plan starting from the modeling data stored in the database and from an intermediate plan called the preproduction plan. The preproduction plan is automatically created and managed by the product. To avoid problems, the database is locked during the generation of the plan and is unlocked when the generation completes or if an error condition occurs. The preproduction plan is used to identify in advance the job stream instances and the external follows job stream dependencies involved in a specified time period.

You use the `JnextPlan` command on the master domain manager to generate the production plan and distribute it across the Tivoli Workload Scheduler network.

To generate and start a new production plan, Tivoli Workload Scheduler performs the following steps:

1. Updates the preproduction plan with the objects defined in the database that were added or updated since the last time the plan was created or extended.
2. Retrieves from the preproduction plan the information about the job streams to run in the specified time period and saves it in an intermediate production plan.
3. Includes in the new production plan the uncompleted job streams from the previous production plan.
4. Creates the new production plan and stores it in a file named Symphony.
5. Distributes a copy of the Symphony file to the workstations involved in the new product plan processing.
6. Logs all the statistics of the previous production plan into an archive.
7. Updates the job stream state in the preproduction plan.

The copy of the newly-generated Symphony file is used starting from the top domain's fault-tolerant agents and domain managers of the child domains and down the tree to all subordinate domains.

Each fault-tolerant agent that receives the production plan can continue processing even if the network connection to its domain manager goes down.

At each destination fault-tolerant agent, Tivoli Workload Scheduler performs the following actions to manage job processing:

1. Accesses the copy of the Symphony file and reads the instructions about which job to run.
2. Makes calls to the operating system to launch jobs as required.
3. Updates its copy of the Symphony file with the job processing results and sends notification back up the tree to the master domain manager and to all full status fault-tolerant agents. The original copy of the Symphony file stored on the master domain manager and the copies stored on the backup master domain managers, if defined, are updated accordingly.

This means that during job processing, each fault-tolerant agent has its own copy of the Symphony file updated with the information about the jobs it is running (or that are running in its domain and child domains if the fault-tolerant agent is full-status or a domain manager), and the master domain manager (and backup master domain manager if defined) has the copy of the Symphony file that contains all updates coming from all fault-tolerant agents. In this way the Symphony file on the master domain manager is kept up-to-date with the jobs still to run, the jobs running, and the jobs already completed.

After the production plan is generated for the first time, it can be extended to the next production period with the `JnextPlan` command. The Symphony file is refreshed with the latest changes and redistributed throughout the network.

Running job streams

Depending on their run cycle definition, job streams are taken from the Tivoli Workload Scheduler database and automatically inserted into the current production plan.

While the job stream is in the plan, and has not completed, you can still modify any of its components. That is, you can modify the job stream properties, the properties of its jobs, their sequence, the workstation or resources they use, and so on, to satisfy last-minute requirements.

You can also hold, release, or cancel a job stream, as well as change the maximum number of jobs within the job stream that can run concurrently. You can change the priority previously assigned to the job stream and release the job stream from all its dependencies.

Last minute changes to the current production plan include the possibility to submit jobs and job streams that are already defined in the Tivoli Workload Scheduler database but were not included in the plan. You can also submit jobs that are being defined ad hoc. These jobs are submitted to the current plan but are not stored in the database.

Starting from version 8.3, you can create and manage multiple instances of the same job stream over a number of days or at different times within the same day. This new feature introduced the possibility to have in the same plan more than one instance of the same job stream with the same name. Each job stream instance is identified by the job stream name, the name of the workstation where it is scheduled to run, and by the start time defined in the preproduction plan.

Monitoring

Monitoring is done by listing plan objects in either graphical user interface. Using lists, you can see the status of all or of subsets of the following objects in the current plan:

- Job stream instances
- Job instances
- Domains
- Workstations
- Resources.
- File dependencies
- Prompt dependencies.

You can use these lists also to manage some of these objects. For example, you can reallocate resources, link or unlink workstations, kill jobs, or switch domain manager.

Additionally, you can monitor the daily plan with Tivoli Business Systems Manager, an object-oriented systems management application that provides monitoring and event management of resources, applications, and subsystems, that is integrated with Tivoli Workload Scheduler.

Network managers can use Tivoli Workload Scheduler/NetView, a NetView application, to monitor and diagnose Tivoli Workload Scheduler networks from a NetView management node. It includes a set of submaps and symbols to view Tivoli Workload Scheduler networks topographically, and determine the status of job scheduling activity and critical Tivoli Workload Scheduler processes on each workstation. Menu actions are provided to start and stop Tivoli Workload Scheduler processing and to run *conman* on any workstation in the network.

Controlling with IBM Tivoli Monitoring

IBM Tivoli Monitoring is a product that applies pre-configured best practices to the automated monitoring of essential system resources. It helps you to detect bottlenecks and other potential problems and provides you with the means for automatic recovery from critical situations. In this way it eliminates the need for system administrators to manually scan through extensive performance data.

Tivoli Workload Scheduler integrates with IBM Tivoli Monitoring. A set of IBM Tivoli Monitoring resource models, tailored to check the status of scheduling resources, is included with Tivoli Workload Scheduler.

By adding this set of resource models to the IBM Tivoli Monitoring default resource models set, you can add these resource models to monitoring profiles and distribute them to the profile subscribers where the scheduling resources are.

Within the monitoring profile you can define the following items:

- Which resource models you want to distribute, and, for each resource model:
  - When an alarm is triggered
  - Which response severity is assigned to the triggered alarm
  - Who is notified about the alarm and how
  - If a program is run in response to a triggered alarm
  - If events are to be sent to Tivoli Enterprise Console in response to triggered alarms
- Who are the subscribers of the monitoring profile distribution
- When the monitoring profile containing the resource model is active

In addition to these resource models a set of additional BAROC files is provided. These BAROC profiles are used to customize the TEC Event Server to manage the events triggered by the new set of resource models.

Reporting

As part of the pre-production and post-production processes, reports are generated which show summary or detail information about the previous or next production day. These reports can also be generated ad-hoc. The following reports are available:

- Job details listing
- Prompt listing
- Calendar listing
- Parameter listing
- Resource listing
- Job History listing
- Job histogram
- Planned production schedule
- Planned production summary
- Planned production detail
- Actual production summary
- Actual production detail
- Cross reference report

In addition, during production, a standard list file (STDLIST) is created for each job instance launched by Tivoli Workload Scheduler. Standard list files contain header and trailer banners, echoed commands, and errors and warnings. These files can be used to troubleshoot problems in job execution.

Auditing

An auditing option helps track changes to the database and the plan.

For the database, all user modifications, except for the delta of the modifications, are logged. If an object is opened and saved, the action is logged even if no modification is made.

For the plan, all user modifications to the plan are logged. Actions are logged whether or not they are successful.

Audit files are logged to a flat text file on individual machines in the Tivoli Workload Scheduler network. This minimizes the risk of audit failure due to network issues and allows a straightforward approach to writing the log. The log formats are basically the same for both the plan and the database. The logs consist of a header portion which is the same for all records, an “action ID”, and a section of data which varies according to the action type. All data is stored in clear text and formatted to be readable and editable from a text editor such as vi or notepad.

Using event-driven workload automation

Use this optional feature to set up and run rules that perform predefined actions in response to particular events occurring on your agents. Your organization can benefit from using this feature by adding on-demand workload automation to plan-based job scheduling, gaining savings in time and resources.

Event-driven workload automation is based on the concept of event rule. In Tivoli Workload Scheduler an event rule is a scheduling object and is made up of events, event-correlating conditions, and actions. When you define an event rule, you specify one or more events, a correlation rule, and one or more actions that are triggered by those events. Moreover, you can specify validity dates, a daily time interval of activity, and a common time zone for all the time restrictions that are set.

You can set up event rules to:

- Trigger the execution of batch jobs and job streams based on the occurrence or combination of real time events
- Reply to prompts
- Notify users when anomalous conditions occur in the Tivoli Workload Scheduler scheduling environment or batch scheduling activity
- Invoke an external product when a particular event condition occurs

Tivoli Workload Scheduler includes a set of predefined event and action plug-ins, but also provides a software development kit with samples and templates for your application programmers to develop their own plug-ins.

Options and security

The Tivoli Workload Scheduler options files determine how Tivoli Workload Scheduler runs on your system. Several performance, tuning, security, logging, and other configuration options are available.

Setting the Tivoli Workload Scheduler options

You can set two types of properties to configure your Tivoli Workload Scheduler runtime environment, properties that are set on the master domain manager and affect processing on all workstations in the Tivoli Workload Scheduler network, and properties that are set locally on a workstation and affect processing on that workstation only. The former are managed using the Tivoli Workload Scheduler command line program named `optman`, and the latter you define locally on the workstation by customizing the configuration files `useropts`, `localopts`, and `jobmanrc`.

Global options are used to:

- Define if the security files of all the workstations of the network can be created and managed only from the master domain manager or if the root user or administrator of each workstation can create and manage their own.
- Select whether to enable or disable database auditing.
- Control which objects in the plan the user is permitted to list when running a query.
- Select whether to enable plan auditing.
- Select whether to enable strong encryption.
- Select whether to enable or disable the fault tolerant switch manager.
- Select whether to enable or disable the time zone option.
- Enter the number of days for which you want to save job statistics.
- Set the minimum and maximum lengths of the preproduction plan in days.
- Determine if uncompleted job streams are carried forward from the old to the new production control file.
- Define the start time of the Tivoli Workload Scheduler processing day.

Local options are used to:

- Specify the name of the local workstation
- Prevent the launching of jobs run by root in UNIX
- Prevent unknown clients from connecting to the system
- Specify a number of performance options
- Specify a number of logging preferences
- Set SSL security options.

Setting security

Security is accomplished with the use of a security file that contains one or more user definitions. Each user definition identifies a set of users, the objects they are permitted to access, and the types of actions they can perform.

A template file is installed with the product. Edit the template to create the user definitions and compile and install it with a utility program to create a new operational security file. After it is installed, you make further modifications by creating an editable copy with another utility.

Each workstation in a Tivoli Workload Scheduler network has its own security file. An individual file can be maintained on each workstation, or a single security file can be created on the master domain manager and copied to each domain manager, fault-tolerant agent, and standard agent.

Secure authentication and encryption

Security is enhanced for connections between protected and non-protected domains by applying the authentication and encryption mechanism based on the Secure Sockets Layer (SSL) protocol. SSL uses digital certificates to authenticate the identity of a workstation.

The Tivoli Workload Scheduler administrator must plan how authentication will be used within the network:

- Use one certificate for the entire Tivoli Workload Scheduler network.
- Use a separate certificate for each domain.
- Use a separate certificate for each workstation.

SSL support is automatically installed with Tivoli Workload Scheduler.

Work across firewalls

For previous versions of Tivoli Workload Scheduler, running the commands to start or stop a workstation or to get the standard list required opening a direct TCP/IP connection between the originator and the destination nodes. In a firewall environment, this forces users to break the firewall to open a direct communication path between the master and each fault-tolerant agent in the network.

Tivoli Workload Scheduler provides a configurable attribute, named *behindfirewall*, in the workstation's definition in the database. You can set this attribute to ON to indicate that a firewall exists between that particular workstation and its domain manager, and that the link between the domain manager and the workstation (which can be another domain manager) is the only allowed link between the domains.

Also, for all the workstations having this attribute set to ON, the commands to start or stop the workstation or to get the standard list will be transmitted through

the domain hierarchy instead of opening a direct connection between the master (or domain manager) and the workstation.

Centralized security mechanism

A new global option makes it possible to change the security model in the Tivoli Workload Scheduler network. If you use this option, then the security files for the fault-tolerant agents in the network can be created or modified only on the master domain manager. The Tivoli Workload Scheduler administrator is responsible for creating, updating, and distributing the security files for all the agents where user access is required. Setting this global option triggers a security mechanism to identify and trust the Tivoli Workload Scheduler network corresponding to that master domain manager.

If you prefer the traditional security model, you can still use it by not activating the global variable.

Using time zones

Tivoli Workload Scheduler supports different time zones. Enabling time zones provides you with the ability to manage your workload across a multiple time zone environment. Both the 3-character and the variable length notations are supported with the current version of Tivoli Workload Scheduler. The variable length notation format is area/city, for example Europe/Paris as equivalent to ECT (European Central Time). The 3-character notation is supported for backward compatibility with previous versions of the product.

Once configured, time zones can be specified for start and deadline times within jobs and job streams.

Using extended agents

With IBM Tivoli Workload Scheduler for Applications, extended agents (XA) are used to extend the job scheduling functions of Tivoli Workload Scheduler to other systems and applications. An extended agent is defined as a workstation that has a host and an access method.

The host is a Tivoli Workload Scheduler fault-tolerant agent (FTA) or standard agent (SA).

The access method is a program that is run by the hosting workstation whenever Tivoli Workload Scheduler, either through its command line or either graphical interface, interacts with the external system. IBM Tivoli Workload Scheduler for Applications includes the following access methods:

- Oracle E-Business Suite access method (MCMAGENT)
- PeopleSoft access method (psagent)
- R/3 access method (r3batch)
- z/OS access method (mvsjes and mvsopc)

To launch and monitor a job on an extended agent, the host runs the access method, passing it job details as command line options. The access method communicates with the external system to launch the job and returns the status of the job.

An extended agent workstation is a logical entity related to an access method hosted by the physical Tivoli Workload Scheduler workstation (a fault-tolerant

agent or standard agent). More than one extended agent workstation can be hosted by the same Tivoli Workload Scheduler workstation and rely on the same access method. The extended agent is defined in a standard Tivoli Workload Scheduler workstation definition, which gives the extended agent a name and identifies the access method.

Figure 5 shows how these elements fit together in the case of a typical extended agent configuration.

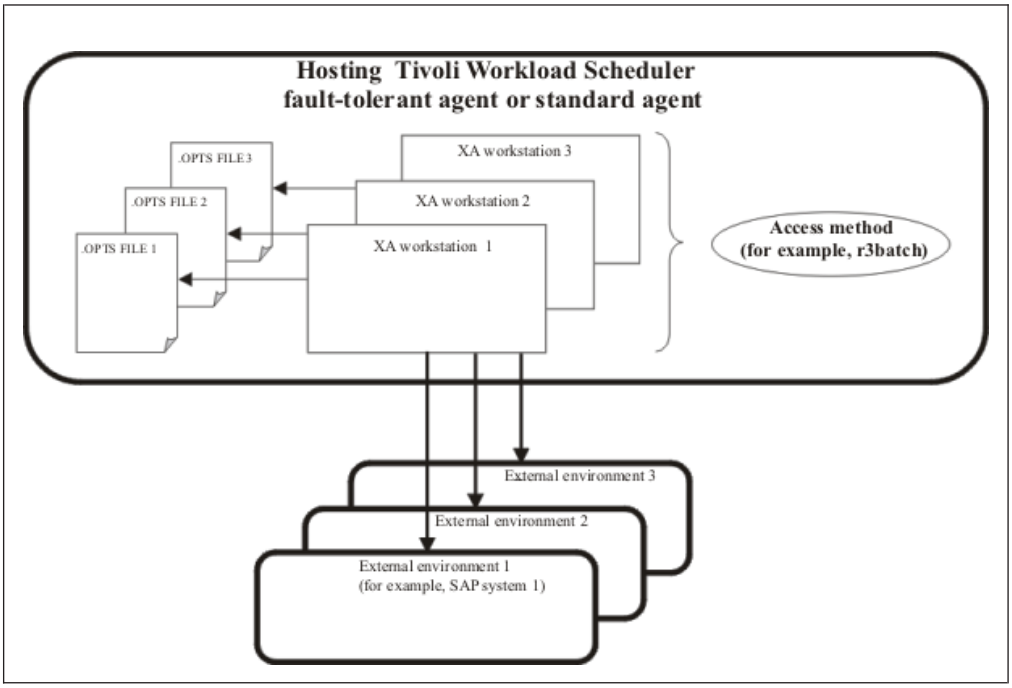


Figure 5. How extended agents work

To launch a job in an external environment, Tivoli Workload Scheduler runs the extended agent access method, providing it with the extended agent workstation name and information about the job. The method looks at the corresponding file named *<WORKSTATIONNAME>_<methodname>*.opts to determine which external environment instance it will connect to. The access method can then launch jobs on that instance and monitor them through completion writing job progress and status information in the standard list file of the job.

Extended agents can be used to run jobs also in an end-to-end environment, where their scheduling and monitoring is performed from a Tivoli Workload Scheduler for z/OS controller.

Chapter 7. Tivoli Workload Scheduler for z/OS

Tivoli Workload Scheduler for z/OS expands the scope for automating your data processing (DP) operations. It plans and automatically schedules the production workload. From a single point of control, it drives and controls the workload processing at both local and remote sites. By using Tivoli Workload Scheduler for z/OS to increase automation, you use your DP resources more efficiently, have more control over your DP assets, and manage your production workload processing better.

How your production workload is managed

How does Tivoli Workload Scheduler for z/OS give you all this? This section describes functions that make your information systems (IS) operations run more efficiently. But first, here is a brief introduction to the structure of the product and some concepts.

Structure

Tivoli Workload Scheduler for z/OS consists of a base product, the *agent* and a number of features. Every z/OS system in your complex requires the base product. One z/OS system in your complex is designated the *controlling* system and runs the *engine* feature. Only one engine feature is required, even when you want to start standby engines on other z/OS systems in a sysplex.

Tivoli Workload Scheduler for z/OS with Tivoli Workload Scheduler addresses your production workload in the distributed environment. You can schedule, control, and monitor jobs in Tivoli Workload Scheduler from Tivoli Workload Scheduler for z/OS. For example, in the current plan, you can specify jobs to run on workstations in Tivoli Workload Scheduler.

The workload on other operating environments can also be controlled with the open interfaces provided with Tivoli Workload Scheduler for z/OS. Sample programs using TCP/IP or an NJE/RSCS (network job entry/remote spooling communication subsystem) combination show you how you can control the workload on environments that at present have no scheduling feature.

Additionally, national language features let you see the dialogs and messages, in the language of your choice. These languages are currently available:
- English
- German
- Japanese
- Spanish

Panel and message text can also be modified to include enterprise-specific instructions or help.

Concepts

In managing production workloads, Tivoli Workload Scheduler for z/OS builds on several important concepts.

Plans. Tivoli Workload Scheduler for z/OS constructs operating *plans* based on user-supplied descriptions of the DP operations department and its production

workload. These plans provide the basis for your service level agreements and give you a picture of the status of the production workload at any point in time. You can simulate the effects of changes to your production workload, calendar, and installation by generating trial plans.

Job streams. A *job stream* is a description of a unit of production work. It can include the following:

- A list of the *jobs* (related tasks) associated with that unit of work, such as:
  - Data entry
  - Job preparation
  - Job submission or started-task initiation
  - Communication with the NetView program
  - File transfer to other operating environments
  - Printing of output
  - Postprocessing activities, such as quality control or dispatch
  - Other tasks related to the unit of work that you want to schedule, control, and track
- A description of dependencies between jobs within a job stream and between jobs in other job streams
- Information about resource requirements, such as exclusive use of a data set
- Special operator instructions that are associated with a job
- How and where each job should be processed
- Run policies for that unit of work; that is, when it should be scheduled or alternatively the name of a group definition that records the run policy

Tivoli Workload Scheduler for z/OS schedules work based on the information you provide in your job stream descriptions.

Workstations. When scheduling and processing work, Tivoli Workload Scheduler for z/OS considers the processing requirements of each job. Some typical processing considerations are:

- Which human or machine resources are required for processing the work, for example, operators, processors, or printers?
- When are these resources available?
- How are these jobs to be tracked?
- Can this work be processed somewhere else if the resources become unavailable?

Tivoli Workload Scheduler for z/OS supports a range of work process types, called *workstations*, that map the processing requirements of any task in your 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
- Communication with the NetView program
- Print jobs
- Manual preprocessing or postprocessing activity

You can plan for maintenance windows in your hardware and software environments. Tivoli Workload Scheduler for z/OS helps you perform a controlled and incident-free shutdown of the environment, preventing last-minute

cancellation of active tasks. You can choose to reroute the workload automatically during any outage, planned or unplanned.

Tivoli Workload Scheduler 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. You can view or modify this status information online using the workstation ready lists in the dialog.

Virtual Workstations.Using virtual workstations improves workload balancing and the monitoring of system availability. This feature automatically directs the submission of workload to different destinations removing the need to associate a workstation to a specific destination. You can define a list of destinations for the submission of workload and the scheduler distributes the workload to automatically-selected active destinations, according to a round-robin scheduling approach.

You can activate this feature by specifying the new virtual option at workstation definition level. This option is allowed for computer workstations with the automatic reporting attribute, and is supported by all the interfaces available to define, modify, and monitor workstations.

Using virtual workstations the scheduler distributes the workload across your trackers evenly, thus avoiding bottlenecks when submitting or running jobs. In fact, the scheduler splits the workload among the available destinations, so that the Job Entry System (JES) and Workload Manager (WLM) do not find overloaded input queues when selecting jobs for their action.

Dependencies. In general, every DP-related activity must occur in a specific order. Activities performed out of order might create invalid output and possibly even corrupt your corporate data. This might cause costly reruns, missed deadlines, and unsatisfied customers.

You can define *dependencies* for jobs when a specific processing order is required. When Tivoli Workload Scheduler for z/OS manages the dependent relationships for you, the jobs are always started in the correct order every time they are scheduled. A dependency is called *internal* when it is between two jobs in the same job stream, and *external* when it is between two jobs in different job streams.

When specifying dependencies, you can use both return code and status of an operation to determine the starting of another operation. Standard logical operators are supported to define the check on status or return code values, to implement dependencies definition with a *conditional logic*. If the predecessor operation is associated to a job with different steps, you can specify a conditional *step-level dependency* on individual step return codes.

Tivoli Workload Scheduler for z/OS lets you serialize work based on the status of any DP resource. A typical example is a job that uses a data set as input, but must not start until the data set is successfully created and loaded with valid data. You can use *resource serialization* support to send availability information about a DP resource to Tivoli Workload Scheduler for z/OS.

Special resources. Special resources are typically defined to represent physical or logical objects used by jobs. A special resource can be used to serialize access to a data set or to limit the number of file transfers on a particular network link. The resource does not have to represent a physical object in your configuration, although it often does.

Tivoli Workload Scheduler for z/OS keeps a record of the state of each resource and its current allocation status. You can choose to hold resources if a job allocating the resources ends abnormally. You can also use the Tivoli Workload Scheduler for z/OS interface with the Resource Object Data Manager (RODM) to schedule jobs according to *real* resource availability. You can subscribe to RODM updates in both local and remote domains.

Tivoli Workload Scheduler for z/OS lets you *subscribe* to data set activity on z/OS systems. The data set triggering function of Tivoli Workload Scheduler for z/OS automatically updates special resource availability when a data set is closed. You can use this notification to coordinate planned activities or to add unplanned work to the schedule.

Calendars. Tivoli Workload Scheduler for z/OS uses information about when the job departments work, so that job streams are not scheduled to run on days when processing resources are not available (for example, Sundays and holidays). This information is stored in a *calendar*. Tivoli Workload Scheduler for z/OS supports *multiple calendars* for enterprises where different departments have different work days and non-working days. Different groups within a business operate according to different calendars.

The multiple calendar function is critical if your enterprise has installations in more than one geographical location (for example, with different local or national holidays).

Business processing cycles. Tivoli Workload Scheduler for z/OS uses business processing cycles, or *periods*, to calculate when your job streams run, for example, weekly, or every 10th working day. Periods are based on the business cycles of your customers. Tivoli Workload Scheduler for z/OS supports a range of periods for processing the different job streams in your production workload.

When you define a job stream, you specify when it is planned to run, using a *run cycle*, which can be:

- A *rule* with a format such as

  ```
  ONLY the SECOND TUESDAY of every MONTH
  EVERY FRIDAY in the user-defined period SEMESTER1
  ```

  where the words in upper case are selected from lists of ordinal numbers, names of days, and common calendar intervals or period names.
- A combination of period and *offset*. For example, an offset of 10 in a monthly period specifies the 10th day of each month.

Using Plans in Tivoli Workload Scheduler for z/OS

Tivoli Workload Scheduler for z/OS plans your production workload schedule. It produces both high-level and detailed plans. These plans both drive the production workload and show you the status of the production workload on your system at any specified time. You can produce trial plans to forecast future workloads.

Long-term planning

The *long-term plan* is a high-level schedule of your anticipated production workload. It lists, by day, the instances of job streams to be run during the period of the plan. Each instance of a job stream is called an *occurrence*. The long-term plan shows when occurrences are to run, as well as the dependencies that exist between the job streams. You can view these dependencies graphically on your workstation as a network, to check that work has been defined correctly. The plan

can help you in forecasting and planning for heavy processing days. The long-term-planning function can also produce histograms showing planned resource use for individual workstations during the plan period.

You can use the long-term plan as the basis for documenting your service level agreements. It lets you relate service level agreements directly to your production workload schedules so that your customers can see when and how their work is to be processed.

The long-term plan provides a window to the future. You can decide how far into the future, from one day to four years. You can also produce long-term plan simulation reports for *any* future date. Tivoli Workload Scheduler for z/OS can automatically extend the long-term plan at regular intervals. You can print the long-term plan as a report, or you can view, alter, and extend it online using the dialogs.

Detailed planning

The *current plan* is the center of Tivoli Workload Scheduler for z/OS processing. It drives the production workload automatically and provides a way to check its status. The current plan is produced by the run of batch jobs that extract from the long-term plan the occurrences that fall within the specified period of time from the job details. The current plan selects a window from the long-term plan and makes the jobs ready to be run. They are started depending on the decided restrictions (for example, dependencies, resources availability, or time-dependent jobs).

The current plan is a rolling plan that can cover several days. A common method is to cover 1 to 2 days with regular extensions each shift. Production workload processing activities are listed by minute.

You can either print the current plan as a report, or view, alter, and extend it online, by using the dialogs.

Automatically controlling the production workload

Tivoli Workload Scheduler for z/OS automatically drives the production workload by monitoring the flow of work and by directing the processing of jobs to follow the business priorities established in the plan.

Through its interface to the NetView program or its management-by-exception ISPF dialog, Tivoli Workload Scheduler for z/OS can alert the production control specialist to problems in the production workload processing. Furthermore, the NetView program can automatically trigger Tivoli Workload Scheduler for z/OS to perform corrective actions in response to these problems.

Tivoli Workload Scheduler for z/OS automatically:

- Starts and stops started tasks
- Edits job statements: z/OS JCL or equivalent job statements for other operating environments before submission
- Submits jobs in the specified sequence to the target operating environment every time
- Tracks each scheduled job in the plan
- Determines the success or failure of the jobs
- Displays status information and instructions to guide workstation operators

- Provides automatic recovery of jobs when they end in error, regardless of the operating environment
- Generates processing dates for your job stream run cycles using rules, such as:
  - Every second Tuesday of the month
  - Only the last Saturday in June, July, and August
  - Every third workday in the user-defined PAYROLL period
- Starts jobs with regard to real resource availability
- Performs data set cleanup in error and rerun situations for the z/OS workload
- Tailors the JCL for step restarts of z/OS jobs and started tasks
- Dynamically schedules additional processing in response to unplannable activities
- Provides automatic notification when an updated data set is closed; this can be used to trigger subsequent processing
- Generates alerts when abnormal situations are detected in the workload

Tivoli Workload Scheduler for z/OS also provides manual control facilities, which are described in "Manual control and intervention" on page 72.

Automatic workload submission

Tivoli Workload Scheduler for z/OS automatically drives work through the system, taking into account work that requires manual or program-recorded completion. Program-recorded completion refers to situations where the status of a scheduler-controlled job is set to "complete" by a user-written program. It also promotes the optimum use of resources, improves system availability, and automates complex and repetitive operator tasks. Tivoli Workload Scheduler for z/OS automatically controls the submission of work according to:

- Dependencies between jobs
- Workload priorities
- Specified time for the submission of particular work
- Availability of resources

By saving a copy of the JCL for each separate run, or occurrence, of a particular job in its plans, Tivoli Workload Scheduler for z/OS prevents the unintentional reuse of temporary JCL changes, such as overrides.

Job tailoring. Tivoli Workload Scheduler for z/OS provides automatic job tailoring functions to automatically edit jobs. This can reduce your dependency on time-consuming and error-prone manual editing of jobs. Tivoli Workload Scheduler for z/OS job tailoring provides:

- Automatic variable substitution
- Dynamic inclusion and exclusion of inline job statements
- Dynamic inclusion of job statements from other libraries or from an exit

For jobs submitted on a z/OS system, these job statements are z/OS JCL. Scheduler JCL tailoring directives can be included in jobs that are submitted on other operating systems, such as AIX®/6000.

Variables can be substituted in specific columns, and you can define verification criteria to ensure that invalid strings are not substituted. Special directives supporting a variety of date formats used by job stream programs let you dynamically define the required format and change the multiple times for the same job. You can define arithmetic expressions to calculate values such as the current date plus four work days. And you can set a temporary variable to a specific value or to an expression composed of other temporary variables.

System Automation commands tailoring. Tivoli Workload Scheduler for z/OS provides a function that edits system automation commands automatically. This helps you to save time and reduce the possibility of editing errors. Tivoli Workload Scheduler for z/OS command tailoring provides automatic variable substitution.

Automatic recovery and restart

Tivoli Workload Scheduler for z/OS provides automatic restart facilities for your production work. You can specify the restart actions to take if work initiated by Tivoli Workload Scheduler for z/OS ends in error (see Figure 6.) You can use these functions to predefine automatic error recovery and restart actions for jobs and started tasks. The scheduler's integration with the NetView program allows it to automatically pass alerts to the NetView program in error situations. Using the z/OS cross-system coupling facility (XCF) enables Tivoli Workload Scheduler for z/OS to maintain production workload processing when system failures occur.

Recovery of jobs and started tasks. Automatic recovery actions for failed jobs are specified in user-defined control statements. Parameters in these statements determine the recovery actions to be taken when a job or started task ends in error.

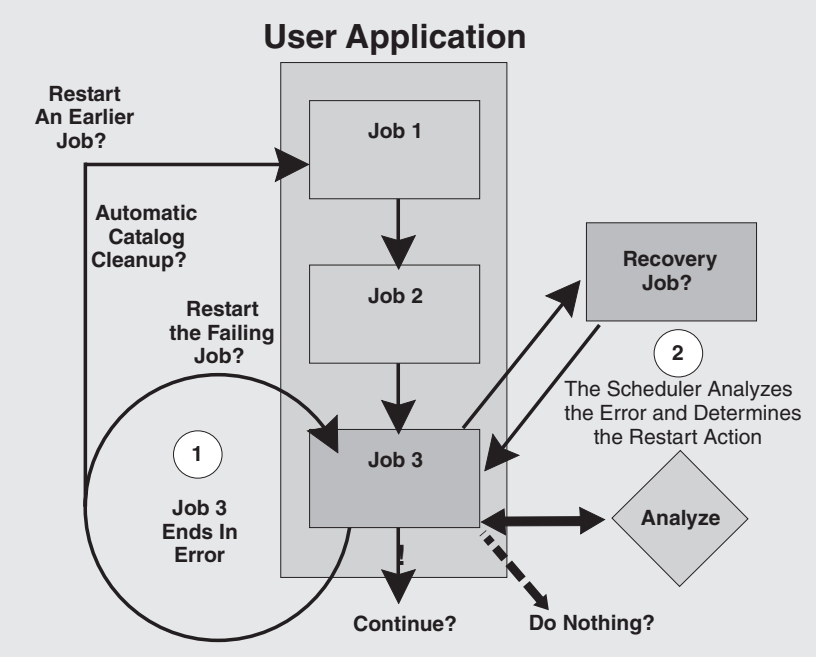


Figure 6. Automatic recovery and restart

Restart and cleanup. You can use restart and cleanup to catalog, uncatalog, or delete data sets when a job ends in error, or when you need to rerun a job. Data set cleanup handles JCL in the form of in-stream JCL, in-stream procedures, and cataloged procedures on both local and remote systems. This function can be initiated automatically by Tivoli Workload Scheduler for z/OS or manually by using the panels. Tivoli Workload Scheduler for z/OS resets the catalog to the status that it was before the job ran for both generation data set groups (GDGs) and for DD allocated data sets contained in JCL. In addition, restart and cleanup supports the use of Removable Media Manager in your environment.

Restart at both the step- and job-level is also provided in the Tivoli Workload Scheduler for z/OS panels. It manages resolution of generation data group (GDG)

names, JCL containing nested INCLUDEs or PROC, and IF-THEN-ELSE statements. Tivoli Workload Scheduler for z/OS also automatically identifies problems that can prevent successful restart, providing a logic of the "best restart step."

You can browse the job log or request a step-level restart for any z/OS job or started task even when there are no catalog modifications. The job-log browse functions are also available for the workload on other operating platforms, which is especially useful for those environments that do not support an SDSF-like facility. If you use a SYSOUT archiver, for example RMDS, you can interface with it from Tivoli Workload Scheduler for z/OS and so prevent duplication of job log information.

These facilities are available to you without the need to make changes to your current JCL.

Tivoli Workload Scheduler for z/OS gives you an enterprise-wide data set cleanup capability on remote agent systems.

Production workload restart. Tivoli Workload Scheduler for z/OS provides a production workload restart, which can automatically maintain the processing of your work if a system or connection fails. Scheduler-controlled production work for the unsuccessful system is rerouted to another system. Because Tivoli Workload Scheduler for z/OS can restart and manage the production workload, the integrity of your processing schedule is maintained, and service continues for your customers.

Tivoli Workload Scheduler for z/OS uses the VTAM® Model Application Program Definition feature and the z/OS defined symbols to ease the configuration and job in a sysplex environment, giving you a single system view of the sysplex.

Starting, stopping, and managing your engines and agents do not require you to know on which sysplex z/OS image they are actually running on.

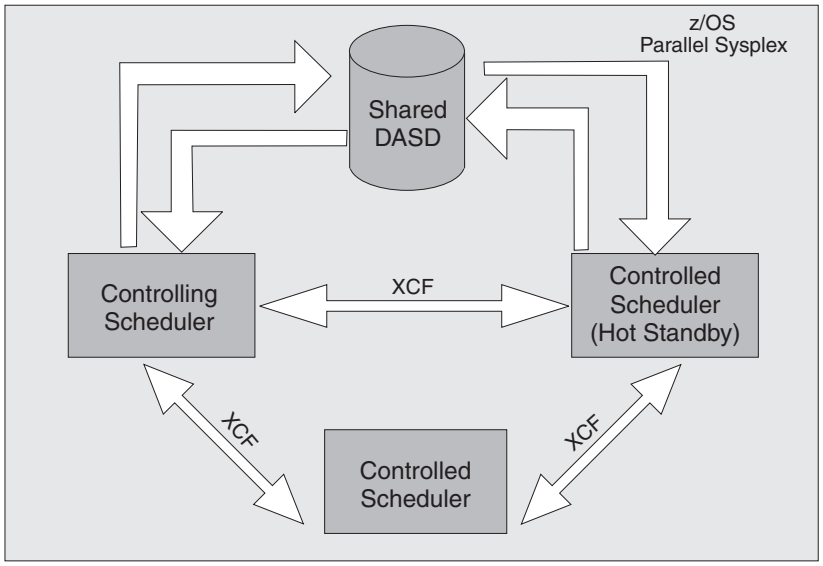


Figure 7. Production workload restart and hot standby

Hot standby. Tivoli Workload Scheduler for z/OS provides a single point of control for your z/OS production workload. If this controlling system fails, Tivoli Workload Scheduler for z/OS can automatically transfer the controlling functions to a backup system within a Parallel Sysplex®, see Figure 7 on page 70. Through XCF, Tivoli Workload Scheduler for z/OS can automatically maintain production workload processing during system or connection failures.

z/OS automatic restart manager support

All the scheduler components are enabled to be restarted by the Automatic Restart Manager (ARM) of the z/OS operating system, in the case of program failure.

Workload Manager (WLM) support

With Workload Manager (WLM), you can make the best use of resources accessed by your scheduled jobs. In addition, your jobs maintain the highest possible throughput with WLM and Tivoli Workload Scheduler for z/OS. When used with WLM, the scheduler can achieve the best possible system response times.

Automatic status checking

To track the work flow, Tivoli Workload Scheduler for z/OS interfaces directly with the operating system, collecting and analyzing status information about the production work that is currently active in the system. Tivoli Workload Scheduler for z/OS can record status information from both local and remote processors. When status information is reported from remote sites in different time zones, Tivoli Workload Scheduler for z/OS makes allowances for the time differences.

Status reporting from heterogeneous environments

The processing on other operating environments can also be tracked by Tivoli Workload Scheduler for z/OS. You can use supplied programs to communicate with the engine from any environment that can establish communications with a z/OS system.

Status reporting from user programs

You can pass status information about production workload processing to Tivoli Workload Scheduler for z/OS from your own user programs through a standard supplied routine.

Additional job-completion checking

If required, Tivoli Workload Scheduler for z/OS provides further status checking by scanning SYSOUT and other print data sets from your processing when the success or failure of the processing cannot be determined by completion codes. For example, Tivoli Workload Scheduler for z/OS can check the text of system messages or messages originating from your user programs. Using information contained in job completion checker (JCC) tables, Tivoli Workload Scheduler for z/OS determines what actions to take when it finds certain text strings. These actions can include:

- Reporting errors
- Requeuing SYSOUT
- Writing incident records to an incident data set

Managing unplanned work

Tivoli Workload Scheduler for z/OS can be automatically triggered to update the current plan with information about work that cannot be planned in advance. This allows Tivoli Workload Scheduler for z/OS to control unexpected work. Because Tivoli Workload Scheduler for z/OS checks the processing status of this work, automatic recovery facilities are also available.

Interfacing with other programs

Tivoli Workload Scheduler for z/OS provides a program interface (PIF). Using this interface, you can automate most actions that you can perform online through the dialogs. This interface can be called from CLISTs, user programs, and using TSO commands.

The application programming interface (API) lets your programs communicate with Tivoli Workload Scheduler for z/OS from any compliant platform. You can use Common Programming Interface for Communications (CPI-C), advanced program-to-program communication (APPC), or your own logical unit (LU) 6.2 verbs to converse with Tivoli Workload Scheduler for z/OS through the API. You can use this interface to query and update the current plan. The programs can be running on any platform that is connected locally, or remotely through a network, with the z/OS system where the engine runs.

Manual control and intervention

Tivoli Workload Scheduler for z/OS lets you check the status of work and intervene manually when priorities change or when you want to run unplanned work. You can query the status of the production workload and then modify the schedule if needed.

Status inquiries

With the ISPF dialogs, you can make queries online and receive timely information on the status of the production workload.

Time information that is displayed by the dialogs is in the local time of the dialog user. Using the dialogs, you can request detailed or summary information on individual job streams, jobs, and workstations, as well as summary information concerning workload production as a whole. You can also display dependencies graphically as a network at both job stream and job level. Status inquiries:

- Provide you with overall status information that you can use when considering a change in workstation capacity or when arranging an extra shift or overtime work.
- Help you supervise the work flow through the installation; for example, by displaying the status of work at each workstation.
- Help you decide whether intervention is required to speed up the processing of specific job streams. You can find out which job streams are the most critical. You can also check the status of any job stream, as well as the plans and actual times for each job.
- Help you to check information before making modifications to the plan. For example, you can check the status of a job stream and its dependencies before deleting it or changing its input arrival time or deadline. See “Modifying the current plan” for more information.
- Provide you with information on the status of processing at a particular workstation. Perhaps work that should have arrived at the workstation has not arrived. Status inquiries can help you locate the work and find out what has happened to it.

Modifying the current plan

Tivoli Workload Scheduler for z/OS makes status updates to the plan automatically, using its tracking functions. However, it lets you change the plan manually to reflect unplanned changes to the workload or to the operations environment, which often occur during a shift. For example, you might want to change the priority of a job stream, add unplanned work, or reroute work from

one workstation to another. You might also want to correct operational errors manually. Modifying the current plan might be the best way to handle these situations.

You can modify the current plan online. For example, you can:

- Include unexpected jobs or last-minute changes to the plan. Tivoli Workload Scheduler for z/OS then automatically creates the dependencies for this work.
- Manually modify the status of jobs.
- Delete occurrences of job streams.
- Graphically display job dependencies before you modify them.
- Modify the data in job streams, including the JCL.
- Respond to error situations by:
  - Rerouting jobs
  - Rerunning jobs or occurrences
  - Completing jobs or occurrences
  - Changing jobs or occurrences
- Change the status of workstations by:
  - Rerouting work from one workstation to another
  - Modifying workstation reporting attributes
  - Updating the availability of resources
  - Changing the way resources are handled
- Replan or extend the current plan

In addition to using the dialogs, you can modify the current plan from your own job streams using the program interface or the application programming interface. You can also trigger Tivoli Workload Scheduler for z/OS to dynamically modify the plan using TSO commands or a batch program. This adds unexpected work automatically to the plan.

Management of critical jobs

Tivoli Workload Scheduler for z/OS uses the capability of the Workload Manager component of z/OS to ensure that critical jobs are completed on time. If a critical job is late, Tivoli Workload Scheduler for z/OS favors it using the Workload Manager interface.

Management of critical path

In addition to the handling of critical jobs based on Workload Manager, Tivoli Workload Scheduler for z/OS provides the dynamic handling of the critical path calculated by the daily planning batch jobs process.

The critical path is the path, within a network of jobs, with the least slack time.

The slack time, in a critical job predecessor path, is the amount of time that processing of the predecessor jobs can be delayed without exceeding the deadline of a critical job. It is the spare time calculated using the deadline, input arrival, and duration settings of predecessor jobs.

The capabilities include:

- Monitoring of critical job predecessors that are late, long running, or ended with an error. This process uses the same internal logic that the scheduler applies to monitor alert conditions.
- Monitoring of the paths that are consuming their slack time, becoming more critical than the paths calculated at plan generation.

- Enhanced critical jobs monitoring, using ISPF dialog flows.
- Back-end support for new views available using the Dynamic Workload Console.

Security

Today, DP operations increasingly require a high level of data security, particularly as the scope of DP operations expands and more people within the enterprise become involved. Tivoli Workload Scheduler for z/OS provides complete security and data integrity within the range of its functions. It provides a shared central service to different user departments even when the users are in different companies and countries. Tivoli Workload Scheduler for z/OS provides a high level of security to protect scheduler data and resources from unauthorized access. With Tivoli Workload Scheduler for z/OS, you can easily organize, isolate, and protect user data to safeguard the integrity of your end-user applications (see Figure 8 on page 74). Tivoli Workload Scheduler for z/OS can plan and control the work of many user groups, and maintain complete control of access to data and services.

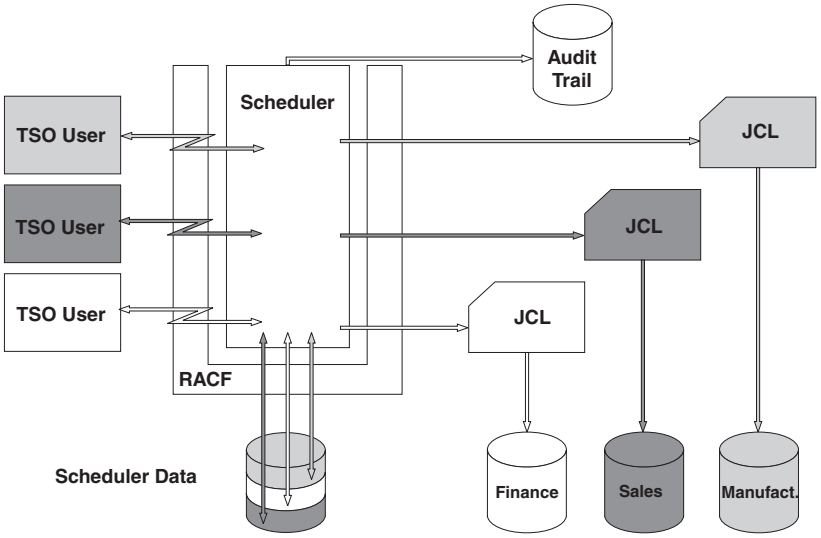


Figure 8. Security

Audit trail

With the audit trail, you can define how you want Tivoli Workload Scheduler for z/OS to log accesses (both reads and updates) to scheduler resources. Because it provides a history of changes to the databases, the audit trail can be extremely useful for staff that work with debugging and problem determination.

A sample program is provided for reading audit-trail records. The program reads the logs for a period that you specify and produces a report detailing changes that have been made to scheduler resources.

System authorization facility

Tivoli Workload Scheduler for z/OS uses the system authorization facility (SAF), a function of z/OS, to pass authorization verification requests to your security system, for example RACF. This means that you can protect your scheduler data objects with any security system that uses the SAF interface.

Protection of data and resources: Each user request to access a function or to access data is validated by SAF. This is some of the information that can be protected:

- Calendars and periods
- Job stream names or job stream owner, by name
- Workstation, by name
- Job stream-specific data in the plan
- Operator instructions
- JCL

To support distributed, multi-user handling, Tivoli Workload Scheduler for z/OS lets you control the level of security you want to implement, right down to the level of individual records. You can define generic or specific RACF resource names to extend the level of security checking.

If you have RACF Version 2 Release 1 installed, you can use the Tivoli Workload Scheduler for z/OS reserved resource class to manage your Tivoli Workload Scheduler for z/OS security environment. This means that you do not have to define your own resource class by modifying RACF and restarting your system.

Data integrity during submission: Tivoli Workload Scheduler for z/OS can ensure the correct security environment for each job it submits, regardless of whether the job is run on a local or a remote system. Tivoli Workload Scheduler for z/OS lets you create tailored security profiles for individual jobs or groups of jobs.

Configurations of Tivoli Workload Scheduler for z/OS

Tivoli Workload Scheduler for z/OS supports many configuration options using a variety of communication methods:

- The controlling system
- Controlled z/OS systems
- Remote panels and program interface applications
- Scheduling jobs that are in Tivoli Workload Scheduler

The controlling system

The controlling system requires both the agent and the engine. One controlling system can manage the production workload across all your operating environments.

The engine is the focal point of control and information. It contains the controlling functions, the dialogs, and the scheduler's own batch programs. Only one engine is required to control the entire installation, including local and remote systems (see Figure 9 on page 76).

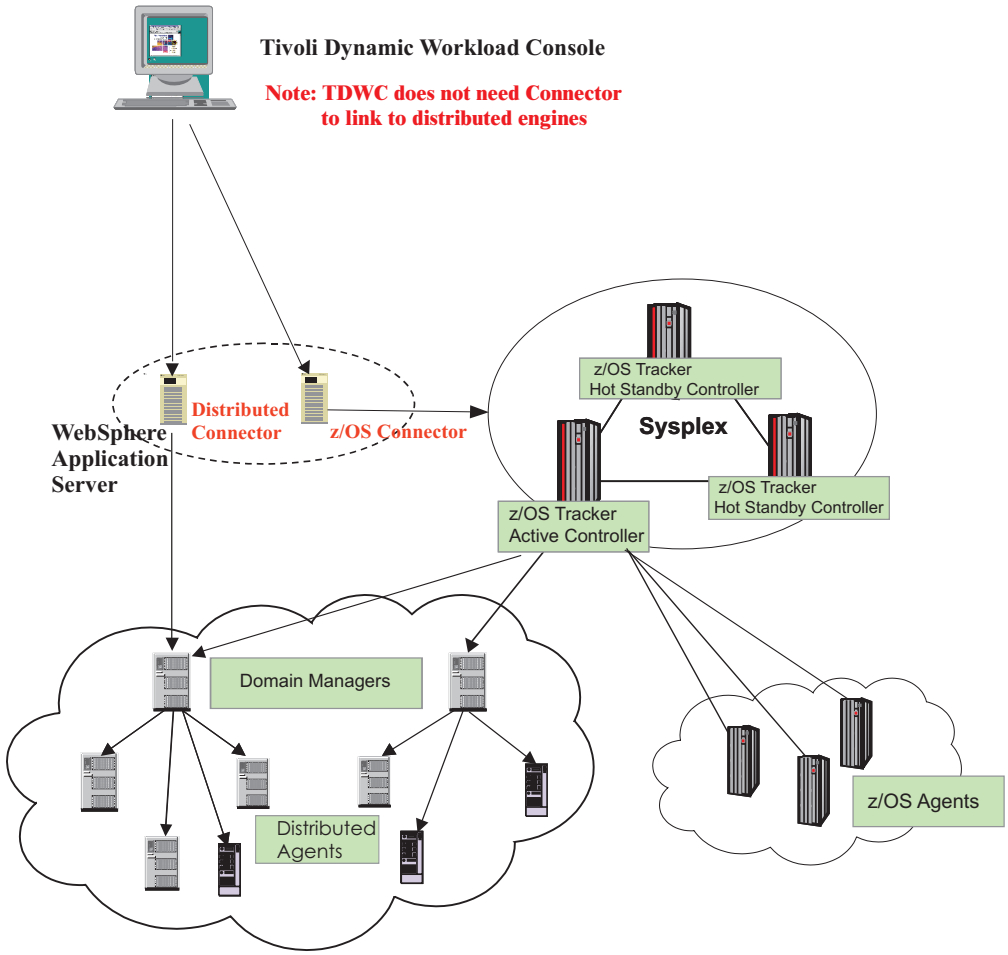


Figure 9. Tivoli Workload Scheduler for z/OS configurations

Controlled z/OS systems

An agent is required for every controlled z/OS system in an configuration. This includes, for example, local controlled systems within shared DASD or sysplex configurations.

The agent runs as a z/OS subsystem and interfaces with the operating system (through JES2 or JES3, and SMF), using the subsystem interface and the operating system exits. The agent monitors and logs the status of work, and passes the status information to the engine via shared DASD, XCF, or ACF/VTAM.

You can use z/OS and cross-system coupling facility (XCF) to connect your local z/OS systems. Instead of being passed to the controlling system using shared DASD, work status information is passed directly through XCF connections. XCF lets you use all the production-workload-restart facilities and its hot standby function. See “Automatic recovery and restart” on page 69.

Remote systems

The agent on a remote z/OS system passes status information about the production work in progress to the engine on the controlling system. All communication between Tivoli Workload Scheduler for z/OS subsystems on the controlling and remote systems is done through ACF/VTAM.

Tivoli Workload Scheduler for z/OS lets you link remote systems using ACF/VTAM networks. Remote systems are frequently used locally "on premises" to reduce the complexity of the data processing (DP) installation.

Remote panels and program interface applications

ISPF panels and program interface (PIF) applications can run in a different z/OS system from the one where the engine is running. Dialogs and PIF applications send requests to and receive data from a Tivoli Workload Scheduler for z/OS server which is running on the same z/OS system where the target engine is running, using advanced program-to-program communications (APPC). The server communicates with the engine to perform the requested actions.

The server is a separate address space, started and stopped either automatically by the engine or by the user through the z/OS start command. There can be more than one server for an engine.

If the dialogs or the PIF applications run on the same z/OS system where the target engine is running, the server might not be involved.

Scheduling jobs that are in Tivoli Workload Scheduler

Tivoli Workload Scheduler for z/OS also allows you to access job streams (schedules in Tivoli Workload Scheduler) and add them to the current plan in Tivoli Workload Scheduler for z/OS. In addition, you can build dependencies among Tivoli Workload Scheduler for z/OS job streams and Tivoli Workload Scheduler jobs. From Tivoli Workload Scheduler for z/OS, you can monitor and control the distributed agent.

Using fault-tolerant workstations

In the Tivoli Workload Scheduler for z/OS current plan, you can specify jobs to run on fault-tolerant agents in Tivoli Workload Scheduler. Tivoli Workload Scheduler for z/OS passes the job information to the Tivoli Workload Scheduler Symphony file, which in turn passes the jobs in the current plan to Tivoli Workload Scheduler to distribute and process. In turn, Tivoli Workload Scheduler reports the status of running and completed jobs back to the current plan for monitoring in Tivoli Workload Scheduler for z/OS.

Using z-centric workstations

z-centric workstations are agents that are installed in a Tivoli Workload Scheduler network and that can be connected to Tivoli Workload Scheduler for z/OS by HTTP or HTTPS. They provide the means to schedule from Tivoli Workload Scheduler for z/OS jobs that need to run on distributed platforms (UNIX, Linux, Windows). They are equivalent to computer automatic workstations in Tivoli Workload Scheduler for z/OS and require less configuration and a smaller supporting infrastructure than fault-tolerant workstations.

Chapter 8. Dynamic Workload Console

The Dynamic Workload Console is a Web-based user interface for:

- Tivoli Workload Scheduler
- Tivoli Workload Scheduler for z/OS
- Tivoli Workload Scheduler for Applications

It is the strategic user interface for the Tivoli Workload Automation suite of products and includes support for the latest functions and enhancements available with the scheduling engines. It has replaced the Job Scheduling Console, whose functional contents have not been extended beyond those of version 8.4.

The Dynamic Workload Console is a light, powerful and user-friendly single point of operational control for the entire scheduling network. It allows for single sign-on and authentication to one or many schedulers, is highly scalable, and provides real-time monitoring, management and reporting of enterprise workloads. It also greatly simplifies report creation and customization.

With Dynamic Workload Console you can:

- Manage your workload to design objects in the database, handle plans, submit jobs or job streams, and monitor objects in the plan.
- Design and control the topology of your scheduling environment, that is workstations and domains.
- Define and run reports to gather historical data or details about your plans. You can also generate and run customized SQL reports.
- Define and manage logical resources or groups of logical resources for use with dynamic scheduling.

You can access the Dynamic Workload Console from any computer in your environment using a web browser through both secure HTTPS or HTTP protocol.

The first and main actions you perform when you connect to the Dynamic Workload Console are:

Creating a connection to a scheduling engine (Tivoli Workload Scheduler or Tivoli Workload Scheduler for z/OS)

You type the details (such as IP address, user name, and password) to access a scheduling engine, and, optionally, a database to operate with objects defined in plans or stored in the database. You can also define new scheduling objects in the database.

From the Dynamic Workload Console you can access the current plan, a trial plan, a forecast plan, or an archived plan for the distributed environment or the current plan for the z/OS environment.

You might want to access the database to perform actions against objects stored in it or generate reports showing historical or statistical data.

In addition, working both on the database and on plans, you can create and run event rules to define and trigger actions that you want to run in response to events occurring on Tivoli Workload Scheduler nodes.

Creating tasks to manage scheduling objects in the plan

You specify some filtering criteria to query a list of scheduling objects

whose attributes satisfy the criteria you specified. Starting from this list, you can navigate and modify the content of the plan, switching between objects, opening more lists and accessing other plans or other Tivoli Workload Scheduler or Tivoli Workload Scheduler for z/OS environments.

The console provides also the following graphical views tools to manage your workload:

Job stream view (for modeling)
A graphical extension to the Workload Designer that shows graphical representations of job stream definitions in the database. It provides an intuitive way to create and maintain them.

Plan view (for monitoring)
A high-level representation of a plan of any type, showing a filtered set of job streams and their mutual dependencies.

Impact view (for monitoring)
An expansible graphical representation of job streams and jobs in plan. It provides a straightforward, multilevel analysis of how job and job stream completion affects plan progress.

Job stream view (for monitoring)
A graphical representation of a single job stream in plan. It provides a direct way to work with it and its dependencies.

From each view, you can take actions on objects, view their properties, and easily switch between the views. Graphics can be exported to SVG files.

You can also launch short demos (visual helps) directly from some Dynamic Workload Console panels. In fact, by clicking the "camera" icon on the toolbar, you open a menu listing some short demos that help you get rapidly familiar with the main functions available from that panel.

Chapter 9. End-to-end scheduling

By using end-to-end scheduling, you can schedule and control jobs on mainframe, Windows, and UNIX environments, for truly distributed scheduling. In the end-to-end configuration, Tivoli Workload Scheduler for z/OS is used as the planner for the job scheduling environment. Tivoli Workload Scheduler domain managers, standard, fault-tolerant, and z-centric agents are used to schedule on the distributed platforms. The agents replace the use of tracker agents.

Tivoli Workload Scheduler for z/OS also allows you to access job streams (schedules in Tivoli Workload Scheduler) and add them to the current plan in Tivoli Workload Scheduler for z/OS. In addition, you can build dependencies among Tivoli Workload Scheduler for z/OS job streams and Tivoli Workload Scheduler jobs. From Tivoli Workload Scheduler for z/OS, you can monitor and control the distributed agents.

You can manage distributed scheduling by activating either of the following features:

- "End-to-end scheduling with fault tolerance capabilities"
- "End-to-end scheduling with z-centric capabilities" on page 83

End-to-end scheduling with fault tolerance capabilities

End-to-end scheduling with fault tolerance capabilities directly connects Tivoli Workload Scheduler standard agents, fault-tolerant agents, and domain managers (with their underlying agents and domains) to Tivoli Workload Scheduler for z/OS. Tivoli Workload Scheduler for z/OS is seen by the distributed network as the master domain manager.

Tivoli Workload Scheduler for z/OS creates the production plan also for the distributed network and sends it to the domain managers and to the directly-connected agents. The domain managers send a copy of the plan to each of their agents and subordinate domain managers for execution.

The Tivoli Workload Scheduler domain managers function as the broker systems for the distributed network by resolving all dependencies for their subordinate managers and agents. They send their updates (in the form of events) to Tivoli Workload Scheduler for z/OS so that it can update the plan accordingly. Tivoli Workload Scheduler for z/OS handles its own jobs and notifies the domain managers of all the status changes of the Tivoli Workload Scheduler for z/OS jobs that involve the Tivoli Workload Scheduler plan. In this configuration, the domain managers and all the distributed agents recognize Tivoli Workload Scheduler for z/OS as the master domain manager and notify it of all the changes occurring in their own plans. At the same time, the agents are not permitted to interfere with the Tivoli Workload Scheduler for z/OS jobs, because they are viewed as running on the master that is the only node that is in charge of them.

In the Tivoli Workload Scheduler for z/OS current plan, you can specify jobs to run on workstations in the Tivoli Workload Scheduler network. Tivoli Workload Scheduler for z/OS passes the job information to the Symphony file in the Tivoli Workload Scheduler for z/OS server, which in turn passes the Symphony file to the Tivoli Workload Scheduler domain managers (DMZ) to distribute and process.

In turn, Tivoli Workload Scheduler reports the status of running and completed jobs back to the current plan for monitoring in the Tivoli Workload Scheduler for z/OS engine.

Figure 10 shows a Tivoli Workload Scheduler network managed by a Tivoli Workload Scheduler for z/OS and the flow of data.

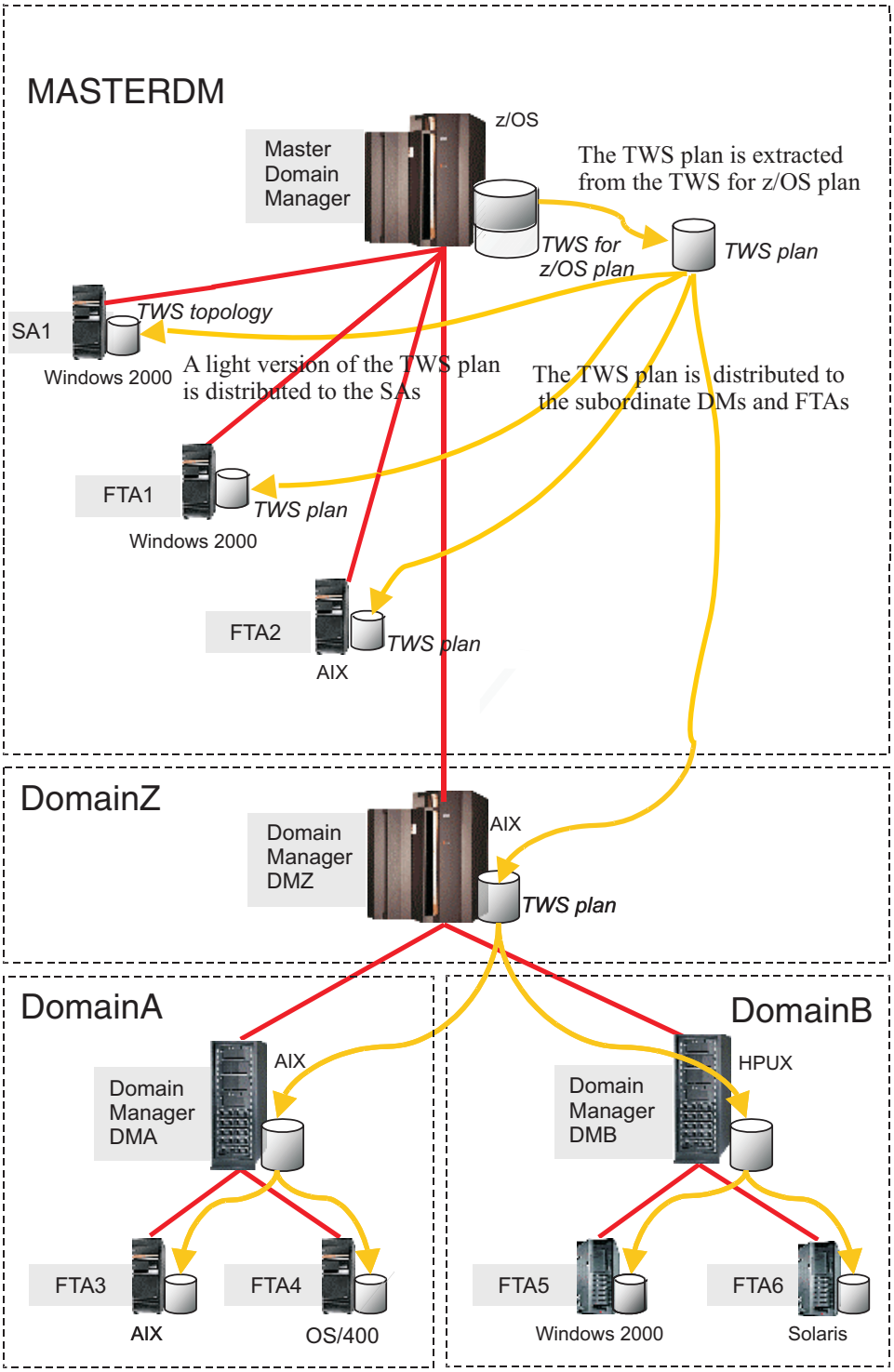


Figure 10. End-to-end with fault tolerance capabilities configuration

End-to-end scheduling with z-centric capabilities

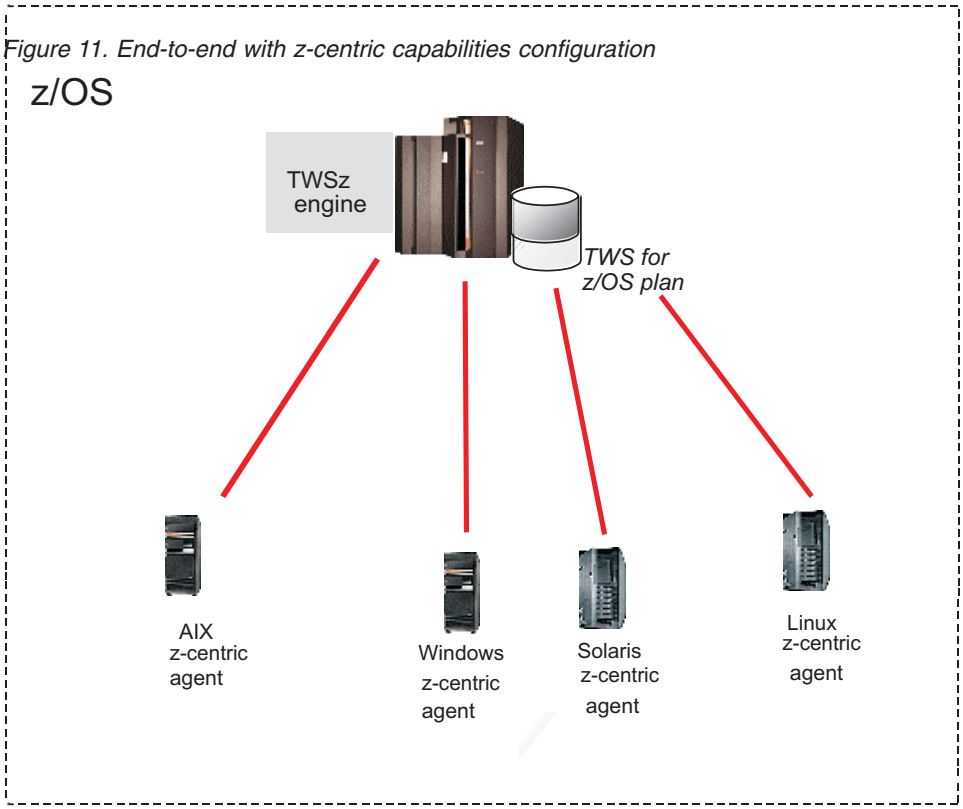
End-to-end scheduling with z-centric capabilities directly connects Tivoli Workload Scheduler z-centric agents to Tivoli Workload Scheduler for z/OS, that is the master domain manager for the distributed network.

Powerful mainframe capabilities, such as standard variable substitution, automatic recovery statements and alternate workstation, are supported to manage distributed workload.

Communication between the z-centric agents and Tivoli Workload Scheduler for z/OS controller is direct, through the HTTP or HTTPS protocol.

Figure 11 shows a network with this configuration.

Figure 11. End-to-end with z-centric capabilities configuration



Distributed agents

A distributed agent is a computer running Tivoli Workload Scheduler on which you can schedule jobs from Tivoli Workload Scheduler for z/OS. Examples of distributed agents are the following: standard agents, extended agents, fault-tolerant agents, and domain managers.

The following is a description of the types of distributed agents:

Domain Manager
The management hub in a domain. All communications to and from the agents in a domain are routed through the domain manager.

Backup Domain Manager
A fault-tolerant agent or domain manager capable of assuming the responsibilities of its domain manager for automatic workload recovery.

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 helps you launch and control jobs on other systems and applications, such as PeopleSoft, Oracle E-Business Suite, SAP, and z/OS JES2 and JES3.

z-centric Agent
A workstation that runs jobs scheduled from Tivoli Workload Scheduler for z/OS. The controller directly handles the communication with this type of agent.

Distributed agents replace tracker agents in Tivoli Workload Scheduler for z/OS. The distributed agents help you schedule on non-z/OS systems with a more reliable and scalable agent.

In the Tivoli Workload Scheduler for z/OS plan, the logical representation of a distributed agent is called a fault-tolerant workstation or a z-centric workstation.

Benefits of end-to-end scheduling

The benefits that can be gained from using end-to-end scheduling are the following:

- Connecting either fault-tolerant or z-centric Tivoli Workload Scheduler agents to Tivoli Workload Scheduler for z/OS.
- Scheduling on additional operating systems.
- Synchronization of work in mainframe and distributed environments.
- The ability for Tivoli Workload Scheduler for z/OS to use multi-tier architecture with domain managers.

Notices

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

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

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

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

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

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

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

Trademarks

IBM, the IBM logo, and ibm.com® are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at http://www.ibm.com/legal/copytrade.shtml.

Adobe, and all Adobe-based trademarks are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, other countries, or both.

Intel, and Itanium, are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

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

Index

R

S

T

U

V

W

X

Z

Product Number: 5698-A17, 5698-WSH, 5698-WSE

Printed in USA

SC32-1256-12

Spine information:

IBM Tivoli Workload Automation **Version 8.6** **Overview**