Tivoli

Tivoli Workload Scheduler

Reference Guide

Version 7.0

Tivoli

Tivoli Workload Scheduler

Reference Guide

Version 7.0

Tivoli Workload Scheduler Reference Manual (May 2000)

Contents

Preface

The Tivoli Workload Scheduler Reference Guide provides detailed information about the command line interface, scheduling language, and utility commands for TWS.

Who Should Read This Guide

This guide is intended for adminstrators and advanced users of TWS.

Related Documents

- *Tivoli Workload Scheduler Planning and Installation Guide*

  Provides information about planning and installing Tivoli Workload Scheduler.

- *Tivoli Workload Scheduler User's Guide*

  Provides information about using the Tivoli Workload Scheduler to manage your production environment.

What This Guide Contains

The Tivoli Workload Scheduler Reference Guide contains the following sections:

- Chapter 1, "Composer Reference"

  Provides an overview of the **composer** CLI. The **composer** CLI is used to create scheduling objects in the TWS database.

- Chapter 2, "Scheduling Language"

  Provides an overview of the scheduling language used to define jobs and job streams in the TWS database.

- Chapter 3, "Conman Reference"

  Provides an overview of the **conman** CLI. The **conman** CLI is used to monitor and manage job execution during the production day.

- Chapter 4, "Utility Commands"

Provides an overview of the TWS utility commands. These commands are used to manage the TWS environment.

- Chapter 5, “Extended Agent Reference”

  Provides a programmers reference for creating Extended Agents.

- Chapter 6, “Network Agent Reference”

  Provides information on creating and using a Network Agent workstation.

Conventions Used in This Guide

The guide uses several typeface conventions for special terms and actions. These conventions have the following meaning:

Bold	Commands, keywords, file names, authorization roles, URLs, or other information that you must use literally appear in **bold**. Names of windows, dialogs, and other controls also appear in **bold**.
Italics	Variables and values that you must provide appear in *italics*. Words and phrases that are emphasized also appear in *italics*.
Bold Italics	New terms appear in ***bold italics*** when they are defined in the text.
`Monospace`	Code examples, output, and system messages appear in a `monospace`font.

Platform-specific Information

The following table identifies the supported platform versions known at the time of publication. For more detailed and up-to-date information, please see the release notes.

Platform	TWS Engine	TWS Connector	JS Console
AIX 4.2	X	X	X
AIX 4.3	X	X	X
HP-UX 10.2	X	X	X

Platform	**TWS Engine**	**TWS Connector**	**JS Console**
HP-UX 11.0	X	X	X
Solaris 2.6	X	X	X
Solaris 2.7	X	X	X
Windows NT 4.0 w/ SP 4 or higher	X	X	X
Windows 2000			X
Digital UNIX 4.0	X		
Intel ABI compliant	X		
MIPS ABI compliant	X		

Contacting Customer Support

For support inside the United States, for this or any Tivoli product, contact Tivoli Customer Support in one of the following ways:

- Send e-mail to **support@tivoli.com**
- Call **1-800-TIVOLI8**
- Navigate our Web site at **http://www.support.tivoli.com**

For support outside the United States, refer to your Customer Support Handbook for phone numbers in your country. The Customer Support Handbook is available online at **http://www.support.tivoli.com**.

When you contact Tivoli Customer Support, be prepared to provide identification information for your company so that support personnel can assist you more readily.

We are very interested in hearing from you about your experience with Tivoli products and documentation. We welcome your suggestions for improvements. If you have comments or suggestions about this documentation, please send e-mail to **pubs@tivoli.com**.

1

Composer Reference

The Composer program is used to manage scheduling objects in Workload Scheduler's database. Scheduling objects are workstations, workstation classes, domains, jobs, job streams, resources, prompts, calendars, and parameters. This chapter describes the Composer command syntax and the syntax used to define scheduling objects.

Managing Scheduling Objects

Scheduling objects are managed with the Composer program and are stored in Workload Scheduler's database. To add a new object, you enter its definition in an edit file, which is checked for correct syntax before it is added to the database.

You can access the database entries for job streams, jobs, users, workstations, domains, and workstation classes and manage them individually. For example, to modify an existing job definition, the definition is copied from the database into an edit file, you modify the edit file, the edit file is checked for correct syntax, and the modified definition is copied back into the database, replacing the existing definition.

You manage the database entries for calendars, parameters, prompts, and resources as complete lists. For example, to modify an existing resource, the entire resource list is copied from the database into an edit file, you modify the edit file, the edit file is checked for correct syntax, and the new list is copied back into the database, replacing the existing resource list.

Workstation Definitions

A workstation is a scheduling object that runs jobs. You can include multiple workstation definitions in the same text file, along with workstation class definitions and domain definitions.

Synopsis

```
#[ comment]
cpuname wkstation [description "text"]
os os-type
node hostname   [tcpaddr port]
[timezone|tz tzname]
[domain domainname]
[for maestro      [host host-wkstation [access method]]
        [type fta | s-agent | x-agent]
        [ignore]
        [autolink on | off]
        [fullstatus on | off]
        [resolvedep on | off]
        [server serverid]]
end

[cpuname ...]

[cpuclass ...]

[domain ...]
```

Arguments

*comment*
: Specifies to treat everything from the pound sign to the end of the line as a comment.

cpuname *wkstation*
: Specifies the name of the workstation. The name must start with a letter, and can contain alphanumeric characters, dashes and underscores. For non-expanded databases, it can contain up to eight characters. For expanded databases, it can contain up to 16 characters.

Note: Workstation names must be unique, and cannot be the same as workstation class and domain names.

description "*text***"**
Provides a description of the workstation. Your text must be enclosed in double quotes.

os *os-type*
Specifies the operating system of the workstation. Valid include **MPEV**, **MPIX**, **UNIX**, **WNT**, and **OTHER**.

node *hostname*
Specifies the host name or the IP address of the workstation. Fully-qualified domain names are accepted.

tcpaddr *port*
Specifies the TCP port number that Workload Scheduler uses for communications on the workstation. The default is **31111**.

timezone|tz *tzname*
Specifies the time zone of the workstation. See "Time Zone Names" on page 309 for valid time zone names. To ensure the accuracy of scheduling times, this time zone must be the same as the computer's operating system time zone.

domain *domainname*
Specifies the name of the Workload Scheduler domain of the workstation. The default for fault-tolerant and standard agents is the master domain, usually named **MASTERDM**. The default for a domain manager is the domain in which it is defined as the manager. The default for an extended agent is the domain of its host workstation.

host *host-wkstation*
Specifies the name of the agent's host workstation. This is required for extended agents. The host is the workstation with which the extended agent communicates and where its access method resides. Note that the host cannot be another extended agent.

access *method*
Specifies an access method for extended and network agents.

This must be the name of a file that resides in the *TWShome*/**methods** directory on the agent's host workstation.

type Specifies the type of workstation. Enter one of the following:

- **fta** Fault-tolerant agent, including domain managers and backup domain managers.
- **s-agent** Standard agent.
- **x-agent** Extended agent.

ignore Specifies that Workload Scheduler will ignore this workstation definition.

autolink
Specifies whether to open the link between workstations at startup. For fault-tolerant and standard agents, enter **on** to have the domain manager open the link to the agent when the domain manager is started. For the domain manager, enter **on** to have its agents open links to the domain manager when they are started.

Autolink is useful primarily during the startup sequence at the beginning of each day. At that time, a new production plan is created and compiled on the master domain manager, and all workstations are stopped and restarted. For each agent with **autolink** turned on, the domain manager automatically sends a copy of the new production plan and starts the agent. If **autolink** is also turned on for the domain manager, the agent, in turn, opens a link back to the domain manager. If the value of **autolink** is **off** for an agent, it is initialized when you execute a Conman **link** command on the agent's domain manager or the master domain manager.

fullstatus
Specifies whether the agent is updated with full or partial status. This is for fault-tolerant agents only. Enter **on** to have the agent operate in **Full Status** mode. In this mode, the

agent is updated about the status of jobs and job streams running on all other workstations in its domain and subordinate domains.

If the value of **fullstatus** is **off**, the agent is informed only about the status of jobs and job streams on other workstations that affect its own jobs and job streams. This can improve performance by reducing network activity.

To keep an agent's production plan at the same level of detail as its domain manager, set **fullstatus** and **resolvedep** to **on**. Always set these modes **on** for backup domain managers.

resolvedep

Specifies whether an agent will track all dependencies or only its own. This is for fault-tolerant agents only. Enter **on** to have the agent operate in **Resolve All Dependencies** mode. In this mode, the agent tracks dependencies for all jobs and job streams, including those running on other workstations. Note that the value of **fullstatus** must also be **on** so that the agent is informed about activity on other workstations. If the value of **resolvedep** is **off**, the agent tracks dependencies for its own jobs and job streams only. This reduces processing overhead.

To keep an agent's production plan at the same level of detail as its domain manager, set **fullstatus** and **resolvedep** to **on**. Always set these modes **on** for backup domain managers.

server *serverid*

Specifies a Mailman server on the domain manager to handle communications with the agent. This is for fault-tolerant and standard agents only. Do not use this option for domain managers. Using servers can reduce the time required to initialize agents and improve the timeliness of messages.

The ID is a single letter or a number (A-Z and 0-9). The IDs are unique to each domain manager, so you can use the same

IDs in other domains without conflict. If more than 36 server IDs are required in a domain, consider dividing it into two or more domains.

If a server ID is not specified, communications with the agent are handled by the main Mailman process on the domain manager.

When it starts up, the domain manager creates a separate server for each unique server ID. If the same ID is used for multiple agents, a single server is created to handle their communications. As a guide, extra servers should be defined to prevent a single server from handling more than eight agents.

Examples

The following creates a master domain manager named **hdq-1**, and a fault-tolerant agent named **hdq-2** in the master domain. Note that a **domain** argument is optional in this example, because the master domain defaults to **masterdm**.

```
cpuname hdq-1 description "Headquarters master DM"
     os unix
     tz pst
     node sultan.unison.com
     domain masterdm
     for maestro type fta
          autolink on
          fullstatus on
          resolvedep on
end
```

```
cpuname hdq-2
     os wnt
     tz pst
     node opera.unison.com
     domain masterdm
     for maestro type fta
          autolink on
end
```

The following example creates a domain named **distr-a** with a domain manager named **distr-a1** and a standard agent named **distr-a2**:

```
domain distr-a
     manager distr-a1
     parent masterdm
end

cpuname distr-a1 description "District A domain mgr"
     os wnt
     tz est
     node pewter.unison.com
     domain distr-a
     for maestro type fta
          autolink on
          fullstatus on
          resolvedep on
end

cpuname distr-a2
     os wnt
     node quatro.unison.com
     tz est
     domain distr-a
     for maestro type s-agent
          host distr-a1
end
```

Workstation Class Definitions

A workstation class is a group of workstations for which common job streams can be written. You can include multiple workstation class definitions in the same text file, along with workstation definitions and domain definitions.

Synopsis

# *comment*
cpuclass *wkstationclass*
    members {*wkstation* | **@**} [...]
end

[**cpuname** ...]

[**cpuclass** ...]

[**domain** ...]

Arguments

*comment*
: Specifies to treat everything from the pound sign to the end of the line as a comment.

cpuclass *wkstationclass*
: Specifies the name of the workstation class. The name must start with a letter, and can contain alphanumeric characters, dashes and underscores. For non-expanded databases, it can contain up to eight characters. For expanded databases, it can contain up to 16 characters.

  Note: You cannot use the same names for workstations, workstation classes, and domains.

members *wkstation*
: Specifies a list of workstation names, separated by spaces, that are members of the class. The at sign (@) wildcard character includes all workstations.

Examples

The following defines a workstation class named **backup**:

```
cpuclass backup
     members
          main
          site1
          site2
end
```

The following defines a workstation class named **allcpus** that contains every workstation:

```
cpuclass allcpus
     members
          @
end
```

Domain Definitions

A domain is a group of workstations consisting of one or more agents and a domain manager. The domain manager acts as the management hub for the agents in the domain. You can include multiple domain definitions in the same text file, along with workstation definitions and workstation class definitions.

Synopsis

# *comment*
domain *domainname*[**description** "*text*"]
    manager *wkstation*
    [**parent** *domainname*]
end

[**cpuname** ...]

[**cpuclass** ...]

[**domain** ...]

Arguments

# *comment*
: Specifies to treat everything from the pound sign to the end of the line as a comment.

domain *domainname*
: Specifies the name of the domain. The name must start with a letter, and can contain alphanumeric characters, dashes and underscores. For non-expanded databases, it can contain up to eight characters. For expanded databases, it can contain up to 16 characters.

  Note: You cannot use the same names for workstations, workstation classes, and domains.

description "*text*"
: Provides a description of the domain. Your text must be enclosed in double quotes.

manager *wkstation*
: Specifies the name of the workstation that is the domain manager. The domain manager must be a fault-tolerant agent with **fullstatus** and **resolvedep** set to **on**.

parent *domainname*
: Specifies the name of the parent domain to which the domain manager is linked. The default is the master domain.

Note: The master domain does not require a domain definition. The master domain is defined by the global options **master** and **master domain**.

Examples

The following defines a domain named **east**, with the master domain as its parent, and two subordinate domains named **northeast** and **southeast**:

```
domain east description "The Eastern domain"
     manager cyclops
end
domain northeast description "The Northeastern domain"
     manager boxcar      parent east
end
domain southeast description "The Southeastern domain"
     manager sedan      parent east
end
```

Job Definitions

A job is an executable file, program, or command that is scheduled and launched by Workload Scheduler. You can write job definitions in edit files and then add them to Workload Scheduler’s database with the Composer program. You can include multiple job definitions in a single edit file. Note that you specify job dependencies when you write job streams. The scheduling language used to write job streams also permits you to add and modify job definitions.

Synopsis

$jobs
[*wkstation*#]*jobname*
    [**description** “*text*”]
    {**scriptname** | **jobfilename**} *filename* | **docommand** \
“*command*”
streamlogon *username*
    [**interactive**]
    [**recovery**
        {**stop** | **continue** | **rerun**}
        [{**recoveryjob** | **after**} [*wkstation*#]*jobname*]
        [{**recoveryprompt** | **abendprompt**} “*text*”]]

[[*wkstation*#]*jobname*...]

Arguments

wkstation#
: Specifies the name of the workstation or workstation class on which the job runs. The default is the workstation on which Composer is running. The pound sign (#) is a required delimiter. If you specify a workstation class, it must match the workstation class of any job stream in which the job is included.

jobname
: Specifies the name of the job. The name must start with a letter, and can contain alphanumeric characters, dashes and

underscores. For non-expanded databases, it can contain up to eight characters. For expanded databases, it can contain up to 40 characters.

description "*text***"**
Provides a description of the job. Your text must be enclosed in double quotes.

{scriptname | jobfilename} *filename*

Specifies the name of the file the job executes. Use **scriptname** for UNIX and Windows NT jobs, and use **jobfilename** for MPE jobs. For an executable file, enter the file name and any options and arguments. For non-expanded databases, it can contain up to 255 characters. For expanded databases, it can contain up to 4095 characters. You can also use Workload Scheduler parameters. See "Using Parameters in Job Definitions" on page 15 for more information.

For Windows NT jobs, include the file extensions. Universal Naming Convention (UNC) names are permitted. Do not specify files on mapped drives.

If spaces or special characters are included, other than slashes (/) and backslashes (\), the entire string must be enclosed in quotes (″).

If the file name contains spaces, enter the name in another file that does not have spaces in its name and use the second file's name for this argument.

docommand *command*
Specifies a command that the job executes. Enter a valid command and any options and arguments (up to 4095 characters) enclosed in quotes (″). A command is executed directly and, unlike **scriptname** or **jobfilename**, the configuration script, **jobmanrc**, is not executed. Otherwise, the command is treated as a job, and all job rules apply. You can also enter Workload Scheduler parameter. See "Using Parameters in Job Definitions" on page 15 for more information.

streamlogon *username*

The user name under which the job runs. The name can contain up to 47 characters. If the name contains special characters it must be enclosed in quotes (″). Specify a user that can log on to the workstation on which the job runs. You can also enter Workload Scheduler parameter. See “Using Parameters in Job Definitions” on page 15 for more information.

For Windows NT jobs, the user must also have a user definition. See “User Definitions” on page 17 for user requirements.

interactive

For Windows NT jobs, include this keyword to indicate that the job runs interactively on the Windows NT desktop.

recovery

Recovery options for the job. The default is **stop** with no recovery job and no recovery prompt. Enter one of the recovery options, **stop**, **continue**, or **rerun**. This can be followed by a recovery job, a recovery prompt or both.

stop If the job abends, do not continue with the next job.

continue
If the job abends, continue with the next job.

rerun If the job abends, rerun the job.

{recoveryjob | after} [*wkstation***#]***jobname*

Specifies the name of a recovery job to run if the parent job abends. Recovery jobs are run only once for each abended instance of the parent job.

You can specify the recovery job’s workstation if it is different than the parent job’s workstation. The default is the parent job’s workstation. Not all jobs are eligible to have recovery jobs run on a different workstation. Follow these guidelines:

- The parent job’s and recovery job’s workstation must be running Workload Scheduler Version 4.5.5 or later.
- If either workstation is an extended agent, it must be hosted by a domain manager or a fault-tolerant agent with a value of **on** for **fullstatus**.
- The recovery job’s workstation must be in the same domain as the parent job’s workstation.
- If the recovery job’s workstation is a fault-tolerant agent, it must have a value of **on** for **fullstatus**.

{recoveryprompt | abendprompt} “*text*”
Specifies the text of a recovery prompt, enclosed in quotes, to be displayed if the job abends. The text can contain up to 64 characters. If the text begins with a colon (:), the prompt is displayed, but no reply is required to continue processing. If the text begins with an exclamation mark (!), the prompt is not displayed, but a reply is required to proceed.

Using Parameters in Job Definitions

Parameters have the following uses and limitations in job definitions:

- Parameters are allowed in the **streamlogon**, **scriptname**, and **docommand** values.
- A parameter can be used as an entire string or a part of it.
- Multiple parameters are permitted in a single variable.
- Enclose parameter names in carets (^), and enclose the entire string in quotation marks.

See the example below in which a parameter named **mis** is used in the **streamlogon** value. For additional examples, see “Parameter Definitions” on page 21.

Examples

The following is a file containing two job definitions:

```
$jobs
cpu1#gl1
     scriptname "/usr/acct/scripts/gl1"
     streamlogon acct
     description "general ledger job1"
bkup
     scriptname "/usr/mis/scripts/bkup"
     streamlogon "^mis^"
     recovery continue after recjob1
```

User Definitions

The user names used as the **streamlogon**

value for Windows NT job definitions must have user definitions. This is not required for users who run jobs on other platforms.

Synopsis

# *comment*
username[*wkstation*#]*username*
    password "*password*"
end

[**username** ...]

Arguments

# *comment*
: Specifies to treat everything from the pound sign to the end of the line as a comment.

username [*wkstation***#]***username*
: Specifies the name of a Windows NT user.

wkstation
: Specifies the workstation on which the user is allowed to launch jobs. The pound sign is required. The default is the blank, meaning all workstations.

username
: Specifies the user name in the following form:

[*domain*\]*user*

where *domain* is the Windows NT domain of the user and *user* is the name of the user.

For non-expanded databases, the name can contain up to eight characters. For expanded databases, the domain name can contain up to 16 characters (including the backslash), and the user name can contain up to 31 characters.

Note that Windows NT user names are case-sensitive. Also, the user must be able to log on to the workstation on which Workload Scheduler will launch jobs, and must have the right to **Log on as batch**.

If the name is not unique in Windows NT, it is considered to be a local user, a domain user, or a trusted domain user, in that order.

password

Specifies the user’s password. The password can contain up to 29 characters, and must be enclosed in quotes. To indicate no password, use two consecutive quotes with no space (″″). Once a user definition is compiled, you cannot read the password. Users with appropriate security privileges can modify or delete a user, but password information is never displayed.

Trusted Domain User

If Workload Scheduler must launch jobs for a trusted domain user, give special attention to defining the user accounts. Assuming Workload Scheduler is installed in **Domain1** for user account **maestro**, and user account **sue** in **Domain2** needs to launch a job, the following must be true:

- There must be mutual trust between **Domain1** and **Domain2**.
- In **Domain1** on the computers where jobs are launched, **sue** must have the right to **Log on as batch**.
- In **Domain1**, **maestro** must be a domain user.
- On the domain controllers in **Domain2**, **maestro** must have the right to **Access this computer from network**.

Examples

The following example defines four users:

```
username joe
     password "okidoki"
end
#
username server#jane
```

```
     password "okitay"
end
#
username dom1\jane
     password "righto"
end
#
username jack
     password ""
end
```

Calendar Definitions

Calendars are lists of dates that you can use to schedule job streams.

Synopsis

$calendar
calendarname [“*description*”]
    date [...]

[*calendarname* ...]

Arguments

calendarname
: Specifies the name of the calendar. The name can contain up to eight alphanumeric characters, including dashes (-) and underscores (_), and must start with a letter.

“*description***”**
: Provides a description of the calendar. It must be enclosed in double quotes.

date **[...]**
: Specifies one or more dates, separated by spaces. The format is *mm/dd/yy*.

Examples

The following example defines three calendars named **monthend**, **paydays**, and **holidays**:

```
$calendar
monthend "Month end dates 1st half '99"
     01/31/99 02/28/99 03/31/99 04/30/99 05/31/99 06/30/99
paydays
     01/15/99 02/15/99
     03/15/99 04/15/99
     05/14/99 06/15/99
holidays
     01/01/99 02/15/99 05/31/99
```

Parameter Definitions

Parameters are values that can be substituted variables in job and job stream definitions.

Synopsis

$parm
parametername *“value”*

[*parametername* ...]

Arguments

parametername
The name of the parameter. The name can contain up to eight alphanumeric characters, including dashes (-) and underscores (_), and must start with a letter.

value Specifies the value assigned to the parameter. Do not include the names of other parameters.

Usage Notes

Parameters are accessible to all jobs, job streams, and prompts. When used in scheduling, parameter names are replaced by their values when the production plan is compiled for a new processing day. See *Examples* below.

Examples

Two parameters, **glpath** and **gllogon**, are defined as follows:

```
$parm
glpath     "/glfiles/daily"
gllogon     "gluser"
```

In a job stream named **glsched**, the **glpath** and **gllogon** parameters are used in a job named **gljob2**:

```
schedule glsched on weekdays
:
gljob2
     scriptname "/usr/gl^glpath^"
```

```
          streamlogon "^gllogon^"
          opens "^glpath^/datafile"
          prompt ":^glpath^ started by ^gllogon^"
end
```

Prompt Definitions

You can use prompts as dependencies in 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:

- ad hoc or local
- predefined or global

A local prompt is defined within the properties of a job or job stream and is unique to that job or job stream.

A global prompt is defined in the TWS database and can be used by any job or job stream.

Synopsis

$prompt
promptname “[**:** | **!**]*text*”

[*promptname* ...]

Arguments

promptname
: Specifies the name of the prompt. The name can contain up to eight alphanumeric characters, including dashes (-) and underscores (_), and must start with a letter.

text
: Provides the text of the prompt. If the text begins with a colon (:), the prompt is displayed, but no reply is required to continue processing. If the text begins with an exclamation mark (!), the prompt is not displayed, but a reply is required to proceed.

  You can use one or more parameters as part or all of the text string for a prompt. If you use a parameter, the parameter string must be enclosed in carets (^). See “Parameter Definitions” on page 21 for an example.

Note: Within a local prompt, when not designating a parameter, carets (^) must be preceded by a backslash (\) or they will cause errors in the prompt. Within global prompts, carets do not have to be preceded by a backslash.

You can include backslash n (**n**) within the text to create a new line.

Examples

The following example defines three prompts:

```
$prompt
     prmt1 "ready for job4? (y/n)"
     prmt2 ":job4 launched"
     prmt3 "!continue?"
```

Resource Definitions

Resources represent physical or logical scheduling resources that can be used as dependencies for jobs and job streams.

Synopsis

$resource
wkstation#resourcename *units* [“*description*”]

[*wkstation#resourcename* ...]

Arguments

wkstation
: Specifies the name of the workstation or workstation class on which the resource is used.

resourcename
: Specifies the name of the resource. The name can contain up to eight alphanumeric characters, including dashes (-) and underscores (_), and must start with a letter.

units
: Specifies the number of available resource units. Values can be **0** through **1024**.

“*description***”**
: Provides a description of the resource. It must be enclosed in double quotes.

Examples

The following example defines four resources:

```
$resource
  ux1#tapes  3 "tape units"
  ux1#jobslots  24 "job slots"
  ux2#tapes  2 "tape units"
  ux2#jobslots  16 "job slots"
```

The Composer Program

The Composer program manages scheduling objects in Workload Scheduler’s database.

Running Composer

To run the program, use the following command:

composer ["*command*[**&**[*command*]][...]"]

The following are examples of the command:

- Runs Composer and prompts for a command:
  ```
  composer
  ```
- Executes **print** and **version** commands, and quits:
  ```
  composer "p parms&v"
  ```
- Executes **print** and **version** commands, and then prompts for a command:
  ```
  composer "p parms&v&"
  ```
- Reads commands from **cmdfile**:
  ```
  composer < cmdfile
  ```
- Pipes commands from **cmdfile** to Composer:
  ```
  cat cmdfile | composer
  ```

Control Characters

You can enter the following control characters in conversational mode to interrupt Composer if your **stty** settings are configured to do so.

Ctrl+c

Composer stops executing the current command at the next step that can be interrupted and returns a command prompt.

Ctrl+d

Composer quits after executing the current command.

Terminal Output

The output to your computer is controlled by shell variables named **MAESTROLINES** and **MAESTROCOLUMNS**. If either is not set, the standard shell variables, **LINES** and **COLUMNS**, are used. At the end of each screen page, Composer prompts to continue. If **MAESTROLINES** (or **LINES**) is set to zero or a negative number, Composer does not pause at the end of a page.

Offline Output for Windows NT

The **;offline** option in Composer commands is used to print the output of a command. When you include it, the following variables control the output:

$MAESTROLP
: Specifies the file into which a command's output is written.The default is **stdout**.

$MAESTROLPLINES
: Specifies the number of lines per page. The default is 60.

$MAESTROLPCOLUMNS
: Specifies the number of characters per line. The default is 132.

Offline Output for UNIX

The **;offline** option in Composer commands is used to print the output of a command. When you include it, the following shell variables control the output:

$MAESTROLP
: Specifies the destination of a command's output. Set it to one of the following:

> *file*
: Redirects output to a file and overwrites the contents of the file. If the file does not exist, it is created.

>> *file*
: Redirects output to a file and appends the output to the end of the file. If the file does not exist, it is created.

| *command*
: Pipes output to a system command or process. The system command is executed whether or not output is generated.

|| *command*

Pipes outut to a system command or process. The system command is not executed if there is no output.

The default is **| lp -tCONLIST** which directs the command output to the printer and places the title “CONLIST” in the banner page of the printout.

$MAESTROLPLINES
Specifies the number of lines per page. The default is **60**.

$MAESTROLPCOLUMNS
Specifies the number of characters per line. The default is **132**.

You must export the variables before you run Composer.

Selecting Composer’s Editor on Windows NT

Several of Composer’s commands automatically open a text editor. The default is the MS-DOS editor. To change the editor, set the variable **EDITOR** to the name of the new editor before running Composer.

Selecting Composer’s Editor on UNIX

Several of Composer’s commands automatically open a text editor. The type of editor is determined by the value of two shell variables. If the variable **VISUAL** is set, it defines the editor, otherwise the variable **EDITOR** defines the editor. If neither of the variables is set, a **vi** editor is opened.

Selecting Composer’s Command Prompt on UNIX

Composer’s command prompt is defined in the message file *TWShome*/**catalog/maestro.msg**. The default command prompt is a dash (-). To select a different prompt, edit the message file, locate set 208, message number 01, and change the character on that line. The prompt can be up to ten characters long, not including the required trailing pound sign (#). After changing the prompt, generate a new catalog file from the changed message file as follows:

```
cd TWShome/catalog
gencat maestro.cat maestro.msg
```

Command Syntax

Composer commands consist of the following elements:

commandname selection arguments

where:

commandname
Specifies the command name.

selection
Specifies the object or set of objects to be acted upon.

arguments
Specifies the command arguments.

Wildcards

The following wildcard characters are permitted in some Composer commands:

@	Replaces one or more alphanumeric characters.
?	Replaces one alphabetic character.
%	Replaces one numeric character.

Delimeters and Special Characters

The following characters have special meanings in Composer commands.

Character	Description
&	Command delimiter. See “Running Composer” on page 26.
;	Argument delimiter. For example: `;info;offline`
=	Value delimiter. For example: `sched=sked5`
: !	Command prefixes that pass the command on to the system. These prefixes are optional; if Composer does not recognize the command, it is passed automatically to the system. For example: `!ls or :ls`

Character	Description
<< >>	Comment brackets. Comments can be placed on a single line anywhere in a job stream. For example: `schedule foo <<`*`comment`*`>> on everyday`

*	Comment prefix. When this prefix is the first character on a line, the entire line is a comment. When the prefix follows a command, the remainder of the line is a comment. For example:
	`*`*`comment`*
	`or`
	`print& *`*`comment`*

>	Redirects command output to a file and overwrites the contents of the file. If the file does not exist, it is created. For example: `display parms > parmlist`
>>	Redirects command output to a file and appends the output to the end of file. If the file does not exist, it is created. For example: `display parms >> parmlist`
|	Pipes command output to a system command or process. The system command is executed whether or not output is generated. For example: `display parms | grep alparm`
||	Pipes command output to a system command or process. The system command is not executed if there is no output. For example: `display parms || grep alparm`

Command Descriptions

The following pages describe Composer’s commands.

Command	Description	Page
add	Adds scheduling objects.	32
build	Builds or rebuilds a Workload Scheduler file.	34
continue	Ignores the next error.	36

Command	Description	Page
create	Creates a file for editing.	37
delete	Deletes scheduling objects.	40
display	Displays scheduling objects.	42
edit	Edits a file.	47
exit	Terminates Composer.	48
list	Lists scheduling objects.	42
modify	Modifies scheduling objects.	49
new	Edits and adds scheduling objects.	52
print	Prints scheduling objects.	42
redo	Edits the previous command.	53
replace	Replaces scheduling objects.	55
validate	Validates a file.	56
version	Displays Composer's program banner.	58
system command	Passes a system command to the system.	59

You can type command names and keywords in either uppercase or lowercase. You can also be abbreviate them to as few leading characters as needed to distinguish them from one another.

add

Adds jobs, job streams, users, workstations, workstation classes, and domains. You must have **add** access for new objects. If an object already exists, you must have **modify** access to the object.

Synopsis

add *filename*

Arguments

filename

Specifies the name of a file that contains the following:

- Job definitions. (The first line of the file must be **$jobs**.)
- Job stream definitions.
- Any combination of workstation, workstation class, and domain definitions.
- User definitions.

Usage Notes

The syntax of the file is always checked before it is written to the database. All errors and warnings are reported. If there are syntax errors, you are asked if you want to edit the file to make corrections. If an object already exists, you are asked whether or not to replace it.

Examples

The following example adds the jobs from the file **myjobs**:

```
add myjobs
```

The following example adds the job streams from the file **mysked**:

```
a mysked
```

The following example adds the workstations, workstation classes, and domains from the file **cpus.src**:

```
a cpus.src
```

The following example adds the user definitions from the file **users_nt**:

```
a users_nt
```

build

Builds or rebuilds Workload Scheduler's database files. You must have **build** access to the file.

Synopsis

build *filename*

Arguments

filename

Specifies one of the following file names:

calendars	The file that contains calendar definitions.
cpudata	The file that contains workstation, workstation class, and domain definitions.
jobs	The file that contains job definitions.
mastsked	The file that contains job stream definitions.
parms	The file that contains parameter definitions.
prompts	The file that contains prompt definitions.
resources	The file that contains resource definitions.
userdata	The file that contains user definitions.

Usage Notes

If a file does not exist, one is created. If the file exists, it is rebuilt. Rebuilding a database file can be useful when the database becomes fragmented from numerous additions and deletions. The rebuild will remove unused records and optimize the keys.

Examples

The following example rebuilds the files containing job stream definitions:

```
build mastsked
```

The following example rebuilds the file containing calendars:

```
build calendars
```

The following example rebuilds the file containing workstations:

```
b cpudata
```

continue

Specifies to ignore the next command error.

Synopsis

continue

Usage Notes

This command is useful when multiple commands are entered on the command line or redirected from a file. It instructs Composer to continue executing commands even if the next command, following **continue**, results in an error. This command is not needed when you enter commands interactively because Composer will not quit on an error.

Examples

The following example tells Composer to continue with the **print** command if the **delete** command results in a error:

```
composer "continue&delete cpu=site4&print cpu=@"
```

create

Creates a file containing object definitions. You must have **display** access to the objects being copied.

Synopsis

create *filename* **from calendars** | **parms** | **prompts** | **resources** |
cpu={*wkstation* | *wkstationclass* | *domain*} |
jobs=[*wkstation#*]*jobname* |
sched=[*wkstation#*]*jstream* |
users=[*wkstation#*]*username*

Arguments

filename
: Specifies a file name.

calendars
: Copies all calendars into the file.

parms Copies all parameters into the file.

prompts
: Copies all prompts into the file.

resources
: Copies all resources into the file.

cpu Copies a workstation, workstation class, or domain into the file.

wkstation
: The workstation name. Wildcard characters are permitted.

wkstationclass
: The workstation class name. Wildcard characters are permitted.

domain
: The domain name. Wildcard characters are permitted.

jobs Copies a job into the file.

wkstation
: The name of the workstation or workstation class on which the job runs. Wildcards are permitted. The default is the workstation on which Composer is running.

jobname
: The job name. Wildcards are permitted.

sched Copies a job stream into the file.

wkstation
: The name of the workstation or workstation class on which the job stream runs. Wildcard characters are permitted. The default is the workstation on which Composer is running.

jstream
: The job stream name. Wildcard characters are permitted.

users Copies a user into the file. The password field is not copied for security reasons.

wkstation
: The name of the workstation on which the user is defined. Wildcard characters are permitted. The default is the workstation on which Composer is running.

username
: The user name. Wildcard characters are permitted.

Usage Notes

After you create a file, you can use the **edit** command to make changes to the file. The **add** or **replace** command can then be used to add or update the definition.

Examples

The following example creates a file containing all calendars:

```
create caltemp from calendars
```

The following example creates a file containing all job streams:

```
cr stemp from sched=@
```

delete

Deletes object definitions from the database. You must have **delete** access to the objects being deleted.

Synopsis

delete cpu={*wkstation* | *wkstationclass* | *domain*} |
jobs=[*wkstation*#]*jobname* |
sched=[*wkstation*#]*jstream* |
users=[*wkstation*#]*username*

Arguments

cpu Deletes a workstation, workstation class, or domain.

wkstation
The workstation name. Wildcards are permitted.

wkstationclass
The workstation class name. Wildcards are permitted.

domain
The domain name. Wildcards are permitted.

jobs Deletes a job.

wkstation
The name of the workstation or workstation class on which the job runs. Wildcards are permitted. The default is the workstation on which Composer is running.

jobname
The job name. Wildcards are permitted.

sched Deletes a job stream.

wkstation
The name of the workstation or workstation class on which the job stream runs. Wildcards are permitted. The default is the workstation on which Composer is running.

jstream
The job stream name. Wildcards are permitted.

users Deletes a user.

wkstation
The name of the workstation on which the user is defined. Wildcards are permitted. The default is the workstation on which Composer is running.

username
The user name. Wildcards are permitted.

Usage Notes

If you use wildcard characters to specify a set of definitions, Composer requires confirmation before deleting each matching definition.

Examples

The following example deletes **job3** that is launched on workstation **site3**:

```
delete jobs=site3#job3
```

The following example deletes all workstations with names starting with **ux**:

```
de cpu=ux@
```

The following example deletes all job streams with names starting with **test** on all workstations:

```
de sched=@#test@
```

display, list, print

Displays, lists, or prints object definitions. For **display** and **print**, you must have **display** access to the object. **Display** actually displays the contents of the database object, while **list** and **print** display on the name and attributes of the database object.

Synopsis

display | **list** | **print** **calendars** | **parms** | **prompts** | **resources** |
 cpu={*wkstation* | *wkstationclass* | *domain*} |
jobs=[*wkstation*#]*jobname* |
sched=[*wkstation*#]*jstream* |
users=[*wkstation*#]*username*

Arguments

calendars
Displays all calendars.

parms Displays all parameters.

prompts
Displays all prompts.

resources
Displays all resources.

cpu Displays a workstation, workstation class, or domain.

wkstation
The workstation name. Wildcards are permitted.

wkstationclass
The workstation class name. Wildcards are permitted.

domain
The domain name. Wildcards are permitted.

jobs Displays a job.

wkstation
The name of the workstation or workstation class on

which the job runs. Wildcards are permitted. The default is the workstation on which Composer is running.

jobname
The job name. Wildcards are permitted.

sched Displays a job stream.

wkstation
The name of the workstation or workstation class on which the job stream runs. Wildcards are permitted. The default is the workstation on which Composer is running.

jstream
The job stream name. Wildcards are permitted.

users Displays a user. (The password field is not copied for security reasons.)

wkstation
The name of the workstation on which the user is defined. Wildcards are permitted. The default is the workstation on which Composer is running.

username
The user name. Wildcards are permitted.

Command Output

The **list** command displays only the object names. The output of the **print** command is controlled by the variable MAESTROLP. See "Offline Output" on page 97 for more information.

Calendars Format

Calendar
The calendar name.

Description
A free-form description of the calendar.

Following these fields is a list of calendar dates.

Cpu Format

CPU id
The of a workstation, workstation class, or domain.

Creator
The name of the user who created the workstation definition.

Last Updated
The date the workstation definition was last updated.

Following these fields is the workstation or workstation class definition.

Jobs Format

CPU id
The name of the workstation on which the job runs.

Job The name of the job.

Logon The name of the logon user for the job.

Last Run Date
The date the job was last run.

Following these fields is the job definition.

Parms Format

Parameter
The name of the parameter.

Value The value of the parameter.

Prompts Format

Prompt
The name of the prompt.

Message
The prompt message text.

Resources Format

CPU id
The name of the workstation on which the resource is defined.

Resource
The name of the resource.

No Avail
The total number of resource units.

Description
The free-form description of the resource.

Sched Format

CPU id
The name of the workstation on which the job stream runs.

Schedule
The name of the job stream.

Creator
The name of the user who created the job stream definition.

Last Updated
The date the job stream definition was last updated.

Following these fields is the job stream definition.

Users Format

CPU id
The name of the workstation on which the user is allowed to run jobs.

User The name of the user.

Creator
The name of the user who created the user definition.

Last Updated
The date the user definition was last updated.

Following these fields is the user definition.

Examples

The following example displays all calendars:

```
display calendars
```

The following example prints all job streams that are launched on workstation **site2**:

```
di sched=site2#@;offline
```

edit

Edits a file.

Synopsis

edit *filename*

Usage Notes

An editor is started and the specified file is opened for editing. See “The Composer Program” on page 26 for more information.

Examples

The following example opens the file **mytemp** for editing:

```
edit mytemp
```

The following example opens the file **resfile** for editing:

```
ed resfile
```

exit

Exits the Composer program.

Synopsis

exit

Usage Notes

When you are running the Composer program in help mode, this command returns Composer to command input mode.

Examples

The following examples exits the Composer program:

```
exit
```

```
e
```

modify

Modifies or adds scheduling objects. You must have **modify** access to the object or affected database file.

Synopsis

modify calendars | **parms** | **prompts** | **resources** |
cpu={*wkstation* | *wkstationclass* | *domain*} |
jobs=[*wkstation#*]*jobname* |
sched=[*wkstation#*]*jstream* |
users=[*wkstation#*]*username*

Arguments

calendars
Modifies all calendars.

parms Modifies all parameters.

prompts
Modifies all prompts.

resources
Modifies all resources.

cpu Modifies a workstation, workstation class, or domain.

wkstation
The workstation name. Wildcards are permitted.

wkstationclass
The workstation class name. Wildcards are permitted.

domain
The domain name. Wildcards are permitted.

jobs Modifies a job.

wkstation
The name of the workstation or workstation class on which the job runs. Wildcards are permitted. The default is the workstation on which Composer is running.

jobname
The job name. Wildcards are permitted.

sched Modifies a job stream.

wkstation
The name of the workstation or workstation class on which the job stream runs. Wildcards are permitted. The default is the workstation on which Composer is running.

jstream
The job stream name. Wildcards are permitted.

users Modifies a user.

wkstation
The name of the workstation on which the user is defined. Wildcards are permitted. The default is the workstation on which Composer is running.

username
The user name. Wildcards are permitted.

Usage Notes

The **modify** command copies the definition or object list into a temporary file, edits the file, and then adds the contents of the file to the appropriate database file. This is equivalent to the following sequence of commands:

```
create file from object-specification
edit file
add file
```

If the add operation is successful, the edit file is purged. For more information, refer to the descriptions of the create, edit, and add commands in this chapter.

For user definitions, if a password field remains empty when you exit the editor, the old password is retained. To specify a null password use two consecutive double quotes (“”).

Examples

The following example modifies all calendars:

```
modify calendars
```

The following example modifies job stream **sked9** that is launched on workstation **site1**:

```
m sched=site1#sked9
```

new

Adds a new scheduling object. You must have **add**

access for new objects on the workstation. For existing objects, you must have **modify**

access to the object or the affected database file.

Synopsis

new

Usage Notes

The **new** command creates a temporary file, edits the file, and then adds the contents of the file. For calendars, parameters, resources, and prompts, the contents of the file overwrites any existing objects. See “modify” on page 49 to create a temporary file of existing objects.

Examples

The following example creates a temporary file, edits the file, and then adds the contents of the file to the database:

```
new
```

or:

```
n
```

redo

Edits and reexecutes the previous command.

Synopsis

redo

Context

When you execute the **redo** command, Composer displays the previous command, so that it can be edited and reexecuted. Use the spacebar to move the cursor under the character to be modified, and enter the following directives.

Directives

d[*dir***]**	Deletes the character above the **d**. This can be followed by other directives.
i*text*	Inserts text before the character above the **i**.
r*text*	Replaces one or more characters with text, beginning with the character above the **r**. Replace is implied if no other directive is entered.
>*text*	Appends text to the end of the line.
>**d[***dir* **|** *text***]**	Deletes characters at the end of the line. This can be followed by another directive or text.
>**r***text*	Replaces characters at the end of the line with text.

Directive Examples

ddd	Deletes the three characters above the **d**s.
iabc	Inserts **abc** before the character above the **i**.
rabc	Replaces the three characters, starting with the one above the **r**, with **abc**.
abc	Replaces the three characters above **abc** with **abc**.
d diabc	Deletes the character above the first **d**, skips one character, deletes the character above the second **d**, and inserts **abc** in its place.

>abc	Appends **abc** to the end of the line.
>ddabc	Deletes the last two characters in the line, and inserts **abc** in their place.
>rabc	Replaces the last three characters in the line with **abc**.

Examples

The following example inserts a character:

```
redo
dislay site1#sa@
    ip
display site1#sa@
```

The following example replaces a character:

```
redo
display site1#sa@
            2
display site2#sa@
```

replace

Replaces scheduling objects. You must have **modify** access to the objects or the affected database file.

Synopsis

replace *filename*

Arguments

filename

Specifies the name of a file containing one of the following:

- Job definitions. The first line of the file must be **$jobs**.
- Job stream definitions.
- Any combination of workstation, workstation class, and domain definitions.
- User definitions.
- A complete set of calendars, parameters, prompts, or resources.

Usage Notes

The **replace** command is similar to the **add** command, except that there is no prompt to replace existing objects. For more information, refer to “add” on page 32.

Examples

The following example replaces the jobs from the file **myjobs**:

```
replace myjobs
```

The following example replaces the job streams from the file **mysked**:

```
rep mysked
```

The following example replaces all resources with those contained in the file **myres**:

```
rep myres
```

validate

Validates a file containing scheduling objects.

Synopsis

validate *filename*[**;syntax**]

Arguments

filename

Specifies the name of a file that contains calendars, workstations, workstation classes, domains, jobs, parameters, prompts, resources, or job streams. Enter **prodsked** to validate the production schedule file created during pre-production day processing.

syntax

Checks the file for syntax error.

Usage Notes

The **validate** command performs the same syntax checks and validation that are performed when you add or modify objects. It can also be used to validate the production schedule file, **prodsked**, created during the pre-production day processing. **Prodsked** contains the job streams to be run on a particular day, and it can be modified with an editor to include ad hoc changes before processing day begins. **Validate** should be used in these cases to ensure that the production schedule is still valid.

For job streams, if the syntax option is omitted, validation and reporting include the following:

- Verify job names against the master job file.
- Examine dependencies to ensure that the objects exist. For example, a **needs** dependency on a non-existent resource is reported. This will also check for references to non-existent calendars.
- Check for circular dependencies. For example, if **job1** follows **job2**, and **job2** follows **job1** there is a circular dependency.

The output of the **validate** command can be redirected to a file as follows:

```
composer "validate filename" > outfile
```

To include error messages in the output file, use the following:

```
composer "validate filename" > outfile 2>&1
```

Examples

The following example checks the syntax of a file containing workstation definitions:

```
validate mycpus;syntax
```

The following example revalidates all job streams. This can be done to verify the integrity of references to other scheduling objects after changes have been made.

```
create allskeds from sched=@#@
validate allskeds
```

version

Displays Composer’s program banner.

Synopsis

version

Examples

The following example displays Composer’s program banner:

```
version
```

or:

```
v
```

System Command

Executes a system command.

Synopsis

[**:** | **!**] *sys-command*

Arguments

sys-command	Specifies any valid system command. The prefix of colon (:) or exclamation mark (!) is required only when the command is spelled the same as a Composer command.

Examples

The following example executes a **ps** command on UNIX:

```
ps -ef
```

The following example executes a **dir** command on Windows NT:

```
dir \bin
```

2

Scheduling Language

This chapter describes the scheduling language format and keywords used to define job streams for Workload Scheduler. These keywords and syntax are used to define job streams using the command line interface. Job streams created using the graphical user interface do not reference the keywords listed in this chapter, but all job streams in Tivoli Workload Scheduler are saved using the same scheduling language syntax and keywords.

Syntax for Job Streams

The following shows the structure of a job stream, with keywords in **bold**. A job stream begins with a **schedule** keyword followed by attributes and dependencies. The colon delimiter introduces the jobs that comprise the job stream. Each job has its own attributes and dependencies.

schedule *name* **on** *dates* [**except** *dates*]
    [**at** *time*]
    [**carryforward**]
    [**follows** *job|jstream*]
    [**in order**]
    [**limit** *number*]
    [**needs** *resource*]
    [**opens** *file*]
    [**priority** *number*]
    [**prompt** *name|text*]
    [**until** *time*]

:
job-statement
    [**at** *time*]
    [**confirmed**]
    [**every** *rate*]
    [**follows** *job|jstream*]
    [**needs** *resource*]
    [**opens** *file*]
    [**priority** *number*]
    [**prompt** *name|text*]
    [**until** *time*]
[*job-statement***...**]
end

Keywords

A brief description of the scheduling keywords is provided in the following table.

Keyword	Description	Page
at	Defines the time of day that job stream or job execution begins.	64
carryforward	Carries the job stream forward if it is not completed.	66
confirmed	Specifies that the completion of this job requires confirmation.	68
end	Marks the end of a job stream.	69
every	Launches the job repeatedly at a specified rate.	70
except	Specifies dates that are exceptions to the dates the job stream is selected for execution.	71
follows	Specifies not to launch this job or job stream until other jobs and job streams have completed successfully.	73
in order	Launches jobs in this job stream in the order they are listed.	75
job-statement	Defines a job and its dependencies.	77
limit	Sets a limit on the number of jobs that can be launched concurrently from the job stream.	82

Keyword	Description	Page
needs	Defines the number of units of a resource required by the job or job stream before it can be launched.	83
on	Defines the dates on which the job stream is selected for execution.	85
opens	Defines files that must be accessible before the job or job stream is launched.	87
priority	Defines the priority for a job or job stream.	89
prompt	Defines prompts that must be replied to before the job or job stream is launched.	90
schedule	Assigns a name to the job stream.	92
until	Defines a time of day after which the job or job stream is not launched.	93

Dependencies

A dependency is a condition that must be satisfied before a job or job stream is launched. They are to job streams with the **follows**, **needs**, **opens** and **prompt** keywords. The maximum number of dependencies permitted for a job or job stream is 40.

Keyword Descriptions

The following pages describe the syntax for scheduling language keywords.

at

Defines the earliest time a job or job stream will be launched.

Synopsis

at *time* [**timezone|tz** *tzname*][+*n***day**[**s**]]

Arguments

time Specifies a time of day. Possible values can range from **0000** to **2359**.

tzname

Specifies the time zone to be used when computing the start time. See “Time Zone Names” on page 309 for time zone names. The default is the time zone of the workstation on which the job or job stream is launched.

Note: If an **at** time and an **until** time are specified, the time zones must be the same.

n Specifies an offset in days from the scheduled start date and time.

Usage Notes

If an **at** time is not specified for a job or job stream, its launch time is determined by its dependencies and priority.

Examples

The following examples assume that Workload Scheduler’s processing day starts at 6:00 A.M.

- The following job stream, selected on Tuesdays, is launched no sooner than 3:00 A.M. Wednesday morning. Its two jobs are launched as soon as possible after that time.

  ```
  schedule sked7 on tu at 0300:
  job1
  job2
  end
  ```

- The time zone of workstation **sfran** is defined as Pacific Standard Time (**pst**), and the time zone of workstation **nycity** is defined as Eastern Standard Time (**est**). The following job stream is selected for execution on Friday. It is launched on workstation **sfran** at 10:00 A.M. **pst** Saturday. **job1** is launched on **sfran** as soon as possible after that time. **job2** is launched on **sfran** at 2:00 P.M. **est** (11:00 A.M. **pst**) Saturday. **job3** is launched on workstation **nycity** at 4:00 P.M. **est** (1:00 P.M. **pst**) Saturday.

```
sfran#schedule sked8 on fr at 1000 + 1 day :
job1
job2 at 1400 tz est
nycity#job3 at 1600
end
```

carryforward

Makes a job stream eligible to be carried forward to the next day’s production plan if it is not completed before the end of the current day’s production plan.

Synopsis

carryforward

Examples

The following job stream is carried forward if its jobs have not completed before pre-production processing begins for a new day.

```
schedule sked43 on th
carryforward
:
job12
job13
job13a
end
```

Comments

Includes comments in a job stream definitition.

Synopsis

*_text_ | <<_text_>>

Arguments

*_text_ Inserts a comment line. The first character in the line must be an asterisk.

<<_text_>>
Inserts comment text on a line. The text must be enclosed in doulble angle brackets.

Examples

The following example includes both types of comments:

```
****************************************
* The weekly cleanup jobs
****************************************
*
schedule wkend on fr at 1830
in order
:
job1 <<final totals and reports>>
job2 <<update database         >>
end
```

confirmed

Specifies that a job’s completion must be confirmed by executing a Conman **confirm** command. See “confirm” on page 131 for more information.

Synopsis

confirmed

Examples

In the following job stream, confirmation of the completion of **job1** must be received before **job2** and **job3** are launched.

```
schedule test1 on fr:
job1 confirmed
job2 follows job1
job3 follows job1
end
```

end

Marks the end of a job stream definition.

Synopsis

end

Examples

```
schedule test1 on monthend
:
job1
job2
job3
end << end of job stream >>
```

every

Defines the repitition rate for a job. The job is launched repeatedly at the specified rate.

Synopsis

every*rate*

Arguments

rate	The repetition rate expressed in hours and minutes, in the format: *hhmm*. (The rate can be greater than 24 hours.)

Examples

The following example launches the job **testjob** every hour:

```
testjob every 100
```

The following example launches the job **testjob1** every 15 minutes, between the hours of 6:00 P.M. and 8:00 P.M.:

```
testjob1 at 1800 every 15 until 2000
```

except

Defines the dates that are exceptions to a job stream's **on** dates. See "on" on page 85 for more information.

Synopsis

except {*date* | *day* | *calendar*} [,...]

Arguments

date A date in the format: *mm*/*dd*/*yy*.

day A day of the week. Specify one or more of the following:

mo	Monday
tu	Tuesday
we	Wednesday
th	Thursday
fr	Friday
sa	Saturday
su	Sunday
weekdays	Everyday except Saturday and Sunday.

calendar
The dates specified on a calendar by this name. The calendar name can be followed by an offset in the following format:

{+ | -}*n* {**day**[**s**] | **weekday**[**s**] | **workday**[**s**]}

Where:

n The number of days, weekdays, or workdays.

days Includes every day of the week.

weekdays
Includes every day of the week, except Saturday and Sunday.

workdays
Includes every day of the week, except the dates that appear on calendar named **holidays**.

Examples

The following example selects job stream **testskd2** to run every weekday except those days whose dates appear on calendars named **monthend** and **holidays**:

```
schedule testskd2 on weekdays
except monthend,holidays :
```

The following example selects job stream **testskd3** to run every weekday except May 15,1999 and May 23, 1999:

```
schedule testskd3 on weekdays
except 05/15/99,05/23/99 :
```

The following example selects job stream **testskd4** to run every day except two weekdays prior to any date appearing on a calendar named **monthend**:

```
schedule testskd4 on everyday
except monthend-2 weekdays :
```

follows

Defines the other jobs and job streams that must complete successfully before a job or job stream is launched.

Synopsis

Use the following syntax for job streams:

follows [*netagent*::][*wkstation*#]*jstream*[.*jobname*| **@**] [**,**...]

Use the following syntax for jobs:

follows [*netagent*::][*wkstation*#]*jstream*{.*jobname* | **@**} | *jobname* [**,**...]

Arguments

netagent
: The name of the network agent where the inter-network dependency is defined.

wkstation
: The workstation on which the job or job stream that must be complete runs. The default is the same workstation as the dependent job or job stream.

  If a *wkstation* is not specified with *netagent*, the default is the workstation to which the network agent is connected.

jstream
: The name of the job stream that must be complete. For a job, the default is the same job stream as the dependent job.

jobname
: The name of the job that must be complete. An at sign (@) can be used to indicate that all jobs in the job stream must complete successfully.

Examples

The following example specifies to not launch job stream **skedc** until job stream **sked4** on workstation **site1**, and job **joba** in job stream **sked5** on workstation **site2** have completed successfully:

```
schedule skedc on fr
follows site1#sked4,site2#sked5.joba
```

Do not launch sked6 until jobx in the job stream skedx on remote network cluster4 has completed successfully:

```
sked6 follows cluster4::site4#skedx.jobx
```

The following example specifies to not launch **jobd** until **joba** in the same job stream, and **job3** in job stream **skeda** have completed successfully:

```
jobd follows joba,skeda.job3
```

The following example specifies to not launch **jobe** until all jobs in job stream **skedb** on workstation **unix1** have completed successfully:

```
jobe follows unix1#skedb.@
```

in order

Specifies that jobs are to be launched in the order they appear in a job stream. Each job must complete successfully before the next job is launched. This keyword must be the last keyword to appear in hte job stream definition before the colon. No dependencies are permitted for the jobs in the job stream.

Synopsis

in order

Usage Notes

If an **in order** job does not complete successfully, and cannot be rerun successfully, the jobs that follow it must be released with a Conman **release** command. For example, to release the jobs in job stream **sked1**, you can enter the following Conman command:

```
rj sked1.@
```

If an **in order** job is cancelled, the next job must be released with a Conman **release** command. The remaining jobs in the job stream will continue as normal. For example, if job **j1** is cancelled, you can release job **j2** by entering the following Conman command:

```
rj sked1.j2
```

The fault-tolerant agent on which an **in order** job runs must have **Full Status** mode turned on if the job follows a job that runs on a different workstation. This is illustrated in “Examples.” For more information about **Full Status** mode, refer to “Workstation Definitions” on page 2.

Examples

The following example launches **joba**, **jobb** and **jobc**, one at a time, in the order they appear in **sked2**:

```
schedule sked2 on everyday  in order :
  joba
  jobb
  jobc
end
```

The following example launches **job1**, **site2#job2** and **job3**, one at a time, in the order they appear in **sked3**:

```
schedule sked3 on tu,th  in order :
  job1
  site2#job2
  job3
end
```

If the job stream is launched on a fault-tolerant agent (**site3**, for example), job **site2#job2** will not be launched when **job1** completes unless **Full Status** mode is turned on for **site2**. In addition, **job3** will not be launched when **site2#job2** completes unless **Full Status** mode is turned on for **site3**.

Job Statement

Job statements place jobs in a job stream and define job dependencies. In a job statement, you can also include job attributes and recovery options that add a new job or modify an existing job in the database. See “Usage Notes” for more information.

Synopsis

[*wkstation*#]*jobname*
[**description** “*text*”]
[{**scriptname** | **jobfilename**} *filename* | **docommand** “*commandline*”**]**
[streamlogon *username*]
[**interactive**]
[**recovery** {**stop | continue | rerun**}
  [{**recoveryjob** | **after**} [*rwkstation*#]*rjobname*]
  [{**recoveryprompt** | **abendprompt**} “*rtext*”]]
[*job-dependency* [**,**...]]

Arguments

wkstation
: Specifies the name of the workstation or workstation class on which the job runs. The default is the workstation on which the job stream runs. The pound sign (#) is a required delimiter. If you specify a workstation class, it must match the workstation class of any job stream in which the job is included.

jobname
: Specifies the name of the job. The name must start with a letter, and can contain alphanumeric characters, dashes and underscores. For non-expanded databases, it can contain up to eight characters. For expanded databases, it can contain up to 40 characters.

description
: A free-form description of the job, enclosed in double quotes.

{scriptname | jobfilename} *filename*

Specifies the name of the file the job executes. Use **scriptname** for UNIX and Windows NT jobs, and use **jobfilename** for MPE jobs. For an executable file, enter the file name and any options and arguments (up to 4095 characters). You can also use Workload Scheduler parameters. See “Using Parameters in Job Definitions” on page 80 for more information.

For Windows NT jobs, include the file extensions. Universal Naming Convention (UNC) names are permitted. Do not specify files on mapped drives.

If spaces or special characters are included, other than slashes (/) and backslashes (\), the entire string must be enclosed in quotes (″).

If the file name contains spaces, enter the name in another file that does not have spaces in its name and use the second file’s name for this argument.

docommand *command*

Specifies a command that the job executes. Enter a valid command and any options and arguments (up to 4095 characters) enclosed in quotes (″). A command is executed directly and, unlike **scriptname** or **jobfilename**, the configuration script, **jobmanrc**, is not executed. Otherwise, the command is treated as a job, and all job rules apply. You can also enter Workload Scheduler parameter. See “Using Parameters in Job Definitions” on page 80 for more information.

streamlogon

The user name under which the job runs. The name can contain up to 47 characters. If the name contains special characters it must be enclosed in quotes (″). Specify a user that can log on to the workstation on which the job runs. You can also enter Workload Scheduler parameter. See “Using Parameters in Job Definitions” on page 80.

For Windows NT jobs, the user must also have a user definition. See "User Definitions" on page 17 for user requirements.

interactive

For Windows NT jobs, include this keyword to indicate that the job runs interactively on the Windows NT desktop.

recovery

Recovery options for the job. The default is **stop** with no recovery job and no recovery prompt. Enter one of the recovery options, **stop**, **continue**, or **rerun**. This can be followed by a recovery job, a recovery prompt or both.

stop If the job abends, do not continue with the next job.

continue
If the job abends, continue with the next job.

rerun If the job abends, rerun the job.

{recoveryjob | after} [*wkstation***#]***jobname*

Specifies the name of a recovery job to run if the parent job abends. Recovery jobs are run only once for each abended instance of the parent job.

You can specify the recovery job's workstation if it is different than the parent job's workstation. The default is the parent job's workstation. Not all jobs are eligible to have recovery jobs run on a different workstation. Follow these guidelines:

- The parent job's and recovery job's workstation must be running Workload Scheduler Version 4.5.5 or later.
- If either workstation is an extended agent, it must be hosted by a domain manager or a fault-tolerant agent with a value of **on** for **fullstatus**.
- The recovery job's workstation must be in the same domain as the parent job's workstation.

- If the recovery job's workstation is a fault-tolerant agent, it must have a value of **on** for **fullstatus**.

{recoveryprompt | abendprompt} "*text*"
Specifies the text of a recovery prompt, enclosed in quotes, to be displayed if the job abends. The text can contain up to 64 characters. If the text begins with a colon (:), the prompt is displayed, but no reply is required to continue processing. If the text begins with an exclamation mark (!), the prompt is not displayed, but a reply is required to proceed.

job-dependency
Specifies scheduling keywords and arguments. The valid keywords for jobs are: **at**, **confirmed**, **every**, **follows**, **needs**, **opens**, **priority**, **prompt**, and **until**.

Using Parameters in Job Definitions

Parameters have the following uses and limitations in job definitions:

- Parameters are allowed in the **streamlogon**, **scriptname**, and **docommand** values.
- A parameter can be used as an entire string or a part of it.
- Multiple parameters are permitted in a single variable.
- Enclose parameter names in carets (^), and enclose the entire string in quotation marks.

See the example below in which a parameter named **mis** is used in the **streamlogon** value. For additional examples, see "Parameter Definitions" on page 21.

Usage Notes

A job needs to be defined only once in the database, and can be used in multiple job streams. When a job stream is added or modified, the attributes or recovery options of its jobs are also added or modified.

When defining jobs, keep the following in mind:

- Jobs can be defined independently (as described in "Job Definitions" on page 12), or as part of job streams. In either case, the changes are made in the database and do not affect the production plan until the start of a new processing day.

- When you add or replace a job stream, any job modifications affect all other job streams that use the jobs. Note that the Cross Reference Report can be used to determine the names of job streams in which a particular job is included.

Examples

The following example defines a job stream with three previously defined jobs:

```
schedule bkup on fr at 20:00 :
  cpu1#jbk1
  cpu2#jbk2
    needs 1 tape
  cpu3#jbk3
    follows jbk1
end
```

The following job stream definition contains job statements that add or modify the definitions of two jobs in the database:

```
schedule sked4 on mo :
  job1  scriptname "d:\apps\maestro\scripts\jcljob1"
    streamlogon jack
    recovery stop abendprompt "continue production"
  site1#job2  scriptname "d:\apps\maestro\scripts\jcljob2"
    streamlogon jack
    follows job1
end
```

limit

Limits the number of jobs that can execute simultaneously in a job stream.

Synopsis

limit *joblimit*

Arguments

joblimit	Specifies the number of jobs that can be running at the same time in the schedule. Possible values are **0** through **1024**. If you specify **0**, you prevent all jobs from being launched.

Examples

The following example limits to five the number of jobs that can execute simultaneously in job stream **sked2**:

```
schedule sked2 on fr
  limit 5 :
```

needs

Defines resources that must be available before a job or job stream is launched.

Synopsis

needs [*n*] [*wkstation*#]*resourcename* [,...]

Arguments

n	Specifies the number of resource units required. Possible values are **0** through **32**. The default is one.
wkstation	Specifies the name of the workstation on which the resource is defined. The default is the workstation of the dependent job or job stream. Resources can be used as dependencies only by jobs and job streams that run on the same workstation as the resource. However, a standard agent and its host can reference the same resources.
resourcename	Specifies the name of the resource.

Usage Notes

Adding a resource dependency to a job that runs on a fault-tolerant agent (fta), or to a job stream that contains a job that runs on an fta, can prevent the job from executing if the resource is not defined on the same fta. To prevent this, ensure that the workstation definition for the fta has **Full Status** turned on, or, alternatively, ensure that the resource is defined on the fta on which the job runs.

Examples

The following example prevents job stream **sked3** from being launched until three units of **cputime**, and two units of **tapes** become available:

```
schedule sked3 on fr
  needs 3 cputime,2 tapes :
```

The **jlimit** resource has been defined with two available units. The following example allows no more than two jobs to execute concurrently in job stream **sked4**:

```
schedule sked4 on mo,we,fr :
  joba needs 1 jlimit
  jobb needs 1 jlimit
  jobc needs 2 jlimit    <<runs alone>>
  jobd needs 1 jlimit
end
```

on

This is a required job stream keyword that defines when and how often a job stream is selected for execution. The **on** keyword must follow the **schedule** keyword. See “except” on page 71 for more information.

Synopsis

on {*date* | *day* | *calendar* | **request**} [,...]

Arguments

date Specifies a date in the format, *mm*/*dd*/*yy*.

day A day of the week. You can specify one or more of the following:

mo	Monday
tu	Tuesday
we	Wednesday
th	Thursday
fr	Friday
sa	Saturday
su	Sunday
weekdays	Everyday except Saturday and Sunday.

calendar
The dates specified on a calendar by this name. The calendar name can be followed by an offset in the following format:

{+ | -}*n* {**day**[**s**] | **weekday**[**s**] | **workday**[**s**]}

Where:

n	The number of days, weekdays, or workdays.
days	Includes every day of the week.
weekdays	Includes every day of the week, except Saturday and Sunday.

workdays

Includes every day of the week, except the dates that appear on calendar named **holidays**.

request

Select the job stream when requested. This is used for job streams that are selected by name rather than date. See the *Tivoli Workload Scheduler Administrator's Guide* for information about the **schedulr** command.

Examples

The following example selects job stream **sked1** on Mondays and Wednesdays:

```
schedule sked1 on mo,we
```

The following example selects job stream **sked3** on June 15, 1999, and on the dates listed on the **apdates** calendar:

```
schedule sked3 on 6/15/99,apdates
```

The following example selects job stream **sked4** two weekdays before each date appearing on the **monthend** calendar:

```
schedule sked4 on monthend -2 weekdays
```

The following example selects job stream **testskd1** every weekday except Wednesdays:

```
schedule testskd1 on weekdays
  except we
```

The following example selects job stream **testskd3** every weekday except May 15, 1999 and May 24, 1999:

```
schedule testskd3 on weekdays
  except 05/16/99,05/24/99
```

The following example selects job stream **testskd4** every day except two weekdays prior to any date appearing on a calendar named **monthend**:

```
schedule testskd4 on everyday
  except monthend -2 weekdays
```

opens

Specifies files that must be available before a job or job stream can be launched.

Synopsis

opens [*wkstation*#]"*filename*" [**(***qualifier***)**] [**,**...]

Arguments

wkstation

Specifies the name of the workstation or workstation class on which the file exists. The default is the workstation or workstation class of the dependent job or job stream. If you use a workstation class, it must be the same as that of the job stream that includes this statement.

filename

Specifies the name of the file, enclosed in quotation marks. You can use Workload Scheduler parameters as part or all of the file name string. If you use a parameter, it must be enclosed in carets (^).

qualifier

Specifies a valid test condition. On UNIX, the qualifier is passed to a **test** command, which executes as **root** in bin/sh.

On Windows NT, the test function is performed as the maestro user. The valid qualifiers are:

-d %p	True if the file exists and is a directory.
-e %p	True if the file exists.
-f %p	True if the file exists and is a regular file.
-r %p	True if the file exists and is readable.
-s %p	True if the file exists and it's size is greater than zero.
-w %p	True if the file exists and is writeable.

On both UNIX and Windows NT, the expression **%p**, inserts the name of the file.

Entering **(notempty)** is the same as entering **(-s %p)**. If no qualifier is specified, the default is **(-f %p)**.

Usage Notes

The combination of *filename* and *qualifier* cannot exceed 148 characters, within which the base name of the file name cannot exceed 28 characters.

Examples

The following example checks to see that file **c:\users\fred\datafiles\file88** on workstation **nt5** is available for read access before launching **ux2#sked6**:

```
schedule ux2#sked6 on tu opens nt5#"c:\users\fred\datafiles\file88"
```

The following example checks to see if three directories, **/john**, **/mary**, and **/roger**, exist before launching job **jobr2**:

```
jobr2  opens "/users"(-d %p/john -a -d %p/mary -a -d %p/roger)
```

The following example checks to see if **cron** has created its FIFO file before launching job **job6**:

```
job6  opens "/usr/lib/cron/FIFO"(-p %p)
```

The following example checks to see that file **d:\work\john\execit1** on workstation **dev3** exists and is not empty, before running job **jobt2**:

```
jobt2  opens dev3#"d:\work\john\execit1"(notempty)
```

The following example checks to see that file **c:\tech\checker\startf** on workstation **nyc** exists with a size greater than zero, and is writable, before running job **job77**:

```
job77  opens nyc#"C:\tech\checker\startf"(-s %p -a -w %p)
```

priority

Sets the priority of a job or job stream.

Synopsis

priority *number* | **hi** | **go**

Arguments

number	Specifies the priority. Possible values are **0** through **99**. A priority of zero prevents the job or job stream from launching.
hi	The equivalent of priority **100**.
go	The equivalent of priority **101**, the highest priority.

Examples

The following example illustrates the relationship between job stream and job priorities. The jobs are launched in the following order: **job1**, **job2**, **joba**, **jobb**.

```
schedule sked1 on tu                schedule sked2 on tu
priority 50                         priority 10
:                                   :
job1 priority 15                    joba priority 60
job2 priority 10                    jobb priority 50
end                                 end
```

If the job stream priorities were the same, the jobs would be launched in the following order: **joba**, **jobb**, **job1**, **job2**.

prompt

Specifies prompts that must be answered affirmatively before a job or job stream is launched.

Synopsis

prompt *promptname* [**,**...]

prompt ″[**:** | **!**]*text*″ [...]

Arguments

promptname	Specifies the name of a prompt in the database.
text	Specifies a literal prompt as a text string enclosed in quotes (″). Multiple strings separated by backlash **n** (\n) can be used for long messages. If the string begins with a colon (:), the message is displayed but no reply is necessary. If the string begins with an exclamation point (!), the message is not displayed but it requires a reply. You can include backslash **n** (\n) within the text for new lines. You can use one or more parameters as part or all of the text string. To use a parameter, place its name between carets (^).

Note: Within a local prompt, when not designating a parameter, carets (^) must be preceded by a backslash (\) or they will cause errors in the prompt. Within global prompts, carets do not have to be preceded by a backslash.

Examples

The following example illustrates both literal and named prompts. The first prompt is a literal prompt that uses a parameter named **sys1**. When a single affirmative reply is received for the named prompt **apmsg**, the dependencies for both **job1** and **job2** are satisfied.

```
schedule sked3 on tu,th
  prompt "All ap users logged out of ^sys1^? (y/n)"
:
  job1 prompt apmsg
  job2 prompt apmsg
end
```

The following example defines a literal prompt that will appear on more than one line. It is defined with backlash **n** (\n) at the end of each line:

```
schedule sked5 on fr
  prompt  "The jobs in this job stream consume \n
an enormous amount of cpu time.\n
Do you want to launch it now? (y/n)"
:
  j1
  j2 follows j1
end
```

schedule

Specifies the job stream name. With the exception of comments, this must be the first keyword in a job stream, and must be followed by the **on** keyword.

Synopsis

schedule [*wkstation*#]*jstreamname* **on** ...

Arguments

wkstation	Specifies the name of the workstation on which the job stream is launched. The default is the workstation on which Composer runs to add the job stream.
jstreamname	Specifies the name of the job stream. The name must start with a letter, and can contain alphanumeric characters and dashes. For non-expanded databases, it can contain up to eight characters. For expanded databases, it can contain up to 16 characters.
on	Specifies when, or how often, the job stream is selected for execution. See “on” on page 85 for more information.

Examples

The following example names a job stream **sked1** that runs on the workstation on which Composer is running:

```
schedule sked1  on tu
```

The following example names a job stream **sked-2** that runs on the workstation on which Composer is running:

```
schedule sked-2  on everyday  except fr
```

The following example names a job stream **skedux7** that runs on workstation **hpux3**:

```
schedule hpux3#skedux7  on monthend
```

until

Specifies the latest time a job or job stream will be launched.

Synopsis

until *time* [**timezone|tz** *tzname*][+*n***day**[**s**]]

Arguments

time Specifies the time of day. The possible values are **0000** through **2359**.

tzname
Specifies the time zone to be used when computing the time. See "Time Zone Names" on page 309 for time zone names. The default is the time zone of the workstation on which the job or job stream is launched.

Note: If an **until** time and an **at** time are specified, the time zones must be the same.

n Specifies an offset, in days, from the scheduled date and time.

Examples

The following example prevents **sked1** from launching after 5:00 P.M. on Tuesdays:

```
schedule sked1  on tu  until 1700 :
```

The following example launches **job1** between 1:00 P.M. and 5:00 P.M. on weekdays:

```
schedule sked2  on weekdays :
  job1  at 1300  until 1700
end
```

The following example launches **joba** every 15 minutes between 10:30 P.M. and 11:30 P.M. on Mondays:

```
schedule sked3 on mo :
  joba  at 2230  every 0015  until 2330
end
```

The following example launches job stream **sked4** on Sundays between 8:00 A.M. and 1:00 P.M. The jobs are also be launched within this interval:

```
schedule sked4  on fr  at 0800 + 2 days
  until 1300 + 2 days
:
  job1
  job2  at 0900  <<launched on sunday>>
  job3  follows job2  at 1200  <<launched on sunday>>
end
```

3

Conman Reference

Workload Scheduler’s production plan environment is managed with the Conman program. Conman is used to start and stop processing, alter and display the production plan (Symphony), and control workstation linking in a network. This chapter contains information about the following:

- Running Conman
- Selecting and qualifying jobs and job streams
- The syntax and usage for Conman commands

Running Conman

To run Conman, enter the following command:

```
conman ["command[&...][&]"]
```

Examples

In the following example, Conman starts and prompts for a command:

```
conman
```

In the following example, Conman executes the **sj** and **sp** commands, and then quits:

```
conman"sj&sp"
```

In the following example, Conman executes the **sj** and **sp** commands, and then prompts for a command:

```
conman"sj&sp&"
```

In the following example, Conman reads commands from **cfile**:

```
conman < cfile
```

In the following example, commands from **cfile** are piped to Conman:

```
cat cfile | conman
```

Control Characters

You can enter the following control characters to interrupt Conman.

Control+c
Conman stops executing the current command at the next step that can be interrupted, and returns a command prompt.

Control+d
Conman quits after executing the current command.

Executing System Commands

When a system command is entered using a pipe or a system command prefix (: or !), it is executed by a child process. The child process's effective user ID is set to the ID of the user running Conman to prevent security breaches.

User Prompting

When you use wildcard characters to select the objects to be acted upon by a command, Conman prompts for confirmation after finding each matching object. Responding with **yes** allows the action to be taken, and **no** skips the object without taking the action.

When Conman is run interactively, the confirmation prompts are issued at your computer. Pressing the **Return** key in response to a prompt is interpreted as a **no** response. Prompting can be disabled by including the **;noask** option in a command.

Although no confirmation prompts are issued when Conman is not running in interactive mode, it acts as though the response had been **no** in each case, and no objects are acted on. It is important, therefore, to include the **;noask** option on commands when Conman is not run in interactive mode.

Terminal Output

The output to your computer is specified by shell variables named MAESTROLINES and MAESTROCOLUMNS. If either is not set, the standard shell variables, LINES and COLUMNS, are used. The variables can be set as follows:

$MAESTROLINES
Specifes the number of lines per screen. The default is **24**. At the end of each screen page, Conman prompts to continue. If MAESTROLINES (or LINES) is set to zero or a negative number, Conman does not pause at the end of a page.

$MAESTROCOLUMNS
Specifes the number of characters per line. The default is **80**.

$MAESTRO_OUTPUT_STYLE
Specifes the method of displaying object names. If set to **LONG**, full names are displayed. If not set, or set to any value other than **LONG**, and the global option **expanded version** is set to **yes**, long names are truncated to eight characters followed by a plus sign (+). If **expanded version** is set to **no**, long names are truncated to eight characters.

Offline Output

The **;offline** option in Conman commands is generally used to print the output of a command. When you include it, the following shell variables control the output:

$MAESTROLP
Specifies the destination of a command's output. Set it to one of the following:

> *file* Redirects output to a file and overwrites the contents of the file. If the file does not exist, it is created.

>> *file*
Redirects output to a file and appends the output to the end of the file. If the file does not exist, it is created.

| *command*
: Pipes output to a system command or process. The system command is executed whether or not output is generated.

|| *command*
: Pipes outut to a system command or process. The system command is not executed if there is no output.

The default is **| lp -tCONLIST** which pipes the command output to the printer and places the title “CONLIST” in the printout’s banner page.

$MAESTROLPLINES
: Specifies the number of lines per page. The default is **60**.

$MAESTROLPCOLUMNS
: Specifies the number of characters per line. The default is **132**.

The variables must be exported before running Conman.

Conman’s Command Prompt

The Conman command prompt is, by default, a percent sign (%). This is defined in the message catalog file *WShome*/**catalog/maestro.msg**. To select a different prompt,

1. Edit the file.
2. Locate set 202 and message numbers 110 and 111.
3. Change the characters on those lines.

   Message number 110 is the standard prompt and message number 111 is the prompt used after you select a different Symphony file with the **setsym** command. The prompts can be up to eight characters in length.
4. After changing the prompts, generate a new catalog file as follows:

   ```
   cd WShome/catalog
   gencat maestro.cat maestro.msg
   ```

Command Syntax

Conman commands consist of the following elements:

commandname selection arguments

where:

commandname
: Specifies the command name.

selection
: Specifies the object or set of objects to be acted upon.

arguments
: Specifies the command arguments.

The following is an example of a Conman command:

```
sj sked1.@+state=hold˜priority=0;info;offline
```

where:

sj The abbreviated form of the **showjobs** command.

sked1.@+state=hold˜priority=0
: Selects all jobs in the job stream **sked1** that are in the **hold** state with a priority other than zero.

;info;offline
: Arguments for the **showjobs** command.

Wildcard Characters

The following wildcard characters are permitted:

@	Replaces one or more alphanumeric characters.
?	Replaces one alphabetic character.
%	Replaces one numeric character.

Delimiters and Special Characters

The following characters have special meanings in Conman commands:

Char.	Description
&	Command delimiter. See “Running Conman” on page 95.
+	A delimiter used to select objects for commands. It adds an attribute the object must have. For example: `sked1.@+priority=0`
˜	A delimiter used to select objects for commands. It adds an attribute the object must not have. For example: `sked1.@˜priority=0`
;	Argument delimiter. For example: `;info;offline`
,	Repetition and range delimiter. For example: `state=hold,sked,pend`
=	Value delimiter. For example: `state=hold`
: !	Command prefixes that pass the command on to the system. These prefixes are optional; if Conman does not recognize the command, it is passed automatically to the system. For example: `!ls or :ls`
<<>>	Comment brackets. For example: `sj @#@.@ <<`*`comment`*`>>`

*	Comment prefix. The prefix must be the first character on a command line or following a command delimiter. For example:
	`*`*`comment`*
	
	or
	`sj& *`*`comment`*

>	Redirects command output to a file and overwrites the contents of the file. If the file does not exist, it is created. For example: `sj> joblist`
>>	Redirects command output to a file and appends the output to the end of file. If the file does not exist, it is created. For example: `sj >> joblist`

Char.	Description
|	Pipes command output to a system command or process. The system command is executed whether or not output is generated. For example: `sj| grep ABEND`
||	Pipes command output to a system command or process. The system command is not executed if there is no output. For example: `sj || grep ABEND`

List of Commands

The following table lists Conman's command set. Command names and keywords can be entered in either uppercase or lowercase characters, and can be abbreviated to as few leading characters as are needed to distinguish them from each other. Some of the command names also have short forms.

Command	Short Form	Description	Type[1]	Page
addep	**adj | ads**	Adds job or job stream dependencies.	M,F	120 122
altpass		Alters a User object definition password.	M,F	124
altpri	**ap**	Alters job or job stream priorities.	M,F	125
cancel	**cj | cs**	Cancels a job or a job stream.	M,F	126, 129
confirm		Confirms job completion.	M,F	131
console		Assigns the Workload Scheduler console.	M,F,A	133
continue		Ignores the next error.	M,F,A	135
deldep	**ddj | dds**	Deletes job or job stream dependencies.	M,F	136 138
display	**df | dj | ds**	Displays files, jobs, and job streams.	M,F,A[2]	140
exit		Terminates Conman.	M,F,A	142

Command	Short Form	Description	Type[1]	Page
fence		Sets Workload Scheduler's job fence.	M,F,A	143
help[5]		Displays command information.	M,F,A	145
kill		Stops an executing job.	M,F	147
limit	**lc | ls**	Changes a workstation or job stream job limit.	M,F,A[3]	148 150
link	**lk**	Opens workstation links.	M,F,A	151
listsym		Displays a list of Symphony log files.	M,F	155
recall	**rc**	Displays prompt messages.	M,F	157
redo		Edits the previous command.	M,F,A	159
release	**rj | rs**	Releases job or job stream dependencies.	M,F	161 163
reply		Replys to prompt message.	M,F	165
rerun	**rr**	Reruns a job.	M,F	167
resource		Changes the number of resource units.	M,F	171
setsym		Selects a Symphony log file.	M,F	172
showcpus	**sc**	Displays workstation and link information.	M,F,A	173
showdomain		Displays domain information.	M,F,A	177
showfiles	**sf**	Displays information about files.	M,F	179
showjobs	**sj**	Displays information about jobs.	M,F	182
showprompts	**sp**	Displays information about prompts.	M,F	190
showresources	**sr**	Displays information about resources.	M,F	193

Command	Short Form	Description	Type[1]	Page
showschedules	**ss**	Displays information about job streams.	M,F	195
shutdown		Stops Workload Scheduler's production processes.	M,F,A	199
start		Starts Workload Scheduler's production processes.	M,F,A	200
status		Displays Workload Scheduler's production status.	M,F,A	204
stop		Stops Workload Scheduler's production processes.	M,F,A	205
submit	**sbd | sbf | sbj | sbs**	Submits a command, file, job, or job stream.	M,F,A[4]	209 212 215 218
switchmgr		Switches the domain manager.	M,F	220
sys-command		Sends a command to the system.	M,F,A	222
tellop	**to**	Sends a message to the console.	M,F,A	223
unlink		Closes workstation links.	M,F,A	224
version	**v**	Displays Conman's program banner.	M,F,A	227

1. Workstation types: domain managers (M), fault-tolerant agents (F), standard agents (S).
2. You can display only files on a standard agent.
3. You can change the limit of only workstations on a standard agent.

4. You can use **submit job** (**sbj**) and **submit sched** (**sbs**) on a standard agent only if the **mozart** directory on the master domain manager is accessible to the standard agent.
5. Not available on Windows NT.

Selecting Jobs in Commands

For commands that operate on jobs, the target jobs are selected by means of attributes and qualifiers. The job selection syntax is shown below, and described on the following pages.

Synopsis

[*wkstation*#] {*jobstream.job* | *jobnumber*} [+ | ˜*jobqualifier*[...]]

or:

netagent::[*wkstation*#] *jobstream.job*

Arguments

wkstation
: When used with *jobstream.job*, this specifies the name of the workstation on which the job stream runs. When used with *jobnumber*, it specifies the workstation on which the job runs. Wildcard characters are permitted.

jobstream
: Specifies the name of the job stream in which the job runs. Wildcard characters are permitted.

job
: Specifies the name of the job. Wildcard characters are permitted.

jobnumber
: Specifies the job number.

jobqualifier
: See the following section, "Job Qualifiers" on page 105.

netagent
: Specifies the name of the Workload Scheduler network agent that interfaces with the remote Workload Scheduler network

containing the target job. The two colons (::) are a required delimiter. Wildcard characters are permitted. For more information refer to “Network Agent Reference” on page 301.

Job Qualifiers

at[=*time* **|** *lowtime* **|** *hightime* **|** *lowtime***,***hightime***]**

Selects or excludes jobs based on scheduled start time.

time

Specifies the scheduled start time expressed as follows:

hhmm[+*n* **days** | +*date*] [**timezone|tz** *tzname*]

hhmm The hour and minute.

+*n* **days**
The next occurrence of *hhmm* in *n* number of days.

+*date* The next occurrence of *hhmm* on *date*, expressed as *mm/dd/yy*.

timezone|tz *tzname*
The name of the time zone of the job. See “Time Zone Names” on page 309 for valid names.

lowtime
Specifies the lower limit of a time range, expressed in the same format as *time*. Jobs are selected that have scheduled start times on or after this time.

hightime
Specifies the upper limit of a time range, expressed in the same format as *time*. Jobs are selected that have scheduled start times on or before this time.

If **at** is used alone, the range is open-ended, and jobs are selected or excluded if they have any scheduled start time.

confirmed

Selects or excludes jobs that were scheduled using the **confirm** keyword.

every[=*rate* **|** *lowrate***, | ,***highrate* **|** *lowrate***,***highrate***]**

Selects or excludes jobs based in whether or not they have a repetition rate.

rate Specifies the scheduled execution rate, expressed as hour and minute (*hhmm*).

lowrate

Specifies the lower limit of a rate range, expressed in the same format as *rate*. Jobs are selected that have repitition rates equal to or greater than this rate.

highrate

Specifies the upper limit of a rate range, expressed in the same format as *rate*. Jobs are selected that have repitition rates less than or equal to this rate.

If **every** is used alone, the range is open-ended, and jobs are selected or excluded if they have any repetition rate.

finished[=*time* **|** *lowtime***, | ,***hightime* **|** *lowtime***,***hightime***]**

Selects or excludes jobs based on whether or not they have finished.

time Specifies the exact time the jobs finished, expressed as follows:

hhmm [*date*] [**timezone|tz** *tzname*]

hhmm The hour and minute.

date The next occurrence of *hhmm* on date, expressed as *mm*/*dd*/*yy*.

timezone|tz *tzname*

The name of the time zone of the job. See “Time Zone Names” on page 309 for valid names.

lowtime

Specifies the lower limit of a time range, expressed in the same format as *time*. Jobs are selected that finished at or after this time.

hightime
: Specifies the upper limit of a time range, expressed in the same format as *time*. Jobs are selected that finished at or before this time.

If **finished** is used alone, the range is open-ended, and jobs are selected or excluded if they have finished executing.

follows[=[*netagent***::][[***wkstation***#]** *jobstream***.]***job***]**
: Selects or excludes jobs based on whether or not they have a **follows** dependency.

wkstation
: Specifies the name of the workstation on which the prerequisite job runs. Wildcard characters are permitted.

jobstream
: Specifies the name of the job stream in which the prerequisite job runs. Wildcard characters are permitted. If you enter *jobstream***.@**, you specify that the target job follows all jobs in the job stream.

job
: Specifies the name of the prerequisite job. When entered without a *jobstream*, it means that the prerequisite job is in the same job stream as the target job. Wildcard characters are permitted.

netagent
: Specifies the name of the Workload Scheduler network agent that interfaces with the remote Workload Scheduler network containing the prerequisite job. Wildcard characters are permitted. For more information refer to “Network Agent Reference” on page 301.

If **follows** is used alone, jobs are selected or excluded if they have any **follows** dependencies.

logon=*username*

Select jobs based on the user names under which they run. If *username* contains special characters it must be enclosed in quotes (″). Wildcard characters are permitted.

needs[=[*wkstation***#]***resourcename***]**

Selects or excludes jobs based on whether or not they have a resource dependency.

wkstation

Specifies the name of the workstation on which the resource is defined. Wildcard characters are permitted.

resourcename

Specifies the name of the resource. Wildcard characters are permitted.

If **needs** is used alone, jobs are selected or excluded if they have any resource dependency.

opens[=[*wkstation***#]***filename***[(***qualifier***)]]**

Select jobs based on whether or not they have a file dependency.

wkstation

Specifies the name of the workstation on which the file exists. Wildcard characters are permitted.

filename

Specifies the name of the file. The name must be enclosed in quotes (″) if it contains characters other than the following: alphanumerics, dashes (-), slashes (/), backslashes (\), and underscores (_). Wildcard characters are permitted.

qualifier

A valid test condition. If omitted, jobs are selected or excluded without regard to a qualifier.

If **opens** is used alone, jobs are selected or excluded if they have any file dependency.

priority=*pri* **|** *lowpri***, | ,***highpri* **|** *lowpri***,***highpri*
Selects or excludes jobs based on their priorities.

pri Specifies the priority value. You can enter **0** through **99**, **hi** or **go**.

lowpri Specifies the lower limit of a priority range. Jobs are selected with priorities equal to or greater than this value.

highpri
Specifies the upper limit of a priority range. Jobs are selected with priorities less than or equal to this value.

prompt[=*promptname* **|** *msgnum***]**
Selects or excludes jobs based on whether or not they have a prompt dependency.

promptname
Specifies the name of a global prompt. Wildcard characters are permitted.

msgnum
Specifies the message number of a local prompt.

If **prompt** is used alone, jobs are selected or excluded if they have any prompt dependency.

recovery=*recv-option*
Selects or excludes jobs based on their recovery options.

recv-option
Specifies the job recovery option as **stop**, **continue**, or **rerun**.

scriptname=*filename*
Selects or excludes jobs based on their executable file names.

filename
Specifies the name of an executable file. The name must be enclosed in quotes (″) if it contains characters other than the following: alphanumerics,

dashes (-), slashes (/), backslashes (\), and underscores (_). Wildcard characters are permitted.

started[=*time* **|** *lowtime***, | ,***hightime* **|** *lowtime***,***hightime***]**

Selects or excludes jobs based on whether or not they have started.

time Specifies the exact time the jobs started, expressed as follows:

hhmm [*date*] [**timezone|tz** *tzname*]

hhmm The hour and minute.

date The next occurrence of *hhmm* on date, expressed as *mm*/*dd*/*yy*.

timezone|tz *tzname*

The name of the time zone of the job. See “Time Zone Names” on page 309 for valid names.

lowtime

Specifies the lower limit of a time range, expressed in the same format as *time*. Jobs are selected that started at or after this time.

hightime

Specifies the upper limit of a time range, expressed in the same format as *time*. Jobs are selected that started at or before this time.

If **started** is used alone, the range is open-ended, and jobs are selected or excluded if they have started.

state=*state***[,...]**

Selects or excludes jobs based on their states.

state Specifies the current state of the job. Valid job states are as follows:

abend The job terminated with a non-zero exit code.

abenp	An **abend** confirmation was received, but the job is not completed.
add	The job is being submitted.
done	The job completed in an unknown state.
error	For internetwork dependencies only, an error occurred while checking for the remote status.
exec	The job is executing.
extrn	For internetwork dependencies only, the status is unknown. An error occurred, a rerun action was just performed on the job in the **external** job stream, or the remote job or job stream does not exist.
fail	Unable to launch the job.
fence	The job's priority is below the fence.
hold	The job is awaiting dependency resolution.
intro	The job is introduced for launching by the system.
pend	The job completed, and is awaiting confirmation.
ready	The job is ready to launch, and all dependencies are resolved.
sched	The job's **at** time has not arrived.
succ	The job completed with an exit code of zero.
succp	A **succ** confirmation was received, but the job is not completed.
susp	The job was suspended by a **breakjob** command. (MPE only)
wait	The job is in the **wait** state. (Extended agent and MPE only)

waitd The job is in the **wait** state, and is deferred. (MPE only)

until[=*time* **|** *lowtime***, | ,***hightime* **|** *lowtime***,***hightime***]**

Selects or excludes jobs based on their scheduled end time.

time Specifies the scheduled end time expressed as follows:

hhmm[+*n* **days** | *date*] [**timezone|tz** *tzname*]

hhmm The hour and minute.

+*n* **days**

The next occurrence of *hhmm* in *n* number of days.

+*date* The next occurrence of *hhmm* on *date*, expressed as *mm/dd/yy*.

timezone|tz *tzname*

The name of the time zone of the job. See “Time Zone Names” on page 309 for valid names.

lowtime

Specifies the lower limit of a time range, expressed in the same format as *time*. Jobs are selected that have scheduled end times on or after this time.

hightime

Specifies the upper limit of a time range, expressed in the same format as *time*. Jobs are selected that have scheduled end times on or before this time.

If **until** is used alone, the range is open-ended, and jobs are selected or excluded if they have any scheduled end time.

Selecting Job Streams in Commands

For commands that operate on job streams, the target job streams are selected by means of attributes and qualifiers. The job stream selection syntax is shown below, and described on the following pages.

Synopsis

[*wkstation*#] *jobstream*[+ | ˜*jobstreamqual*[...]]

Arguments

wkstation
: Specifies the name of the workstation on which the job stream runs. Wildcard characters are permitted.

jobstream
: Specifies the name of the job stream. Wildcard characters are permitted.

jobstreamqual
: See "Job Stream Qualifiers" below.

Job Stream Qualifiers

at[=*time* **|** *lowtime***, | ,***hightime* **|** *lowtime***,***hightime***]**

Selects or excludes job streams based on scheduled start time.

time Specifies the scheduled start time expressed as follows:

hhmm[+*n* **days** | *date*] [**timezone|tz** *tzname*]

hhmm The hour and minute.

+*n* **days**
: The next occurrence of *hhmm* in *n* number of days.

+*date* The next occurrence of *hhmm* on *date*, expressed as *mm/dd/yy*.

timezone|tz *tzname*
: The name of the time zone of the job stream. See "Time Zone Names" on page 309 for valid names.

lowtime
: Specifies the lower limit of a time range, expressed in the same format as *time*. Job streams are selected that have scheduled start times on or after this time.

hightime
Specifies the upper limit of a time range, expressed in the same format as *time*. Job streams are selected that have scheduled start times on or before this time.

If **at** is used alone, the range is open-ended, and job streams are selected or excluded if they have any scheduled start time.

carriedforward
Selects or excludes job streams that were carried forward.

carryforward
Selects or excludes job streams that were scheduled using the **carryforward** keyword.

finished[=*time* **|** *lowtime***, | ,***hightime* **|** *lowtime***,***hightime***]**
Selects or excludes job streams based on whether or not they have finished.

time Specifies the exact time the job streams finished, expressed as follows:

hhmm [*date*] [**timezone|tz** *tzname*]

hhmm The hour and minute.

date The next occurrence of *hhmm* on date, expressed as *mm*/*dd*/*yy*.

timezone|tz *tzname*
The name of the time zone of the job stream. See "Time Zone Names" on page 309 for valid names.

lowtime
Specifies the lower limit of a time range, expressed in the same format as *time*. Job streams are selected that finished at or after this time.

hightime
Specifies the upper limit of a time range, expressed

in the same format as *time*. Job streams are selected that finished at or before this time.

If **finished** is used alone, the range is open-ended, and job streams are selected or excluded if they have finished executing.

follows[=[*netagent***::][***wkstation***#]** *jobstream***[.***job***]**
Selects or excludes job streams based on whether or not they have a **follows** dependency.

netagent
Specifies the name of the network agent that interfaces with the remote Workload Scheduler network containing the prerequisite job or job stream. Wildcard characters are permitted. For more information about network agents, refer to “Network Agent Reference” on page 301.

wkstation
Specifies the name of the workstation on which the prerequisite job or job stream runs. Wildcard characters are permitted.

jobstream
Specifies the name of the prerequisite job stream. Wildcard characters are permitted.

job Specifies the name of the prerequisite job. Wildcard characters are permitted.

If **follows** is used alone, job streams are selected or excluded if they have any **follows** dependency.

limit[=*limit* **|** *lowlimit***, | ,***highlimit* **|** *lowlimit***,***highlimit***]**
Selects or excludes job streams based on whether or not they have a job limit.

limit Specifies the exact job limit value.

lowlimit
: Specifies the lower limit of range. Job streams are selected that have job limits equal to or greater than this limit.

highlimit
: Specifies the upper limit of a range. Job streams are selected that have job limits less than or equal to this limit.

If **limit** is used alone, the range is open-ended, and job streams are selected or excluded if they have any job limit.

needs[=[*wkstation***#]***resourcename***]**
Selects or excludes job streams based on whether or not they have a resource dependency.

wkstation
: Specifies the name of the workstation on which the resource is defined. Wildcard characters are permitted.

resourcename
: Specifies the name of the resource. Wildcard characters are permitted.

If **needs** is used alone, job streams are selected or excluded if they have any resource dependency.

opens[=[*wkstation***#]***filename***[(***qualifier***)]]**
Selects or excludes job streams based on whether or not they have a file dependency.

wkstation
: Specifies the name of the workstation on which the file exists. Wildcard characters are permitted.

filename
: Specifies the name of the file. The name must be enclosed in quotes (″) if it contains characters other than the following: alphanumerics, dashes (-), slashes (/), backslashes (\), and underscores (_). Wildcard characters are permitted.

qualifier
A valid test condition. If omitted, job streams are selected or excluded without regard to a qualifier.

If **opens** is used alone, job streams are selected or excluded if they have any file dependency.

priority=*pri* **|** *lowpri***, |** **,***highpri* **|** *lowpri***,***highpri*
Selects or excludes job streams based on their priorities.

pri Specifies the priority value. You can enter **0** through **99**, **hi** or **go**.

lowpri Specifies the lower limit of a priority range. Job streams are selected with priorities equal to or greater than this value.

highpri
Specifies the upper limit of a priority range. Job streams are selected with priorities less than or equal to this value.

prompt[=*promptname* **|** *msgnum***]**
Selects or excludes job streams based on whether or not they have a prompt dependency.

promptname
Specifies the name of a global prompt. Wildcard characters are permitted.

msgnum
Specifies the message number of a local prompt.

If **prompt** is used alone, job streams are selected or excluded if they have any prompt dependency.

started[=*time* **|** *lowtime***, |** **,***hightime* **|** *lowtime***,***hightime***]**
Selects or excludes job streams based on whether or not they have started.

time Specifies the exact time the job streams started, expressed as follows:

hhmm [*date*] [**timezone|tz** *tzname*]

hhmm	The hour and minute.
date	The next occurrence of *hhmm* on date, expressed as *mm/dd/yy*.

timezone|tz *tzname*
The name of the time zone of the job stream. See “Time Zone Names” on page 309 for valid names.

lowtime
Specifies the lower limit of a time range, expressed in the same format as *time*. Job streams are selected that started at or after this time.

hightime
Specifies the upper limit of a time range, expressed in the same format as *time*. Job streams are selected that started at or before this time.

If **started** is used alone, the range is open-ended, and job streams are selected or excluded if they have started executing.

state=*state***[,...]**
Selects or excludes job streams based on their states.

state Specifies the current state of the job stream. Valid job stream states are as follows:

abend	The job stream terminated abnormally.
exec	The job stream is executing.
hold	The job stream is awaiting dependency resolution.
ready	The job stream is ready to launch, and all dependencies resolved.
stuck	Execution is interrupted. No jobs are launched without operator intervention.
succ	The job stream completed successfully.

until[=*time* **|** *lowtime***, | ,***hightime* **|** *lowtime***,***hightime***]**
Selects or excludes job streams based on scheduled end time.

time Specifies the scheduled end time expressed as follows:

hhmm[+*n* **days** | *date*] [**timezone|tz** *tzname*]

hhmm The hour and minute.

+*n* **days**
The next occurrence of *hhmm* in *n* number of days.

+*date* The next occurrence of *hhmm* on *date*, expressed as *mm/dd/yy*.

timezone|tz *tzname*
The name of the time zone of the job stream. See "Time Zone Names" on page 309 for valid names.

lowtime
Specifies the lower limit of a time range, expressed in the same format as *time*. Job streams are selected that have scheduled end times on or after this time.

hightime
Specifies the upper limit of a time range, expressed in the same format as *time*. Job streams are selected that have scheduled end times on or before this time.

If **until** is used alone, the range is open-ended, and job streams are selected or excluded if they any scheduled end time.

Command Descriptions

The following pages contain detailed descriptions of Conman commands.

Note: In the commands, the terms sched and schedule refer to jobstreams, and the term cpu refers to workstations.

adddep job

Adds dependencies to a job.

You must have **adddep** access to the job. To include **needs** and **prompt** dependencies, you must have **use** access to the resources and global prompts.

Synopsis

adj *jobselect*[**;***dependency*[**;**...]][**;noask**]

Arguments

jobselect
: See “Selecting Jobs in Commands” on page 104.

dependency
: The type of dependency. Specify one of the following. Wildcard characters are not permitted.

  at=*hhmm*[**+***n* **days** | *mm/dd/yy*]

  confirmed

  every=*rate*

  follows=[*netagent***::**][*wkstation***#**]*jstream*{*.job* | **@**} | *job*[**,**...]

  needs=[*num*] [*wkstation***#**]*resource*[**,**...]

  opens=[*wkstation***#**]*"filename"*[**(***qualifier***)**][**,**...] **priority**[=*pri* | **hi** | **go**]

  prompt="[**:** | **!**]*text*" | *promptname*[**,**...]

  until=*hhmm*[**+***n* **days** | *mm/dd/yy*]

noask Specifies not to prompt for confirmation before taking action on each qualifying job.

Usage Notes

If you do not specify a value for **priority**, the job reverts to its original scheduled priority. If you do not specify a workstation in **follows**, **needs**, or **opens**, the default is the workstation on which the job runs. To add prompt dependencies while running Conman on a fault tolerant agent, you must have access to the *WShome*/**mozart** directory on the master domain manager.

Examples

The following example adds a resource dependency to **job3** in job stream **sked9**:

```
adddep sked9.job3;needs=2 tapes
```

The following example adds a file dependency, and an **until** time to **job6** in job stream **sked2**:

```
adj sked2.job6;opens="/usr/lib/prdata/file5"(-s %p);
until=2330
```

adddep sched

Adds dependencies to a job stream.

You must have **adddep** access to the job stream. To include **needs** and **prompt** dependencies, you must have **use** access to the resources and global prompts.

Synopsis

ads *jstreamselect*[**;***dependency*[**;**...]][**;noask**]

Arguments

jstreamselect
: See “Selecting Job Streams in Commands” on page 112.

dependency
: The type of dependency. Specify one of the following. Wildcard characters are not permitted.

at=*hhmm*[**+***n* **days** | *mm/dd/yy*]

carryforward

follows=[*netagent***::**][*wkstation***#**]*jstream*{**.***job* | **@**} | *job*[**,**...]

limit=*limit*

needs=[*num*] [*wkstation***#**]*resource*[**,**...]

opens=[*wkstation***#**]*″filename″*[**(***qualifier***)**][**,**...] **priority**[=*pri* | **hi** | **go**]

prompt=″[**:** | **!**]*text*″ | *promptname*[**,**...]

until=*hhmm*[**+***n* **days** | *mm/dd/yy*]

noask
: Specifies not to prompt for confirmation before taking action on each qualifying job stream.

Usage Notes

If you do not specify a value for **priority**, the job stream reverts to its original scheduled priority. If you do not specify a workstation in **follows**, **needs**, or **opens**, the default is the workstation on which the job stream runs. To add prompt dependencies while running Conman on a fault tolerant agent, you must have access to the *WShome*/**mozart** directory on the master domain manager.

Examples

The following example adds a prompt dependency to job stream **sked3**:

```
adddep sched=sked3;prompt=msg103
```

The following example adds a follows dependency and a job limit to job stream **sked4**:

```
ads sked4;follows=sked3;limit=2
```

altpass

Alters the password of a user object in the current production plan.

You must have **altpass** access to the user object.

Synopsis

altpass [*wkstation*#]*username*[**;**″*password*″]

Arguments

wkstation	Specifies the workstation on which the user is defined. The default is the workstation on which you are running Conman.
username	Specifies the name of a user.
password	Specifies the new password. It must be enclosed in quotes. To indicate no password for the user, use two consecutive quotes (″″).

Usage Notes

If you do not specify a *password*, Conman prompts for a password and a confirmation. In this case, the password is not displayed as it is entered and should not be enclosed in quotes. Note that the change is made only in the current production plan, and is therefore temporary. To make a permanent change see “User Definitions” on page 17.

Examples

The following example changes the password of user **jim** on workstation **mis5** to **giraffe**:

```
altpass mis5#jim;"giraffe"
```

The following example changes the password of user **jim** on workstation **mis5** to **giraffe** without displaying the password:

```
altpass mis5#jim
password: xxxxxxxx
confirm: xxxxxxxx
```

altpri

Alters the priority of a job or job stream.

You must have **altpri** access to the job or job stream.

Synopsis

ap *jobselect* | *jstreamselect*[**;***pri*][**;noask**]

Arguments

jobselect	See “Selecting Jobs in Commands” on page 104.
jstreamselect	See “Selecting Job Streams in Commands” on page 112.
pri	Specifies the priority level. You can enter a value of **0** through **99**, **hi**, or **go**.
noask	Specifies not to prompt for confirmation before taking action on each qualifying job or job stream.

Examples

The following example changes the priority of the **balance** job in job stream **glmonth**:

```
altpri glmonth.balance;55
```

The following example changes the priority of all jobs in job stream **mis5**:

```
ap mis5.@;25
```

The following example changes the priority of job stream **glmonth**:

```
altpri glmonth;10
```

The following example changes the priority of job stream **mis5**:

```
ap mis5;30
```

cancel job

Cancels a job.

You must have **cancel** access to the job.

Synopsis

cj *jobselect*[**;pend**][**;noask**]

Arguments

jobselect	See *“Selecting Jobs in Commands” on page 104.*
pend	Cancels the job only after its dependencies are resolved.
noask	Specifies not to prompt for confirmation before taking action on each qualifying job.

Usage Notes

If a job is cancelled before it is launched, it will not launch. If you cancel a job after it is launched, the job continues to execute. If an executing job is cancelled and it completes in the **abend** state, no automatic job recovery steps are attempted.

If you do not use the **;pend** option, jobs and job streams that are dependent on the cancelled job are released immediately from the dependency.

If you include the **;pend** option, and the job has not been launched, cancellation is deferred until all of the dependencies, including an **at** time, are resolved. Once all the dependencies are resolved, the job is cancelled and any jobs or job streams that are dependent on the cancelled job are released from the dependency. During the period the cancel is deferred, the notation **[Cancel Pend]** is listed in the Dependencies column of the job in a **showjobs** display.

If you include the **;pend** option and the job has already been launched, the option is ignored, and any jobs or job streams that are dependent on the cancelled job are immediately released from the dependency.

You can use the **rerun** command to rerun jobs that have been cancelled, or that are marked **[Cancel Pend]**. You can also add and delete dependencies on jobs that are marked **[Cancel Pend]**.

To immediately cancel a job that is marked **[Cancel Pend]**, you can either enter a **release** command for the job or enter another **cancel** command without the **;pend** option.

For jobs with expired **until** times, the notation **[Until]** is listed in the Dependencies column in a **showjobs** display, and their dependencies are no longer evaluated. If such a job is also marked **[Cancel Pend]**, it is not cancelled until you release or delete the **until** time, or enter another **cancel** command without the **;pend** option.

To stop evaluating dependencies, set the priority of a job to zero with the **altpri** command. To resume dependency evaluation, set the priority to a value greater than zero.

Note: In the case of internetwork dependencies, cancelling a job in the **external** job stream releases all local jobs and job streams from the dependency. Jobs in the **external** job stream represent jobs and job streams that have been specified as internetwork dependencies. The status of an internetwork dependency is not checked after a **cancel** is performed. For more information see “Internetwork Dependencies” on page 304.

Examples

The following example cancels job **report** in job stream **apwkly** on workstation **site3**:

```
cancel site3#apwkly.report
```

The following example cancels job **setup** in job stream **mis5** if it is not in the **abend** state:

```
cj mis5.setup~state=abend
```

The following example cancels job **job3** in job stream **sked3** only after its dependencies are resolved:

```
cj sked3.job3;pend
```

cancel sched

Cancels a job stream.

You must have **cancel** access to the job stream.

Synopsis

cs *jstreamselect*[**;pend**][**;noask**]

Arguments

jstreamselect	See *"Selecting Job Streams in Commands" on page 112*.
pend	Cancels the job stream only after its dependencies are resolved.
noask	Specifies not to prompt for confirmation before taking action on each qualifying job stream.

Usage Notes

If a job stream is cancelled before it is launched, it will not launch. If cancelled after it is launched, jobs that have started are allowed to complete, but no other jobs are launched.

If you do not use the **;pend** option, jobs and job streams that are dependent on the cancelled job stream are released immediately from the dependency.

If you use the **;pend** option and the job stream has not been launched, cancellation is deferred until all of its dependencies, including an **at** time, are resolved. Once all dependencies are resolved, the job stream is cancelled and any dependent jobs or job streams are released from the dependency. During the period the **cancel** is deferred, the notation **[Cancel Pend]** is listed in the Dependencies column of a **showschedules** display.

If you include the **;pend** option and the job stream has already been launched, any remaining jobs in the job stream are cancelled, and any dependent jobs and job streams are released from the dependency.

To immediately cancel a job stream that is marked **[Cancel Pend]**, you can either enter a **release** command for the job stream or enter another **cancel** command without the **;pend** option.

For job streams with expired **until** times, the notation **[Until]** appears in the Dependencies column of the **showschedules** display and their dependencies are no longer evaluated. If such a job stream is also marked **[Cancel Pend]**, it is not cancelled until you release or delete the **until** time or enter another **cancel** command without the **;pend** option.

To stop evaluating of dependencies, set the job stream's priority to zero with the **altpri** command. To resume dependency evaluation, set the priority to a value greater than zero.

Examples

The following example cancels job stream **sked1** on workstation **site2**:

```
cancel site2#sked1
```

The following example cancels job stream **mis2** if it is in the **stuck** state:

```
cs mis2+state=stuck
```

The following example cancels job stream **sked3** only after its dependencies are resolved:

```
cs sked3;pend
```

confirm

Confirms the completion of a job that was scheduled with the **confirmed**

keyword. An exception is noted in the “Usage Notes” section.

You must have **confirm** access to the job.

Synopsis

confirm *jobselect***;**{**succ** | **abend**}[**;noask**]

Arguments

jobselect	See *“Selecting Jobs in Commands” on page 104.*
succ	Confirms that the job ended successfully.
abend	Confirms that the job ended unsuccessfully.
noask	Specifies not to prompt for confirmation before taking action on each qualifying job.

Usage Notes

Changing the state of a job from **abend** to **succ** does not require that the **confirmed** keyword be used to schedule the job. For more information about job confirmation, see “confirmed” on page 68. For more information about **external** jobs, see “Internetwork Dependencies” on page 304.

The following table shows the affect of the **confirm** command on the various states of jobs:

Initial Job State	State after confirm ;succ	State after confirm ;abend
ready	no affect	no affect
hold	no affect	no affect
exec	**succp**	**abenp**
abenp	**succp**	no affect
succp	no affect	no affect
pend	**succ**	**abend**

Initial Job State	**State after confirm ;succ**	**State after confirm ;abend**
done	**succ**	**abend**
succ	no affect	no affect
abend	**succ**	no affect
fail	no affect	no affect
susp	no affect	no affect
skel	no affect	no affect
any **external** job	**succ**	**abend**

Examples

The following example issues a **succ** confirmation for **job3** in job stream **misdly**:

```
confirm misdly.job3;succ
```

The following example issues an **abend** confirmation to job number **234**:

```
conf 234;abend
```

console

Assigns the Workload Scheduler console and sets the message level.

You must have **console** access to the workstation.

Synopsis

console [**sess** | **sys**][**;level=***msglevel*]

Arguments

sess	Sends Workload Scheduler's console messages and prompts to standard output.
sys	Stops sending Workload Scheduler's console messages and prompts to standard output. This occurs automatically when you exit Conman.
msglevel	The level of Workload Scheduler messages that are sent to the console. Specify one of the following levels:

0	No messages. This is the default on fault-tolerant agents.
1	Exception messages such as operator prompts, and job abends.
2	Level 1, plus job stream successful messages.
3	Level 2, plus job successful messages. This is the default on the master domain manager.
4	Level 3, plus job launched messages.

Usage Notes

If you enter the **console** command with no options, the current state of the console is displayed.

By default, Workload Scheduler's control processes write console messages and prompts to standard list files. On UNIX, you can also have them sent to the **syslog** daemon.

Examples

The following example begins writing console messages and prompts to standard output and changes the message level to **1**:

```
console sess;level=1
```

The following example stops writing console messages and prompts to standard output and changes the message level to **4**:

```
cons sys;l=4
```

The following example displays the current state of the console:

```
cons
Console is #J675, level 2, session
```

675 is the process ID of the user's shell.

continue

Ignores the next command error.

Synopsis

continue

Usage Notes

This command is useful when commands are entered non-interactively. It instructs Conman to continue executing commands even if the next command, following **continue**, results in an error.

Examples

In the following example, Conman continues with the **rerun** command even if the **cancel** commands fails:

```
conman "continue&cancel=176&rerun job=sked5.job3"
```

deldep job

Deletes dependencies from a job.

You must have **deldep** access to the job.

Synopsis

ddj *jobselect***;***dependency*[**;**...][**;noask**]

Arguments

jobselect

See *"Selecting Jobs in Commands" on page 104.*

dependency

The type of dependency. Specify at lease one of the following. You can use wildcard characters in *wkstation, jstream, job, resource, filename,*and *promptname*.

at

confirmed

every

follows[=[*netagent***::**][*wkstation*#]*jstream*{*.job* | **@**} | *job*[**,**...]]

needs[=[*num*] [*wkstation*#]*resource*[**,**...]]

opens[=[*wkstation*#]″*filename*″[**(***qualifier***)**][**,**...]]

priority

prompt[=″[**:** | **!**]*text*″ | *promptname*[**,**...]]

until

noask Specifies not to prompt for confirmation before taking action on each qualifying job.

Usage Notes

If **priority** is deleted, the job reverts to its original scheduled priority. When you delete an **opens** dependency, you can include only the base file name and Conman performs a case-insensitive search for matching files, ignoring the directory names. Dependencies on all matching files are deleted.

Examples

The following example deletes a resource dependency from **job3** in **sked5**:

```
deldep sked5.job3;needs=2 tapes
```

The following example deletes all **follows** dependencies from **job4** in **sked3**:

```
ddj sked3.job4;follows
```

deldep sched

Deletes dependencies from a job stream.

You must have **deldep** access to the job stream.

Synopsis

dds *jstreamselect***;***dependency*[**;**...][**;noask**]

Arguments

jstreamselect
: See *"Selecting Jobs in Commands" on page 104.*

dependency
: The type of dependency. Specify at least one of the following. You can use wildcard characters in *wkstation, jstream, job, resource, filename,*and *promptname*.

  at

  carryforward

  follows[=[*netagent***::**][*wkstation*#]*jstream*{.*job* | **@**} | *job*[**,**...]]

  limit

  needs[=[*num*] [*wkstation*#]*resource*[**,**...]]

  opens[=[*wkstation*#]″*filename*″[**(***qualifier***)**][**,**...]]

  priority

  prompt[=″[**:** | **!**]*text*″ | *promptname*[**,**...]]

  until

noask
: Specifies not to prompt for confirmation before taking action on each qualifying job stream.

Usage Notes

If **priority** is deleted, the job reverts to its original scheduled priority. When you delete an **opens** dependency, you can include only the base file name, and Conman performs a case-insensitive search for matching files, ignoring the directory names. Dependencies on all matching files are deleted.

Examples

The following example deletes a resource dependency from **job3** in **sked5**:

```
deldep sked5.job3;needs=2 tapes
```

The following example deletes all **follows** dependencies from **job4** in **sked3**:

```
ddj sked3.job4;follows
```

display

Displays a job file or a job stream definition.

If you specify a file by name, you must have read access to the file. For job files and job stream definitions, you must have **display** access to the job or job stream.

Synopsis

df*filename*[**;offline**]

dj*jobselect*[**;offline**]

ds*jstreamselect*[**;offline**]

Arguments

filename	Specifies the name of the file, usually a job script file. The name must be enclosed in quotes (″) if it contains characters other than the following: alphanumeric characters, dashes (-), slashes (/), backslashes (\), and underscores (_). Wildcard characters are permitted. The file must be accessible from your login workstation.
jobselect	The job whose job file is displayed. See *“Selecting Jobs in Commands” on page 104*. The job file must be accessible from your login workstation.
jstreamselect	The job stream whose definition is displayed. See *“Selecting Job Streams in Commands” on page 112*. Workload Scheduler’s **mozart** directory on the master domain manager must be accessible from your login workstation.
offline	Sends the output of the command to the Conman output device. For information about this device, see “Offline Output” on page 97.

Examples

The following example displays the file **c:\maestro\jclfiles\arjob3**:

```
df c:\apps\maestro\jclfiles\arjob3
```

The following example displays the script file for **job6** in **sked3** offline:

```
dj sked3.job6;off
```

The following example displays the job stream definition for **sked9**:

```
ds sked9
```

exit

Exits the Conman program.

Synopsis

e

Usage Notes

When you are in help mode on UNIX, this command returns Conman to command-input mode.

Examples

The following examples exit the Conman program:

```
exit
```

```
e
```

fence

Changes the job fence on a workstation. Jobs are not launched on the workstation if their priorities are less than or equal to the job fence.

You must have **fence** access to the workstation.

Synopsis

fence *workstation*;*pri*[**;noask**]

Arguments

workstation	Specifies the workstation name. The default is your login workstation.
pri	Specifies the priority level. You can enter **0** through **99**, **hi**, or **go**. Entering **system** sets the job fence to zero.
noask	Specifies not to prompt for confirmation before taking action on each qualifying workstation.

Usage Notes

The job fence prevents low priority jobs from being launched, regardless of the priorities of their job streams. It is possible, therefore, to hold back low priority jobs in high priority job streams, while allowing high priority jobs in low priority job streams to be launched.

When you first start Workload Scheduler following installation, the job fence is set to zero. Once you change the job fence, it is carried forward during pre-production processing to the next day's production plan.

To display the current setting of the job fence, use the **status** command.

Examples

The following example changes the job fence on workstation **site4**:

```
fence site4;20
```

The following example changes the job fence on the workstation on which you are running Conman:

```
f ;40
```

The following example prevents all jobs from being launched by Workload Scheduler on workstation **tx3**:

```
f tx3;go
```

The following example changes the job fence to zero on the workstation on which you are running Conman:

```
f ;system
```

help

Displays help information about commands. Not available on Windows NT.

Synopsis

help *command*

Arguments

command

Specifies the name of a Conman or system command. For Conman commands, enter the full command name, abbreviations and short forms are not supported. In the case of commands consisting of two words, enter the first word, and help for all versions of the command is displayed. For example, entering **help display** will display information about the **display file**, **display job**, and **display sched**commands. You can also enter the following keywords:

commands
Lists all Conman commands.

jobselect
Lists information about selecting jobs for commands.

jobstates
Lists information about job states.

schedselect
Lists information about selecting job streams for commands.

schedstates
Lists information about job stream states.

Examples

The following example displays a list of all Conman commands:

```
help commands
```

The following example displays information about the **fence** command:

```
help fence
```

The following example displays information about the **altpri job** and **altpri sched** commands:

```
h altpri
```

kill

Stops an executing job. On UNIX, this is accomplished with a UNIX **kill**

command.

Synopsis

kill *jobselect*[**;noask**]

Arguments

jobselect	See *"Selecting Jobs in Commands" on page 104.*
noask	Specifies not to prompt for confirmation before taking action on each qualifying job.

Usage Notes

The **kill** operation is not performed by Conman, it is executed by a Workload Scheduler production process, so there might be a short delay.

Killed jobs terminate in the **abend** state. Any jobs or job streams that are dependent on a killed job are not released. Killed jobs can be rerun.

Examples

The following example kills the job **report** in job stream **apwkly** on workstation **site3**:

```
kill site3#apwkly.report
```

The following example kills job number **456**:

```
k 456
```

limit cpu

Changes the job limit for a workstation.

You must have **limit** access to the workstation.

Synopsis

lc *wkstation***;***limit*[**;noask**]

Arguments

wkstation	Specifies the name of the workstation. Wildcard characters are permitted. The default is your login workstation.
limit	Specifies the job limit. You can enter **0** through **1024**. Entering **system** sets the job limit to zero. If a limit of zero is set, no jobs, other than **hi** and **go** priority jobs, are launched on the workstation.
noask	Specifies not to prompt for confirmation before taking action on each qualifying workstation.

Usage Notes

To display the current job limit on your login workstation, use the **status** command.

When you first start Workload Scheduler following installation, the workstation job limit is set to zero, and must be raised before any jobs are launched. Once you change the limit, it is carried forward during pre-production processing to the next day’s production plan.

Workload Scheduler attempts to launch as many jobs as possible within the job limit. There is a practical limit to the number of processes that can be started on a workstation. If the limit is reached, the system responds with a message indicating that system resources are not available. When a job cannot be launched for this reason, it enters the **fail** state. Lowering the job limit can prevent this from occuring.

Examples

The following example changes the job limit on workstation **site3**:

```
limit cpu=site3;25
```

The following example changes the job limit on the workstation on which you are running Conman:

```
lc ;12
```

The following example changes the job limit on workstation **rx12**:

```
lc rx12;6
```

limit sched

Changes the job limit for a job stream.

You must have **limit** access to the job stream.

Synopsis

ls *jstreamselect*;*limit*[**;noask**]

Arguments

jstreamselect	See *"Selecting Job Streams in Commands" on page 112*.
limit	Specifies the job limit. You can enter **0** through **1024**. If you specify a limit of zero, no further jobs are launched from the job stream.
noask	Specifies not to prompt for confirmation before taking action on each qualifying job stream.

Examples

The following example changes the job limit on all job streams that include **sales** in the name:

```
limit sched=sales@;4
```

The following example changes the job limit on job stream **apwkly**:

```
ls apwkly;6
```

link

Opens communications links between workstations. In a Workload Scheduler network, fault-tolerant and standard agents are linked to their domain managers, and domain managers are linked to their parent domain managers. Extended agents are not linked; they communicate through a host.

You must have **link**access to the target workstation.

Synopsis

lk[*domain***!**]*wkstation*[**;noask**]

Arguments

domain	Specifies the name of the domain in which links are opened. Wildcard characters are permitted. This argument is useful when linking more than one workstation in a domain. For example, to link all the agents in domain **stlouis**, use the following command: `link stlouis!@` The domain is not needed if you do not include wildcard characters in *wkstation*. If you do not include *domain*, and you include wildcard characters in *wkstation*, the default domain is the one in which Conman is running.
wkstation	Specifies the name of the workstation to be linked. Wildcard characters are permitted.
noask	Specifies not to prompt for confirmation before taking action on each qualifying workstation.

Usage Notes

If the **autolink** option is set to **on** in a workstation definition, its link is opened automatically each time Workload Scheduler is started. If

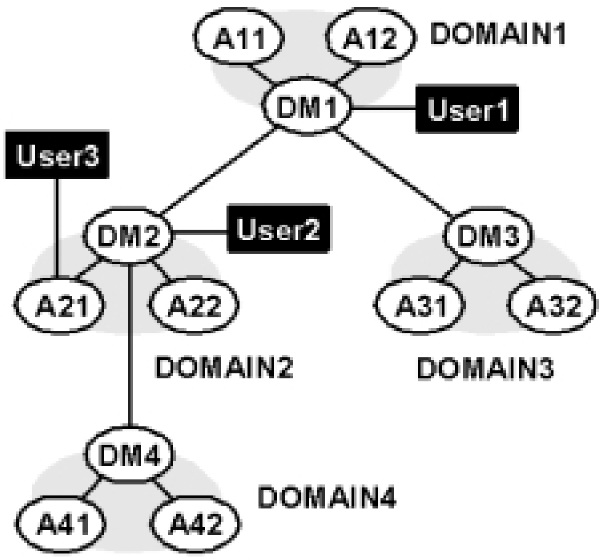
autolink is set to **off**, you must use **link** and **unlink** commands to control linking. For information about **autolink** see “Workstation Definitions” on page 2.

Assuming that a user has **link** access to the workstations being linked, the following rules apply:

- A user running Conman on the master domain manager can link any workstation in the network.
- A user running Conman on a domain manager other than the master can link any workstation in its own domain and subordinate domains. The user cannot link workstations in peer domains.
- A user running Conman on an agent can link any workstation in its local domain.
- To link a subordinate domain while running Conman in a higher domain, it is not necessary that the intervening links be open.

Examples

The illustration and table below show the links opened by **link** commands executed by users in various locations in the network.



DMn are domain managers and **A***nn* are agents.

Command	Links Opened by User1	Links Opened by User2	Links Opened by User3
link @!@	All links are opened.	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4 DM4-A41 DM4-A42	DM2-A21 DM2-A22
link @	DM1-A11 DM1-A12 DM1-DM2 DM1-DM3	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4	DM2-A21 DM2-A22
link DOMAIN3!@	DM3-A31 DM3-A32	Not allowed.	Not allowed.
link DOMAIN4!@	DM4-A41 DM4-A42	DM4-A41 DM4-A42	Not allowed.
link DM2	DM1-DM2	Not applicable.	DM2-A21
link A42	DM4-A42	DM4-A42	Not allowed.

Command	Links Opened by User1	Links Opened by User2	Links Opened by User3
link A31	DM3-A31	Not allowed.	Not allowed.

listsym

Lists the production plan (Symphony) log files.

You must have **display** access to the Symphony file.

Synopsis

listsym [**;offline**]

Arguments

offline	Sends the output of the command to the Conman output device. For information about this device, see "Offline Output" on page 97.

Command Output

Schedule Date
The date used by the **schedulr** command to select job streams for execution.

Actual Date
The date **batchman** began executing the Symphony file.

Start Time
The time **batchman** began executing the Symphony file.

Log Date
The date the plan (Symphony) was logged by the **stageman** command.

Run Num
The run number assigned to the plan (Symphony). These are used internally for Workload Scheduler network synchronization.

Size The size of the log file in records.

Log Num
The log number indicating the chronological order of log files. This number can be used in a **setsym** command to switch to a specific log file.

Filename

The name of the log file assigned by the **stageman** command.

Examples

The following example lists the production plan log files:

```
listsym
```

The following example lists the production plan log files to Conman's output device:

```
lis ;off
```

recall

Displays prompts that are waiting for a response.

You must have **display** access to the prompts.

Synopsis

rc [*wkstation*][**;offline**]

Arguments

wkstation	Specifies the name of the workstation on which the prompt was issued. If you do not specify a workstation, only prompts for the login workstation and global prompts are displayed.
offline	Sends the output of the command to the Conman output device. For information about this device, see "Offline Output" on page 97.

Command Output

State The state of the prompt. The state of pending prompts is always **ASKED**.

Message or Prompt
For named prompts, the message number, the name of the prompt, and the message text. For unnamed prompts, the message number, the name of the job or job stream, and the message text.

Examples

The following example displays pending prompts issued on the workstation on which you are running Conman:

```
recall
```

or:

```
rc
```

The following example displays pending prompts on workstation **site3**:

```
rc site3
```

The following example displays pending prompts on all workstations and the output is sent to Conman’s offline device:

```
rc @;offline
```

redo

Edits and reexecutes the previous command.

Synopsis

redo

Context

When you execute the **redo** command, Conman displays the previous command, so that it can be edited and reexecuted. Use the spacebar to move the cursor under the character to be modified, and enter the following directives.

Directives

d[*dir***]**	Deletes the character above the **d**. This can be followed by other directives.
i*text*	Inserts text before the character above the **i**.
r*text*	Replaces one or more characters with text, beginning with the character above the **r**. Replace is implied if no other directive is entered.
>*text*	Appends text to the end of the line.
>**d[***dir* **|** *text***]**	Deletes characters at the end of the line. This can be followed by another directive or text.
>**r***text*	Replaces characters at the end of the line with text.

Directive Examples

ddd	Deletes the three characters above the **d**s.
iabc	Inserts **abc** before the character above the **i**.
rabc	Replaces the three characters, starting with the one above the **r**, with **abc**.
abc	Replaces the three characters above **abc** with **abc**.
d diabc	Deletes the character above the first **d**, skips one character, deletes the character above the second **d**, and inserts **abc** in its place.

>abc	Appends **abc** to the end of the line.
>ddabc	Deletes the last two characters in the line, and inserts **abc** in their place.
>rabc	Replaces the last three characters in the line with **abc**.

Examples

The following example inserts a character:

```
redo
setsm 4
    iy
setsym 4
```

The following example replaces a character:

```
redo
setsym 4
       5
setsym 5
```

release job

Releases jobs from dependencies.

You must have **release** access to the job.

Synopsis

rj *jobselect*[**;***dependency*[**;**...]][**;noask**]

Arguments

jobselect
: Specifies the job or jobs to be released. See *"Selecting Jobs in Commands" on page 104*.

dependency
: The type of dependency. You can specify one of the following. You can use wildcard characters in *wkstation, jstream, job, resource, filename,*and *promptname*.

  at

  confirmed

  every

  follows[=[*netagent*::][*wkstation*#]*jstream*{.*job* | **@**} | *job*[**,**...]]

  needs[=[*num*] [*wkstation*#]*resource*[**,**...]]

  opens[=[*wkstation*#]"*filename*"[**(***qualifier***)**][**,**...]]

  priority

  prompt[="[**:** | **!**]*text*" | *promptname*[**,**...]]

  until

noask Specifies not to prompt for confirmation before taking action on each qualifying job.

Usage Notes

When you release an **opens** dependency, you can include only the base file name, and Conman performs a case-insensitive search for matching files, ignoring the directory names. Dependencies on all matching files are released.

For **needs** dependencies, the released job is given the required number of units of the resource, even though they might not be available. This can cause the **Available** units in a **showresources** to display a negative number.

Examples

The following example releases **job3** in job stream **ap** from all of its dependencies:

```
release job=ap.job3
```

The following example releases **job2** in job stream **skedr** from all of its **opens** dependencies:

```
rj skedr.job2;opens
```

The following example releases all jobs on workstation **site4** from their dependencies on a prompt named **glprmt**:

```
rj site4#@.@;prompt=glprmt
```

release sched

Releases job streams from dependencies.

You must have **release** access to the job stream.

Synopsis

rs *jstreamselect*[**;***dependency*[**;**...]][**;noask**]

Arguments

jstreamselect
: See *"Selecting Job Streams in Commands" on page 112*.

dependency
: The type of dependency. Specify one of the following. You can use wildcard characters in *wkstation, jstream, job, resource, filename,*and *promptname.*

  at

  carryforward

  follows[=[*netagent***::**][*wkstation***#**]*jstream*{*.job* | **@**} | *job*[**,**...]]

  limit

  needs[=[*num*] [*wkstation***#**]*resource*[**,**...]]

  opens[=[*wkstation***#**]"*filename*"[**(***qualifier***)**][**,**...]]

  priority

  prompt[="[**:** | **!**]*text*" | *promptname*[**,**...]]

  until

noask Specifies not to prompt for confirmation before taking action on each qualifying job stream.

Usage Notes

When deleting an **opens** dependency, you can include only the filename (basename), and Conman performs a case-insensitive search for matching files, ignoring the directory names. Dependencies on all matching files are released.

For **needs** dependencies, the released job stream is given the required number of units of the resource, even though they might not be available. This can cause the Available units in a **showresources** to display a negative number.

Examples

The following example releases job stream **ap** from all of its dependencies:

```
release sched=ap
```

The following example releases job stream **sked5** from all of its **opens** dependencies:

```
rs sked5;opens
```

The following example releases all job streams on workstation **site3** from their dependencies on job stream **main#sked23**:

```
rs site3#@;follows=main#sked23
```

reply

Replies to a job or job stream prompt.

You must have **reply** access to the named or global prompt. To reply to an unnamed prompt, you must have **reply** access to prompts, and **reply** access to the associated job or job stream.

Synopsis

reply{ *promptname* | [*wkstation***#**]*msgnum*}**;***reply*[**;noask**]

Arguments

promptname	Specifies the name of a global prompt. Wildcard characters are permitted.
wkstation	Specifies the name of the workstation on which an unnamed prompt was issued.
msgnum	Specifies the message number of an unnamed prompt. You can display message numbers with the **recall** and **showprompts** commands.
reply	Specifies the reply, either **Y** for yes or **N** for no.
noask	Specifies not to prompt for confirmation before taking action on each qualifying prompt.

Usage Notes

If the reply is **Y**, dependencies on the prompt are satisfied. If the reply is **N**, the dependencies are not satisfied and the prompt is not re-issued.

Prompts can be replied to before they are issued. You can use the **showprompts** command to display all prompts, whether they have been issued or not.

Examples

The following example replies **Y** to the global prompt **arprmt**:

```
reply arprmt;y
```

The following example replies **N** to message number **24** on workstation **site4**:

```
rep site4#24;n
```

rerun

Reruns a job.

You must have **rerun**access to the job.

Synopsis

rr *jobselect*[**;from=**[*wkstat*#]*job*[**;at=***time*][**;pri=***pri*]][**;noask**]

rr *jobselect*[**;step=***step*][**;noask**]

Arguments

jobselect
: Specifies the name of the global prompt. Wildcard characters are permitted.

from=[*wkstat*#**]***job*
: Specifies the name of a job defined in Workload Scheduler's database whose job file or command will be run in place of the job specified by *jobselect*.

  wkstat#
  : Specifies the name of the workstation on which the **from** job runs. The default is the workstation on which Conman is running.

  job
  : Specifies the name of the **from** job definition The following types of job names are not permitted:
    - The names of jobs submitted using the **submit file** and **submit docommand** commands.
    - The alias names of jobs submitted using the **submit job** command.

  To use the **from** argument, you must have access to Workload Scheduler's database from the computer on which you are running Conman

at=*time*
: Specifies the rerun job's start time, expressed as follows:

  *hhmm [***timezone|tz** *tzname]*[+*n* **days** | *date*]

  where:

hhmm The hour and minute.

+*n* **days**
The next occurrence of *hhmm* in *n* number of days.

+*date* The next occurrence of *hhmm* on *date*, expressed as *mm/dd/yy*.

timezone|tz *tzname*
The name of the time zone of the job. See “Time Zone Names” on page 309 for valid names.

pri=*pri*
Specifies the priority to be assigned to the rerun job. If you do not specify a priority, the job is given the same priority as the original job.

step=*step*
Specifies that the job is rerun using this name in place of the original job name. See “Usage Notes” for more information.

noask Specifies not to prompt for confirmation before taking action on each qualifying job.

Usage Notes

You can rerun jobs that are in the **succ**, **fail** or **abend** state. A rerun job is placed in the same job stream as the original job, and inherits the original job’s dependencies. If you rerun a repetitive (**every**) job, the rerun job is scheduled to run at the same rate as the original job.

Note: You can issue **rerun** for jobs in the **external** job stream that are in the **error** state. Jobs in the **external** job stream represent jobs and job streams that have been specified as internetwork dependencies. The job state is initially set to **extrn** immediately after a **rerun** is executed, and Conman begins checking the state.

When **;from** is used, the name of the rerun job depends on the value of the Global Option **retain rerun job names**. If the option is set to **Y**, rerun jobs retain the original job names. If the option is set to **N**, rerun jobs are given the **from** job names. See “Global Options” in chapter 2 of the *Tivoli Workload Scheduler User’s Guide* for more information.

In Conman displays, rerun jobs are displayed with the notation **>>rerun as**. To refer to a rerun job in another command, such as **altpri**, you must use the original job name.

When a job is rerun with the **;step** option, the job runs with *step* in place of its original name. Within a job script, you can use the **jobinfo** command to return the job name and the execute the script differently for each iteration. For example, in the following UNIX script, the **jobinfo** command is used to set a variable named **STEP** to the name that was used to run the job. The **STEP** variable is then used to determine how the script is executed.

```
...
MPATH=`maestro`
STEP=`$MPATH/bin/jobinfo job_name`
if [$STEP = JOB3]
  then
  ...
  STEP=JSTEP1
  fi
if [$STEP = JSTEP1]
  then
  ...
  STEP=JSTEP2
  fi
if [$STEP = JSTEP2]
  then
  ...
  fi
...
```

In Conman displays, jobs rerun with the **;step** option are displayed with the notation **>>rerun step**.

For information about **jobinfo**, see “jobinfo Command” on page 252.

Examples

The following example reruns **job4** in **sked1** on workstation **main**:

```
rerun main#sked1.job4
```

The following example reruns **job5** in **sked2** using the job definition for **jobx**. The job's **at** time is set to 6:30 P.M. and its priority is set to **25**:

```
rr sked2.job5;from=jobx;at=1830;pri=25
```

The following example reruns **job3** in **sked4** using the job name **jstep2**:

```
rr sked4.job3;step=jstep2
```

resource

Changes the number of total units of a resource.

You must have **resource** access to the resource.

Synopsis

resource [*wkstation*#]*resource***;***num*[**;noask**]

Arguments

wkstation	Specifies the name of the workstation on which the resource is defined. The default is the workstation on which Conman is running.
resource	Specifies the name of the resource.
num	Specifies the total number of resource units. Valid values are **0** through **1024**.
noask	Specifies not to prompt for confirmation before taking action on each qualifying resource.

Examples

The following example changes the number of units of resource **tapes** to **5**:

```
resource tapes;5
```

The following example changes the number of units of resource **jobslots** on workstation **site2** to **23**:

```
res site2#jobslots;23
```

setsym

Selects a production plan archive file. Subsequent display commands display the contents of the archived production plan. You cannot modify the information in a production plan archive file.

Synopsis

setsym [*filenum*]

Arguments

filenum	Specifies the number of the production plan archive file. If you do not specify a log file number, the pointer returns to zero, the current production plan (Symphony). Use the **listsym** command to list archive file numbers.

Examples

The following example selects production plan archive file **5**:

```
setsym 5
```

The following example selects the current production plan (Symphony):

```
set
```

showcpus

Displays information about workstations and links.

Synopsis

sc [[*domain*!]*wkstation*][**;info**][**;link**][**;offline**]

Arguments

domain	Specifies the name of a domain. The default is the domain of *wkstation*. If *wkstation* includes wildcard characters, the default is the domain in which Conman is running. Wildcard characters are permitted.
wkstation	Specifies the name of a workstation. The default is the workstation on which Conman is running. Wildcard characters are permitted.
info	Displays information in the **info** format.
link	Displays information in the **link** format.
offline	Sends the output of the command to the Conman output device. For information about this device, see "Offline Output" on page 97.

Command Output

The output of the command is produced in three formats, standard, **info**, and **link**.

Standard Format

CPUID
The name of the workstation to which this information applies.

RUN The run number of the Production Control file (Symphony).

NODE
The node type and workstation type. Node types are as follows:

- UNIX
- WINT
- MPEV
- MPIX

- OTHER

Workstation types are as follows:
- MASTER
- MANAGER
- FTA
- S-AGENT
- X-AGENT

LIMIT
The Workload Scheduler job limit.

FENCE
The Workload Scheduler job fence.

DATE TIME
The date and time Workload Scheduler started executing the current production plan (Symphony).

STATE
The state of the workstation's links. Up to five characters are displayed as follows:

```
[L] [D|T|H|X] [I] [J] [W|M|H|X]
```

L	The link is open (linked).
D	The link is DS. For MPEV and MPIX only.
T	The link is TCP/IP.
I	The **jobman** program has completed startup initialization.
J	The **jobman** program is running.
W	The workstation is linked by TCP/IP.
M	The workstation is linked by DS. For MPEV and MPIX only.
X	The workstation is linked as an extended agent (x-agent).
H	The workstation is linked through its host.

Note: If the workstation running Conman is the extended agent's host, the state of the x-agent is

```
LXI JX
```

If the workstation running Conman is not the extended agent's host, the state of the x-agent is

```
LHI JH
```

METHOD
The name of the access method specified in the workstation definition. For extended agents only.

DOMAIN
The name of the domain in which the workstation is a member.

info Format

CPUID
The name of the workstation to which this information applies.

VERSION
The version of Workload Scheduler's **jobman** program.

TIMEZONE
The time zone of the workstation. It is the same as the value of the TZ environment variable. For an extended agent, this is the time zone of its host.

INFO The Operating System version and hardware model. For extended agents, no information is listed.

link Format

CPUID
The name of the workstation to which this information applies.

HOST
The name of the workstation acting as the host to a standard agent or extended agent. For domain managers and fault-tolerant agents, this is the same as CPUID. For standard agent workstations, this is the name of the domain manager. For extended agents, this is the name of the host workstation.

FLAGS
The state of the workstation's links. Up to five characters are displayed as follows:

```
[A] [F] [s] [D|T|B]
```

A	Autolink is turned on in the workstation definition.
F	Full Status mode is turned on in the workstation definition.

s	The ID of **mailman** server for the workstation.
D	The link is defined as DS. For MPEV and MPIX only.
T	The link is defined as TCP/IP.
B	The link is defined as both DS and TCP/IP. For MPEV and MPIX only.

ADDR

The TCP port number for the workstation.

NODE

The node name of the workstation.

Examples

The following example displays information about the workstation on which you are running Conman in the **info** format:

```
showcpus ;info
```

The following example displays link information offline for all workstations:

```
sc @!@;link;off
```

showdomain

Displays domain information.

Synopsis

showdomain [*domain*][**;info**][**;offline**]

Arguments

domain	Specifies the name of the domain. The default is the domain in which Conman is running. Wildcard characters are permitted.
info	Displays information in the **info** format.
offline	Sends the output of the command to the Conman output device. For information about this device, see “Offline Output” on page 97.

Command Output

The output of the command is produced in two formats, standard, and **info**.

Standard Format

DOMAIN
The name of the domain to which this information applies.

MANAGER
The name of the domain manager.

PARENT
The name of the parent domain.

info Format

DOMAIN
The name of the domain to which this information applies.

MEMBER-CPUS
The names of the workstations in the domain.

CPU-TYPE
The type of each workstation: MASTER, MANAGER, FTA, S-AGENT, or X-AGENT.

Examples

The following example displays information about the **masterdm** domain:

```
showdomain masterdm
```

The following example displays the member workstations in all domains in the **info** format:

```
showdomain @;info
```

showfiles

Displays information about file dependencies.

Synopsis

sf [[*wkstation#*]*file*][**;***state*[**;**...]][**;keys**][**;offline**]

sf [[*wkstation#*]*file*][**;***state*[**;**...]][**;deps**[**;keys | info | logon**]][**;offline**]

Arguments

wkstation	Specifies the name of the workstation on which the file exists. The default is the workstation on which Conman is running. Wildcard characters are permitted.
file	Specifies the name of the file. The name must be enclosed in quotes (″) if it contains characters other than the following: alphanumerics, dashes (-), slashes (/), backslashes (\), and underscores (_). The default is to display all file dependencies. Wildcard characters are permitted.
state	Specifies the state of the file dependencies to be displayed. The default is to display file dependencies in all states. The states are as follows:

yes	File exists and is available.
no	File is unavailable, or does not exist.
?	Availability is being checked.
<blank>	The file has not yet been checked, or the file was available and used to satisfy a job or job stream dependency.

keys	Displays a single column list of the objects selected by the command.
deps	Displays information in the **deps** format. Use **keys**, **info**, or **logon** to modify the display.
offline	Sends the output of the command to the Conman

output device. For information about this device, see “Offline Output” on page 97.

Command Output

The output of the command is produced in three formats: standard, **keys**, and **deps**. The arguments **keys**, **info**, and **logon** modify the **deps** display.

Standard Format

Exists The state of the file dependency.

File Name
The name of the file.

keys Format

Files are listed with one file on each line. Directory names are not included. Each file is listed in the following format:

wkstation#file

deps Format

Files are listed followed by the dependent jobs and job streams. Jobs are listed in the standard **showjobs** format. Job streams are listed in the standard **showschedules** format.

deps;keys Format

Jobs and job streams that have file dependencies are listed with one on each line, in the following format:

wkstation#jstream[.*job*]

deps;info Format

Files are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;info** format. Job streams are listed in the standard **showschedules** format.

deps;logon Format

Files are listed followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;logon** format. Job streams are listed in the standard **showschedules** format.

Examples

The following example displays the status of a file dependency for **d:\apps\mis\lib\data4**:

```
showfiles d:\apps\mis\lib\data4
```

The following example displays **offline** the status of all file dependencies on all workstations in the **deps** format:

```
sf @#@;deps;offline
```

showjobs

Displays information about jobs.

Synopsis

sj[*jobselect*] [**;keys | info | step | logon**][**;short | single**][**;offline**]

sj[*jobselect*] [**;deps**[**;keys | info | logon**]][**;short | single**][**;offline**]

sj[*jobselect*] [**;stdlist**[**;keys**]][**;short | single**][**;offline**]

Arguments

jobselect	See *"Selecting Jobs in Commands" on page 104.*
keys	Displays a single column list of the objects selected by the command.
info	Displays information in the **info** format.
step	Displays information in the **step** format.
logon	Displays information in the **logon** format.
stdlist	Displays information in the **stdlist** format. Use the **keys** argument to modify the display.
deps	Displays information in the **deps** format. Use **keys**, **info**, or **logon** to modify the display.
short	Shortens the display for **every** and **rerun** jobs to include only the following: ■ The first iteration ■ Jobs in different states ■ Exactly matched jobs
single	Selects only the parent job in a chain that can include reruns, repetitions, and recovery jobs. The job must be identified by job number in *jobselect*. This is particularly useful with the **stdlist** option.
offline	Sends the output of the command to the Conman output device. For information about this device, see "Offline Output" on page 97.

Command Output

The output of the **showjobs** command is produced in seven formats: standard, **keys**, **info**, **step**, **logon**, **deps**, and **stdlist**. The arguments **keys**, **info**, and **logon** modify the displays.

Standard Format

CPU The workstation on which the job runs.

Schedule
The name of the job stream.

Job The name of the job. The following notation may precede a job name:

>> **rerun as**
A job that was rerun with the **rerun** command, or as a result of automatic recovery.

>> **rerun step**
A job that was rerun with the **rerun ;step** command.

>> **every run**
The second and subsequent runs of an every job.

>> **recovery**
The run of a recovery job.

State The state of the job or job stream. Job states are follows:

abend The job terminated with a non-zero exit code.

abenp An **abend** confirmation was received, but the job is not completed.

add The job is being submitted.

done The job completed in an unknown state.

error For internetwork dependencies only, an error occurred while checking for the remote status.

exec The job is executing.

extrn For internetwork dependencies only, the status is unknown. An error occurred, a rerun action was just performed on the job in the **external** job stream, or the remote job or job stream does not exist.

fail	Unable to launch the job.
fence	The job's priority is below the fence.
hold	The job is awaiting dependency resolution.
intro	The job is introduced for launching by the system.
pend	The job completed, and is awaiting confirmation.
ready	The job is ready to launch, and all dependencies are resolved.
sched	The job's **at** time has not arrived.
succ	The job completed with an exit code of zero.
succp	A **succ** confirmation was received, but the job is not completed.
susp	The job was suspended by a **breakjob** command. (MPE only)
wait	The job is in the **wait** state. (Extended agent and MPE only)
waitd	The job is in the **wait** state, and is deferred. (MPE only)

Job stream states are as follows:

abend	The job stream terminated with a non-zero exit code.
add	The job stream was added with operator intervention.
cancel	The job stream was canceled.
cancel pend	The job stream is pending cancellation. Cancellation is deferred until all of the dependencies, including an at time, are resolved.
error	For internetwork dependencies only, an error occurred while checking for the remote status.
exec	The job stream is executing.

extrn For internetwork dependencies only, the job stream is in a remote TWS network and its status is unknown. An error occurred, a rerun action was just performed on the EXTERNAL job stream, or the INET job or job stream does not exist.

hold The job stream is awaiting dependency resolution.

ready The job stream is ready to launch and all dependencies are resolved.

stuck Execution of the job stream was interrupted. No jobs are launched without operator intervention.

succ The job stream completed successfully.

Pr The priority of the job stream or job. A plus sign (+) preceding the priority means the job has been launched.

(Est)Start
The start time of the job stream or job. Parentheses indicate an estimate of the start time. If the start time is more than 24 hours in the past or future, the date is listed instead of the time.

(Est)Elapse
The run time of the job stream or job. Parentheses indicate an estimate based on logged statistics.

dependencies
A list of job dependencies and comments. Any combination of the following can be listed:

- For a **follows** dependency, a job stream or job name is displayed.
- For an **opens** dependency, the file name is displayed. If the file resides on an extended agent and its name is longer than 25 characters, only the last 25 characters are displayed.
- For a **needs** dependency, a resource name enclosed in hyphens (-) is displayed. If the number of units requested is greater than one, the number is displayed before the first hyphen.

- For an **every** rate, the repetition rate preceded by an ampersand (&) is displayed.
- For an **until** time, the time preceded by an angle bracket (<) is displayed.
- For a **prompt** dependency, the prompt number is displayed in the format #*num*. For global prompts, the prompt name follows in parentheses.
- For executing jobs, the process identification number (PID) is displayed in the format **#J***nnnnn*.
- Jobs submitted on UNIX using Workload Scheduler's **at** and **batch** commands are labeled **[Userjcl]**.
- Cancelled jobs are labeled **[Cancelled]**.
- Jobs cancelled with **;pend** option are labeled **[Cancel Pend]**.
- Jobs with expired **until** times, including jobs cancelled with **;pend**option, are labeled **[Until]**.
- **[Recovery]** means that operation intervention is required.
- **[Confirm]** means that confirmation is required because the job was scheduled using the **confirm** keyword.

keys Format

Job names are listed one on each line in the following format:

wkstation#jstream.job

info Format

CPU The workstation on which the job runs.

Schedule
The name of the job stream.

Job The name of the job. The following notation may precede a job name:

>> rerun as
A job that was rerun with the **rerun** command, or as a result of automatic recovery.

>> rerun step
A job that was rerun with the **rerun ;step** command.

>> every run
The second and subsequent runs of an every job.

>> recovery
The run of a recovery job.

Job File
The name of the job's script or executable file. Long file names might wrap, causing incorrect paging. To avoid this, pipe the output to **more**. For example:

```
conman "sj;info | more
```

Opt The job recovery option, if any. The recovery options are **RE** for rerun, **CO** for continue, and **ST** for stop.

Job The name of the recovery job, if any.

Prompt
The number of the recovery prompt, if any.

step Format

This format is not supported on Windows NT.

CPU The workstation on which the job runs.

Schedule
The name of the job stream.

Job The name of the job. The following notation might precede a job name:

>> rerun as
A job that was rerun with **rerun** command, or as a result of automatic recovery.

>> repeated as
The second and subsequent runs of an **every** job.

State The state of the job or job stream. See "Standard Format" for information about state.

Job# The process identification number displayed as **#J***nnnnn*.

Step A list of descendant processes that are associated with the job. For extended agent jobs, only host processes are listed.

logon Format

CPU The workstation on which the job runs.

Schedule
The name of the job stream.

Job The name of the job. The following notation may precede a job name:

>> rerun as
A job that was rerun with **rerun** command, or as a result of automatic recovery.

>> repeated as
The second and subsequent runs of an **every** job.

State The state of the job or job stream. See “Standard Format” for information about state.

Job# The process identification number displayed as **#J***nnnnn*.

Logon The user name under which the job executes.

stdlist Format

The standard list files for the selected jobs are displayed.

stdlist;keys Format

The names of the standard list files for the selected jobs are listed, one on each line.

deps Format

Jobs used in **follows** dependencies are listed followed by the dependent jobs and job streams. Jobs are listed in the standard **showjobs** format. Job streams are listed in the standard **showschedules** format.

deps;keys Format

Jobs and job streams that have **follows** dependencies are listed, one on each line.

deps;info Format

Jobs used in **follows** dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;info** format. Job streams are listed in the standard **showschedules** format.

deps;logon Format

Jobs used in **follows** dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;logon** format. Job streams are listed in the standard **showschedules** format.

Examples

The following example displays the status of all jobs in all **acctg** job streams on workstation **site3**:

```
showjobs site3#acctg@.@
```

The following example displays the status of all jobs on the workstation on which you are running Conman in the **logon** format:

```
sj ;logon
```

The following example displays the status of all jobs in the **hold** state on all workstations in the **deps** format:

```
sj @#@.@+state=hold;deps
```

The following example displays all of the standard list files for the job **gl** in the job stream **arec** on workstation **site2**:

```
sj site2#arec.gl;stdlist
```

showprompts

Displays information about prompts.

Synopsis

sp[*promptselect*][**;keys**][**;offline**]

sp[*promptselect*] [**;deps**[**;keys | info | logon**]][**;offline**]

Arguments

promptselect

[*promptname* | [*wkstation#*]*msgnum*][**;***state*[**;**...]]

promptname
Specifies the name of a global prompt. Wildcard characters are permitted.

wkstation
Specifies the name of the workstation on which an unnamed prompt is issued. The default is the workstation on which Conman is running.

msgnum
Specifies the message number of an unnamed prompt.

state Specifies the state of prompts to be displayed. The states are as follows:

YES The prompt was replied to with **y**.

NO The prompt was replied to with **n**.

ASKED
The prompt was issued, but no reply was given.

INACT
The prompt has not been issued.

keys Displays a single column list of the objects selected by the command.

deps Displays information in the **deps** format. Use **keys**, **info**, or **logon** to modify the display.

info Displays information in the **info** format.

logon Displays information in the **logon** format.

offline Sends the output of the command to the Conman output device. For information about this device, see “Offline Output” on page 97.

Command Output

The output of the command is produced in three formats: standard, **keys**, and **deps**. The arguments **keys**, **info**, and **logon** modify the **deps** display.

Standard Format

State The state of the prompt.

Message or Prompt
For named prompts, the message number, the name, and the text of the prompt. For unnamed prompts, the message number, the name of the job or job stream, and the text of the prompt.

keys Format

The prompts are listed one on each line. Named prompts are listed with their message numbers and names. Unnamed prompts are listed with their message numbers, and the names of the jobs or job streams in which they appear as dependencies.

deps Format

Prompts used as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the standard **showjobs** format. Job streams are listed in the standard **showschedules** format.

deps;keys Format

Jobs and job streams that have prompt dependencies are listed one on each line.

deps;info Format

Prompts used as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;info** format. Job streams are listed in the standard **showschedules** format.

deps;logon Format

Prompts used as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;logon** format. Job streams are listed in the standard **showschedules** format.

Examples

The following example displays the status of all prompts issued on the workstation on which you are running Conman:

```
showprompts
```

The following example displays, in the **deps** format, the status of all **mis** prompts that have been issued:

```
sp mis@;asked;deps
```

The following example displays the status of prompt number 34 on workstation **main**:

```
sp main#34
```

showresources

Displays information about resources.

Synopsis

sr[[*wkstation#*]*resourcename*][**;keys**][**;offline**]

sr[[*wkstation#*]*resourcename*][**;deps**[**;keys | info | logon**]][**;offline**]

Arguments

wkstation	Specifies the name of the workstation on which the resource is defined. The default is the workstation on which Conman is running.
resourcename	Specifies the name of the resource. Wildcard characters are permitted.
keys	Displays a single column list of the objects selected by the command.
deps	Displays information in the **deps** format. Use **keys**, **info**, or **logon** to modify the display.
info	Displays information in the **info** format.
logon	Displays information in the **logon** format.
offline	Sends the output of the command to the Conman output device. For information about this device, see “Offline Output” on page 97.

Command Output

The output of the command is produced in three formats: standard, **keys**, and **deps**. The arguments **keys**, **info**, and **logon** modify the **deps** display.

Standard Format

CPU	The workstation on which the resource is defined.
Resource	The name of the resource.
Total	The total number of defined resource units.

Available	The number of resource units that have not been allocated.
Qty	The number of resource units allocated to a job or job stream.
Used By	The name of the job or job stream.

keys Format

The resources are listed one on each line.

deps Format

Resources used as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the standard **showjobs** format. Job streams are listed in the standard **showschedules** format.

deps;keys Format

Jobs and job streams that have resource dependencies are listed one on each line.

deps;info Format

Resources used as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;info** format. Job streams are listed in the standard **showschedules** format.

deps;logon Format

Resources used as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;logon** format. Job streams are listed in the standard **showschedules** format.

Examples

The following example displays information about all resources on the workstation on which you are running Conman:

```
showresources
```

The following example displays information about the **dbase** resource on workstation **main** in the **deps** format:

```
sr main#dbase;deps
```

showschedules

Displays information about job streams.

Synopsis

ss[*jstreamselect*][**;keys**][**;offline**]

ss[*jstreamselect*][**;deps**[**;keys | info | logon**]][**;offline**]

Arguments

jstreamselect	See "Selecting Job Streams in Commands" on page 112.
keys	Displays a single column list of the objects selected by the command.
deps	Displays information in the **deps** format. Use **keys**, **info**, or **logon** to modify the display.
info	Displays information in the **info** format.
logon	Displays information in the **logon** format.
offline	Sends the output of the command to the Conman output device. For information about this device, see "Offline Output" on page 97.

Command Output

The output of the command is produced in three formats: standard, **keys**, and **deps**. The arguments **keys**, **info**, and **logon** modify the **deps** display.

Standard Format

CPU The workstation on which the job stream runs.

Schedule
The name of the job stream.

State The state of the job stream. The states are as follows:

add	The job stream was added with operator intervention.
abend	The job stream terminated with a non-zero exit code.

cancel The job stream was canceled.

cancel pend
The job stream is pending cancellation. Cancellation is deferred until all of the dependencies, including an at time, are resolved.

error For internetwork dependencies only, an error occurred while checking for the remote status.

exec The job stream is executing.

extrn For internetwork dependencies only, the job stream is in a remote TWS network and its status is unknown. An error occurred, a rerun action was just performed on the EXTERNAL job stream, or the INET job or job stream does not exist.

hold The job stream awaiting dependency resolution.

ready The job stream ready to launch and all dependencies are resolved.

stuck Job stream execution was interrupted. No jobs are launched without operator intervention.

succ The job stream completed successfully.

Pr The priority of the job stream.

(Est)Start
The start time of the job stream. Parentheses indicate an estimate of the start time. If the start time is more than 24 hours in the past or future, the date is listed instead of the time.

(Est)Elapse
The run time of the job stream. Parentheses indicate an estimate based on logged statistics.

Jobs #
The number of jobs in the job stream.

Jobs OK
The number of jobs that have completed successfully.

Sch Lim

The job stream's job limit. If one is not listed, no limit is in effect.

dependencies

A list of job stream dependencies and comments. Any combination of the following may be listed:

- For a **follows** dependency, a job stream or job name is displayed.
- For an **opens** dependency, the file name displayed. If the file resides on an extended agent, and its name is longer than 25 characters, only the last 25 characters are displayed.
- For a **needs** dependency, a resource name enclosed in hyphens (-) is displayed. If the number of units requested is greater than one, the number is displayed before the first hyphen.
- For an **until** time, the time preceded by an angled bracket (<).
- For a **prompt** dependency, the prompt number displayed as #*num*. For global prompts, the prompt name in parentheses follows.
- Cancelled job streams are labeled **[Cancelled]**.
- Job streams cancelled with the **;pend** option are labeled **[Cancel Pend]**.
- Job streams with expired **until** times, including job streams cancelled with the **;pend**option, are labeled: **[Until]**.
- Job streams that contain the **carryforward** keyword are labeled **[Carry]**.
- For job streams that were carried forward from the previous day, the original name and date are displayed in brackets.

keys Format

The job streams are listed one on each line.

deps Format

Job streams used in as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the standard **showjobs** format. Job streams are listed in the standard **showschedules** format.

deps;keys Format

Job streams that have **follows** dependencies are listed one on each line.

deps;info Format

Job streams used in as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;info** format. Job streams are listed in the standard **showschedules** format.

deps;logon Format

Job streams used in as dependencies are listed, followed by the dependent jobs and job streams. Jobs are listed in the **showjobs;logon** format. Job streams are listed in the standard **showschedules** format.

Examples

The following example displays the status of all job streams in the **hold** state on the workstation on which you are running Conman:

```
showschedules @+state=hold
```

The following example displays the status of all **corp** job streams on workstation **site2** in the **deps;info** format:

```
ss site2#corp@;deps;info
```

The following example displays **offline** the status of all job streams in the **abend** state on all workstations:

```
ss @#@+state=abend;off
```

shutdown

Unconditionally stops all of Workload Scheduler's production processes, including Batchman, Jobman, Netman, Mailman, all Mailman servers, and all writers.

You must have **shutdown** access to the workstation.

Synopsis

shutdown [**;wait**]

Arguments

wait	Waits until all processes have stopped before prompting for another command.

Usage Notes

The **shutdown** command stops the processes only on the workstation on which Conman is running. To restart Netman only, execute the **StartUp** command. For information about the **StartUp** command, see "StartUp Command" on page 273. To restart the entire process tree, execute a Conman **start** command.

You must execute a Conman **unlink @** command before executing a **shutdown** command.

Examples

The following example shuts down production on the workstation on which you are running Conman:

```
unlink @
shutdown
```

The following example shuts down production on the workstation on which you are running Conman and waits for all processes to stop:

```
unlink@;noask
shut ;wait
```

start

Starts Workload Scheduler’s production processes.

You must have **start** access to the workstation.

Synopsis

start[*domain***!**]*wkstation*[**;mgr**][**;noask**]

Arguments

domain	Specifies the name of the domain in which workstations are started. Because workstations have unique names, the domain is not needed when starting a specific workstation. Wildcard characters are permitted. This argument is useful when starting more than one workstation in a domain. For example, to start all the agents in domain **stlouis**, use the following command: `start stlouis!@` If *domain* is omitted, and *wkstation* contains wildcard characters, the default domain is the one in which Conman is running.
wkstation	Specifies the name of the workstation to be started. Wildcard characters are permitted.
mgr	This can be entered only on the workstation on which Conman is running. It starts the local workstation as the domain manager. The workstation becomes the new domain manager and the current domain manager becomes a fault-tolerant agent. This form of the command usually follows a **stop** command. Note that the preferred method of switching a domain manager is to use a **switchmgr** command. See “switchmgr” on page 220 for more information.

noask	Specifies not to prompt for confirmation before taking action on each qualifying workstation.

Usage Notes

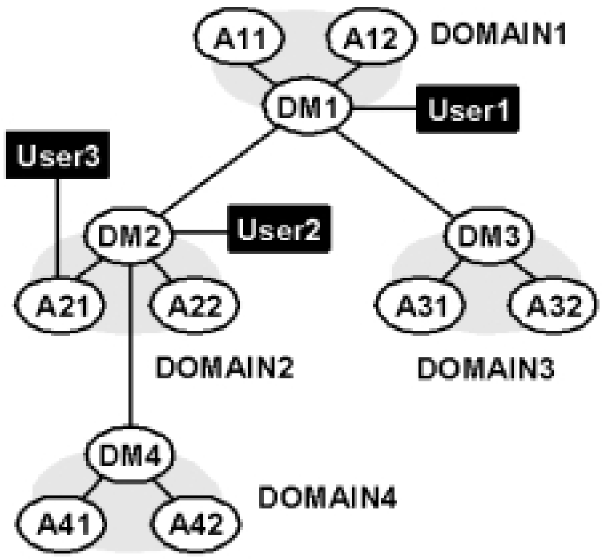
The **start** command is used at the start of each day to restart Workload Scheduler following pre-production processing. At that time it causes the autolinked fault-tolerant agents and standard agents to be initialized and started automatically. Agents that are not autolinked are initialized and started when you execute a **link** command.

Assuming the user has **start** access to the workstations being started, the following rules apply:

- A user running Conman on the master domain manager can start any workstation in the network.
- A user running Conman on a domain manager other than the master can start any workstation in that domain and subordinate domains. The user cannot start workstations in peer domains.
- A user running Conman on an agent can start any workstation in the local domain.

Examples

The illustration and table below show the workstations started by **start** commands executed by users in various locations in the network.



DM*n* are domain managers and **A***nn* are agents.

Command	Started by User1	Started by User2	Started by User3
start @!@	All workstations are started.	DM2 A21 A22 DM4 A41 A42	DM2 A21 A22
start @	DM1 A11 A12	DM2 A21 A22	DM2 A21 A22
start DOMAIN3!@	DM3 A31 A32	Not allowed.	Not allowed.
start DOMAIN4!@	DM4 A41 A42	DM4 A41 A42	Not allowed.
start DM2	DM2	DM2	DM2
start A42	A42	A42	Not allowed.

Command	Started by User1	Started by User2	Started by User3
start A31	A31	Not allowed.	Not allowed.

status

Displays Conman's banner and Workload Scheduler's production status.

Synopsis

status

Command Output

Following the word **Schedule** on the second line of output, the production plan (Symphony) mode is shown in parentheses. **Def** means that the production plan is in non-expanded mode, and **Exp** means it is in expanded mode. The mode of the production plan is determined by the setting of the Global option **expanded version**. See Global Options in chapter 2 of the *Tivoli Workload Scheduler Administrator's Guide* for more information.

Examples

The following example displays the status of the current production plan. Then it sets the production plan pointer to 2 and displays the status of that production plan.

```
%status
MAESTRO for UNIX (HPUX)/CONMAN 6.0 (3.34)  (C) Tivoli Systems 1998
Schedule (Exp) 11/30/98 (#7) on DEMOCPU.  Batchman down.
Limit: 6, Fence: 0
%setsym 2
Schedule 11/29/98 (#5) on DEMOCPU.  Symphony switched.
(2)%status
MAESTRO for UNIX (HPUX)/CONMAN 6.0 (3.34)  (C) Tivoli Systems 1998
Schedule (Exp) 11/29/98 (#5) on DEMOCPU.  Symphony switched.
```

stop

Stops Workload Scheduler's production processes. To stop the Netman process, use the **shutdown**

command.

You must have **stop** access to the workstation.

Synopsis

stop[*domain***!**]*wkstation*[**;wait**][**;noask**]

Arguments

domain	Specifies the name of the domain in which workstations are stopped. Because workstations have unique names, the domain is not needed when stopping a specific workstation. Wildcard characters are permitted. This argument is useful when stopping more than one workstation in a domain. For example, to stop all the agents in domain **stlouis**, use the following command: `stop stlouis!@` If *domain* is omitted, and *wkstation* contains wildcard characters, the default domain is the one in which Conman is running.
wkstation	Specifies the name of the workstation to be stopped. Wildcard characters are permitted.
wait	Specifies not to accept another command until all processes have stopped.
noask	Specifies not to prompt for confirmation before taking action on each qualifying workstation.

Usage Notes

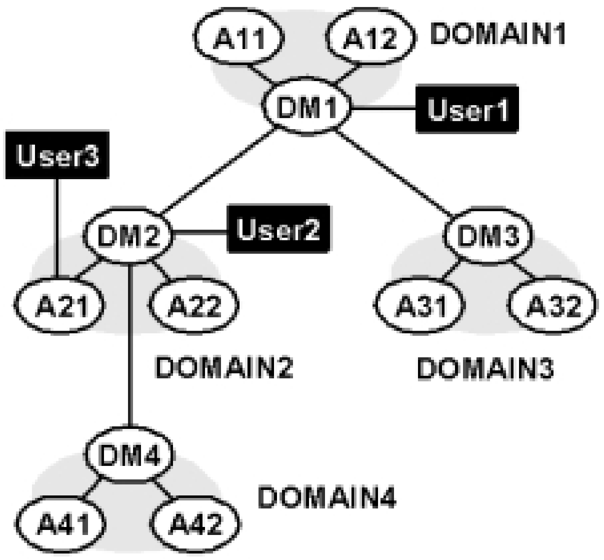
If the **stop** command cannot be applied to a distant workstation (if the TCP/IP path not available, for example), the command is stored locally in a **pobox** file, and is mailed to the workstation when it becomes linked.

Assuming the user has **stop** access to the workstations being stopped, the following rules apply:

- A user running Conman on the master domain manager can stop any workstation in the network.
- A user running Conman on a domain manager other than the master can stop any workstation in that domain and subordinate domains. The user cannot stop workstations in peer domains.
- A user running Conman on an agent can stop any workstation in the local domain.

Examples

The illustration and table below show the workstations stopped by different **stop** commands executed by users in different locations in the network.



DM*n* are domain managers and **A***nn* are agents.

Command	Stopped by: User1	Stopped by User2	Stopped by User3
stop @!@	All workstations are stopped.	DM2 A21 A22 DM4 A41 A42	DM2 A21 A22
stop @	DM1 A11 A12	DM2 A21 A22	DM2 A21 A22
stop DOMAIN3!@	DM3 A31 A32	Not allowed.	Not allowed.
stop DOMAIN4!@	DM4 A41 A42	DM4 A41 A42	Not allowed.
stop DM2	DM2	DM2	DM2
stop A42	A42	A42	Not allowed.

Command	Stopped by: User1	Stopped by User2	Stopped by User3
stop A31	A31	Not allowed.	Not allowed.

submit docommand

Submits a command to be launched as a Workload Scheduler job.

You must have **submit** access to the job. To include **needs** and **prompt** dependencies, you must have **use** access to the resources and global prompts.

Synopsis

sbd [*wkstation*#]″*cmd*″[**;alias**[=*name*]][**;into**=*jobstream*]
[**;***joboption*[**;**...]]

Arguments

wkstation
: Specifies the name of the workstation on which the job will be launched. Wildcard characters are permitted, in which case, the job is launched on all qualifying workstations. The default is the workstation on which Conman is running. You cannot specify a domain or workstation class.

cmd
: Specifies a valid system command of up to 255 characters. The entire command must be enclosed in quotes (″). The command is treated as a job, and all job rules apply.

alias=*name*
: Specifies a unique name to be assigned to the job. If you enter the **alias** keyword without specifying a name, a name is constructed using the first two alphanumeric characters of the command followed by a six digit random number. The name is always upshifted. For example, if the command is **rm apfile**, the generated name will be similar to **RM123456**.

: If you do not include **alias**, a job name is constructed using the first eight alphanumeric characters of the command, upshifted.

into=*jobstream*
: Specifies the name of the job stream into which the job will be placed for launching. Enter the name as follows:

[*wkstation*#]*jstream* If you do not specify a *wkstation*, the default is the workstation on which Conman is running. If **into** is not used, the job is added to a job stream named **JOBS**.

joboption

Specify one of the following:

at=*hhmm* [**timezone|tz** *tzname*] [**+***n* **days** | *mm/dd/yy*]

confirmed

every=*rate*

follows=[*netagent***::**][*wkstation*#]*jstream*{.*job* | **@**} | *job*[**,**...]

interactive

logon=*user*

needs=[*num*] [*wkstation*#]*resource*[**,**...]

opens=[*wkstation*#]"*filename*"[**(***qualifier***)**][**,**...] **priority**[=*pri* | **hi** | **go**]

prompt="[**:** | **!**]*text*" | *promptname*[**,**...]

recovery=stop | continue | rerun

recoveryjob | after=[*wkstation*#]*job*

recoveryprompt | abendprompt="*text*"

until=*hhmm* [**timezone|tz** *tzname*] [**+***n* **days** | *mm/dd/yy*]

Usage Notes

If you do not specify a *workstation* with **follows**, **needs**, or **opens**, the default is the workstation of the job. If the job is executed on a fault-tolerant agent, and you want to include a **prompt** dependency or a **recoveryprompt**, the mozart directory (*WShome*/**mozart**) on the master domain manager must be accessible, either mounted or shared.

Examples

The following example submits an **rm** command into the **JOBS** job stream with a **follows** dependency:

```
submit docommand="rm apfile";follows sked3
```

The following example submits a **sort** command with the alias **sortit** and places the job in the **reports** job stream with an **at** time of 5:30 P.M.:

```
sbd "sort < file1 > file2";alias=sortit;into=reports;at=1730
```

The following example submits **chmod** commands on all workstations with names beginning with **site**:

```
sbd site@#"chmod 444 file2";alias
```

submit file

Submits a file to be launched as a Workload Scheduler job.

You must have **submit** access to the job. To include **needs** and **prompt** dependencies, you must have **use** access to the resources and global prompts.

Synopsis

sbf
filename[**;alias**[=*name*]][**;into=***jobstream*][**;***joboption*[**;**...]][**;noask**]

Arguments

filename
: Specifies the name of the file, up to 255 characters. Wildcard characters are permitted. The name must be enclosed in quotes (″) if it contains characters other than alphanumeric characters, dashes (**-**), slashes (/), backslashes (\), and underscores (_).

alias=*name*
: Specifies a unique name to be assigned to the job. If you enter the **alias** keyword without specifying a name, a name is constructed using the first two alphanumeric characters of the file's base name followed by a six digit random number. The name is always upshifted. For example, if the file name is **jclttx5**, the generated name will be similar to **JC123456**.

  If you do not include **alias**, a job name is constructed using the first eight alphanumeric characters of the file's base name, upshifted.

  In either of the above cases, if the file name does not start with a letter, you are prompted to use **alias=** *name*.

into=*jobstream*
: The name of the job stream into which the job will be placed for launching. Enter the name as:

[*wkstation*#]*jstream* If you do not specify a *wkstation*, the default is the workstation on which Conman is running. If **into** is not used, the job is added to a job stream named **JOBS**.

joboption
Specify one of the following:

at=*hhmm* [**timezone|tz** *tzname*] [**+***n* **days** | *mm/dd/yy*]

confirmed

every=*rate*

follows=[*netagent***::**][*wkstation*#]*jstream*{.*job* | **@**} | *job*[**,**...]

interactive

logon=*user*

needs=[*num*] [*wkstation*#]*resource*[**,**...]

opens=[*wkstation*#]"*filename*"[**(***qualifier***)**][**,**...] **priority**[=*pri* | **hi** | **go**]

prompt="[**:** | **!**]*text*" | *promptname*[**,**...]

recovery=stop | continue | rerun

recoveryjob | after=[*wkstation*#]*job*

recoveryprompt | abendprompt="*text*"

until=*hhmm* [**timezone|tz** *tzname*] [**+***n* **days** | *mm/dd/yy*]

noask Specifies not to prompt for confirmation before taking action against each qualifying file.

Usage Notes

If you do not specify a workstation with **follows**, **needs**, or **opens**, the default is the workstation on which Conman is running. If the job is executed on a fault-tolerant agent, and you want to include a **prompt** dependency or a **recoveryprompt**, the mozart directory (*WShome*/**mozart**) on the master domain manager must be accessible, either mounted or shared.

Examples

The following example submits a file into the **jobs** job stream. The job name is **myjcl**.

```
submit file=d:\jobs\lib\daily\myjcl
```

The following example submits a file, with a job name of **misjob4**, into a job stream named **missked**. The job needs two units of the **slots** resource.

```
sbf /usr/lib/mis/jcl4;alias=misjob4;into=missked ;needs=2 slots
```

The following example submits all files that have names beginning with **back** into a job stream named **bkup**.

```
sbf "/usr/lib/backup/back@";into=bkup
```

submit job

Submits a job to be launched by Workload Scheduler.

You must have **submit** access to the job. To include **needs** and **prompt** dependencies, you must have **use** access to the resources and global prompts.

To submit a job, you must be running Conman on the master domain manager, or have access to the Workload Scheduler databases on the master domain manager.

Synopsis

sbj [*wkstation*#]*jobname*[**;alias**[=*name*]][**;into**=*jobstream*]
[**;***joboption*[**;**...]][**;noask**]

Arguments

wkstation
: Specifies the name of the workstation on which the job will be launched. Wildcard characters are permitted, in which case, the job is launched on all qualifying workstations. The default is the workstation on which Conman is running. You cannot specify a domain or workstation class.

jobname
: Specifies the name of the job. Wildcard characters are permitted, in which case, all qualifying jobs are submitted. If the job is already in the production plan, and is being submitted into the same job stream, you must use the **alias** argument to assign a unique name.

alias=*name*
: Specifies a unique name to be assigned to the job in place of *jobname*. If you enter the **alias** keyword without specifying a name, a name is constructed using the first two alphanumeric characters of *jobname* followed by a six digit random number. The name is always upshifted. For example, if *jobname* is **jcrttx5**, the generated name will be similar to **JC123456**.

into=*jobstream*
: Specifies the name of the job stream into which the job will be placed for launching. Enter the name as:

  [*wkstation*#]*jstream* If you do not specify a *wkstation*, the default is the workstation on which Conman is running. If **into** is not used, the job is added to a job stream named **jobs**.

joboption
: Specify one of the following:

  at=*hhmm* [**timezone|tz** *tzname*] [**+***n* **days** | *mm/dd/yy*]

  confirmed

  every=*rate*

  follows=[*netagent***::**][*wkstation*#]*jstream*{.*job* | **@**} | *job*[**,**...]

  needs=[*num*] [*wkstation*#]*resource*[**,**...]

  opens=[*wkstation*#]"*filename*"[**(***qualifier***)**][**,**...] **priority**[=*pri* | **hi** | **go**]

  prompt="[**:** | **!**]*text*" | *promptname*[**,**...]

  recovery=stop | continue | rerun

  recoveryjob | after=[*wkstation*#]*job*

  recoveryprompt | abendprompt="*text*"

  until=*hhmm* [**timezone|tz** *tzname*] [**+***n* **days** | *mm/dd/yy*]

noask
: Specifies not to prompt for confirmation before taking action against each qualifying job.

Usage Notes

If you do not specify a workstation with **follows**, **needs**, or **opens**, the default is the workstation of the job. If the job is executed on a fault-tolerant agent, and you want to include a **prompt** dependency or a **recoveryprompt**, the mozart directory (*WShome*/**mozart**) on the master domain manager must be accessible, either mounted or shared.

Examples

The following example submits the **test** jobs into the **JOBS** job stream:

```
submit job=test@
```

The following example submits a job with an alias of **rptx4** and places the job in the **reports** job stream with an **at** time of 5:30 P.M.:

```
sbj rjob4;alias=rptx4;into=reports;at=1730
```

The following example submits job **txjob3** on all workstations whose names begin with **site**:

```
sbj site@#txjob3;alias
```

submit sched

Submits a job stream to be launched by Workload Scheduler.

You must have **submit** access to the job stream. To include **needs** and **prompt** dependencies, you must have **use** access to the resources and global prompts.

To submit a job stream, you must be running Conman on the master domain manager or have access to the Workload Scheduler databases on the master domain manager.

Synopsis

sbs [*wkstation*#]*jstreamname*[**;alias**[=*name*]]
[**;***jstreamoption*[**;**...]][**;noask**]

Arguments

wkstation
: Specifies the name of the workstation on which the job stream will be launched. Wildcard characters are permitted, in which case, the job stream is launched on all qualifying workstations. The default is the workstation on which Conman is running. You cannot specify a domain or workstation class.

jstreamname
: Specifies the name of the job stream. Wildcard characters are permitted, in which case, all qualifying job streams are submitted. If the job stream is already in the production plan, you must use the **alias** argument to assign a unique name.

alias=*name*
: Specifies a unique name to be assigned to the job stream in place of *jstreamname*. If you enter the **alias** keyword without specifying a name, a name is constructed using the first two alphanumeric characters of *jstreamname* followed by a six digit random number. The name is always upshifted. For example, if *jstreamname* is **sttrom**, the generated name will be similar to **ST123456**.

jstreamoption

Enter one of the following:

at=*hhmm* [**timezone|tz** *tzname*] [+*n* **days** | *mm/dd/yy*]

carryforward

follows=[*netagent***::**][*wkstation***#**]*jstream*{*.job* | **@**} | *job*[**,**...]

limit=*joblimit*

needs=[*num*] [*wkstation***#**]*resource*[**,**...]

opens=[*wkstation***#**]"*filename*"[**(***qualifier***)**][**,**...] **priority**[=*pri* | **hi** | **go**]

prompt="[**:** | **!**]*text*" | *promptname*[**,**...]

until=*hhmm* [**timezone|tz** *tzname*] [+*n* **days** | *mm/dd/yy*]

noask Specifies not to prompt for confirmation before taking action against each qualifying job stream.

Usage Notes

If you do not specify a workstation with **follows**, **needs**, or **opens**, the default is the workstation of the job stream. If the job stream is executed on a fault-tolerant agent, and you want to include a **prompt** dependency or a **recoveryprompt**, the mozart directory (*WShome*/**mozart**) on the master domain manager must be accessible, either mounted or shared.

Examples

The following example submits the **adhoc** job stream on workstation **site1** and flags it as a **carryforward** job stream:

```
submit sched=site1#adhoc;carryforward
```

The following example submits job stream **fox4** with a job limit of **2**, a priority of **23**, and an **until** time of midnight:

```
sbs fox4;limit=2;pri=23;until=0000
```

The following example submits job stream **sched3** on all workstations with names that start with **site**:

```
sbs site@#sched3
```

switchmgr

Switches domain management from the current domain manager to a backup domain manager.

You must have **start** and **stop** access to the backup domain manager.

Synopsis

switchmgr *domain***,***newmgr*

Arguments

domain	Specifies the domain in which you want to switch managers.
newmgr	Specifies the name of the new domain manager. This must be a workstation in the same domain, and should be defined beforehand as a fault-tolerant agent with Resolve Dependencies and Full Status enabled.

Usage Notes

The command stops a specified workstation and restarts it as the domain manager. All domain member workstations are informed of the switch, and the old domain manager is converted to a fault-tolerant agent in the domain.

The identification of domain managers is carried forward to each new day's Symphony file, so that switches remain in effect until a subsequent **switchmgr** command is executed. However, if new day processing (the **Jnextday** job) is performed on the old domain manager, the domain will act as though another **switchmgr** command had been executed and the old domain manager will automatically resume domain management responsibilities.

Examples

The following example switches the domain manager to workstation **orca** in the **masterdm** domain:

```
switchmgr masterdm,orca
```

The following example switches the domain manager to workstation **ruby** in the **bldg2** domain:

```
switchmgr bldg2,ruby
```

System Command

Executes a system command.

Synopsis

[**:** | **!**] *sys-command*

Arguments

sys-command	Specifies any valid system command. The prefix (: or !) is required only when a command name has the same spelling as a Conman command.

Examples

The following example executes a **ps** command on UNIX:

```
ps -ef
```

The following example executes a **dir** command on Windows NT:

```
dir \bin
```

tellop

Sends a message to the Workload Scheduler console.

Synopsis

to [*text*]

Arguments

text	Specifies the text of the message. The message can contain up to 900 characters.

Usage Notes

If **tellop** is issued on the master domain manager, the message is sent to all linked workstations. If issued on a domain manager, the message is sent to all of the linked agents in its domain and subordinate domains. If issued on a workstation other than a domain manager, the message is sent only to its domain manager if it is linked. The message is displayed only if the console message level is greater than zero. See “console” on page 133.

If **tellop** is entered alone, it prompts for the message text. At the prompt, type each line and press the Return key. At the end of the message, type two slashes (//) or a period (.), and press the Return key. You can use the new line sequence (\n) to format messages. Typing **Control+c** at any time will exit the **tellop** command without sending the message.

Examples

The following example sends a message:

```
tellop TWS will be stopped at\n4:30 for 15 minutes.
```

The following example prompts for text before sending a message:

```
to
TELLOP>*******************************
TELLOP>*  TWS will be stopped at      *
TELLOP>*  4:30 for 15 minutes.        *
TELLOP>*******************************
TELLOP>//
```

unlink

Closes communications links between workstations.

You must have **unlink** access to the target workstation.

Synopsis

unlink[*domain*!]*wkstation*[**;noask**]

Arguments

domain	Specifies the name of the domain in which to close links. Because workstations have unique names, a domain is not needed when unlinking a specific workstation. Wildcard characters are permitted. This argument is useful when unlinking more than one workstation in a domain. For example, to unlink all the agents in domain **stlouis**, use the following command: `link stlouis!@` If you do not specify *domain*, and *wkstation* includes wildcard characters, the default domain is the one in which Conman is running.
wkstation	Specifies the name of the workstation to be unlinked. Wildcard characters are permitted.
noask	Specifies not to prompt for confirmation before taking action on each qualifying workstation.

Usage Notes

Assuming that a user has **unlink** access to the workstations being unlinked, the following rules apply:

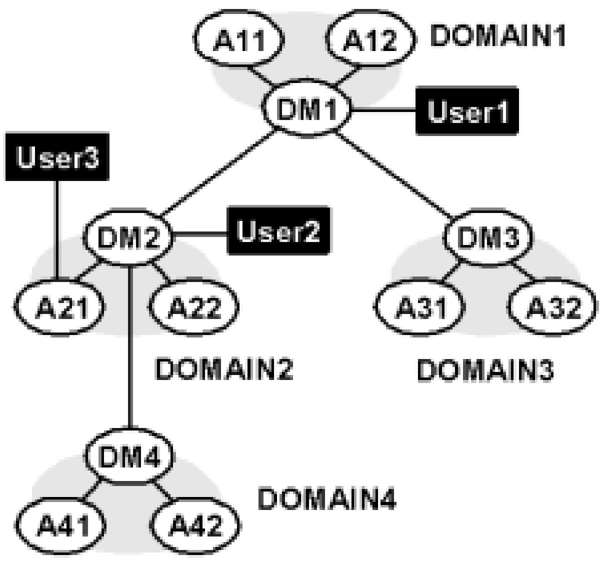
- A user running Conman on the master domain manager can unlink any workstation in the network.
- A user running Conman on a domain manager other than the master can unlink any workstation in its own domain and subordinate domains. The user cannot unlink workstations in peer domains.

- A user running Conman on an agent can unlink any workstation in its local domain.

For additional information see “link” on page 151.

Examples

The illustration and table below show the links closed by **unlink** commands executed by users in various locations in the network.



DM*n* are domain managers and **A***nn* are agents.

Command	Closed by User1	Closed by User2	Closed by User3
unlink @!@	All links are closed.	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4 DM4-A41 DM4-A42	DM2-A21 DM2-A22

Command	Closed by User1	Closed by User2	Closed by User3
unlink @	DM1-A11 DM1-A12 DM1-DM2 DM1-DM3	DM1-DM2 DM2-A21 DM2-A22 DM2-DM4	DM2-A21 DM2-A22
unlink domain3!@	DM3-A31 DM3-A32	Not allowed.	Not allowed.
unlink domain4!@	DM4-A41 DM4-A42	DM4-A41 DM4-A42	Not allowed.
unlink DM2	DM1-DM2	Not applicable.	DM2-A21
unlink A42	DM4-A42	DM4-A42	Not allowed.
unlink A31	DM3-A31	Not allowed.	Not allowed.

version

Displays Conman's program banner.

Synopsis

version

Examples

The following example displays Conman's program banner:

```
%version
MAESTRO for UNIX (HPUX)/CONMAN 6.0 (3.34)  (C) Tivoli Systems 1998
Schedule 5/16/98 (#7) on SFO.  Batchman down.  Limit: 6, Fence: 0
```

4

Utility Commands

This chapter describes Tivoli Workload Scheduler's utility commands. These commands are tools to help you manage Tivoli Workload scheduler. The commands, with the exception of **StartUp** and **version**, are installed in *TWShome*/**bin** directory. **StartUp** is installed in the *TWShome* directory, and **version** is installed in the *TWShome*/**version** directory.

Command Descriptions

Command	Description
at | mat	For UNIX only. Submits a job to be executed at a specific time.
batch | mbatch	For UNIX only. Submits a job to be executed as soon as possible.
caxtract	Extracts information about calendars.
cpuinfo	Returns information from a workstation definition.
datecalc	Converts date and time to a desired format
dbexpand	Expands Tivoli Workload Scheduler's databases.
delete	Removes script files and standard list files by name.
evtsize	Defines the maximum size of event message files.
jbxtract	Extracts information about jobs.
jobinfo	Returns information about a job.
jobstdl	Returns the pathnames of standard list files.

Command	Description
listproc	For Windows NT only. Lists processes. This command is unsupported.
killproc	For Windows NT only. Kills processes. This command is unsupported.
maestro	Returns Tivoli Workload Scheduler’s home directory.
makecal	Creates custom calendars.
morestdl	Displays the contents of standard list files.
parms	Displays, changes, and adds parameters.
paxtract	Extracts information about parameters.
prxtract	Extracts information about prompts.
r11xtr	Extracts information about job streams.
release	Releases units of a resource.
rextract	Extracts information about resources.
rmstdlist	Removes standard list files based on age.
showexec	For UNIX only. Displays information about executing jobs.
StartUp	Starts the Netman process.
version	For UNIX only. Displays version information.
xrxtrct	Extracts information about cross-references.
wmaeutil	Extracts information about cross-references.

at | batch Commands

For UNIX only. Submits ad hoc commands and jobs to be launched by TWS. When you install TWS, the following links are created by default:

/usr/bin/mat —>*TWShome*/**bin/at**
/usr/bin/mbatch —> *TWShome*/**bin/batch**

See **at.allow** and **at.deny** below for information about the availability to users.

Synopsis

at | mat -v | -u

at | mat -s*jstream* | **-q***queuetime-spec*

batch | mbatch -v | -u

batch | mbatch [**-s** *jstream*]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

-s *jstream*

Specifies the name of a job stream into which the job is submitted. If the job stream does not exist, it is created. The name must start with a letter, and can contain alphanumeric characters and dashes. For non-expanded databases, it can contain up to eight characters. For expanded databases, it can contain up to 16 characters.

If the **-s** and **-q** arguments are omitted, a job stream name is selected based on the value of the environment variable ATSCRIPT. If ATSCRIPT contains the word **maestro**, the job stream name will be the first eight characters of the user’s group name. If ATSCRIPT is not set, or is set to a value other than **maestro**, the job stream name will be **at** (for jobs submitted with **at**), or **batch** (for jobs submitted with **batch**).

See “Other Considerations” on page 234 for more information about job streams.

-q*queue*

Specifies to submit the job into a job stream with the name *queue*, which can be a single letter (a through z). See “Other Considerations” on page 234 for more information about job streams.

time-spec

Specifies the time at which the job will be launched. For **at** jobs only. The syntax is the same as that used with the UNIX **at** command.

Usage Notes

After entering **at** or **batch**, enter the commands that constitute the job. End each line of input by pressing the Return key. The entire sequence is ended with end-of-file (usually **Control+d**), or by entering a line with a period (.). Alternatively, use an angle bracket (<) to read commands from a file. See the example below “Other Considerations” on page 234.

Information about **at** and **batch** jobs is sent to the master domain manager, where the jobs are added to job streams in the production plan, Symphony. The jobs are launched based on the dependencies included in the job streams.

The UNIX shell used for jobs submitted with Tivoli Workload Scheduler’s **at** and **batch** commands is determined by the SHELL_TYPE variable in the **jobmanrc** configuration script. Do not use the C shell. For more information, see the *Tivoli Workload Scheduler User’s Guide*.

Once submitted, jobs are launched in the same manner as other scheduled jobs. Each job executes in the submitting user’s environment. To ensure that the environment is complete, **set** commands are inserted into the script to match the variable settings in the user's environment.

Replacing the UNIX Commands

The standard UNIX **at** and **batch** commands can be replaced with the TWS commands. The following commands illustrate how to replace the UNIX **at** and **batch** commands:

```
$ mv /usr/bin/at /usr/bin/uat
$ mv /usr/bin/batch /usr/bin/ubatch
$ ln -s TWShome/bin/at /usr/bin/at
$ ln -s TWShome/bin/batch /usr/bin/batch
```

Note that the **customize** script installs links to its **at** and **batch** commands by default. This permits the commands to be run as follows:

```
/usr/bin/mat
```

```
/usr/bin/mbatch
```

The at.allow and at.deny Files

The **at** and **batch** commands use the files **/usr/lib/cron/at.allow** and **/usr/lib/cron/at.deny** to restrict usage. If the **at.allow** file exists, only users listed in the file are allowed to use **at** and **batch**. If the file does not exist, **at.deny** is checked to see if the user is explicitly denied permission. If neither of the files exists, only the **root** user is permitted to use the commands. If the commands are executed as **mat** and **mbatch**, **at.allow** and **at.deny** are ignored, and no restrictions apply.

Script Files

The commands entered with **at** or **batch** are stored in script files. The file are created by TWS using the following naming convention:

TWShome/**atjobs**/*epoch.sss*

where:

epoch	The number of seconds since 00:00, 1/1/70.
sss	The first three characters of the job stream name.

Note: TWS removes script files for jobs that are not carried forward. However, Tivoli recommends that you monitor the disc space in the **atjobs** directory and remove older files if necessary.

Job Names

All **at** and **batch** jobs are given unique names by TWS when they are submitted. The names consist of the user's process ID (PID) preceded by the user's name truncated so as not to exceed eight characters. The resulting name is upshifted.

Other Considerations

Tivoli recommends that the job streams into which **at** and **batch** jobs are submitted be created beforehand with Composer. The job streams can contain dependencies that determine when the jobs will be launched. At a minimum, the job streams should contain the **carryforward** keyword. This will ensure that jobs that do not complete, or are not launched, during the current day will be carried forward to the next day's production plan. Some other suggestions regarding **at** and **batch** job streams:

- Include the expression **on everyday** to have the job streams selected every day.
- Use the **limit** keyword to limit the number of submitted jobs that can be run concurrently.
- Use the **priority** keyword to set the priority of submitted jobs relative to other jobs.

Examples

The following example submits a job into job stream **sched8** to be launched as soon as possible:

```
mbatch -s sched8
command <Return>
...
<Control d>
```

The following example submits a job into job stream **k** to be launched at 9:45 P.M.:

```
mat -qk 21h45
command <Return>
...
<Control d>
```

The following example submits a job to be launched two hours from the time at which the command was entered:

```
TWShome/bin/at now + 2 hours
command <Return>
...
<Control d>
```

If the variable ATSCRIPT is null, the job is submitted into a job stream having the same name of the user’s group. Otherwise, it is submitted into a job stream named **at**.

The following example submits a job into job stream **sked-mis** to be launched at 5:30 P.M.:

```
mat -s sked-mis 17h30
command <Return>
...
<Control d>
```

The following example is the same as the previous example, except that the job’s commands are read from a file:

```
mat -s sked-mis 17h30 < ./myjob
```

The fact that the commands are read from a file does not change the way they are processed. That is, the commands are copied from the **./myjob** file into a script file.

caxtract Command

Extracts information about calendars from the TWS database.

Synopsis

caxtract[**-v** | **-u**] [**-o** *file*]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.
-o *file*	Specifies the output file. The default is **stdout**.

Command Output

Each calendar record contains tab-delimited, variable length fields. The fields are described in the following table.

Field	Description	Max Length (bytes)
1	calendar name	8
2	calendar description	64

Examples

Extract information about all calendar definitions and direct the output to the file **cainfo**:

```
caxtract -o cainfo
```

cpuinfo Command

Returns information from a workstation definition.

Synopsis

cpuinfo -v | -u

cpuinfo *wkstation*[*infotype*] [...]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

wkstation

The name of the workstation.

infotype
The type of information to display. Specify one or more of the following:

os_type
Returns the value of the **os** field: MPEV, MPIX, UNIX, WNT, or OTHER.

node Returns the value of the **node** field.

port Returns the value of the **port** field.

autolink
Returns the value of the **autolink** field: ON or OFF.

fullstatus
Returns the value of the **fullstatus** field: ON or OFF.

resolvedep
Returns the value of the **resolvedep** field: ON or OFF.

host Returns the value of the **host** field.

method
Returns the value of the **access** field.

server Returns the value of the **server** field.

type Returns the type of workstation: MASTER, MANAGER, FTA, S-AGENT, and X-AGENT.

time_zone
Returns the time zone of the workstation. For an extended agent, the field is blank.

version
Returns the TWS version that is running on the workstation. For an extended agent, the field is blank.

info Returns the operating system version and workstation model. For an extended agent, the field is blank.

Usage Notes

The values are returned, separated by new lines, in the same order that the arguments were entered on the command line. If no arguments are specified, all applicable information is returned with labels, separated by new lines.

Examples

The examples below are based on the following workstation definition:

```
cpuname  oak
os       UNIX
node     oak.tivoli.com
tcpaddr  31111
     for maestro
          autolink    on
          fullstatus  on
          resolvedep  on
end
```

The following example prints the **os** type for workstation **oak**:

```
>cpuinfo oak os_type
UNIX
```

The following example prints the **node** and **port** for workstation **oak**:

```
>cpuinfo oak node port
oak.tivoli.com
31111
```

The following example prints all information for workstation **oak**:

```
>cpuinfo oak
OS TYPE:UNIX
NODE:oak.tivoli.com
PORT:31111
AUTOLINK:ON
FULLSTATUS:ON
RESOLVEDEP:ON
HOST:
METHOD:
SERVER:
TYPE: FTA
TIME ZONE:US/Pacif
VERSION:6.1
INFO:SunOS 5.3 Generic 1016 sun4m
```

datecalc Command

Resolves date expressions and returns dates in desired formats.

Synopsis

datecalc -v | -u

datecalc *base-date* [*offset*] [**pic** *format*]

datecalc -t *time* [*base-date*] [*offset*] [**pic** *format*]

datecalc*yyyymmddhhtt* [*offset*] [**pic** *format*]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

base-date

Specify one of the following:

day | *date* | **today** | **tomorrow** | **scheddate**

where:

day Specifies a day of the week. Valid values are: **su**, **mo**, **tu**, **we**, **th**, **fr**, or **sa**.

date Specifies a date, in the format *element*/*element*[/*element*], where *element* is: *d*[*d*], *m*[*m*], and *yy*[*yy*].

If two digits are used for the year (*yy*), a number greater than 70 is a 20th century date, and a number less than 70 is a 21st century date.

Valid values for the month (*m*[*m*]) are **jan**, **feb**, **mar**, **apr**, **may**, **jun**, **jul**, **aug**, **sep**, **oct**, **nov**, or **dec**.

The slashes (/) can be replaced by dashes (-), periods (.), commas (,), or spaces. For example, any of the following can be entered for March 28, 1999:

03/28/99
3-28-1999
28.mar.99
99,28,3
mar 28 1999
28 3 99

If numbers are used, it is possible to enter an ambiguous date, for example, 2,7,98. In this case, **datecalc** uses the date format defined in the TWS message catalog to interpret the date. If the date does not match the format, **datecalc** generates an error message.

today Specifies the current system date.

tomorrow
Specifies the current system date plus one day, or, in the case of time calculations, plus 24 hours.

scheddate
Specifies the date of the production plan.

-t *time* **[***base-date***]**
Specify *time* in one of the following formats:

now | **noon** | **midnight** | [*h*[*h*][[:]*mm*] [**am** | **pm**] [**zulu**]

where:

now Specifies the current system date and time.

noon Specifies 12:00 P.M. (or 1200).

midnight
Specifies 12:00 A.M. (or 0000).

*h***[***h***][[:]***mm***]**
Specifies the hour and minute in 12-hour time (if **am** or **pm** are used), or 24-hour time. The optional colon (:) delimiter can be replaced by a period (.), a comma (,), an apostrophe (’), the letter **h**, or a space. For example, any of the following can be entered for 8:00 P.M.:

8:00pm
20:00
0800pm
2000
8pm
20
8,00pm
20.00
8\'00pm
20 00

zulu Specifies that the time you entered is Greenwich Mean Time (Universal Coordinated Time). **datecalc** will convert it to the local time.

yyyymmddhhtt
Specifies the year, month, day, hour, and minute expressed in exactly twelve digits. For example, for 1999, May 7, 9:15 A.M., enter the following: **199905070915**

offset Specifies an offset from *base-date* in the following format:

{[+ | > | - | < *number* | **nearest**] | **next**} **day[s]** | **weekday[s]** | **workday[s]** | **week[s]** | **month[s]** | **year[s]** | **hour[s]** | **minute[s]** | *day* | *calendar*

where:

+ | > (Plus) Specifies an offset to a later date or time. Be sure to escape the angle bracket (″\>″).

- | < (Minus) Specifies an offset to an earlier date or time. Be sure to escape the angle bracket (″\<″).

number
The number of units of the specified type.

nearest
Specifies an offset to the nearest occurrence of the unit type (earlier or later).

next Specifies the next occurrence of the unit type.

day[s] Specifies every day.

weekday[s]
Specifies every day except Saturday and Sunday.

workday[s]
Same as **weekday[s]**, but also excludes the dates on the **holidays** calendar.

week[s]
Specifies seven days.

month[s]
Specifies calendar months.

year[s]
Specifies calendar years.

hour[s]
Specifies clock hours.

minute[s]
Specifies clock minutes.

day Specifies a day of the week. Valid values are: **su**, **mo**, **tu**, **we**, **th**, **fr**, or **sa**.

calendar
Specifies the entries in a calendar by this name.

pic *format*
Specifies the format in which the date and time are returned. The *format* characters are as follows:

m	Month number.
d	Day number.
y	Year number.
j	Julian day number.
h	Hour number.
t	Minute number.
^	One space. (This character must be escaped (\) in the Bourne shell.)

You can also include punctuation characters. These are the same as the delimiters used in *date* and *time*.

If a format is not defined, **datecalc** returns the date and time in the format defined by the Native Language Support (NLS) enviroment variables. If the NLS variables are not defined, the native language defaults to C.

Examples

The following example returns the next date, from today, on the monthend calendar:

```
>datecalc today next monthend
```

In the following examples, the current system date is Friday, April 9, 1999.

```
>datecalc today +2 days pic mm/dd/yy
04/11/99
```

```
>datecalc today next tu pic yy\^mm\^dd
99 04 13
```

```
>LANG=american;export LANG
>datecalc -t 14:30 tomorrow
Sat, Apr 10, 1999 02:30:00 PM
```

```
>LANG=french;datecalc -t 14:30 tomorrow
Samedi 10 avril 1999 14:30:00
```

In the following example, the current system time is 10:24.

```
>datecalc -t now \> 4 hours pic hh:tt
14:24
```

dbexpand Command

Converts the databases on the master domain manager from non-expanded mode to expanded mode. The command sets the **expanded version** global option to **yes**, and makes backup copies of your old database files that you can use to return to non-expanded mode if necessary.

If you update your network in stages and it contains workstations running Tivoli Maestro Version 5.x or earlier, you must use non-expanded databases until all of your computers have been updated to Tivoli Maestro 6.x or Tivoli Workload Scheduler. When all of the computers are updated, run **dbexpand** on the master domain manager to convert the databases to expanded mode.

Synopsis

dbexpand -v | -u

dbexpand -n[**-b***backup-dir*]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

-n

Specifies not to prompt for a backup directory name. If **-b** is included, the named directory is used for backup. If **-b** is not included, the default directory is used. In either case, if the directory exists, it is overwritten.

-b *backup-dir*

Specifies a directory in which to backup the database files. The default directory is:

TWShome/**mozart/mozart.old**

If **-n** is omitted and the backup directory already exists, you are prompted for a backup directory name.

Usage Notes

You can run the **dbexpand** command without stopping TWS. However, you cannot submit jobs or job streams into the current production plan until after the new day turnover occurs. For this reason, Tivoli recommends that you run **dbexpand** shortly before the **Jnextday** job runs.

Examples

The following example expands the databases and backs up the current files in the **/usr/lib/maestro/temp** directory. If the directory already exists, it overwrites its contents.

```
dbexpand -n -b /usr/lib/maestro/temp
```

The following example expands the databases and backs up the current files in the **c:\programs\wsched\temp** directory. A prompt is displayed if the directory already exists.

```
dbexpand -b c:\programs\wsched\temp
```

delete Command

Removes files. This command is intended to remove standard list files. The users **maestro** and **root** can remove any file. Other users can remove only files associated with their own jobs.

Synopsis

delete -v | -u

delete *filename*

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

filename

Specifies the name of the file or group of files to be removed. The name must be enclosed in quotes (″) if it contains characters other than the following: alphanumerics, dashes (-), slashes (/), backslashes (\), and underscores (_). Wildcard characters are permitted.

CAUTION:
Use this command carefully. Improper use of wildcard characters can result in removing files accidentally.

Examples

The following example removes all the standard list files for 4/11/99:

```
delete d:\win32app\maestro\stdlist\1999.4.11\@
```

The following script, included in a scheduled job on UNIX, removes the job’s standard list file if there are no errors:

```
...
#Remove the stdlist for this job:
if grep -i error $UNISON_STDLIST
then
exit 1
```

```
else
`maestro`/bin/delete $UNISON_STDLIST
fi
...
```

Note that the standard configuration script, **jobmanrc**, sets the variable UNISON_STDLIST to the name of the job's standard list file. For more information about **jobmanrc** refer to the *Tivoli Workload Scheduler User's Guide*.

evtsize Command

Defines the size of the TWS event files. This command is used to increase the size of an event file after receiving the message, “End of file on events file.” You must be the **maestro** or **root** user to run **evtsize**.

Synopsis

evtsize -v | -u

evtsize *filename size*

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.
filename	The name of the event file. Specify one of the following: **Courier.msg** **Intercom.msg** **Mailbox.msg** **maestro.msg** **netman.msg** **unison.msg** **pobox/***wkstation***.msg**
size	The maximum size of the event file in bytes. When first built by TWS, the maximum size is set to 1 MB.

Examples

The following example sets the maximum size of the **Intercom** file to 2 MB:

```
evtsize Intercom.msg 2000000
```

The following example sets the maximum size of the **pobox** file for workstation **chicago** to 1.5 MB:

```
evtsize pobox\chicago.msg 1500000
```

jbxtract Command

Extracts information about jobs from the TWS database.

Synopsis

jbxtract[**-v** | **-u**] [**-j** *job*] [**-c** *wkstat*] [**-o** *file*]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

-j *job* Specifies the job for which extraction is performed. The default is all jobs.

-c *wkstat*
Specifies the workstation of jobs for which extraction is performed. The default is all workstations.

-o *file* Specifies the output file. The default is **stdout**.

Command Output

The **MAESTRO_OUTPUT_STYLE** variable specifies the the output style for long object names. Set the variable to **LONG** to use full length (long) fields for object names. If the variable is not set or is set to anything other than **LONG**, and the global option **expanded version** is set to **yes**, long names are truncated to eight characters and a plus sign. For example: **A1234567+**. If the **expanded version** option is set to **no**, long names are truncated to eight characters.

Each job record contains tab-delimited, variable length fields. The fields are described in the following table.

Field	**Description**	**Max Length (bytes) ***
1	workstation name	8/16
2	job name	8/16
3	job script file name	256/4096
4	job description	65
5	recovery job name	16

Field	Description	Max Length (bytes) *
6	recovery option (0=stop, 1=rerun, 2=continue)	5
7	recovery prompt text	64
8	auto-documentation flag (0=disabled, 1=enabled)	5
9	job login user name	36
10	job creator user name	36
11	number of successful runs	5
12	number of abended runs	5
13	total elapsed time of all job runs	8
14	total cpu time of all job runs	8
15	average elapsed time	8
16	last run date (yymmdd)	8
17	last run time (hhmm)	8
18	last cpu seconds	8
19	last elapsed time	8
20	maximum cpu seconds	8
21	maximum elapsed time	8
22	maximum run date (yymmdd)	8
23	minimum cpu seconds	8
24	minimum elapsed time	8
25	minimum run date (yymmdd)	8
* non-expanded databases/expanded databases		

Examples

Extract information about job **myjob** on workstation **main** and direct the output to the file **jinfo**:

```
jbxtract -j myjob -c main -o jinfo
```

jobinfo Command

Used in a job script to return information about the job.

Synopsis

jobinfo -v | -u

jobinfo *job-option* [...]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

job-option

The job option. Specify one or more of the following:

confirm_job
Returns **YES** if the job requires confirmation.

is_command
Returns **YES** if the job was scheduled or submitted using the **docommand** construct.

job_name
Returns the job's name without the workstation and job stream names.

job_pri
Returns the job's priority level.

programmatic_job
Returns **YES** if the job was submitted with the TWS **at** or **batch** command.

re_job
Returns **YES** if the job is being rerun as the result of a Conman **rerun** command, or the rerun recovery option.

re_type
Returns the job's recovery option (**stop**, **continue**, or **rerun**).

rstrt_flag
: Returns **YES** if the job is being run as the recovery job.

time_started
: Returns the time the job started executing.

Usage Notes

Job option values are returned, separated by new lines, in the same order they were requested.

Examples

The script file **/jcl/backup** is documented twice, giving it the job names **partback** and **fullback**. If the job runs as **partback**, it performs a partial backup. If it runs as **fullback**, it performs a full backup. Within the script, commands like the following are used to make the determination:

```
#Determine partial (1) or full (2):
if [ "`\`maestro\`/bin/jobinfo job_name`" = "PARTBACK" ]
then
bkup=1
else
bkup=2
fi
...
```

jobstdl Command

Returns the names of standard list files.

Synopsis

jobstdl -v | -u

jobstdl [**-day** *num*] [**-first** | **-last** | **-num** *n* | **-all**] [**-name** *jstream.job* | *jobnum*]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

-day *num*
Returns the names of standard list files that are the specified number of days old (1 for yesterday, 2 for the day before yesterday, and so on). The default is zero (today).

-first Returns the name of the first qualifying standard list file.

-last Returns the name of the last qualifying standard list file.

-num *n*
Returns the name of the standard list file for the specified run of a job.

-all Returns the name of all qualifying standard list files.

-name *jstream.job*
Specifies the name of the job stream and job for which standard list file names are returned.

jobnum
Specifies the job number of the job for which standard list file names are returned.

Usage Notes

File names are returned in a format suitable for input to other commands. Multiple names are returned separated by one space.

Examples

The following example returns the path names of all standard list files for the current day:

```
jobstdl
```

The following example returns the path name of the standard list for the first run of job **mailxhg1.getmail** on the current day:

```
jobstdl -first -name mailxhg1.getmail
```

The following example returns the path name of the standard list for the second run of job **mailxhg1.getmail** on the current day:

```
jobstdl -num 2 -name mailxhg1.getmail
```

The following example returns the path names of the standard list files for all runs of job **mailxhg1.getmail** from three days ago:

```
jobstdl -day 3 -name mailxhg1.getmail
```

The following example returns the path name of the standard list for the last run of job **mailxhg1.getmail** from four days ago:

```
jobstdl -day 4 -last -name mailxhg1.getmail
```

The following example returns the path name of the standard list for job number **455**:

```
jobstdl 455
```

The following example prints the contents of the standard list file for job number **455**:

```
cd `maestro`/bin
lp -p 6 `jobstdl 455`
```

maestro Command

Returns the path name of the Tivoli Workload Scheduler's home directory, referred to as *TWShome*

.

Synopsis

maestro[**-v** | **-u**]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.

Examples

The following example displays the TWS home directory:

```
$ maestro
/usr/lib/maestro
```

The following example changes the directory to the TWS home directory:

```
$ cd `maestro`
```

makecal Command

Creates a custom calendar. On UNIX, the Korn shell is required to execute this command.

Synopsis

makecal [**-v** | **-u**]

makecal [**-c** *name*] **-d** *n* | **-e** | {**-f 1** | **2** | **3 -s** *date*} | **-l** | **-m** | **-p** *n* | {**-r** *n* **-s** *date*} | **-w** *n* [**-i** *n*] [**-x** | **-z**]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.

-c *name*
Specifies a name for the calendar. The name can contain up to eight alphanumeric characters and must start with a letter. The default name is: **C***hhmm*, where *hhmm* is the current hour and minute.

-d *n*	Specifies the *n*th day of every month.
-e	Specifies the last day of every month.

-f 1 | 2 | 3
Creates a fiscal month-end calendar containing the last day of the fiscal month. Specify one of the following formats:

1	4-4-5 week format
2	4-5-4 week format
3	5-4-4 week format

This argument requires the **-s** argument.

-i *n*	Specifies to put *n* dates in the calendar.
-l	Specifies the last workday of every month. For this argument to work properly, the production plan (Symphony file) and the **holidays** calendar must already exist.
-m	Specifies the first and fifteenth days of every month.

-p *n* Specifies the workday before the *n*th day of every month. For this argument to work properly, the production plan (Symphony file) and the **holidays** calendar must already exist

-r *n* Specifies every *n*th day. This argument requires the **-s** argument.

-s *date*

Specifies the starting date for the **-f** and **-r** arguments. The date must be enclosed in quotation marks, and must be valid and unambiguous, for example, use **JAN 10 1999**, not **1/10/99**. See *base-date* for **datecalc** below “datecalc Command” on page 240 for more information about date formats.

-w *n* Specifies the workday after the *n*th of the month. For this argument to work properly, the production plan (Symphony file) and the **holidays** calendar must already exist.

-x Sends the calendar output to **stdout** rather than adding it to the calendar database.

-z Adds the calendar to the calendar database and compiles the production plan (Symphony file). WARNING: This argument re-submits jobs and job streams from the current day’s production plan. It may be necessary to cancel job streams and jobs.

Examples

The following example makes a two-year calendar with the last day of every month selected:

```
makecal -e -i24
```

The following example makes a calendar wtih 30 days that starts on May 30, 1999, and has every third day selected:

```
makecal -r 3 -s "30 MAY 1999" -i30
```

morestdl Command

Displays the contents of standard list files.

Synopsis

morestdl -v | -u

morestdl [**-day** *num*] [**-first** | **-last** | **-num** *n* | **-all**] [**-name** *jstream.job* | *jobnum*]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

-day *num*
Displays standard list files that are the specified number of days old (1 for yesterday, 2 for the day before yesterday, and so on). The default is zero (today).

-first Displays the first qualifying standard list file.

-last Displays the last qualifying standard list file.

-num *n*
Displays the standard list file for the specified run of a job.

-all Displays all qualifying standard list files.

-name *jstream.job*
Specifies the name of the job stream and job for which the standard list file is displayed.

jobnum
Specifies the job number of the job for which the standard list file is displayed.

Examples

The following example displays the standard list file for the first run of job **mailxhg1.getmail** on the current day:

```
morestdl -first -name mailxhg1.getmail
```

The following example displays the standard list file for the second run of job **mailxhg1.getmail** on the current day:

```
morestdl -num 2 -name mailxhg1.getmail
```

The following example displays the standard list files for all runs of job **mailxhg1.getmail** from three days ago:

```
morestdl -day 3 -name mailxhg1.getmail
```

The following example displays the standard list file for the last run of job **mailxhg1.getmail** from four days ago:

```
morestdl -day 4 -last -name mailxhg1.getmail
```

The following example prints the standard list file for job number 455:

```
morestdl 455 | lp -p 6
```

parms Command

Returns the current value of a parameter, changes the value of a parameter, or adds a new parameter.

Synopsis

parms [**-v** | **-u**]

parms *name*

parms -c *name value*

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

name Specifies the name of the parameter whose value is displayed.

-c *name value*
Specifies the name and the value of a parameter. The value can contain up to 72 characters. Quotation marks are required if the value contains special characters. If the parameter does not exist, it is added to the database. If the parameter already exists, its value is changed.

Usage Notes

When **parms** is run at the command line without arguments, it prompts for parameter names and values.

Examples

The following example returns the value of **myparm**:

```
parms myparm
```

The following example changes the value of **myparm**:

```
parms -c myparm "item 123"
```

The following example creates a new parameter named **hisparm**:

```
parms -c hisparm "item 789"
```

The following example changes the value of **myparm** and adds **herparm**:

```
parms
Name of parameter ? myparm < Return>
Value of parameter? "item 456" < Return>
Name of parameter ? herparm < Return>
Value of parameter? "item 123" < Return>
Name of parameter ? < Return>
```

paxtract Command

Extracts information about parameters from the TWS database.

Synopsis

paxtract[**-v** | **-u**] [**-o** *file*]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.
-o *file*	Specifies the output file. The default is **stdout**.

Command Output

Each parameter record contains tab-delimited, variable length fields. The fields are described in the following table.

Field	Description	Max Length (bytes)
1	parameter name	8
2	parameter value	64

Examples

Extract information about all parameter definitions and direct the output to the file **painfo**:

```
paxtract -o painfo
```

prxtract Command

Extracts information about prompts from the TWS database.

Synopsis

prxtract[**-v** | **-u**] [**-o** *file*]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.
-o *file*	Specifies the output file. The default is **stdout**.

Command Output

Each prompt record contains tab-delimited, variable length fields. The fields are described in the following table.

Field	Description	Max Length (bytes)
1	prompt name	8
2	prompt value	200

Examples

Extract information about all prompt definitions and direct the output to the file **prinfo**:

```
prxtract -o prinfo
```

r11xtr Command

Extracts information about job streams from the TWS database.

Synopsis

prxtract[**-v** | **-u**] [**-m** *mm*[*yy*]] [**-c** *wkstat*] [**-o** *file*]

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

-m *mm***[***yy***]**
Specifies the month (*mm*) and, optionally, the year (*yy*) of the job streams. The default is the current month and year.

-c *wkstat*
Specifies the workstation of the job streams. The default is all workstations.

-o *file* Specifies the output file. The default is **stdout**.

Command Output

The **MAESTRO_OUTPUT_STYLE** variable specifies the the output style for long object names. Set the variable to **LONG** to use full length (long) fields for object names. If the variable is not set or is set to anything other than **LONG**, and the global option **expanded version** is set to **yes**, long names are truncated to eight characters and a plus sign. For example: **A1234567+**. If the **expanded version** option is set to **no**, long names are truncated to eight characters.

Each job stream record contains tab-delimited, variable length fields. The fields are described in the following table.

Field	**Description**	**Max Length (bytes) ***
1	workstation name	8/16
2	job stream name	8/16
3	job stream date (yymmdd)	6
4	estimated cpu seconds	6

Field	Description	Max Length (bytes) *
5	multiple workstation flag (* means some jobs run on other workstations)	1
6	number of jobs	4
7	day of week (Su, Mo, Tu, We, Th, Fr, Sa)	2
* non-expanded databases/expanded databases		

Examples

Extract information about job streams on June 1999 for workstation **main**:

```
r11xtr -m 0699 -c main
```

Extract information about job streams on June of this year for all workstations, and direct the output to file **r11out**:

```
r11xtr -m 06 -o r11out
```

release Command

Releases units of a resource at the job stream or job level.

Synopsis

release -v | -u

release [**-s**] [*wkstation*#]*resourcename*

Arguments

-v Displays the command version and exits.

-u Displays command usage information and exits.

-s Releases resource units only at the job stream level.

If **-s** is not used, resource units are released at the job level, or at the job stream level if the resource is not found at the job level.

wkstation#
Specifies the name of the workstation or workstation class on which the resource is defined. The default is the local workstation.

resourcename
Specifies the name of the resource.

Usage Notes

Units of a resource are acquired by a job or job stream at the time it is launched and are released automatically when the job or job stream completes. The **release** command can be used in a job script to release resources before job or job stream completion. Units of a resource are released in the same order that they were acquired.

Examples

In the following job stream, two units of the **dbase** resource are required by stream **sked5**:

```
schedule ux1#sked5 on tu
needs 2 dbase :
job1
jobrel follows job1
job2 follows jobrel
end
```

To release the **dbase** resource before **job2** begins, the script file for **jobrel** contains the following command:

```
`maestro`/bin/release -s dbase
```

Note that the **-s** argument can be omitted, since no resources were reserved at the job level.

In the following job stream, eight units of the **discio** resource are required by **job2**. This is defined in two blocks of 5 and 3 so that they can be released incrementally in the same order they were acquired.

```
schedule ux1#sked7 on weekdays
:
job1
job2 follows job1 needs 5 discio,3 discio
job3 follows job2
end
```

To release the **discio** resource incrementally while **job2** is executing, the script for **job2** contains the following command lines:

```
...
# Release 5 units of discio:
`maestro`/bin/release discio
...
# Release 3 units of discio:
`maestro`/bin/release discio
...
```

rextract Command

Extracts information about resources from the TWS database.

Synopsis

rextract[**-v** | **-u**] [**-o** *file*]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.
-o *file*	Specifies the output file. The default is **stdout**.

Command Output

Each resource record contains tab-delimited, variable length fields. The fields are described in the following table.

Field	Description	Max Length (bytes) *
1	workstation name	8/16
2	resource name	8
3	total resource units	4
4	resource description	72
* non-expanded databases/expanded databases		

Examples

Extract information about all resource definitions and direct the output to the file **reinfo**:

```
rextract -o reinfo
```

rmstdlist Command

Removes or displays standard list files based on the age of the file.

Synopsis

rmstdlist -v | -u

rmstdlist [**-p**] [*age*]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.
-p	Displays the names of qualifying standard list file directories. No directories or files are removed. If you do not specify **-p**, the qualifying standard list files are removed.
age	The minimum age, in days, of standard list file directories to be displayed or removed. The default is 10 days.

Examples

The following example displays the names of standard list file directories that are more than 14 days old:

```
rmstdlist -p 14
```

The following example removes all standard list files (and their directories) that are more than 7 days old:

```
rmstdlist 7
```

showexec Command

Displays the status of executing jobs. For UNIX only. This command is for standard agents. On domain managers and fault-tolerant agents, use the **conman showjobs**

command instead.

Synopsis

showexec [**-v** | **-u** | **-info**]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.
-info	Displays the name of the job file name instead of the user, date, and time.

Command Output

The output of the command is available in two formats: standard, and **info**.

Standard Format

CPU	The workstation on which the job executes.
Schedule	The name of the job stream in which the job executes.
Job	The job name.
Job#	The job number.
User	The user name of the job.
Start Date	The date the job started executing.
Start Time	The time the job started executing.
(Est) Elapse	The estimated time, in minutes, that the job will execute.

info Format

CPU	The workstation on which the job executes.
Schedule	The name of the job stream in which the job executes.

Job	The job name.
Job#	The job number.
JCL	The file name of the job.

Examples

The following example displays executing jobs in the standard format:

```
showexec
```

The following example displays executing jobs in the **info** format:

```
showexec -info
```

StartUp Command

Starts the TWS network management process, Netman. You must have **start**

access to the workstation.

Synopsis

StartUp [**-v** | **-u**]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.

Usage Notes

On Windows NT, the Netman service is started automatically when a computer is restarted. **StartUp** can be used to restart the service if it is stopped for any reason.

On UNIX, the **StartUp** command is usually installed in the **/etc/rc** file, so that Netman is started each time a computer is rebooted. **StartUp** can be used to restart Netman if it is stopped for any reason.

The remainder of the process tree can be restarted with a **conman start** command. See “start” on page 200 for more information.

Examples

The following example displays the command name and version:

```
StartUp -v
```

The following example starts the Netman process:

```
StartUp
```

version Command

Displays Tivoli Workload Scheduler version information. For UNIX only. The information is extracted from a version file.

Synopsis

version/version -V | -u | -h

version/version [**-a**] [**-f** *vfile*] [**-p** *product*] [*file* [...]]

Arguments

-V Displays the command version and exits.

-u Displays command usage information and exits.

-h Displays command help information and exits.

-a Displays information about all product files. The default is to display information only about the specified files.

-f *vfile* Specifies the name of the version file. The default is a file named **version.info** in the current working directory, or the product directory specified with **-p**.

-p *product*
Specifies the Tivoli product name whose directory is directly below the current working directory, and contains a **version.info** file. If omitted, **-f**, or its default, is used.

file Specifies the names of product files, separated by spaces, for which version information is displayed. The default is to display no file information, or, if **-a** is used, all file information.

Command Output

The output header contains the product name, version, platform, patch level, and installation date . The remaining display lists information about the file or files specified. The files are listed in the following format:

File	The name of the file.
Revision	The revision number of the file.

Patch	The patch level of the file, if any.
Size (bytes)	The size of the file in bytes.
Checksum	The checksum for the file. Checksums are calculated using the UNIX **sum** command. On AIX, **sum** is used with the **-o** argument.

Usage Notes

TWS file information is contained in the **version.info** file. This file is placed in the *TWShome*/**version** directory during installation. The **version.info** file is in a specific format and should not be altered.

You can move the **version.info** file to another directory. However, you must then include the **-f** argument to locate the file.

You can use **-p** argument if your current directory contains the directories of multiple Tivoli products. This enables you to access version information by specifying the product name.

Examples

The following example displays information about all files:

```
version/version -a -f version/version.info
```

The following example displays information about the file **customize**:

```
cd version
./version customize
```

The following example displays information about the file **customize**, when **version.info** is located in **/apps/maestro**:

```
cd version
./version -f /apps/maestro/version.info customize
```

wmaeutil Command

Used to stop the connector server for the plan, database, and engine. The makesec command will not run successfully on Windows NT until the connectors are stopped.

Synopsis

wmaeutil*instance_name* **[-stop DB | PL | EG | "*"] [-version DB | PL | EG | "*"] [-dbinfo DB | PL | "*"] [-sethome] [-gethome] [ALL -stop]**

Arguments

instance_name
: The name of the TWS instance. This refers to the instance name you entered during installation of the TWS engine, and the installation of the connector.

-stop DB | PL | EG | "*"
: This option can be used to shut down the specified connector server. The (*) asterisk can be used to shut down all three connector servers. If used, it must be enclosed by double quotes.

-version DB | PL | EG | "*"
: This option is used to obtain the version number of the connector server for the plan, database, engine and installed on the system. The (*) asterisk can be used to obtain versions for all three connector servers at once. If used, it must be enclosed by double quotes.

-dbinfo DB | PL | "*"
: This option is used to find out if the TWS database and plan to which this connector is linked is expanded or unexpanded. The (*) asterisk can be used to obtain versions for both database and plan. If used, it must be enclosed by double quotes.

-sethome
: This option is used to set MaestroHomeDir attribute of the TWS objects (Engine, Database, and Plan) in Tivoli's object database. This attribute value links connectors for the

specified object instance to the core TWS product. It takes fully qualified name of the TWS home directory as an arguments. Also the pathname string should be enclosed in quotes in order to prevent any shell interpretation.

-gethome

This option does not require any arguments and it prints the value of TWShome attribute for Engine, Database, and Plan object instances as set in the object database.

ALL -stop

This option stops the connector servers for all TWS connector instances connected to the current TWS installation, that is, it stops the connector servers for all instances whose TWShome attribute matches the home directory of the TWS current installation.

Usage Notes

Set Environment Variables

Before wmaeutil can be run successfully, you must execute following file in order to set framework environment.

On Windows NT:

```
c: \> %SystemRoot%\system32\drivers\etc\Tivoli\setup_env.cmd
```

For UNIX:

```
$ . /etc/Tivoli/setup_env.sh
```

You can update your UNIX profile to run this file, in order to avoid having to run the command manually.

Makesec Considerations

The wmaeutil command must be run before running the makesec command. The makesec command will not run successfully on Windows NT until the connectors are stopped. You should also stop the connectors when using the makesec command on UNIX.

TWS Instance Name

If you do not remember the instance name that was entered at installation time, perform the following steps:

1. Source the Tivoli environment variables:
   ```
   . /etc/Tivoli/setup_env.sh (for UNIX)
   C:\winnt\system32\drivers\etc\Tivoli\setup_env.cmd (for NT)
   ```
2. Run the **wlookup** command to get the TWS instance name:
   ```
   wlookup -ar MaestroEngine
   maestro2  1697429415.1.596#Maestro::Engine#
   ```
   where *maestro2* is the TWS instance name.

Examples

Stop the connectors for the database, plan, and engine for an instance called maestro:

```
wmaeutil maestro -stop *
```

Stop the connectors for the database for an instance called tws:

```
wmaeutil tws -stop DB
```

Stop the connector versions for the database, plan and engine for an instance called maestro2:

```
wmaeutil maestro2 -version *
```

xrxtrct Command

Extracts information about cross-references from the TWS database.

Synopsis

xrxtrct[**-v** | **-u**]

Arguments

-v	Displays the command version and exits.
-u	Displays command usage information and exits.

Command Output

The **MAESTRO_OUTPUT_STYLE** variable specifies the the output style for long object names. Set the variable to **LONG** to use full length (long) fields for object names. If the variable is not set or is set to anything other than **LONG**, and the global option **expanded version** is set to **yes**, long names are truncated to eight characters and a plus sign. For example: **A1234567+**. If the **expanded version** option is set to **no**, long names are truncated to eight characters.

The command output is written to eight files, **xdep_job,** **xdep_sched**, xfile, **xjob**, **xprompt**, xresources, **xsched**, and **xwhen**.

xdep_job File

The **xdep_job** file contains two record types. The first contains information about jobs and job streams that are dependent on a job. Each dependent job and job stream record contains the fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**03**	2
2	workstation name	8/16
3	job name	8/40
4	job stream name	8/16
5	not used	240
6	dependent job stream workstation name	8/16

Field	Description	Length (bytes) *
7	dependent job stream name	8/16
8	dependent job workstation name	8/16
9	dependent job name	8/40
10	not used	6
11	not used	6
12	not used	8
13	end-of-record (null)	1
* non-expanded databases/expanded databases		

The second record type contains information about jobs and job streams that are dependent on an internetwork dependency. Each dependent job and job stream record contains fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**08**	2
2	workstation name	8/16
3	job name	120
4	not used	128
5	dependent job stream workstation name	8/16
6	dependent job stream name	8/16
7	dependent job workstation name	8/16
8	dependent job name	8/40
9	not used	6
10	not used	6
11	not used	8
12	end-of-record (null)	1
* non-expanded databases/expanded databases		

xdep_sched File

The **xdep_sched** file contains information about job streams that are dependent on a job stream. Each dependent job stream record contains fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**02**	2
2	workstation name	8/16
3	job stream name	8/16
4	not used	248
5	dependent job stream workstation name	8/16
6	dependent job stream name	8/16
7	not used	8
8	not used	8
9	not used	6
10	not used	6
11	not used	8
12	end-of-record (null)	1
* non-expanded databases/expanded databases		

xfile File

The **xfile** file contains information about jobs and job streams that are dependent on a file. Each record contains fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**07**	2
2	workstation name	8/16
3	file name	256
4	dependent job stream workstation name	8/16
5	dependent job stream name	8/16
6	dependent job workstation name	8/16
7	dependent job name	8/40
8	not used	6
9	not used	6

Field	Description	Length (bytes) *
10	not used	8
11	end-of-record (null)	1
* non-expanded databases/expanded databases		

xjob File

The **xjob** file contains information about the job streams in which each job appears. Each job record contains fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**04**	2
2	workstation name	8/16
3	job name	8/40
4	not used	248
5	job stream workstation name	8/16
6	job stream name	8/16
7	not used	8
8	not used	8
9	not used	6
10	not used	6
11	not used	8
12	end-of-record (null)	1
* non-expanded databases/expanded databases		

xprompt File

The **xprompt** file contains information about jobs and job streams that are dependent on a prompt. Each prompt record contains fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**05**	2

Field	Description	Length (bytes) *
2	workstation name	8/16
3	prompt name or text	20
4	not used	236
5	dependent job stream workstation name	8/16
6	dependent job stream name	8/16
7	dependent job workstation name	8/16
8	dependent job name	8/40
9	not used	6
10	not used	6
11	not used	8
12	end-of-record (null)	1
* non-expanded databases/expanded databases		

xresource File

The **xresource** file contains information about jobs and job streams that are dependent on a resource. Each resource record contains fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**06**	2
2	workstation name	8/16
3	resource name	8
4	not used	248
5	dependent job stream workstation name	8/16
6	dependent job stream name	8/16
7	dependent job workstation name	8/16
8	dependent job name	8/40
9	units allocated	6
10	not used	6
11	not used	8

Field	Description	Length (bytes) *
12	end-of-record (null)	1
* non-expanded databases/expanded databases		

xsched File

The **xsched** file contains information about job streams. Each job stream record contains fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**00**	2
2	workstation name	8/16
3	job stream name	8/16
4	not used	248
5	workstation name (same as 2 above)	8/16
6	job stream name (same as 3 above)	8/16
7	not used	8
8	not used	8
9	not used	6
10	not used	6
11	not used	8
12	end-of-record (null)	1
* non-expanded databases/expanded databases		

xwhen File

The **xwhen** file contains information about when job streams will run. Each job stream record contains the following fixed length fields, with no delimiters. The fields are described in the following table.

Field	Description	Length (bytes) *
1	**01**	2
2	workstation name	8/16

Field	Description	Length (bytes) *
3	ON/EXCEPT name or date	8
4	except flag (*=EXCEPT)	1
5	not used	247
6	workstation name	8/16
7	job stream name	8/16
8	not used	8
9	not used	8
10	not used	6
11	offset num	6
12	offset unit	8
13	end-of-record (null)	1
* non-expanded databases/expanded databases		

Examples

Extract information about all cross-references:

```
xrxtrct
```

Unsupported Commands

The following unsupported utility commands provide functions on Windows NT that are similar to UNIX **ps**

and **kill**

commands. They can be used if similar Windows NT utilities are not available.

Synopsis

listproc

killproc*pid*

Usage Notes

listproc

Displays a tabular listing of processes on the system.

killproc

Kills the process with the process ID *pid*.

CAUTION:
When executed by the Administrator, killproc is capable of killing system processes.

5

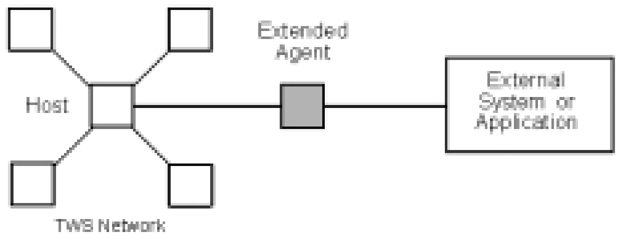
Extended Agent Reference

This chapter describes the extended agent interface and provides information for programmers who need to create custom access methods.

Tivoli Systems has several extended agents that can be integrated with Tivoli Workload Scheduler. For specific information on a particular extended agent, refer to the documentation for that extended agent.

What are Extended Agents?

Extended agents are used to extend the job scheduling functions of 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 any other workstation, except another extended agent. The access method is a Tivoli-supplied or user-supplied script or program that is executed by the host whenever the extended agent is referenced in the production plan. For example, to launch a job on an extended agent, the host executes the access method, passing it job details as command line options. The access method communicates with the external system or application to launch the job and return the status of the job.

Workstation Definition

Each extended agent must have a logical workstation definition. This logical workstation must be hosted by a TWS physical workstation, either a Master, domain manager, or FTA workstation. The extended agent workstation definition references the name of the access method and the host workstation. When jobs are launched on the extended agent workstation, the access method is called and passes the job information to the external system. For an example of defining an extended agent workstation, see the *Tivoli Workload Scheduler User's Guide*.

Access Method Interface

The interface between Workload Scheduler and an access method consists of information passed to the method on the command line, and messages returned to Workload Scheduler in **stdout**.

Method Command Line Syntax

The Workload Scheduler host runs an access method using the following command line syntax:

methodname **-t** *task* *options* **--** *taskstring*

where:

methodname
: Specifies the file name of the access method. All access methods must be stored in the directory: *WShome*/**methods**

-t *task*
: Specifies the task to be performed, where *task* is one of the following:

  LJ Launches a job.

MJ Manages a previously launched job. Use this option to resynchronize if a prior **LJ** task terminated unexpectedly.

CF Checks the availability of a file. Use this option to check file **opens** dependencies.

GS Gets the status of a job. Use this option to check job **follows** dependencies.

options

Specifies the options associated with the task. See "Task Options" for more information.

taskstring

A string of up to 255 characters associated with the task. See "Task Options."

Task Options

The task options are listed in the following table. An X means that the option is valid for the task.

Task	**Options**											**Task String**
-t	**-c**	**-n**	**-p**	**-r**	**-s**	**-d**	**-l**	**-o**	**-j**	**-q**	**-w**	
LJ	X	X	X	X	X	X	X	X	X			*ljstring*
MJ	X	X	X	X	X	X	X	X	X			*mjstring*
CF	X	X	X							X		*cfstring*
GS	X	X	X	X		X					X	*gsstring*

-c *xagent***,***host***,***master*

Specifies the Workload Scheduler names of the extended agent, the host, and the master domain manager separated by commas.

-n *nodename*

Specifies the node name of the computer associated with the extended agent, if any. This is defined in the extended agent's workstation definition **Node** field.

-p *portnumber*
: Specifies the TCP port number associated with the extended agent, if any. This is defined in the extended agent's workstation definition **TCP Address** field.

-r *currentrun***,***specificrun*
: Specifies the current run number of Workload Scheduler and the specific run number associated with the job separated by a comma. The current and specific run numbers might be different if the job was carried forward from an earlier run.

-s *jstream*
: Specifies the name of the job's job stream.

-d *scheddate***,***epoch*
: Specifies the schedule date (*yymmdd*) and the epoch equivalent, separated by a comma.

-l *user*
: Specifies the job's user name. This is defined in the job definition **Logon** field.

-o *stdlist*
: Specifies the full path name of the job's standard list file. Any output from the job must be written to this file.

-j *jobname***,***id*
: Specifies the job's name and the unique identifier assigned by Workload Scheduler, separated by a comma. The name is defined in the job definition **Job Name** field.

-q *qualifier*
: Specifies the qualifier to be used in a test command issued by the method against the file.

-w *timeout*
: Specifies the amount of time, in seconds, that Workload Scheduler waits to get a reply on an external job before sending a SIGTERM signal to the access method. The default is 300.

-- *ljstring*
: Used with the **LJ** task. The string from the **Script File** or **Command** field of the job definition.

-- *mjstring*
: Used with the **MJ** task. The information provided to Workload Scheduler by the method in a **%CJ** response to an **LJ** task. Usually, this identifies the job that was launched. For example, a UNIX method can provide the process identification (PID) of the job it launched, which is then sent by Workload Scheduler as part of an **MJ** task.

-- *cfstring*
: Used with the **CF** task. For a file **opens** dependency, the string from the **Opens Files** field of the job stream definition.

-- *gsstring*
: Used with the **GS** task. Specifies the job whose status is checked. The format is as follows:

  followsjob[**,***jobid*]

  where:

  followsjob
  : The string from the **Follows Sched/Job** list of the job stream definition.

  jobid
  : An optional job identifier received by Workload Scheduler in a **%CJ** response to a previous **GS** task.

Method Response Messages

Methods return information to Workload Scheduler in messages written to **stdout**. Each line starting with a percent sign (%) and ending with a new line is interpreted as a message. The messages have the following format:

%CJ *state* [*mjstring* | *jobid*]

%JS [*cputime*]

%UT [*errormessage*]

where:

CJ Changes the job state.

state The state to which the job is changed. All Workload Scheduler job states are valid except **hold** and **ready**. For the **GS** task, the following states are also valid:

ERROR
An error occurred.

EXTRN
Status is unknown.

mjstring
A string of up to 255 characters that Workload Scheduler will include in any **MJ** task associated with the job. See “mjstring” below “Task Options” on page 289.

jobid A string of up to 64 characters that Workload Scheduler will include in any **GS** task associated with the job. See “gsstring” below “Task Options” on page 289.

JS [*cputime***]**
Indicates successful completion of a job and provides its elapsed run time in seconds.

UT [*errormessage***]**
Indicates that the requested task is not supported by the method. Displays a string of up to 255 characters that Workload Scheduler will include in its error message.

Method Options File

You can use a method options file to specify special login information and other options. Workload Scheduler reads the file, if it exists, before executing a method. If the file is modified after Workload Scheduler is started, the changes take effect when it is stopped and restarted.

The file can contain Workload Scheduler options and any other method information. The options recognized by Workload Scheduler are as follows:

LJuser=*username*

CFuser=*username*

GSuser=*username*

GStimeout=*seconds*

where:

LJuser=*username*
: Specifies the login to use for the **LJ** and **MJ** tasks. The default is the login from the job definition.

CFuser=*username*
: Specifies the login to use for the **CF** task. The default is **root** for UNIX, and for Windows NT it is the user name of the account in which Workload Scheduler was installed.

GSuser=*username*
: Specifies the login to use for the **GS** tasks. The default is **root** for UNIX, and for Windows NT it is the user name of the account in which Workload Scheduler was installed.

GStimeout=*seconds*
: Specifies the amount of time, in seconds, Workload Scheduler waits for a response before killing the access method. The default is 300 seconds.

Note: If the extended agent's host is a Windows NT computer, these users must be defined as Workload Scheduler user objects.

The options file must have the same path name as its access method, with an **.opts** file extension. For example, the Windows NT path name of an options file for a method named **netmth** is *WShome***methods\netmth.opts**.

Method Execution

The following topics describe the interchange between Workload Scheduler and an access method.

Launch Job (LJ) Task

The **LJ** task instructs the extended agent's method to launch a job on an external system or application. Before running the method, Workload Scheduler establishes an execution environment. The **LJuser** parameter is read from the method options file to determine the user account with which to run the method. If the parameter is not present or the options file does not exist, the user account specified in the **Logon** field of the job's definition is used. In addition, the following environment variables are set:

HOME
The login user's home directory.

LOGNAME
The login user's name.

PATH For UNIX, it is set to **/bin:/usr/bin**. For Windows NT, it is set to **%SYSTEM%\SYSTEM32**.

TZ The time zone.

If the method cannot be executed, the job is placed in the **fail** state.

Once a method is running, it writes messages to its **stdout** that indicate the state of the job on the external system. The messages are summarized in the following table.

Task	Method Response	Workload Scheduler Action
LJ and **MJ**	**%CJ** *state* [*mjstring*]	Sets job state to *state*. Includes *mjstring* in any subsequent **MJ** task.
	%JS [*cputime*]	Sets job state to **succ**.
	Exit code=non-zero	Sets job state to **abend**.
	%UT [*errormessage*] and Exit code=2	Sets job state to **abend** and displays *errormessage*.

A typical sequence consists of one or more **%CJ** messages indicating changes to the job state and then a **%JS** message before the method exits to indicate that the job ended successfully. If the job is unsuccessful, the method must exit without writing the **%JS** message. A method that does not support the **LJ** task, writes a **%UT** message to **stdout** and exits with an exit code of **2**.

Manage Job (MJ) Task

The **MJ** task is used to synchronize with a previously launched job if Workload Scheduler determines that the **LJ** task terminated unexpectedly. Workload Scheduler sets up the environment in the same manner as for the **LJ** task and passes in the *mjstring*. See “Launch Job (LJ) Task” on page 294 for more information.

If the method locates the specified job, it responds with the same messages as an **LJ** task. If the method is unable to locate the job, it exits with a non-zero exit code, causing Workload Scheduler to place the job in the **abend** state.

Killing a Job

While an **LJ** or **MJ** task is running, the method must trap a SIGTERM signal (signal **15**). The signal is sent when an operator issues a **Kill** command through the Workload Scheduler’s console manager. Upon receiving the signal, the method must attempt to stop (**kill**) the job and then exit without writing a **%JS** message.

Check File (CF) Task

The **CF** task requests the extended agent’s method to check the availability of a file on the external system. Before running the method, Workload Scheduler establishes an execution environment. The **CFuser** parameter is read from the method options file to determine the user account with which to run the method. If the parameter is not present or the options file does not exist, the **root** user is used on UNIX and, on Windows NT, the user name of the account in which Workload Scheduler was installed is used. If the method cannot be executed, the file **opens** dependency is marked as failed, that is, the file status is set to **NO** and any dependent job or job stream is not allowed to execute.

Once it is running, the method executes a test command, or the equivalent, against the file using the qualifier passed to it in the **-q** command line option. If the file test is true, the method exits with an exit code of zero. If the file test is false, the method exits with a non-zero exit code. This is summarized in the following table.

Task	Method Response	Workload Scheduler Action
CF	Exit code=0	Set file state to **YES**.
	Exit code=non-zero	Set file state to **NO**.
	%UT [*errormessage*] and Exit code=2	Set file state to **NO**.

A method that does not support the **CF** task writes a **%UT** message to **stdout** and exits with an exit code of **2**.

Get Status (GS) Task

The **GS** task tells the extended agent's method to check the status of a job. This is necessary when another job is dependent on the successful completion of an external job. Before running the method, the **GSuser** parameter is read from the method options file to determine the user account with which to run the method. If the parameter is not present or the options file does not exist, the **root** user is used on UNIX, and, on Windows NT, the user name of the account in which Workload Scheduler was installed is used. If the method cannot be executed, the dependent job or job stream is not allowed to execute. If a *jobid* is available from a prior **GS** task, it is passed to the method.

The method checks the state of the specified job, and returns it in a **%CJ** message written to **stdout**. It then exits with an exit code of zero. At a rate set by the **bm check status** local option, the method is re-executed with a **GS** task until one of the following job states is returned:

abend	The job ended abnormally.
succ	The job completed successfully.
cancl	The job was cancelled.
done	The job is done, but its success or failure is not known.
fail	The job could not be run.
error	An error occurred in the method while checking job status.

extrn The job check failed or the job status could not be determined.

Note that **GStimeout** in the method options file specifies how long Workload Scheduler will wait for a response before killing the method. See “Method Options File” on page 292 for more information.

Method responses are summarized in the following table:

Task	Method Response	Workload Scheduler Action
GS	**%CJ** *state* [*jobid*]	Sets job state to *state* and includes *jobid* in any subsequent **GS** task.
	%UT [*errormessage*] and Exit code=2	Job state is unchanged.

A method that does not support the **GS** task writes a **%UT** message to **stdout** and exits with an exit code of **2**.

The cpuinfo Command

The **cpuinfo** command can be used in an access method to return information from a workstation definition. See “cpuinfo Command” on page 237 for complete command information.

Troubleshooting

The following topics are provided to help troubleshoot and debug extendend agent and access method problems.

Job Standard List Error Messages

All output messages from an access method, except those that start with a percent sign (%), are written to the job’s standard list (**stdlist**) file. For **GS** and **CF** tasks that are not associated with Workload Scheduler jobs, messages are written to Workload Scheduler’s standard list file. For information regarding a problem of any kind, check these files.

Method Not Executable

If an access method cannot be executed, the following will occur:

- For **LJ** and **MJ** tasks, the job is placed in the **fail** state.
- For the **CF** task, the file dependency is unresolved and the dependent job remains in the **hold** state.
- For the **GS** task, the job dependency is unresolved and the dependent job remains in the **hold** state.

To get more information, review the standard list files (**stdlist**) for the job and for Workload Scheduler.

Console Manager Messages

This error message is displayed if you issue a **start**, **stop**, **link** or **unlink** command for an extended agent:

```
Error executing command: Not implemented for extended
agents. [2202.58]
```

This error is not displayed if an extended agent is selected as the result of using wildcard characters.

Composer and Compiler Messages

The following error messages are generated when Composer encounters invalid syntax in a workstation definition:

```
ACCESS METHOD is syntactically invalid [1116.45]

Duplicate ACCESS keyword [1116.46]

Missing or invalid ACCESS METHOD [1116.47]
```

If an extended agent is defined with an access method but without a host, the following message is displayed:

```
"Method needs a Host CPU"
```

Jobman Messages

For extended agents, error, warning, and information messages are written to Jobman’s **stdlist** file.

A successful job launch generates the following message:

`Launched job` *`jobname`* `for` *`wkstation`*`, #J`*`jobid`* `for user` *`username`*`.`

Failure to launch a job generates the following message:

```
Error launching jobname for wkstation: errortext
```

Failure of a check file task generates the following message:

```
Error invoking methodname for wkstation: errortext
```

Failure of a manage job task generates the following message:

```
Error managing jobname for wkstation using methodname: errortext
```

When a method sends a message to Jobman that is not recognized, the following message is generated:

```
Error: message invalid for jobname, #jjobnumber for
wkstation using methodname.
"first 64 characters of the offending message"
```

6

Network Agent Reference

TWS internetwork dependencies permit jobs and job streams in a local TWS network to use jobs and job streams in a remote network as follows dependencies. This chapter describes how to use internetwork dependencies.

Overview

A network agent is a logical workstation definition that allows follows dependencies between a local TWS network and a remote TWS network. Remote follows dependencies are assigned to jobs and job streams in the same manner as local follows dependencies, except that the network agent's name is included to identify that the job or job stream is in an external TWS network. These type of dependencies are called internetwork dependencies.

If any job streams with internetwork dependencies are included in the plan, a special job stream named **EXTERNAL** is created to display the status of these dependencies. This **EXTERNAL** job stream contains place holder jobs to represent each remote follows dependency.

The workstation definition for a network agent contains the name of the network access method, **netmth**. An options file is used to specify the user under which the method runs, and the rate that the remote job or job stream dependency is checked. This options file must have the same path name as the access method, and must be called **netmth.opts**.

The access method is invoked by TWS each time it needs to check the status of a remote job or job stream. The access method queries the remote network for the status of the predecessor job or job stream. TWS continues checking until the remote job or job stream reaches the SUCC, CANCL, or ERROR state.

You can monitor the status of internetwork dependencies with the Job Scheduling Console by displaying the **EXTERNAL** job stream.

Configuring a Network Agent Workstation

Before you can specify an internetwork dependency, you must create a TWS workstation definition for the remote network. The workstation definition for a remote network is called a Network Agent. Network agent workstation definitions are defined in the standard manner as Extended Agents and require a host workstation and an access method, **netmeth**.

To define a Network Agent workstation in the TWS database:

1. Highlight the TWS Engine.
2. From the Selected Menu, click New Workstation. The Properties - Workstation in Database window is displayed.
3. In the Name field, specify a name for this workstation. The workstation name can have up to 8 alphanumeric characters for unexpanded database or 16 alphanumeric characters for expanded database. The name can including a dash and underscore, but must begin with a letter.
4. In the Node field, specify the node name or the IP address for this workstation. Fully qualified domain names are accepted. This is a required field.
5. In the TCP Port field, enter 0.
6. In the Operating System field, enter OTHER.
7. In the Domain field, specify the domain of the host workstation.
8. In the Time Zone field, specify the time zone of this workstation (optional).

9. In the Description field, specify a description of the workstation. The description can be up to 40 characters long, and must begin with a letter.
10. In the Workstation Type field, select Extended Agent from the pull down list.
11. Leave the Auto Link checkbox blank.
12. Leave the Full Status checkbox blank.
13. Leave the Ignore checkbox blank.
14. Leave the Resolve Dependencies checkbox blank.
15. Leave the Server field blank.
16. In the Access Method field, enter **netmeth**.
17. In the Host field, specify the node name of the host workstation or select one from the list provided by clicking the Workstations button.
18. Click **OK**. The Network Agent workstation is saved in the TWS database.

Network Agent Command Line Example

The following example shows a command line workstation definition for the network agent:

```
CPUNAME NETAGT
DESCRIPTION "NETWORK AGENT"
OS OTHER
NODE MAIN
TCPADDR 31111
FOR maestro
HOST MASTER
ACCESS NETMTH
END
```

Options File

An options file must be created to specify the user under which the access method runs, and how often a remote job or job stream dependency is checked. Changes to this file do not take effect until you stop and restart TWS.

This options file must have the same path name as the access method, and must be called **netmth.opts**.

```
TWShome/methods/netmth.opts
```

Syntax

```
GSuser=login_name
GStimeout=seconds
```

where:

login_name
: The login used to run the method. If the network agent's host is a Windows NT computer, this user must be defined in TWS.

seconds
: The amount of time, in seconds, TWS waits for a response before killing the access method. The default is 300 seconds. If the access method is called again it will start up automatically.

Example Options File

An example options file for method **netmth** is as follows:

```
GSuser=bunyon
GStimeout=400
```

Internetwork Dependencies

Using the Job Scheduling Console, internetwork dependencies are specified in the Job Stream Editor. Use the **Add Dependency to Internetwork** button to add an Internetwork job icon to the job stream (representing the predecessor) and then create the follows dependency from this icon to any other job using the **Add Link** button.

Note: Remote jobs and job streams are defined and run on their local network in the standard manner. Their use as internetwork dependencies has no effect on their local behavior.

When remote jobs and job streams are specified as Follows dependencies in local job streams, they are tracked by Conman in a specially created EXTERNAL job stream. Names are generated for the dependencies and they are treated as jobs in the EXTERNAL job stream.

Creating an Internetwork Dependency

To add an internetwork dependency to a job:

1. Open the job stream in the Job Stream Editor.
2. From the Graph View of the Job Stream Editor, click **Add Dependency on Internetwork** button on the toolbar and then click the cursor in the window where you want the job stream icon to be placed. The Internetwork Dependency window is displayed.
3. In the Network Agent field, enter the name of the Network Agent.
4. In the Workstation field, enter the name of the workstation that owns the predecessor job. This workstation must reside in the network that your Network Agent is connecting too.
5. In the Job Stream field, enter the name of the predecessor job stream.
6. In the Job field, enter the name of the predecessor job.
7. Click OK. An icon representing the predecessor job is displayed in the window. Note that you can drag and drop the job stream icon to position it in the window.
8. Click the Add Link button to create a follows dependency from the INET job icon to the successor job. An internetwork job can be a predecessor, but not a successor, to any job in your job stream.
9. Save the job stream.

Using the Command Line

Internetwork dependencies can be included in job streams composed with the command line **Composer**. For example, use the **follows** keyword as in the following examples:

```
jstream6 follows netagent1::remotewrkstn#skedx.@
```

```
apjob follows netagent2::ahab#qa5.jobc1
```

Internetwork Dependencies and Conman

Internetwork dependencies are displayed and manipulated in the plan in several ways.

- Adhoc scheduling
- the EXTERNAL job stream

Adhoc Scheduling and Internetwork Dependencies

Internetwork dependencies can be used as follows dependencies for jobs and job streams submitted into the plan. The dependencies are specified as they are for other follows dependencies. Refer to “Creating an Internetwork Dependency” on page 305 for more information.

EXTERNAL Job Stream

All internetwork dependencies are displayed in a job stream named EXTERNAL. The dependencies are listed as jobs regardless of whether they are defined for jobs or job streams. There is an EXTERNAL job stream for every network agent in the plan.

Unique job names are generated as follows:

`E`*`nnnmmss`*

where:

nnn	is a random number.
mm	is the current minutes.
ss	is the current seconds.

The actual name of the job or job stream is stored in the JCL portion of the job record.

External Job States

The release status of the jobs is determined by the access method and listed in the Release Status field of the EXTERNAL job stream. The status is only as current as the last time the remote network was polled. Jobs may appear to skip states that occur between polls.

All states for jobs and job streams are listed, except FENCE. In addition there are two states that are unique to EXTERNAL jobs:

ERROR
An error occurred while checking for the remote status.

EXTRN
Unknown status. An error occurred, a Rerun action was performed on the EXTERNAL job stream, or the INET job or job stream does not exist.

A

Time Zone Names

This appendix provides a list of time zone names support by TWS.

Time Zone Names and Descriptions

The following table shows the time zone names supported by TWS:

Name	Description	Relative to GMT
GMT	Greenwich Mean Time	GMT
UTC	Universal Coordinated Time	GMT
ECT	European Central Time	GMT+1:00
EET	Eastern European Time	GMT+2:00
ART	(Arabic) Egypt Standard Time	GMT+2:00
EAT	Eastern African Time	GMT+3:00
MET	Middle East Time	GMT+3:30
NET	Near East Time	GMT+4:00
PLT	Pakistan Lahore Time	GMT+5:00
IST	India Standard Time	GMT+5:30
BST	Bangladesh Standard Time	GMT+6:00
VST	Vietnam Standard Time	GMT+7:00
CTT	China Taiwan Time	GMT+8:00
JST	Japan Standard Time	GMT+9:00
ACT	Australia Central Time	GMT+9:30
AET	Australia Eastern Time	GMT+10:00

Name	Description	Relative to GMT
SST	Solomon Standard Time	GMT+11:00
NST	New Zealand Standard Time	GMT+12:00
MIT	Midway Islands Time	GMT-11:00
HST	Hawaii Standard Time	GMT-10:00
AST	Alaska Standard Time	GMT-9:00
PST	Pacific Standard Time	GMT-8:00
PNT	Phoenix Standard Time	GMT-7:00
MST	Mountain Standard Time	GMT-7:00
CST	Central Standard Time	GMT-6:00
EST	Eastern Standard Time	GMT-5:00
IET	Indiana Eastern Standard Time	GMT-5:00
PRT	Puerto Rico and US Virgin Islands Time	GMT-4:00
CNT	Canada Newfoundland Time	GMT-3:30
AGT	Argentina Standard Time	GMT-3:00
BET	Brazil Eastern Time	GMT-3:00
CAT	Central African Time	GMT-1:00

Glossary

A

Access method
An access method is an executable used by extended agents to connect and control job execution on other operating systems (for example, MVS) and applications (for example, Oracle Applications, Peoplesoft, and Baan). The access method must be specified in the workstation definition for the extended agent.

B

Batchman
Batchman is a process started at the beginning of each TWS processing day to launch jobs in accordance with the information in the Symphony file.

C

Calendar
A calendar is a defined object in the Tivoli Workload Scheduler database that contains a list of scheduling dates. Because it is a unique object defined in database, it can be assigned to multiple job streams. Assigning a calendar to a job stream causes that job stream to be executed on the days specified in the calendar. Note that a calendar can be used as an inclusionary or exclusionary run cycle.

Conman
Conman (console manager) is a legacy command-line application for managing the production environment. Conman performs the following tasks: start and stop production processes, alter and display schedules and jobs in the plan, and control workstation linking in a network.

Composer
Composer is a legacy command-line application for managing the definitions of your scheduling objects in the database.

D

Database
The database contains all the definitions you have created for scheduling objects (for example, jobs, job streams, resources, workstations, etc). In addition, the database holds other important information such as statistics of job and job stream execution, information on the user ID who created an object, and an object's last modified date. In contrast, the plan contains only those jobs and job streams (including dependent objects) that are scheduled for execution in today's production.

Deadline

The last moment in time that a job or job stream can begin execution. This corresponds to the Until time in legacy Maestro.

Dependency

A dependency is a prerequisite that must be satisfied before the execution of a job or job stream can proceed. The maximum number of dependencies permitted for a job or job stream is 40. The four types of dependencies used by Tivoli Workload Scheduler are follows dependencies, resource dependencies, file dependencies, and prompt dependencies.

Domain

A domain is a named group of TWS workstations consisting of one or more agents and a domain manager acting as the management hub. All domains have a parent domain except for the master domain.

Domain Manager

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

Duration

The time you expect the job to take to complete. In the Timeline view of jobs in the database, the duration is represented by a light blue bar at the center of the activity bar or by a light blue diamond.

E

Earliest start time

The time before which the job or job stream cannot start. The earliest start time is an estimate based on previous experiences running the job or job stream. However, the job or job stream can start after the time you specify as long as all other dependencies are satisfied. In the timeline, the start time is represented by the beginning (left edge) of the navy blue activity bar. For job instances, the start time that OPC calculates is represented by a light blue bar. See also “Actual start time” and “Planned start time”.

Exclusionary run cycle

A run cycle that specifies the days a job stream cannot be run. Exclusionary run cycles take precedent over inclusionary run cycles.

Expanded database

Expanded databases allow longer names for database objects such as jobs, job streams, workstations, domains, and users. Expanded databases are configured using the dbexpand command or as an option during installation. Do not expand your database before understanding the implications and impact of this command.

Extended agent

Extended agents are used to integrate Tivoli Workload Scheduler's job control features with other operating systems (for example, MVS) and applications (for example, Oracle Applications, Peoplesoft, and Baan). Extended agents use scripts called access methods to communicate with external systems.

External job

A job from one job stream that is a predecessor for a job in another job stream. An external job is represented by a place holder icon in the Graph view of the job stream.

F

Fault-tolerant agent

An agent workstation in the Tivoli Workload Scheduler network capable of resolving local dependencies and launching its jobs in the absence of a domain manager.

Fence

The job fence is a master control over job execution on a workstation. The job fence is a priority level that a job or job stream's priority must exceed before it can execute. For example, setting the fence to 40 prevents jobs with priorities of 40 or less from being launched.

Final Job Stream

The FINAL job stream should be the last job stream that is executed in a production day. It contains a job that runs the script file Jnextday.

Follows dependency

A dependency where a job or job stream cannot begin execution until other jobs or job streams have completed successfully.

G

Global options

The global options are defined on the master domain manager in the globalopts file, and these options apply to all workstations in the TWS network. See also "Local options".

H

Host

A Workload Scheduler workstation required by extended agents. It can be any TWS workstation except another extended agent.

I

Inclusionary Run Cycle
A run cycle that specifies the days a job stream is scheduled to run. Exclusionary run cycles take precedent over inclusionary run cycles.

Interactive jobs
A job that runs interactively on a Windows NT desktop.

Internal status
Internal status reflects the current status of jobs and job streams in the TWS engine. Internal status is unique to TWS. See also Status.

Internetwork (INET) dependencies
A dependency between jobs or job streams in separate Tivoli Workload Scheduler networks. See also “Network agent”.

Internetwork (INET) job / job stream
A job or job stream from a remote Tivoli Workload Scheduler network that is a predecessor to a job or job stream in the local network. An Internetwork job is represented by a place holder icon in the Graph view of the job stream. See also “Network agent”.

J

Jnextday job
Pre- and post-production processing can be fully automated by scheduling the **Jnextday** job to run at the end of each day. A sample jnextday job is provided as *TWShome***Jnextday**. The **Jnextday** job does the following: sets up the next day’s processing (contained in the Symphony file), prints reports, carries forward unfinished job streams, and stops and restarts TWS.

Job
A job is a unit of work that is processed at a workstation. The job definition consists of a unique job name in the TWS database along with other information necessary to run the job. When you add a job to a job stream, you can define its dependencies and its time restrictions such as the estimated start time and deadline.

Job Instance
A job scheduled for a specific run date in the plan. See also “Job”.

Job status
See “Status”.

Job Stream

A Job Stream consists of a list of jobs that execute as a unit (such as a weekly backup application), along with times, priorities and other dependencies that determine the exact order of job execution.

Job stream instance

A job stream that is scheduled for a specific run date in the plan. See also “Job stream”.

L

Limit

Job limits provide a means of allocating a specific number of job slots into which Tivoli Workload Scheduler is allowed to launch jobs. A job limit can be set for each job stream, and for each workstation. For example, setting the workstation job limit to 25 permits TWS to have no more than 25 jobs executing concurrently on the workstation.

List

A list displays job scheduling objects. You must create separate lists for each job scheduling object. For each job scheduling object, there are two types of lists: one of definitions in the database and another of instances in the plan.

Local options

The local options are defined in the localopts file. Each workstation in the Tivoli Workload Scheduler network must have a localopts file. The settings in this file apply only to that workstation. See also “Global options”.

M

Master Domain Manager

In a Tivoli Workload Scheduler network, the master domain manager maintains the files used to document the scheduling objects. It creates the plan at the start of each day, and performs all logging and reporting for the network.

N

Network agent

A type of extended agent used to create dependencies between jobs and job streams on separate Tivoli Workload Scheduler networks. See also “Internetwork (INET) dependency”.

P

Parameter
Parameters are used to substitute values into your jobs and job streams. When using a parameter in a job script, the value is substituted at run time. In this case, the parameter must be defined on the workstation where it will be used. Parameters cannot be used when scripting extended agent jobs.

Plan
The plan contains all job scheduling activity planned for a period of one day. In TWS, the plan is created every 24 hours and consists of all the jobs, job streams, and dependency objects that are scheduled to execute for that day. All job streams for which you have created run cycles are automatically scheduled and included in the plan. As the production cycle progresses, the jobs and job streams in the plan are executed according to their time restrictions and other dependencies. Any jobs or job streams that do not execute successfully are rolled over into the next day's plan.

Planned Start Time
The time that TWS estimates a job instance will start. This estimate is based on start times of previous executions.

Predecessor
A job that must complete successfully before successor jobs can begin execution.

Priority
TWS has a queuing system for jobs and job streams in the plan. You can assign a priority level for each job and job stream from **0** to **101**. A priority of **0** will not execute.

Prompt
Prompts can be used as dependencies 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 TWS database and can be used by any job or job stream.

R

Resource
Resources can represent either physical or logical resources on your system. Once defined in Tivoli Workload Scheduler database, they 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.

Run cycle

A run cycle specifies the days that a job stream is scheduled to run. In TWS there are three types of run cycles you can specify for a job stream: a Simple run cycle, a Weekly run cycle, or a Calendar run cycle (commonly called a calendar). Note that each type of run cycle can be inclusionary or exclusionary. That is, each run cycle can define the days a job stream is included in the production cycle, or the days a job stream is excluded from the production cycle. When you define multiple run cycles to a job stream, and inclusionary and exclusionary run cycles specify the same days, the exclusionary run cycles take precedent.

S

Simple Run Cycle

A simple run cycle is a specific set of user-defined days a job stream is executed. A simple run cycle is defined for a specific job stream and cannot be used by multiple job streams. For more information see Run Cycle.

Status

Status reflects the current job or job stream status within the Job Scheduling Console. The Job Scheduling Console status is common to TWS and OPC. See also Internal status.

stdlist file

A standard list file is created for each job launched by Tivoli Workload Scheduler. Standard list files contain header and trailer banners, echoed commands, errors, and warnings. These files can be used to troubleshoot problems in job execution.

Successor

A job that cannot start until all of the predecessor jobs on which it is dependent are completed successfully.

Symphony file

This file contains the scheduling information needed by the Production Control process (batchman) to execute the plan. The file is built and loaded during the pre-production phase. During the production phase, it is continually updated to indicate the current status of production processing: work completed, work in progress, work to be done. To manage production processing, the contents of the Symphony file (plan) can be displayed and altered with the Job Scheduling console.

T

Time restrictions

Time restrictions can be specified for both jobs and job streams. A time can be specified for execution to begin, or a time can be specified after which execution will not be attempted. By specifying both, you can define a window within which a job or job stream will execute. For jobs, you can also specify a repetition rate. For example, you can have Tivoli Workload Scheduler launch the same job every 30 minutes between the hours of 8:30 a.m. and 1:30 p.m.

Tivoli Management Framework (TMF)
The base software that is required to run the applications in the Tivoli product suite. This software infrastructure enables the integration of systems management applications from Tivoli Systems Inc. and the Tivoli Partners. The Tivoli Management Framework includes the following: •Object request broker (oserv) •Distributed object database •Basic administration functions •Basic application services •Basic desktop services such as the graphical user interface In a Tivoli environment, the Tivoli Management Framework is installed on every client and server. However, the TMR server is the only server that holds the full object database.

Tivoli Management Region (TMR)
In a Tivoli environment, a Tivoli server and the set of clients that it serves. An organization can have more than one TMR. A TMR addresses the physical connectivity of resources whereas a policy region addresses the logical organization of resources.

Tree view
The view on the left side of the Job Scheduling Console that displays the TWS server, groups of default lists, and groups of user created lists.

U

User
For Windows NT only, the user name specified in a job definition's "Logon" field must have a matching user definition. The definitions furnish the user passwords required by Tivoli Workload Scheduler to launch jobs.

Utility commands
A set of command-line executables for managing Tivoli Workload Scheduler.

W

Weekly Run Cycle
A run cycle that specifies the days of the week that a job stream is executed. For example, a job stream can be specified to execute every Monday, Wednesday, and Friday using a weekly run cycle. A weekly run cycle is defined for a specific job stream and cannot be used by multiple job streams. For more information see Run Cycle.

Wildcards
The wildcards for Tivoli Workload Scheduler are: ? Replaces one alpha character. % Replaces one numeric character. * Replaces zero or more alphanumeric characters.

Wildcards are generally used to refine a search for one or more objects in the database. For example, if you want to display all workstations, you can enter the asterisk (*) wildcard. To get a listing of workstations site1 through site8, you can enter site%.

Workstation
A workstation is usually an individual computer on which jobs and job streams are executed. They 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 Workload Scheduler network.

Workstation class
A workstation class is 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.

X

X-agent
See “Extended agent”.

Index

GC32-0424-00