Note

Before using this information and the product it supports, read the information in “Notices” on page 331.

Third Edition (December 2004)

This edition applies to version 8, release 2, modification level 0 of IBM Tivoli Workload Scheduler (program number 5698-WSH) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC32-1274-01.

© Copyright International Business Machines Corporation 1999, 2004. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
## Contents

### Figures

| Figures | vii |

### Tables

| Tables | ix |

### About this guide

<table>
<thead>
<tr>
<th>About this guide</th>
<th>xi</th>
</tr>
</thead>
<tbody>
<tr>
<td>What is new in this guide</td>
<td>xi</td>
</tr>
<tr>
<td>Who should read this guide</td>
<td>xi</td>
</tr>
<tr>
<td>What this guide contains</td>
<td>xi</td>
</tr>
<tr>
<td>Publications</td>
<td>xii</td>
</tr>
<tr>
<td>IBM Tivoli Workload Scheduler library</td>
<td>xii</td>
</tr>
<tr>
<td>Related publications</td>
<td>xv</td>
</tr>
<tr>
<td>Accessing publications online</td>
<td>xvi</td>
</tr>
<tr>
<td>Ordering publications</td>
<td>xvi</td>
</tr>
<tr>
<td>Providing feedback about publications</td>
<td>xvi</td>
</tr>
<tr>
<td>Accessibility</td>
<td>xvii</td>
</tr>
<tr>
<td>Tivoli technical training</td>
<td>xvii</td>
</tr>
<tr>
<td>Support information</td>
<td>xvii</td>
</tr>
<tr>
<td>Conventions used in this guide</td>
<td>xvii</td>
</tr>
<tr>
<td>Typeface conventions</td>
<td>xvii</td>
</tr>
<tr>
<td>Operating system-dependent variables and paths</td>
<td>xviii</td>
</tr>
<tr>
<td>Command syntax</td>
<td>xviii</td>
</tr>
</tbody>
</table>

### Chapter 1. Quick start

<table>
<thead>
<tr>
<th>Chapter 1. Quick start</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a plan</td>
<td>1</td>
</tr>
<tr>
<td>Managing jobs and job streams</td>
<td>5</td>
</tr>
</tbody>
</table>

### Chapter 2. The production cycle

<table>
<thead>
<tr>
<th>Chapter 2. The production cycle</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Main interfaces and processes of the product</td>
<td>7</td>
</tr>
<tr>
<td>User interfaces</td>
<td>7</td>
</tr>
<tr>
<td>Pre and post production commands</td>
<td>8</td>
</tr>
<tr>
<td>Security commands</td>
<td>8</td>
</tr>
<tr>
<td>Production processes</td>
<td>8</td>
</tr>
<tr>
<td>The process tree on UNIX platforms</td>
<td>9</td>
</tr>
<tr>
<td>The process tree on Windows platforms</td>
<td>11</td>
</tr>
<tr>
<td>Automating the production cycle</td>
<td>13</td>
</tr>
<tr>
<td>Customizing the final job stream</td>
<td>13</td>
</tr>
<tr>
<td>Adding the final job stream</td>
<td>13</td>
</tr>
<tr>
<td>Starting a production cycle</td>
<td>14</td>
</tr>
<tr>
<td>Managing the production environment</td>
<td>14</td>
</tr>
<tr>
<td>Choosing the IBM Tivoli Workload Scheduler</td>
<td>14</td>
</tr>
<tr>
<td>Start of Day</td>
<td>14</td>
</tr>
<tr>
<td>Changing the start of day</td>
<td>15</td>
</tr>
<tr>
<td>Creating a plan for future or past dates</td>
<td>15</td>
</tr>
<tr>
<td>Using reports</td>
<td>16</td>
</tr>
<tr>
<td>Launching jobs</td>
<td>16</td>
</tr>
<tr>
<td>Job environment variables</td>
<td>16</td>
</tr>
<tr>
<td>Standard configuration script - jobmanrc</td>
<td>17</td>
</tr>
<tr>
<td>Local configuration script - $HOME/jobmanrc</td>
<td>18</td>
</tr>
<tr>
<td>Production processing commands</td>
<td>19</td>
</tr>
<tr>
<td>The schedulr command</td>
<td>20</td>
</tr>
<tr>
<td>The compiler command</td>
<td>21</td>
</tr>
<tr>
<td>The stageman command</td>
<td>23</td>
</tr>
<tr>
<td>The logman command</td>
<td>26</td>
</tr>
</tbody>
</table>

### Chapter 3. Composer reference

<table>
<thead>
<tr>
<th>Chapter 3. Composer reference</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing scheduling objects</td>
<td>31</td>
</tr>
<tr>
<td>Workstation definitions</td>
<td>33</td>
</tr>
<tr>
<td>Workstation class definitions</td>
<td>39</td>
</tr>
<tr>
<td>Domain definitions</td>
<td>40</td>
</tr>
<tr>
<td>Job definitions</td>
<td>42</td>
</tr>
<tr>
<td>User definitions</td>
<td>47</td>
</tr>
<tr>
<td>Calendar definitions</td>
<td>49</td>
</tr>
<tr>
<td>Parameter definitions</td>
<td>50</td>
</tr>
<tr>
<td>Prompt definitions</td>
<td>52</td>
</tr>
<tr>
<td>Resource definitions</td>
<td>54</td>
</tr>
<tr>
<td>The composer command line program</td>
<td>55</td>
</tr>
<tr>
<td>Running composer</td>
<td>55</td>
</tr>
<tr>
<td>Command syntax</td>
<td>57</td>
</tr>
<tr>
<td>Command descriptions</td>
<td>58</td>
</tr>
<tr>
<td>add</td>
<td>60</td>
</tr>
<tr>
<td>build</td>
<td>61</td>
</tr>
<tr>
<td>continue</td>
<td>62</td>
</tr>
<tr>
<td>create</td>
<td>63</td>
</tr>
<tr>
<td>delete</td>
<td>65</td>
</tr>
<tr>
<td>display, list, print</td>
<td>67</td>
</tr>
<tr>
<td>edit</td>
<td>70</td>
</tr>
<tr>
<td>exit</td>
<td>71</td>
</tr>
<tr>
<td>modify</td>
<td>72</td>
</tr>
<tr>
<td>new</td>
<td>74</td>
</tr>
<tr>
<td>redo</td>
<td>75</td>
</tr>
<tr>
<td>replace</td>
<td>77</td>
</tr>
<tr>
<td>system command</td>
<td>78</td>
</tr>
<tr>
<td>validate</td>
<td>79</td>
</tr>
<tr>
<td>version</td>
<td>80</td>
</tr>
</tbody>
</table>

### Chapter 4. The scheduling language

<table>
<thead>
<tr>
<th>Chapter 4. The scheduling language</th>
<th>81</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax for job streams</td>
<td>81</td>
</tr>
<tr>
<td>Keywords</td>
<td>82</td>
</tr>
<tr>
<td>Dependencies</td>
<td>83</td>
</tr>
<tr>
<td>Case sensitivity</td>
<td>83</td>
</tr>
<tr>
<td>Keyword descriptions</td>
<td>83</td>
</tr>
<tr>
<td>at</td>
<td>84</td>
</tr>
<tr>
<td>carryforward</td>
<td>85</td>
</tr>
<tr>
<td>comments</td>
<td>86</td>
</tr>
<tr>
<td>confirmed</td>
<td>87</td>
</tr>
<tr>
<td>deadline</td>
<td>88</td>
</tr>
<tr>
<td>end</td>
<td>89</td>
</tr>
<tr>
<td>every</td>
<td>90</td>
</tr>
<tr>
<td>except</td>
<td>91</td>
</tr>
<tr>
<td>follows</td>
<td>93</td>
</tr>
<tr>
<td>freedays</td>
<td>94</td>
</tr>
<tr>
<td>job statement</td>
<td>96</td>
</tr>
<tr>
<td>keyjob</td>
<td>101</td>
</tr>
<tr>
<td>keysched</td>
<td>102</td>
</tr>
<tr>
<td>limit</td>
<td>103</td>
</tr>
<tr>
<td>needs</td>
<td>104</td>
</tr>
<tr>
<td>on</td>
<td>105</td>
</tr>
</tbody>
</table>

© Copyright IBM Corp. 1999, 2004
## Figures

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Process tree on UNIX</td>
<td>10</td>
</tr>
<tr>
<td>2</td>
<td>Process tree on Windows</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>Network links</td>
<td>160</td>
</tr>
<tr>
<td>4</td>
<td>Started workstations in network</td>
<td>199</td>
</tr>
<tr>
<td>5</td>
<td>Stopped workstations in network</td>
<td>202</td>
</tr>
<tr>
<td>6</td>
<td>Unlinked network workstations</td>
<td>216</td>
</tr>
<tr>
<td>7</td>
<td>Local and remote networks</td>
<td>290</td>
</tr>
<tr>
<td>8</td>
<td>Dead zones</td>
<td>319</td>
</tr>
</tbody>
</table>
## Tables

1. Command syntax ............................................. xvi
2. Job environment variables ............................... 16
3. Variables of jobmanrc ...................................... 17
4. List of reserved words ..................................... 32
5. Values for the supported operating systems .......... 34
6. Type of communication depending on the security level value .............................................. 36
7. Comparison operator ......................................... 43
8. Logical operators ............................................. 44
9. Comparison operator ......................................... 97
10. Logical operators ........................................... 98
11. Object attributes ........................................... 301
12. Actions ....................................................... 302
13. Actions (continued) ......................................... 302
About this guide

IBM® Tivoli® Workload Scheduler simplifies systems management across distributed environments by integrating systems management functions. IBM Tivoli Workload Scheduler plans, automates, and controls the processing of your enterprise's entire production workload. The IBM Tivoli Workload Scheduler Reference Guide provides detailed information about the command line interface, scheduling language, and utility commands for IBM Tivoli Workload Scheduler.

What is new in this guide


These are the main enhancements to the guide:

- A new “See Also” section has been provided in the description of several commands.
- More examples have been provided for the commands.
- A new section describing the product interfaces and processes has been included in Chapter 2, see [Main interfaces and processes of the product] on page 7
- A new chapter called “Quick start” has been added for new users.
- More information about the Security File has been added, see Chapter 10, “Setting User Security Definitions,” on page 297
- A new appendix has been added giving you information about how to contact IBM Software Support, see Appendix A, “Support information,” on page 315
- A new appendix has been added to describe the Time Zone feature, see Appendix B, “Managing time zones,” on page 319
- The information in this guide has been updated to reflect the two APARs IY58702 and IY53459.

Who should read this guide

This guide is intended for administrators and advanced users of IBM Tivoli Workload Scheduler.

What this guide contains

This manual contains the following chapters:

- Chapter 1, “Quick start,” on page 1
  Explains the basic steps a new user should follow to begin to use IBM Tivoli Workload Scheduler for the first time.
- Chapter 2, “The production cycle,” on page 7
  Describes how IBM Tivoli Workload Scheduler determines at the end of each day which schedules have to be run the next day based on the information stored in the database and on the results of the processing of the current production day.
- Chapter 3, “Composer reference,” on page 31
Describes the scheduling objects that you can define in the IBM Tivoli Workload Scheduler database and explains the usage and syntax of the commands used in the composer program to manage these objects in the database.

- Chapter 4, “The scheduling language,” on page 81
  Explains how to create a job stream, based on the scheduling objects defined in the IBM Tivoli Workload Scheduler database, using the composer command.

- Chapter 5, “Conman reference,” on page 115
  Describes the conman command line interface. This is used to monitor and manage jobs and job streams during a production day.

- Chapter 6, “Utility commands,” on page 219
  Describes the IBM Tivoli Workload Scheduler utility commands that manage the environment.

- Chapter 7, “The report commands,” on page 267
  Describes how to print different types of reports.

- Chapter 8, “The Extended Agent reference,” on page 279
  Describes how to create and use extended agents to extend IBM Tivoli Workload Scheduler job scheduling functions to other systems and applications such as local or remote UNIX Systems, Peoplesoft, SAP R/3, z/OS, OPC, Oracle CCM, and VMS.

- Chapter 9, “The Network Agent reference,” on page 289
  Describes how to create and use a network agent workstation to manage scheduling internetwork dependencies.

  Describes how to manage the Security File.

- Appendix A, “Support information,” on page 315
  Describes the different options for obtaining support for IBM products.

- Appendix B, “Managing time zones,” on page 319
  Lists the time zones supported by IBM Tivoli Workload Scheduler.

- Appendix C, “The auditing feature,” on page 325
  Describes how to enable and use the auditing option to track changes applied to the database and to the plan.

---

**Publications**

This section lists publications in the *Tivoli Workload Scheduler* library and any other related documents. It also describes how to access Tivoli publications online and how to order Tivoli publications.

**IBM Tivoli Workload Scheduler library**

IBM Tivoli Workload Scheduler comprises several separate products available on a variety of platforms, and the library is similarly divided:

**IBM Tivoli Workload Scheduling suite library**

This library contains all cross-platform and cross-product publications for IBM Tivoli Workload Scheduler.

**IBM Tivoli Workload Scheduler distributed library**

This library contains all of the publications that refer to using IBM Tivoli Workload Scheduler on platforms other than z/OS®.
IBM Tivoli Workload Scheduler for z/OS library
This library contains all publications that apply only to IBM Tivoli Workload Scheduler for z/OS.

IBM Tivoli Workload Scheduler for Applications library
This library contains all publications that apply only to IBM Tivoli Workload Scheduler for Applications.

IBM Tivoli Workload Scheduler for Virtualized Data Centers library
This library contains all publications that apply only to IBM Tivoli Workload Scheduler for Virtualized Data Centers.

IBM Tivoli Workload Scheduling suite library
The following publications are available in the IBM Tivoli Workload Scheduling suite library. This includes publications which are common to all products, platforms, and components.
- IBM Tivoli Workload Scheduler: General Information, SC32-1256
  Provides general information about all IBM Tivoli Workload Scheduler products. It gives an overview of how they can be used together to provide workload management solutions for your whole enterprise.
- IBM Tivoli Workload Scheduler: Job Scheduling Console User’s Guide, SC32-1257
  Describes how to work with IBM Tivoli Workload Scheduler, regardless of platform, using a common GUI called the Job Scheduling Console.
- IBM Tivoli Workload Scheduler: Job Scheduling Console Release Notes, SC32-1258
  Provides late-breaking information about the Job Scheduling Console.
  Provides information about enabling IBM Tivoli Workload Scheduler for Tivoli Data Warehouse.

Note: This guide is only available on the product CD. It is not possible to access it online, as you can the other books (see "Accessing publications online" on page xvi).

IBM IBM Tivoli Workload Scheduler distributed library
The following publications are available in the IBM IBM Tivoli Workload Scheduler distributed library. This includes publications which refer to using the product on all platforms except z/OS.
- IBM Tivoli Workload Scheduler: Release Notes, SC32-1277
  Provides late-breaking information about IBM Tivoli Workload Scheduler on platforms other than z/OS.
- IBM Tivoli Workload Scheduler: Planning and Installation Guide, SC32-1273
  Describes how to plan for and install IBM Tivoli Workload Scheduler on platforms other than z/OS, and how to integrate IBM Tivoli Workload Scheduler with NetView®, Tivoli Data Warehouse, and IBM IBM Tivoli Business Systems Manager.
  Describes the IBM Tivoli Workload Scheduler command line used on platforms other than z/OS, and how extended and network agents work.
- IBM Tivoli Workload Scheduler: Administration and Troubleshooting, SC32-1275
Provides information about how to administer IBM Tivoli Workload Scheduler on platforms other than z/OS, and what to do if things go wrong. It includes help on many messages generated by the main components of IBM Tivoli Workload Scheduler.

- **IBM Tivoli Workload Scheduler: Limited Fault-tolerant Agent for OS/400®, SC32-1280**
  Describes how to install, configure, and use IBM Tivoli Workload Scheduler limited fault-tolerant agents on AS/400®.

  Describes how to set up and use the IBM Tivoli Workload Scheduler Plus module.


**IBM IBM Tivoli Workload Scheduler for z/OS library**

The following documents are available in the Tivoli Workload Scheduler for z/OS library:

- **IBM Tivoli Workload Scheduler for z/OS: Getting Started, SC32-1262**
  Discusses how to define your installation data for IBM Tivoli Workload Scheduler for z/OS and how to create and modify plans.

- **IBM Tivoli Workload Scheduler for z/OS: Installation Guide**
  Describes how to install Tivoli Workload Scheduler for z/OS.

- **IBM Tivoli Workload Scheduler for z/OS: Customization and Tuning, SC32-1265**
  Describes how to customize Tivoli Workload Scheduler for z/OS.

- **IBM Tivoli Workload Scheduler for z/OS: Managing the Workload, SC32-1263**
  Explains how to plan and schedule the workload and how to control and monitor the current plan.

- **IBM Tivoli Workload Scheduler for z/OS: Quick Reference, SC32-1268**
  Provides a quick and easy consultation reference to operate Tivoli Workload Scheduler for z/OS.

- **IBM Tivoli Workload Scheduler for z/OS: Diagnosis Guide and Reference, SC32-1261**
  Provides information to help diagnose and correct possible problems when using Tivoli Workload Scheduler for z/OS.

- **IBM Tivoli Workload Scheduler for z/OS: Messages and Codes, SC32-1267**
  Explains messages and codes in Tivoli Workload Scheduler for z/OS.

- **IBM Tivoli Workload Scheduler for z/OS: Programming Interfaces, SC32-1266**
  Provides information to write application programs for Tivoli Workload Scheduler for z/OS.

- **IBM Tivoli Workload Scheduler for z/OS: Licensed Program Specifications, GI11-4208**
  Provides planning information about Tivoli Workload Scheduler for z/OS.

- **IBM Tivoli Workload Scheduler for z/OS: Memo for program 5697-WSZ, GI11-4209**
  Provides a summary of changes for the current release of the product.

- **IBM Tivoli Workload Scheduler for z/OS: Program Directory for program 5697-WSZ, GI11-4203**
  Provided with the installation tape for Tivoli Workload Scheduler for z/OS (program 5697-WSZ), describes all of the installation materials and gives installation instructions specific to the product release level or feature number.

- **IBM Tivoli Workload Scheduler for z/OS: Program Directory for program 5698-WSZ, GI11-4207**
Provided with the installation tape for Tivoli Workload Scheduler for z/OS (program 5698-WSC), describes all of the installation materials and gives installation instructions specific to the product release level or feature number.


**IBM IBM Tivoli Workload Scheduler for Applications library**

The following manuals are available in the IBM IBM Tivoli Workload Scheduler for Applications library:

- *IBM Tivoli Workload Scheduler for Applications: Release Notes, SC32-1279*
  Provides late-breaking information about the IBM Tivoli Workload Scheduler extended agents.

- *IBM Tivoli Workload Scheduler for Applications: User’s Guide, SC32-1278*
  Describes how to install, use, and troubleshoot the IBM Tivoli Workload Scheduler extended agents.


**IBM IBM Tivoli Workload Scheduler for Virtualized Data Centers library**

The following manuals are available in the IBM IBM Tivoli Workload Scheduler for Virtualized Data Centers library:

- *IBM Tivoli Workload Scheduler for Virtualized Data Centers: Release Notes, SC32-1453*
  Provides late-breaking information about IBM Tivoli Workload Scheduler for Virtualized Data Centers.

- *IBM Tivoli Workload Scheduler for Virtualized Data Centers: User’s Guide, SC32-1454*
  Describes how to extend the scheduling capabilities of IBM Tivoli Workload Scheduler to workload optimization and grid computing by enabling the control of IBM LoadLeveler® and IBM Grid Toolbox jobs.


**Related publications**

The following documents also provide useful information:

- *IBM Redbooks™: High Availability Scenarios with IBM Tivoli Workload Scheduler and IBM Tivoli Framework*
  This IBM Redbook, shows you how to design and create highly available IBM Tivoli Workload Scheduler and IBM Tivoli Management Framework (TMR server, Managed Nodes and Endpoints) environments. It presents High Availability Cluster Multiprocessing (HACMP®) for AIX® and Microsoft® Windows® Cluster Service (MSCS) case studies.
  This Redbook can be found on the Redbooks Web site at [http://www.redbooks.ibm.com/abstracts/sg246632.html]

- *IBM Redbooks: Customizing IBM Tivoli Workload Scheduler for z/OS V8.2 to Improve Performance*
  This IBM Redbook covers the techniques that can be used to improve the performance of Tivoli Workload Scheduler for z/OS (including end-to-end scheduling).
  This Redbook can be found on the Redbooks Web site at [http://www.redbooks.ibm.com/abstracts/sg246352.html]
IBM Redbooks: End-to-End Scheduling with IBM Tivoli Workload Scheduler Version 8.2

This IBM Redbook considers how best to provide end-to-end scheduling using Tivoli Workload Scheduler Version 8.2, both distributed (previously known as Maestro™) and mainframe (previously known as OPC) components.

This Redbook can be found on the Redbooks Web site at [http://www.redbooks.ibm.com/abstracts/sg246624.html](http://www.redbooks.ibm.com/abstracts/sg246624.html)

The Tivoli Software Glossary includes definitions for many of the technical terms related to Tivoli software. The Tivoli Software Glossary is available at the following Tivoli software library Web site:


Accessing publications online

The product CD contains the publications that are in the product library. The format of the publications is PDF, HTML, or both. To access the publications using a Web browser, open the infocenter.html file. The file is in the appropriate publications directory on the product CD.

IBM posts publications for this and all other Tivoli products, as they become available and whenever they are updated, to the Tivoli software information center Web site. Access the Tivoli software information center by first going to the Tivoli software library at the following Web address:


Scroll down and click the Product manuals link. In the Tivoli Technical Product Documents Alphabetical Listing window, click the appropriate Tivoli Workload Scheduler product link to access the product’s libraries at the Tivoli software information center. All publications in the IBM Tivoli Workload Scheduler suite library, distributed library and z/OS library can be found under the entry IBM Tivoli Workload Scheduler.

Note: If you print PDF documents on other than letter-sized paper, set the option in the File → Print window that allows Adobe Reader to print letter-sized pages on your local paper.

Ordering publications

You can order many Tivoli publications online at the following Web site:


You can also order by telephone by calling one of these numbers:

- In the United States: 800-879-2755
- In Canada: 800-426-4968

In other countries, see the following Web site for a list of telephone numbers:


Providing feedback about publications

If you have comments or suggestions about Tivoli products and documentation, complete the customer feedback survey at the following Web site:
Accessibility

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

For additional information, see the Accessibility Appendix in the IBM Tivoli Job Scheduling Console User’s Guide.

Tivoli technical training

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

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

Support information

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

• Searching knowledge bases: You can search across a large collection of known problems and workarounds, Technotes, and other information.
• Obtaining fixes: You can locate the latest fixes that are already available for your product.
• Contacting IBM Software Support: If you still cannot solve your problem, and you need to work with someone from IBM, you can use a variety of ways to contact IBM Software Support.

For more information about these three ways of resolving problems, see Appendix A, “Support information,” on page 315.

Conventions used in this guide

This guide uses several conventions for special terms and actions, operating system-dependent commands and paths, common syntax, and margin graphics.

Typeface conventions

This guide uses the following typeface conventions:

Bold

• Lowercase commands and mixed case commands that are otherwise difficult to distinguish from surrounding text
• Interface controls (check boxes, push buttons, radio buttons, spin buttons, fields, folders, icons, list boxes, items inside list boxes, multicolumn lists, containers, menu choices, menu names, tabs, property sheets), labels (such as Tip, and Operating system considerations)
• Keywords and parameters in text

Italic

• Words defined in text
• Emphasis of words (words as words)
• New terms in text (except in a definition list)
• Variables and values you must provide

Monospace
• Examples and code examples
• File names, programming keywords, and other elements that are difficult
to distinguish from surrounding text
• Message text and prompts addressed to the user
• Text that the user must type
• Values for arguments or command options

Operating system-dependent variables and paths
This guide uses the UNIX® convention for specifying environment variables and for directory notation.

When using the Windows command line, replace $variable with %variable% for environment variables and replace each forward slash (/) with a backslash (\) in directory paths. The names of environment variables are not always the same in Windows and UNIX. For example, %TEMP% in Windows is equivalent to $tmp in UNIX.

Note: If you are using the bash shell on a Windows system, you can use the UNIX conventions.

Command syntax
This guide uses the following syntax wherever it describes commands:

<table>
<thead>
<tr>
<th>Syntax convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brackets ([ ] )</td>
<td>The information enclosed in brackets ([ ] ) is optional. Anything not enclosed in brackets must be specified.</td>
</tr>
<tr>
<td>Braces ( { } )</td>
<td>Braces ( { } ) identify a set of mutually exclusive options, when one option is required.</td>
</tr>
<tr>
<td>Underscore ( _ )</td>
<td>An underscore (_) connects multiple words in a variable.</td>
</tr>
<tr>
<td>Vertical bar (</td>
<td>)</td>
</tr>
<tr>
<td>Bold</td>
<td>Bold text designates literal information that must be entered on the command line exactly as shown. This applies to command names and non-variable options.</td>
</tr>
<tr>
<td>Italic</td>
<td>Italic text is variable and must be replaced by whatever it represents.</td>
</tr>
</tbody>
</table>
Chapter 1. Quick start

The IBM Tivoli Workload Scheduler product suite provides you with the ability to manage your production environment and automate many operator activities. IBM Tivoli Workload Scheduler prepares jobs for execution, resolves interdependencies, and launches and tracks each job. Because jobs begin as soon as their dependencies are satisfied, idle time is minimized and throughput significantly improved. Jobs never run out of sequence, and, if a job fails, IBM Tivoli Workload Scheduler handles the recovery process with little or no operator intervention.

The section "Creating a plan" provides you with a step by step path of basic information and operations to implement IBM Tivoli Workload Scheduler in your environment using the command line interface. Alternatively, you can use the Job Scheduling Console to perform the same tasks. For more information, refer to Job Scheduling Console User’s Guide.

There are two basic concepts to perform job scheduling with IBM Tivoli Workload Scheduler:

the database

Resides on the Master Domain Manager and contains all the definitions for scheduling objects, such as jobs, job streams, resources, and workstations. It also holds statistics of job and job stream execution, as well as information on the user ID that created an object and when an object was last modified. The program used to manage objects in the database is the composer. For additional information, refer to Chapter 3, “Composer reference,” on page 31.

the plan

Contains all job scheduling activity planned for a period of one day. In IBM Tivoli Workload Scheduler, the plan is created on the Master Domain Manager every 24 hours and consists of all the jobs, job streams, and dependency objects that are scheduled to run on that day. At the end of the production day, any job streams that did not complete successfully or that are either in running or still waiting to be run, can be carried forward into the next day's plan. For additional information on how the plan is produced by the Master Domain Manager, refer to Chapter 2, “The production cycle,” on page 7. The program used to interact with the plan during the current processing day is conman. For additional information, refer to Chapter 5, “Conman reference,” on page 115.

When you have installed your scheduling environment by following the instructions in the IBM Tivoli Workload Scheduler Planning and Installation Guide, you are ready to create a plan that runs daily with the necessary information related to your environment. See “Creating a plan” for how to do this.

Creating a plan

To create a plan that runs daily, perform the following steps:

1. Log in to a shell on the Master Domain Manager

   Log in using the user ID you specified at installation time and from that shell perform all the operations described in the following steps.
2.

Start the IBM Tivoli Workload Scheduler processes
Perform this step from the user command line by running the `StartUp` command to start the scheduler network management process. For more details about the `StartUp` command, refer to “StartUp” on page 255.

3.

Add in the database the definitions to describe the topology of your scheduling environment
This step can be divided as follows:

a.

Define the workstations in your environment
A workstation is usually a physical workstation on which jobs and job streams are run, however, in the case of extended agents, the workstations are logical definitions hosted by a physical workstation. Define a workstation for each machine belonging to your scheduling environment with the exception of the Master Domain Manager which is automatically defined during the IBM Tivoli Workload Scheduler installation. For additional information, refer to “Workstation definitions” on page 33.

b.

Define domains
Use this step if you want to create a hierarchical tree of the path through the domains. IBM Tivoli Workload Scheduler works downwards through this tree when distributing the plan at the beginning of the production day, and upwards, when sending information back to the Master Domain Manager about jobs and job streams run on the target workstations. For additional information refer to “Domain definitions” on page 40.

4.

Define users allowed to run jobs on Windows workstations
On Windows workstations only, define any user allowed to run jobs using IBM Tivoli Workload Scheduler by specifying username and password. For additional information refer to “User definitions” on page 47.

5.

Define dependencies
Dependencies are conditions that must be met before a job can start. They can be defined for the job stream itself, for a job within the job stream, or for both. The maximum number of dependencies that can be set for a job or job stream is 40. For additional information refer to Chapter 4, “The scheduling language,” on page 81. These are some of the dependencies that you can set:

a.

Time dependencies
Set a time dependency to specify the earliest or latest time a job or job stream can be started or the time within which a job or job stream must complete. You can also identify
day when the job stream should not run. For more details, refer to "Keywords" on page 82.

b. **Open file dependencies**
   Use open file dependencies to specify files that must be available before a job or job stream can be started. For more details, refer to "opens" on page 108.

c. **Follow dependencies**
   Set a follow dependency to define other jobs and job streams that must complete successfully before a job or job stream. For more details, refer to "follows" on page 93.

d. **Resources**
   Use resources to specify physical or logical objects belonging to a workstation that are required to run a job on that workstation. Because resource availability is checked before running jobs on a specific workstation, resources can be used as dependencies to make job and job stream management more flexible. For example, you can define a resource called TAPES, with a value of 2, identifying the two system tape drives and then define two jobs that require that both tape drives are available as a dependency. Jobs with this dependency cannot run concurrently, because each time a job is run the TAPES resource is in use. For additional information refer to "Resource definitions" on page 54.

e. **Prompts**
   Use a prompt when you want to associate a unique name to a textual message requesting an answer from the operator. Prompts are used as dependencies to prevent jobs from starting until an affirmative response is received. For additional information refer to "Prompt definitions" on page 52.

f. **Limit**
   Set Limit to set the maximum number of jobs that can run simultaneously in a job stream. For additional information, refer to "limit" on page 103.

g. **Priority**
   Use this dependency to set the priority of a job or job stream. For additional information refer to "priority" on page 110.

6. **Define jobs**
   A job is a unit of work that runs on workstations, such as an executable file, program, or command scheduled and launched by IBM Tivoli Workload Scheduler as part of a job stream processing. It usually includes all necessary computer programs, links, files, and instructions to the operating system. It does not include scheduled...
dates and times to run, because these are defined as arguments of the job streams definition. For additional information refer to "Job definitions" on page 42.

7. Define job streams
A job stream is an ordered sequence of jobs to be run respecting set dependencies. Jobs might be grouped together in a job stream because they all run on the same day, share a common function, or share common dependencies. For additional information refer to Chapter 4, "The scheduling language," on page 81.

8. Define calendars
A calendar defines if and when a job stream has to run. Use it to include or exclude days in a daily run cycle. A calendar definition can be assigned to one or more job streams. For additional information refer to "Calendar definitions" on page 49.

9. Define parameters that represent variables inside jobs or job streams
A parameter definition is the mapping between a name and a value, this value being the value of a variable used in a job or in a job stream for a specific argument. Parameter names are replaced with the corresponding values inside the job or job stream definitions at the end of a processing day when the production plan is created for the next processing day. For additional information refer to "Parameter definitions" on page 50.

10. Automate the plan generation at the end of the current production day
Add the final job stream to the database to perform pre-production and post-production processing to ensure full automation of the plan generation at the end of each current production day. For additional information refer to "Automating the production cycle" on page 13.

11. Generate the plan
Run the \texttt{Jnextday} command to generate the plan. This command starts the processing of the scheduling information stored in the database and creates the plan for the next production day. If you automated the plan generation as described in step 10, you only need to run the \texttt{Jnextday} command the first time.

This plan is stored in a file named \texttt{Symphony} which is distributed from the Master Domain Manager down through the child domains at the start of day time (by default at 6:00 AM). Once the new production day has started, any modification you make in the objects stored in the database has no effect on the current production day processing except for the jobs and job streams or files submitted using \texttt{conman s bj} or \texttt{conman s bs}. For additional information refer to Chapter 2, "The production cycle," on page 7.

When you complete this step by step process, your scheduling environment is up and running, with batch processing of an ordered sequence of jobs and job streams being performed against resources defined on a set of workstations. By default, the first time you run the \texttt{Jnextday} command, the number of jobs that can run simultaneously on a workstation is zero, so make sure that you increase this value.
by changing the \texttt{cpu limit} to allow job execution on that workstation, see the section \texttt{”limit cpu” on page 157} for more details. This batch processing revolves around a 24 hour cycle and is generated again at the end of each production day by the Master Domain Manager.

\section*{Managing jobs and job streams}

If you now want to modify something during the day, while the production day plan is already processing, use the conman program. You can define or modify a job and add it to the plan using the command \texttt{conman sbj} only if the job will run on a workstation that has already received the plan. See \texttt{Chapter 5, “Conman reference,” on page 115} for more details about the conman program. The actions you can perform using the conman command are the following:

\textbf{Add or delete dependencies}

Use this command to add or delete dependencies to a job stream or to job within a job stream. For additional information on adding dependencies refer to \texttt{”Conman command processing” on page 134} and for additional information on deleting dependencies refer to \texttt{”deldep job” on page 148}.

\textbf{Change limit}

Perform this operation to change the limit of the number of jobs that can be run simultaneously on a workstation or the job limit for a job stream. See \texttt{”limit cpu” on page 157} for more details.

\textbf{Change priority}

Use this option to alter the priority for a job or a job stream. See \texttt{”altpri” on page 140} for more details.

\textbf{Change fence}

The fence is a value that can be set on workstations. Set the job fence on a workstation to prevent jobs whose priority is less than or equal to the value set for fence from running on that workstation. See \texttt{”fence” on page 154} for more details.

\textbf{Release a job or a job stream}

Perform this operation to release specific or all dependencies from a job or a job stream. For additional information on how to do this refer to \texttt{”release job” on page 165}.

\textbf{Cancel a job or a job stream}

Perform this operation if you want to cancel a job or a job stream either immediately or after its dependencies are resolved. For additional information on how to do this refer to \texttt{”cancel job” on page 141}.

\textbf{Stop job execution or rerun a job}

Perform this operation if you want to stop job execution by running the \texttt{kill} command. For more information refer to \texttt{”kill” on page 156}. Use rerun job to run the job again. For information refer to \texttt{”rerun” on page 170}.

\textbf{Note:} When you rerun a job it is placed in the same job stream as the original job, and it inherits the original job’s dependencies.

\textbf{Link or unlink workstations in your scheduling environment}

Perform this operation to connect or disconnect a workstation from the scheduling network. Because IBM Tivoli Workload Scheduler uses a store-and-forward technology for maintaining consistency and fault-tolerance across the network, when you unlink a workstation, all the messages are stored in the messages file and then sent all at once as soon as the link is reestablished. This command does not impact any other
processes running on the system. For additional information on linking
workstations, refer to “link” on page 159 and, on unlinking workstations
refer to “unlink” on page 215.

Stop, start or shutdown the IBM Tivoli Workload Scheduler processes
Perform this operation to act on production processes, for example, the
IBM Tivoli Workload Scheduler processes, except for the network
management process. The shutdown operation stops all IBM Tivoli
Workload Scheduler processes, including the network management
process. For additional information refer to “stop” on page 201 “start” on
page 198 and “shutdown” on page 197.

Submit a command
Perform this operation to submit a command on a workstation as a
scheduler job. If not specified in the submit options, the job name defaults
to a string that begins with the command name and the job stream defaults
to JOBS. The events reporting the result of the execution are logged in the
database. For additional information refer to “submit dcommand” on
page 204.

Submit a job
Perform this operation to submit a command to run a job defined in the
database within a job stream during the current production day on a
workstation which has received the plan. If not specified in the submit
options, the job stream defaults to JOBS. This command can only be run on
the Master Domain Manager. The events reporting the result of the
execution are logged in the database. For additional information refer to
“submit job” on page 208.

Submit a job stream
Perform this operation to submit a command to run job stream defined in
the database during the current production day on a workstation which
has received the plan. This command can only be run on the Master
Domain Manager. The events reporting the result of the execution are
logged into the database. For additional information refer to “submit
sched” on page 210.

Submit a file
Perform this action if you want to submit a script file as a job within a job
stream during the current production day on a workstation which has
received the plan. If not specified in the submit options, the job name
defaults to the file name and the job stream defaults to JOBS. The events
reporting the result of the execution are logged in the database. For
additional information refer to “submit file” on page 206.
Chapter 2. The production cycle

The IBM Tivoli Workload Scheduler plan is the to-do list that tells IBM Tivoli Workload Scheduler what jobs to run, and what dependencies must be satisfied before each job is launched. The plan covers 24 hours. The period covered by the plan is referred to as the production day. The plan starts at the time defined by the start global option, which is set by default to 6:00 a.m.

A new plan is created at the start of the production day and is placed in a production control file named Symphony. Once the plan has been created, a copy of this file is sent to all subordinate workstations. The subordinate domain managers distribute their copy to all the FTAs in their domain and to all the domain managers that are subordinate to them, and so on down the line. This enables fault tolerant agents throughout the network to continue processing even if the network connection to their domain manager is down.

At each destination FTA the IBM Tivoli Workload Scheduler processes access the copy of the Symphony file, read the instructions about which job is to be run and make calls to the operating system to launch jobs as required. The operating system runs the job, and in return informs IBM Tivoli Workload Scheduler whether the job has completed successfully or not. This information is entered into the Symphony file to indicate the status of the job. In this way the Symphony file is continuously updated with the status of the jobs: the work that needs to be done, the work in progress, and the work that has been completed. If the FTA is not full-status it only monitors what concerns its own job processing. If it is full-status, it also monitors what happens in its domain and in its domain’s child domains.

Every FTA receives the same symphony file and every FTA updates the parts of the symphony file to which it is related, while the master domain manager (and its backup) contains a copy of the symphony file having all these updates.

To turn over a new day, pre-production setup is performed for the upcoming day, and post-production logging and reporting is performed for the day just ended. This chapter describes the procedures and commands you use to perform these tasks.

---

Main interfaces and processes of the product

This section describes the main interfaces and processes used by IBM Tivoli Workload Scheduler.

**User interfaces**

A combination of graphical and command line interface programs is provided to run IBM Tivoli Workload Scheduler. The command line interface (CLI) is available for certain advanced features which are not available in the graphical user interface (Job Scheduling Console). The user interfaces are:

**Composer**

A command line program used to define scheduling objects and compose schedules.
Conman
A command line program used to monitor and control the IBM Tivoli Workload Scheduler’s production environment.

Job Scheduling Console
An interactive graphical interface used to create, modify, and delete objects in the product database and plan.

Pre and post production commands
The following commands are used to set up the IBM Tivoli Workload Scheduler’s processing day. To automate the process, the commands are normally placed in a schedule that runs at the start of each day.

  schedulr
  The command which selects schedules for execution.

  compiler
  The command which compiles the production control file.

  stageman
  The command which carries forward uncompleted schedules, and installs the production control file.

  logman
  The command which logs job statistics.

Security commands
The following commands are used to define and maintain user privileges.

  dumpsec
  The command which creates an editable copy of the product security file.

  makesec
  The command which compiles and installs the product security file.

Production processes
The following are the IBM Tivoli Workload Scheduler production processes:

  Netman
  The process which receives service requests and invokes appropriate programs.

  Mailman
  The process which routes messages to either local or remote workstations.

  Batchman
  The process which communicates directly with the plan and updates it.

  Jobman (on Windows and UNIX), Jobmon (on Windows), joblnch.exe (on Windows)
  The processes which control the actual execution of jobs. They are responsible for launching and tracking jobs using the scripts jobmanrc and .jobmanrc. For information about these scripts, see "Launching jobs" on page 16.

  Writer
  The process, generated by Netman, which establishes the link between workstations in the IBM Tivoli Workload Scheduler network.
The process tree on UNIX platforms

Figure 1 shows the IBM Tivoli Workload Scheduler process tree on UNIX platforms:
Figure 1. Process tree on UNIX
The process tree on Windows platforms

Figure 2 shows the IBM Tivoli Workload Scheduler process tree on Windows platforms:
Figure 2. Process tree on Windows
Automating the production cycle

Pre-production and post-production processing can be automated fully by adding the final job stream supplied with the product, or a user-supplied equivalent, to the database. A copy of the supplied job stream is in the Sfinal file in the TWS/home/Sfinal directory. A copy of the job script is in the TWS/home/Jnextday directory. You might find it useful to print copies to help you understand the turnover process.

The final job stream is placed in production every day, and results in running a job named Jnextday prior to the start of a new day. The job performs the following tasks:

1. Runs the schedulr command to select job streams for the new day's production plan. For more information, see "The schedulr command" on page 20.
2. Runs the compiler command to compile the production plan. For more information, see "The compiler command" on page 21.
3. Runs the reptr command to print pre-production reports. For more information, see "reptr command" on page 273.
4. Stops the scheduler.
5. Runs the stageman command to carry forward uncompleted job streams, log the old production plan, and install the new plan. For more information, see "The stageman command" on page 23.
6. Run the wmaeutil command to stop all connector instances and then restart them to refresh the symphony file. For more information, see "The wmaeutil command" on page 28.
7. Starts the scheduler for the new day.
8. Runs the reptr command to print post-production reports for the previous day. For more information, see "reptr command" on page 273.
9. Runs the logman command to log job statistics for the previous day. For more information, see "The logman command" on page 26.

Customizing the final job stream

Before using the final job stream, you can modify it to meet your needs, or you can create a different job stream to use in its place.

When creating your own job stream, model it on the one supplied with the product. If you choose to do so, consider the following:

- If you choose to change the way stageman generates log file names, remember that reptr and logman must use the same names.
- If you would like to print the pre-production reports in advance of a new day, you can split the Jnextday job into two jobs. The first job will run schedulr, compiler, and reptr. The second job will stop the scheduler, run stageman, start the scheduler, and run reptr and logman. The first job can then be scheduled to run at any time prior to the end of day, while the second job is scheduled to run just prior to the end of day.

Adding the final job stream

After you have installed a master domain manager, regardless of the method of installation you used, you must add the final job stream to the database and run Jnextday. This job stream is placed in production everyday, and results in running a job named Jnextday prior to the start of a new day. The installation creates an Sfinal file on your workstation containing the final job stream definition. You can
use this **Sfinal** file or create and customize a new one. See the *IBM Tivoli Workload Scheduler Planning and Installation Guide version 8.2* for details about customizing the final job stream.

The following is an example of configuring a master domain manager after the installation:

1. Log in as **TWS user**.
2. Run the `tws_env` script to set the IBM Tivoli Workload Scheduler environment as follows:
   - UNIX: on C shells launch `source/tws_home/tws_env.sh`
   - UNIX: on Korn shells launch `.tws_home/tws_env.sh`
   - From a Windows command line: launch `\tws_home\tws_env.cmd`
   Where `tws_home` represents the product installation directory.
3. Run the `composer` command.
4. Add the final job stream definition to the database by running the following command:
   `composer add Sfinal`

If you did not use the Sfinal file provided with the product but created a new one, use its name in place of Sfinal.

**Starting a production cycle**

To start a production cycle, follow these steps:

1. Log in as **TWS user** on the master domain manager.
2. At a command prompt, run the `Jnextday` job by entering the following command:
   `Jnextday`
   This performs pre-production processing and starts the scheduler production processes.

   **Note:** Do not update or build the Tivoli Workload Scheduler database while `Jnextday` is running, as this could damage the database.
3. When the `Jnextday` job completes, check the status of IBM Tivoli Workload Scheduler:
   `conman status`
   If the IBM Tivoli Workload Scheduler started correctly, the status is `Batchman=LIVES`.  
4. Raise the limit to allow jobs to run. The default job limit after installation is zero. This means no jobs will run, so you might want to raise the job limit now:
   `conman *limit;10`

**Managing the production environment**

This section describes how to change the start of day for IBM Tivoli Workload Scheduler and create a plan to process future or past days processing.

**Choosing the IBM Tivoli Workload Scheduler Start of Day**

You can choose from three common options for the start of the production day:

- Early morning
• Late afternoon
• Midnight

These are a few of the scheduling implications:

**Start Time and Latest Start Time**
Start times (at keyword) are always specified in relation to the scheduler production day start time. You may need to add “+ 1 day” to job streams whose jobs run across production days. Also, ensure that the latest start time (until keyword) is a time later than the start time.

**On keyword**
Production and calendar days might not be the same. If your production day starts at 06:00 a.m. (the default setting), 05:59 a.m. will be the last minute of the production day. A job stream defined to run ON MONDAY at 05:30 will be selected on Monday and will run on the calendar day Tuesday at 5:30 a.m.

**Carryforward keyword**
Placing the start of day near midnight to correspond with the calendar day will tend to produce a large number of carried forward job streams. This may increase the complexity of managing the data center.

**Deadline**
Notifications are sent when jobs and job streams have reached their deadline but have not yet started, or have not yet finished running. A deadline specifies the time within which a job or job stream must complete.

**Changing the start of day**
The start of day for IBM Tivoli Workload Scheduler is when the final job stream is run and the scheduler processes are stopped and restarted. To specify the start of day for the scheduler:

1. Modify the start option in the globalopts file. This is the start time of the processing day in 24 hour format: `hhmm` *(0000-2359)*. The default start time is 6:00 a.m.
2. Modify the start time (at keyword) of the final job stream to run one minute before the end of day.

**Creating a plan for future or past dates**
You can create a plan that runs processing normally scheduled for a future or past day of processing. This procedure effectively recreates any specified day of processing. You may need to use this procedure if you lost a day of processing due to an emergency.

1. Unlink and stop all workstations in your scheduler network. This stops all processing in the network.
2. Run the `schedulr` command with the date option to create a prodsked file:
   ```bash
   schedulr -date ddmmyyyy
   ```
   With the date option you can specify to create a plan based on a future or past day of processing.
3. Run the compiler command to create a Symnew file:
   ```bash
   compiler (-date ddmmyyyy)
   ```
   You can use the date option with the compiler to specify today’s date or the date of the day you are trying to re-create. This option may be necessary if you
have job streams that contain date sensitive input parameters. The scheddate
parameter is keyed off the date specified with the compiler command.

4. Run console manager to stop scheduler processes:
    conman stop @!@;wait

If you have defined at least one workstation as behind firewall in an IBM Tivoli
Workload Scheduler Version 8.2 network, you must enter the following
command:
    conman stop ;progressive

5. Run stageman to create the new Symphony file:
    stageman

6. Run console manager to start scheduler processes:
    conman start

Using reports

Use reports to help you manage the production cycle. For more information, see

Launching jobs

Jobs are launched under the direction of the production control process Batchman.
Batchman resolves all job dependencies to ensure the correct order of execution,
and then issues a job launch message to the Jobman process.

Jobman spawns a job monitor process that begins by setting a group of
environment variables, and then it runs the standard configuration script
(\$TWS\_HOME/\$JOBMANRC). If the user is allowed to use a local configuration script, and
the script \$HOME/\$JOBMANRC exists, the local configuration script is also run. The
job is then run either by the standard configuration script, or by the local one.

Each of the processes launched by Jobman, including the configuration scripts
and the jobs, retain the user name recorded with the stream logon of the job. In case of
submitted jobs, they retain the submitting user’s name. To have the jobs run with
the user’s environment, be sure to add the user’s .profile environment to the local
configuration script.

Job environment variables

The variables listed in the table below are set and exported by Jobman.

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>HOME</td>
<td>The login user’s name directory</td>
</tr>
<tr>
<td>LOGNAME</td>
<td>The login user’s name</td>
</tr>
<tr>
<td>PATH</td>
<td>For Windows: %SYSTEMROOT\SYSTEM32.</td>
</tr>
<tr>
<td></td>
<td>For UNIX: /bin:/usr/bin</td>
</tr>
<tr>
<td>TZ</td>
<td>The time zone</td>
</tr>
<tr>
<td>UNISON_SHELL</td>
<td>The user’s login shell</td>
</tr>
<tr>
<td>UNISON_CPU</td>
<td>The name of this CPU</td>
</tr>
<tr>
<td>UNISON_HOST</td>
<td>The name of the master/host CPU</td>
</tr>
<tr>
<td>UNISON_JOB</td>
<td>The fully qualified job name: cpu#sched.job</td>
</tr>
</tbody>
</table>
Table 2. Job environment variables (continued)

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNISON_JOBNUM</td>
<td>The job number (ppid)</td>
</tr>
<tr>
<td>UNISON_MASTER</td>
<td>The name of the master CPU</td>
</tr>
<tr>
<td>UNISON_RUN</td>
<td>Tivoli Workload Scheduler’s current production run number</td>
</tr>
<tr>
<td>UNISON_SCHED</td>
<td>The schedule name</td>
</tr>
<tr>
<td>UNISON_SCHED_DATE</td>
<td>Tivoli Workload Scheduler’s production date (yymmd)</td>
</tr>
<tr>
<td>UNISON_SCHED_EPOCH</td>
<td>Tivoli Workload Scheduler’s production date, expressed in epoch form</td>
</tr>
<tr>
<td>UNISON_SYM</td>
<td>The Symphony record number</td>
</tr>
<tr>
<td>UNISON_EXEC_PATH</td>
<td>The jobmanrc fully qualified path</td>
</tr>
<tr>
<td>TIVOLI_JOB_DATE</td>
<td>The scheduled date for a job</td>
</tr>
</tbody>
</table>

Standard configuration script - jobmanrc

A standard configuration script template named `tws_home/config/jobmanrc` is supplied with IBM Tivoli Workload Scheduler. It is installed automatically as `tws_home/jobmanrc`. This script can be used by the system administrator to establish a desired environment before each job is run. If you wish to alter the script, make your modifications in the working copy (`tws_home/jobmanrc`), leaving the template file intact. The file contains variables which can be configured, and comments to help you understand the methodology. Table 3 describes the jobmanrc variables.

Table 3. Variables of jobmanrc

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNISON_JCL</td>
<td>The path name of the job’s script file.</td>
</tr>
<tr>
<td>UNISON_STDLIST</td>
<td>The path name of the job’s standard list file.</td>
</tr>
<tr>
<td>UNISON_EXIT</td>
<td>(Settable) If set to yes, the job is terminated immediately if any command returns a non zero exit code. If set to no, the job continues to run if a command returns a non zero exit code. Any other setting is interpreted as no.</td>
</tr>
<tr>
<td>LOCAL_RC_OK</td>
<td>(Settable) If set to yes, the user’s local configuration script is run (if it exists), passing $UNISON_JCL as the first argument. The user may be allowed or denied this option. See “Local configuration script - $HOME/.jobmanrc” on page 18 for more information. If set to no, the presence of a local configuration script is ignored, and $UNISON_JCL is run. Any other setting is interpreted as no.</td>
</tr>
</tbody>
</table>
Table 3. Variables of jobmanrc (continued)

<table>
<thead>
<tr>
<th>Variable Name</th>
<th>Value</th>
</tr>
</thead>
</table>
| MAIL_ON_ABEND    | (Settable) If set to yes, a message is mailed to the login user’s mailbox if the job terminates with a non-zero exit code. This can also be set to one or more user names, separated by spaces, and a message is mailed to each user. For example, “root mis sam mar”. If set to no, no messages are mailed if the job abends. Abend messages have the following format:  

```
cpu#sched.job
jcl-file failed with exit-code
Please review standard-list-filename
```

| SHELL_TYPE       | (Configurable) If set to standard, the first line of the jcl file is read to determine which shell to use to run the job. If the first line does not start with #!, then /bin/sh is used to run the local configuration script or $UNISON_JCL. Commands are echoed to the job’s standard list file. If set to user, the local configuration script or $UNISON_JCL is run by the user’s login shell ($UNISON_SHELL). Commands are echoed to the job’s standard list file. If set to script, the local configuration script or $UNISON_JCL is run directly, and commands are not echoed unless the local configuration script or $UNISON_JCL contains a set -x command. Any other setting is interpreted as standard. |
| USE_EXEC         | (Settable) If set to yes, the job, or the user’s local configuration script is run using the exec command, thus eliminating an extra process. This option is overridden if MAIL_ON_ABEND is also set to yes. Any other setting is interpreted as no, in which case the job or local configuration script is run by another shell process. |

**Local configuration script - $HOME/.jobmanrc**

Using the local configuration script, you can establish a desired environment for the running of your own jobs. The script only runs under the following conditions:

1. The standard configuration script, jobmanrc, must be installed, and the environment variable LOCAL_RC_OK must be set to yes (see Table 3).
2. If the file $tws_home/localrc.allow exists, the user’s name must appear in the file. If the allow file does not exist, the user’s name must not appear in the file, $tws_home/localrc.den. If neither of these files exists, the user is permitted to use a local configuration script.
3. The local configuration script must be installed in the user’s home directory ($HOME/.jobmanrc), and it must have execute permission.
If you intend to use a local configuration script, it must, at a minimum, run the job’s script file ($UNISON_JCL). The Tivoli-supplied standard configuration script, jobmanrc, runs your local configuration script as follows:

```bash
$EXECIT $USE_SHELL $HOME/.jobmanrc "$UNISON_JCL" $IS_COMMAND
```

The value of USE_SHELL is set to the value of the jobmanrc SHELL_TYPE variable (see Table 3 on page 17). IS_COMMAND is set to yes if the job was scheduled or submitted using the docommand construct. EXECIT is set to exec if the variable USE_EXEC is set to yes (see Table 3 on page 17), otherwise it is null.

The following example shows how to run a job’s script file, or command, in your local configuration script:

```bash
#!/bin/ksh
PATH=tws_home:tws_home/bin:$PATH
export PATH
/bin/sh -c "$UNISON_JCL"
```

The following is an example of a .jobmanrc that does processing based on the exit code of the user’s job:

```bash
#!/bin/sh
PATH=tws_home:tws_home/bin:$PATH
export PATH
/bin/sh -c "$UNISON_JCL"

# or use eval "$UNISON_JCL" and the quotes are required
RETVAL=$?
if [ $RETVAL -eq 1 ]
then
  echo "Exit code 1 - Non Fatal Error"
  exit 0
elsif [ $RETVAL -gt 1 -a $RETVAL -lt 100 ]
then
  conman "tellop This is a database error - page the dba"
elseif [ $RETVAL -ge 100 ]
then
  conman "tellop Job aborted. Please page the admin"
fi
```

Production processing commands

The pre- and post-production processing commands run by the jnextday job are described in the following sections.
The schedulr command

The schedulr command selects job streams for a specific date from the database file mastsked, and copies them to a new production schedule file named prodsked.

You must have build access to the scheduler database files.

Synopsis

schedulr -v | -u

 schedulr [-date date | -autodate] [-schedules {in-file | -}] [-prodsked {out-file | -}]

Arguments

-u Display command usage information and exit.
-v Display the command version and exit.
-date Select job streams for a specific date. The date is entered as mm/dd/[yy]yy.
-autodate Select job streams for the current system date.
-schedules In addition to those selected by -date or -autodate, if any, select the job streams named in in-file. The names must appear in the file as [workstation#]jobstream, with one name per line. If a dash is entered instead of a file name, schedulr prompts for job stream names at stdin.
-prodsked Direct schedulr output to out-file. If a dash is entered instead of a file name, the output is directed to stdout. If the argument is omitted, the output is written to a file named prodsked.

Description

If -autodate and -date are omitted, schedulr prompts for a date. If you respond to the prompt by pressing Return, job streams are selected only from the in-file.

Examples

Select job streams for today’s date, plus the job streams named in the file myskeds:
schedulr -autodate -schedules myskeds

Select job streams for February 15, 2001, do not prompt for extra job stream names, and write the output to the file myprodsked:
schedulr -date 2/15/01 -prodsked myprodsked

Select job streams for February 15, 2001, and prompt for extra job streams:
schedulr -date 2/15/2001 -schedules -

Prompt for the production date, and extra job streams (note that “schedule” is the same as “job stream”):
schedulr
Enter schedule date: 4/14/01
Enter a list of extra schedules
Schedule name: site1#sked2
Schedule name: <Return>
<list of job streams selected>
End of Program
The compiler command

The compiler command compiles the production schedule file, and creates an interim production plan file.

Synopsis

```
compiler [-v|-u]
compiler [-date date] [-input in-file] [-output out-file]
```

Arguments

- **-u** Display command usage information and exit.
- **-v** Display the command version and exit.
- **-date** The production date to be recorded in the interim production plan file. The date is entered as mm/dd/yyyy.
- **-input** The name of the file containing the production schedule. If this option is omitted, the default name is prodsked.
- **-output** Direct compiler output to out-file. If the argument is omitted, the output is written to a file named Symnew.

Description

If you omit the -date argument, Symnew is given the same date as that recorded in the production schedule file created by schedulr. If there is no date in production schedule file, the current system date is used. The date in Symnew is the date that the scheduler will begin executing the production plan. The ability to enter a different date can be used to set up processing for past or future dates.

Missing object messages: The following messages are produced by compiler to indicate missing scheduling objects. The messages are normally found in the standard list file for the Jnextday job.

```
job. 5 ... Undefined parameter in "schedule"; not replaced.

A parameter called for in a job stream does not exist in the scheduler database. No substitution occurs and the parameter string itself is used.

AWSBIC102W ... Job name is not found in database. Added a dummy job in FAIL state.

A job named in a job stream does not exist in the scheduler database. A dummy job of the same name is placed in the production schedule with a priority of zero and a state of FAIL.

AWSBIC103W ... Prompt name not found. Added prompt name in Symphony.

A prompt named in a job stream does not exist in the scheduler database. A dummy prompt containing the following text is used instead: Prompt name was not found in database. This is dummy text. Do you want to continue (Y/N).

AWSBIC104W ...Resource name for cpu name not found in database. Added resource name with 0 units.
```
A resource named in a job stream does not exist in the scheduler database. A dummy resource with zero available units is used instead:

AWSBIC106W ... Cpu name does not exist in cpu database. Ignoring schedule name.

A job stream is defined to run on a cpu that does not exist. The job stream is ignored and not placed in the production schedule.

**Examples**

Compile `prodsked` into `Symnew`:
```
compiler
```

Compile `prodsked` into `Symnew`, and enter a production date of May 15, 2002:
```
compiler -date 5/15/02
```

Compile the file `mysked` into a file named `mysym`:
```
compiler -input mysked -output mysym
The stageman command

The `stageman` command carries forward uncompleted job streams, logs the old production plan, and installs the new production plan. The new production plan file is named `Symphony`. A copy of `Symphony`, named `Sinfonia`, is also created. `Sinfonia` is sent to domain managers and agents as part of the initialization process for the new day.

You must have `build` access to the `Symphony` file.

Synopsis

```
stageman -v
```

```
stageman [-carryforward {yes | no | all}] [-log log-file| -nolog] [symnew]
```

Arguments

- `-v` Display the command version and exit.
- `-carryforward` Define the type of carry forward as follows:
  - `no` Do not carry forward any job streams.
  - `yes` Carry forward only those uncompleted job streams that are Carry Forward enabled.
  - `all` Ignore Carry Forward enabling in job streams, and carry forward all uncompleted job streams.
- `-log` Log the old production plan, and give the log file this name. See "Log file names" on page 24 for more information.
- `-nolog` Do not log the old production plan.
- `symnew` The name of the interim production plan file created by `compiler`. If omitted, the file `Symnew` is used.

Description

If you omit `-carryforward`, the default for carry forward is determined by the `carryforward` global option.

In UNIX only, `stageman` also determines which executable files can be deleted for jobs submitted with the `at` and `batch` commands. These are jobs that were not carried forward. The files are actually deleted when the scheduler starts up for the new day.

If scheduler processes are still running and accessing the `Symphony` file, `stageman` displays the message:

```
Unable to get exclusive access to Symphony.
Shutdown batchman and mailman.
```

To continue, stop the scheduler and rerun `stageman`. If `stageman` aborts for any reason, you must rerun both `compiler` and `stageman`.

Users accessing the plan through the CLI during the time `Symphony` is being switched are sent the message:

```
Current Symphony file is old. Switching to new Symphony.
Schedule mm/dd/yy (nnnn) on cpu, Symphony switched.
```
Some user commands run during the switch may not run properly because the target jobs or job streams were not carried forward.

**Log file names:** Production plan log files are stored in the `TWShome/schedlog` directory. The default naming convention used by `stageman`, when the `-log` and `-nolog` arguments are omitted, is as follows:

```
TWShome/schedlog/yyyyymmddhh$t
```

where `yyyyymmddhh$t` is the year, month, day, hour, and minute the log file was created.

The above naming convention is coded in the `Jnextday` script supplied by Tivoli. If you wish, you can change the naming convention when you automate the production cycle. For more information see “Automating the production cycle” on page 13.

**Note:** Be sure to monitor the disc space in the `schedlog` directory and remove older log files on a regular basis.

**Job streams carried forward:** The carry forward option remains enabled on job streams that are carried forward, so they may be carried forward again. If an unsuccessful job stream is carried forward and it continues to terminate in a state other than SUCC, it may be carried forward indefinitely unless its Until time expires or it is cancelled.

Carried forward job streams maintain their original production date internally. Any job within these job streams that utilizes `datecalc` will use this production date when using the “scheddate” keyword. For more information see “datecalc” on page 227.

For carry forward to work properly in a network, the master domain manager’s production plan file, `Symphony`, must be updated with the latest job stream status from its agents and subordinate domain managers. This can be accomplished by entering the following at a command prompt on the master domain manager prior to executing `stageman`:

```
conman "link 0"
```

**Job stream names:** Job streams that are carried forward are renamed as follows:

```
CFyyjjjnnxxxxxxxxx
```

where `yy` are the last two digits of the year, `jjj` is the Julian date, `nn` is a sequence number (00-99, AA-ZZ), and `xxxxxxxxx` is a random alpha string.

**Carry forward prompts:** To retain continuity when carrying job streams forward, `stageman` creates special prompts in the new production plan to account for disconnected Follows dependencies. These prompts are issued after the new processing day begins, when the scheduler checks to see if the job or job stream is ready to launch, and are replied to as standard prompts. The following is an example of a Carry Forward prompt:

```
INACT 12 (SYS1#CF9123AA) follows SYS1#SKED3 satisfied?
```

This prompt indicates that a job stream from the previous day was carried forward as CF9123AA, and that it follows a job stream named sked3 which was not carried forward. The state of the prompt— INACT in this case— defines the state of the corresponding Follows dependency. The possible states are:
INACT
The prompt has not been issued and the dependency is not satisfied.

ASKED
The prompt has been issued, and is awaiting a reply. The dependency is not satisfied.

NO
Either a “no” reply was received, or it was determined before Carry Forward occurred that the followed job stream (sked3) had not completed successfully. The dependency is not satisfied.

YES
Either a “yes” reply was received, or it was determined before Carry Forward occurred that the followed job stream (sked3) had completed successfully. The dependency is satisfied.

Note: A prompt is generated if the schedule is carried forward and it has a follows dependency on another schedule which is not carried forward. In the case of Internetwork dependencies, even when both schedules are carried forward, a prompt is generated. This is because in the case of Internetwork dependencies there is no available mechanism to keep them connected. No way is provided for the successor to know if the schedule on the other side has been carried forward or not, and the dependency link is broken in any case. For more details, refer to Chapter 9, “The Network Agent reference,” on page 289.

Examples
Carry forward all uncompleted job streams (regardless of the status of the Carry Forward option), log the old Symphony file, and create the new Symphony file:
DATE='datecalc today pic YYYMMDHHMT'
stageman -carryforward all -log schedlog/M$DATE

Carry forward uncompleted job streams as defined by the carryforward Global Option, do not log the old Symphony file, and create a new production control file named mysym:
stageman -nolog mysym
The logman command

The logman command log job statistics from a production plan log file.

Synopsis

logman -v [-u]

logman [-smooth percent] [-minmax [elapsed | cpu]] log-file

Arguments

-u        Display command usage information and exit.
-v        Display the command version and exit.
-smooth   Use a weighting factor that favors the most recent job run when calculating the
          normal (average) run time for a job. This is expressed as a percentage.
          For example, -smooth 40 will apply a weighting factor of 40% to the most recent
          job run, and 60% to the existing average. The default is zero.
-minmax   Define how the minimum and maximum job run times are logged and reported.

elapsed   Base the minimum and maximum run times on elapsed time.

cpu       Base the minimum and maximum run times on CPU time.

log-file  The name of the production plan file or log file from which job statistics
          are extracted.

Description

Jobs that have already been logged, cannot be logged again. Attempting to do so generates a 0 jobs logged error message.

Elapsed time compared to CPU time: Elapsed time, expressed in minutes, is greatly affected by system activity. It includes both the amount of time a job made use of the CPU and the intervals the job had to wait for other processes to release the CPU. In periods of high system activity, for example, a job may have a long elapsed time, and yet use no more CPU time than in periods of low system activity. On the other hand, CPU time, expressed in seconds, is a measure of the actual time a job made use of the CPU, and does not include the intervals when the job was waiting.

If you run logman with the -minmax elapsed argument, the maximum and minimum run times and dates are based solely on a job’s elapsed time. The values are updated only if the latest job run has an elapsed time greater than the existing maximum, or less than the existing minimum. The CPU times, in this case, will not necessarily indicate their maximum and minimum extremes.

If you run logman with the -minmax CPU argument, the maximum and minimum run times and dates are based solely on a job’s CPU time. The values are updated only if the latest job run has a CPU time greater than the existing maximum, or less than the existing minimum. The elapsed times, in this case, will not necessarily indicate their maximum and minimum extremes.

If you run logman without the -minmax argument, the elapsed time and CPU time values are updated independently to indicate their maximum and minimum
extremes, but the run dates correspond only to the elapsed time values. No record is kept, in this case, of the run dates for maximum and minimum CPU times.

Examples
Log job statistics from the log file M199903170935:
logman schedlog/M199903170935

Log job statistics from the log file M$DATE based on elapsed time, giving the most recent job runs a weight of 40% when calculating normal (average) run times:
logman -smooth 40 -minmax elapsed schedlog/M$DATE

The $DATE variable contains the date and time stamp used by stageman to create the log file name. See "The stageman command" on page 23 for more information.
The wmaeutil command

The **wmaeutil** command is used to stop the connector server for the plan, database, and engine. The **makesec** command will not run successfully on Windows platforms until the connectors are stopped.

**Note:** If you re-create a plan file manually (not using **Jnextday**), you must stop the connectors by running the **wmaeutil** command and then refresh the views in the Job Scheduling Console to view the new production day. Otherwise, the views in the Job Scheduling Console will remain on the prior production day.

**Synopsis**

UNIX:

```
wmameutil.sh instance_name [-stop DB | PL | EG | "*" ] [-version DB | PL | EG | "*" ] [-dbinfo DB | PL] [-sethome] [-gethome] [ALL -stop]
```

Windows:

```
wmameutil.cmd instance_name [-stop DB | PL | EG | "*" ] [-version DB | PL | EG | "*" ] [-dbinfo DB | PL] [-sethome] [-gethome] [ALL -stop]
```

**Arguments**

*instance_name*

The name of the scheduler instance. This refers to the instance name you entered during installation of the scheduler engine, and the installation of the connector.

*stop DB | PL | EG | *

This option can be used to shut down specified connector server. The (*) asterisk can be used to shut down all three connector server.

*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 server at once.

*dbinfo DB | PL*

This option is used to find out if the scheduler database and plan to which this connector is linked is expanded or unexpanded. With IBM Tivoli Workload Scheduler, Version 8.2, databases and plans are always expanded, but this option exists for backward compatibility.

*sethome*

This option is used to set **TWShome** attribute of the scheduler objects (Engine, Database, and Plan) in Tivoli’s object database. This attribute value links connectors for the specified object instance to the core scheduler product. It takes the fully qualified name of the scheduler home directory as an argument. Also the pathname string should be enclosed in quotes to prevent any shell interpretation.

*gethome*

This option does not require any arguments and it prints the value of **TWShome** attribute for the Engine, Database, and Plan object instances as set in the object database.

**ALL -stop**

This option stops the connector servers for all scheduler connector
instances connected to the current scheduler installation, that is, it stops the
cconnector servers for all instances whose TWSHome attribute matches the
home directory of the scheduler current installation.

Usage Notes

Set environment variables: Before wmaeutil can be run successfully, you must
run the following file to set the framework environment.

For Windows:
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, to avoid having to run the
command manually.

Makesec considerations: The wmaeutil.sh command must be run before running
the makesec command. The makesec command will not run successfully on
Windows platforms until the connectors are stopped. You should also stop the
connectors when using the makesec command on UNIX.

IBM Tivoli Workload Scheduler instance name: If you are not sure of the
instance name that was entered at installation, 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 Windows)
2. Run the wlookup command to get the scheduler instance name:
   wlookup -ar MaestroEngine
   maestro2 1697429415.1.596#Maestro::Engine#
   where maestro2 is the scheduler instance name.

Examples
Stop the connectors for the database, plan, and engine for an instance called
maestro:
  wmaeutil.sh maestro -stop ALL

Stop the connectors for the database for an instance called tws:
  wmaeutil.sh tws -stop ALL DB

Stop the connector versions for the database, plan and engine for an instance called
maestro2:
  wmaeutil.sh maestro2 -version *
Chapter 3. Composer reference

The composer command line program is used to manage scheduling objects in the IBM Tivoli Workload Scheduler database. Scheduling objects are workstations, workstation classes, domains, jobs, job streams, resources, prompts, calendars, and parameters. This chapter describes the following:

- Managing scheduling objects
- Composer command line program
- Command descriptions

Managing scheduling objects

Scheduling objects are managed with the composer command line program and are stored in the scheduler database. The composer command line program uses edit files to update the scheduling database, for additional information on the format of the edit file used for a specific object please refer to that specific object definition described at the end of this part. For example, to add a new object you have to enter its definition in an edit file, then, using the composer command line program you can create or modify that object specifying the edit file that contains the definition, that composer command line program checks for correct syntax inside that edit file and, if correct, adds the definition the database.

According to the kind of objects that you want to manage using the composer command line program you manage that specific object individually in the database or you access the list of objects of the same kind defined in the database and you modify the entry in the list that represents that specific object.

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, you have to use the composer command line program with the create option to have the definition of that job copied from the database into an edit file, you have to modify that edit file and then using the composer command line program with the modify option to modify the job definition, the composer command line program checks that the syntax of the edit file is correct and then copies back the job definition 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, you have to use composer command line program with the modify resources option, the entire resources list is copied from the database into an edit file that you can edit directly from within the composer interface and then save, as you save the edit file the composer command line program checks if the syntax used inside the file is correct and then copies the new list back into the database, replacing the existing one.

Composer does not issue specific warnings if scheduling language keywords are used as names of scheduling objects. However, the use of keywords can result in errors. Avoid using the following keywords when defining scheduling objects:
<table>
<thead>
<tr>
<th>Table 4. List of reserved words</th>
</tr>
</thead>
<tbody>
<tr>
<td>abendprompt</td>
</tr>
<tr>
<td>after</td>
</tr>
<tr>
<td>at</td>
</tr>
<tr>
<td>autodocoff</td>
</tr>
<tr>
<td>autodocon</td>
</tr>
<tr>
<td>behindfirewall</td>
</tr>
<tr>
<td>carryforward</td>
</tr>
<tr>
<td>confirmed</td>
</tr>
<tr>
<td>cpuname</td>
</tr>
<tr>
<td>dateval</td>
</tr>
<tr>
<td>day(s)</td>
</tr>
<tr>
<td>day_of_week</td>
</tr>
<tr>
<td>deadline</td>
</tr>
<tr>
<td>docommand</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td></td>
</tr>
</tbody>
</table>
Workstation definitions

A workstation is a scheduling object that runs jobs. A workstation is usually an individual computer, on which jobs and job streams are run. A workstation definition is required for every computer that runs jobs in the IBM Tivoli Workload Scheduler network. Primarily workstation definitions refer to physical workstations. However, in the case of extended agents, the workstations are logical definitions that must be hosted by a physical workstation. You can include multiple workstation definitions in the same text file, along with workstation class definitions and domain definitions. Each workstation definition has the following format and arguments:

Synopsis

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

[cpuname ...]
[cpuclass ...]
[domain ...]
```

Arguments

```plaintext
# 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. It can contain up to 16 characters. See Table 4 on page 32 for a list of words to avoid using when specifying the cpuname.

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 values include UNIX, WNT, and OTHER where,
Table 5. Values for the supported operating systems

<table>
<thead>
<tr>
<th>Use these values...</th>
<th>For these operating systems...</th>
</tr>
</thead>
<tbody>
<tr>
<td>UNIX</td>
<td>HP-UX, Sun Solaris, AIX, Linux™ for Intel™, Linux for S/390® and z/Series, Linux for iSeries and pSeries, Irix, OSF, Dynix, Compaq</td>
</tr>
<tr>
<td>OTHER</td>
<td>Used in connection with extended agents.</td>
</tr>
</tbody>
</table>

**Note:** Refer to the IBM Tivoli Workload Scheduler Release Notes® for an up-to-date list of supported platforms.

**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 the scheduler uses for communications on the workstation. The default is **31111**.

**secureaddr**
Defines the port used to listen for incoming SSL connections. This value must match the one defined in the **nm SSL port** local option of the workstation. It must be different from the **nm port** local option that defines the port used for normal communications. If **securitylevel** is specified but this attribute is missing, 31113 is used as the default value. See the IBM Tivoli Workload Scheduler Planning and Installation Guide for reference on SSL authentication.

**timezone tz tzname**
Specifies the time zone of the workstation. See Appendix B, “Managing time zones,” on page 319 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 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 TWS_home/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 the 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 run a Conman link command on the agent’s domain manager or the master domain manager.

behindfirewall
If this attribute is set to ON, this means that there is a firewall between the workstation and the master domain manager. Only a direct connection between the workstation and its domain manager is allowed. For all the workstations with behindfirewall set to ON, the start wkstation, stop wkstation, and showjobs commands are sent following the domain hierarchy, instead of making the master or the domain manager open a direct connection with the workstation.

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 with the status of jobs and job streams running on all other workstations in its domain and in subordinate domains, but not on peer or parent 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.

securitylevel
Specifies the type of SSL authentication for the workstation. It must have one of the following values:

enabled
The workstation uses SSL authentication only if its domain manager workstation or another fault-tolerant agent below it in the domain hierarchy requires it.

on
The workstation uses SSL authentication when it connects with its domain manager. The domain manager uses SSL authentication
when it connects to its parent domain manager. The fault-tolerant agent refuses any incoming connection from its domain manager if it is not an SSL connection.

**force** The workstation uses SSL authentication for all of its connections and accepts connections from both parent and subordinate domain managers. It will refuse any incoming connection if it is not an SSL connection.

If this attribute is omitted, the workstation is not configured for SSL connections. In this case, any value for **secureaddr** will be ignored. You should also set the **nm ssl port** local option to 0 to be sure that this port is not opened by netman. See the *IBM Tivoli Workload Scheduler Planning and Installation Guide* for reference on SSL authentication. The following table describes the type of communication used for each type of **securitylevel** setting.

<table>
<thead>
<tr>
<th>Fault-Tolerant Agent (Domain Manager)</th>
<th>Domain Manager (Parent Domain Manager)</th>
<th>Connection Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>-</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>Enabled</td>
<td>-</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>On</td>
<td>-</td>
<td>No connection</td>
</tr>
<tr>
<td>Force</td>
<td>-</td>
<td>No connection</td>
</tr>
<tr>
<td>-</td>
<td>On</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>Enabled</td>
<td>On</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>On</td>
<td>On</td>
<td>SSL</td>
</tr>
<tr>
<td>Force</td>
<td>On</td>
<td>SSL</td>
</tr>
<tr>
<td>-</td>
<td>Enabled</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>Enabled</td>
<td>Enabled</td>
<td>TCP/IP</td>
</tr>
<tr>
<td>On</td>
<td>Enabled</td>
<td>SSL</td>
</tr>
<tr>
<td>Force</td>
<td>Enabled</td>
<td>SSL</td>
</tr>
<tr>
<td>-</td>
<td>Force</td>
<td>No connection</td>
</tr>
<tr>
<td>Enabled</td>
<td>Force</td>
<td>SSL</td>
</tr>
<tr>
<td>On</td>
<td>Force</td>
<td>SSL</td>
</tr>
<tr>
<td>Force</td>
<td>Force</td>
<td>SSL</td>
</tr>
</tbody>
</table>

**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 example 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.ibm.com
domain masterdm
for maestro type fta
    autolink on
    fullstatus on
    resolvedep on
end

end

cpuname hdq-2
    os wnt
tz pst
node opera.ibm.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.ibm.com
domain distr-a
for maestro type fta
    autolink on
    fullstatus on
    resolvedep on
end
The following example creates a workstation with SSL authentication. The securitylevel security definition specifies that the connection between the workstation and its domain manager can be only of the SSL type. Port 32222 is reserved for listening for incoming SSL connections.

cpuname ENNETI3
    os WNT
    node apollo
tcpaddr 30112
    secureaddr 32222
    for maestro
        autolink off
        fullstatus on
        securitylevel on
end

See Also

For the equivalent Job Scheduling Console task, see "Creating a z/OS Workstation" and "Creating a Distributed Workstation" in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
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. Each workstation class definition has the following format and arguments:

Synopsis

```
# comment
cpuclass workstationclass
    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 workstationclass
    Specifies the name of the workstation class. The name must start with a letter, and can contain alphanumeric characters, dashes and underscores. 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 example defines a workstation class named `backup`:

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

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

```
cpuclass allcpus
    members @
end
```

See Also

For the equivalent Job Scheduling Console task, see “Creating a Workstation Class” in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
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. Each domain
definition has the following format and arguments:

Synopsis
# comment
domain domainname[description "text"]
   manager workstation
   [parent domainname]
end

cpuname ...

cpuclass ...

domain ...

Arguments
# comment
Treat all from the pound sign to the end of the line as a comment.

domain domainname
   The name of the domain. It must start with a letter, and can contain
   alphanumeric characters, dashes, and underscores. It can contain up to 16
   characters. 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 workstation
   Specifies the name of the workstation that is the domain manager. This
   workstation must belong to the domain that is being defined. The domain
   manager must be a fault-tolerant agent with fullstatus and resolvedep set
   to on.

parent domainname
   The name of the parent domain to which the domain manager is linked.
   The default is the master domain, which does not require a domain
   definition. The master domain is defined by the global options master and
   master domain.

Examples
The following example 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
See Also

For the equivalent Job Scheduling Console task, see "Creating a Domain" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
Job definitions

A job is an executable file, program, or command that is scheduled and launched by IBM Tivoli Workload Scheduler. You can write job definitions in edit files and then add them to the scheduler database with the composer command line program. You can include multiple job definitions in a single edit file.

Note: A job itself has no dependencies, these must be added to a job in a job stream. The scheduling language used to write job streams also permits you to add and modify job definitions. Please refer to next chapter for additional information on how to use scheduling language to write job streams.

Each job definition has the following format and arguments:

Synopsis

$jobs [workstation#]jobname

   scriptname filename |
   docommand "command" streamlogon username

[description "text"]
[interactive]
[rcondsucc "Success Condition"]
[recovery
   [stop | continue | rerun]
   [after [workstation#]jobname]
   [abendprompt "text"]]

[ [workstation#]jobname... ]

Arguments

workstation#

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. It can contain up to 40 characters.

scriptname filename

Specifies the name of the file the job runs. Use scriptname for UNIX and Windows jobs. For an executable file, enter the file name and any options and arguments. The length of filename plus the length of Success Condition (of the rcondsucc keyword) must not exceed 4095 characters. You can also use IBM Tivoli Workload Scheduler parameters. See “Using parameters in job definitions” on page 46 for more information.

For Windows 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 runs. Enter a valid command and any options and arguments enclosed in quotes ("."). The length of *command* plus the length of Success Condition (of the rccondsucc keyword) must not exceed 4095 characters. A command is run directly and, unlike scriptname, the configuration script, jobmanrc, is not run. Otherwise, the command is treated as a job, and all job rules apply. You can also enter IBM Tivoli Workload Scheduler parameters. See “Using parameters in job definitions” on page 46 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 IBM Tivoli Workload Scheduler parameters. See “Using parameters in job definitions” on page 46 for more information.

For Windows jobs, the user must also have a user definition. See “User definitions” on page 47 for user requirements.

**description** "*text*

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

**interactive**

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

**rccondsucc** "Success Condition"

An expression which determines the return code (RC) required to consider a job successful. The success condition maximum length must be 256 characters. This expression can contain a combination of comparison and boolean expressions:

**Comparison expression**

Specifies the job return codes. The syntax is:

(RC operator operand)

- **RC** The RC keyword.
- **operator** Comparison operator. It can have the following values:

<table>
<thead>
<tr>
<th>Example</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC&lt;a</td>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>RC&lt;=a</td>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>RC&gt;a</td>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>RC&gt;=a</td>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>RC=a</td>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>RC!=a</td>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>RC&lt;&gt;a</td>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>

Chapter 3. Composer reference 43
operand

An integer between -2147483647 and 2147483647.

For example, you can define a successful job as a job that ends with a return code less than or equal to 3 as follows:

rccondsucc "(RC <= 3)"

Boolean expression

Specifies a logical combination of comparison expressions. The syntax is:

comparison_expression operator comparison_expression

comparison_expression

The expression is evaluated from left to right. You can use parentheses to assign a priority to the expression evaluation.

operator

Logical operator. It can have the following values:

<table>
<thead>
<tr>
<th>Example</th>
<th>Operator</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr_a and expr_b</td>
<td>And</td>
<td>TRUE if both expr_a and expr_b are TRUE.</td>
</tr>
<tr>
<td>expr_a or expr_b</td>
<td>Or</td>
<td>TRUE if either expr_a or expr_b is TRUE.</td>
</tr>
<tr>
<td>Not expr_a</td>
<td>Not</td>
<td>TRUE if expr_a is not TRUE.</td>
</tr>
</tbody>
</table>

For example, you can define a successful job as a job that ends with a return code less than or equal to 3 or with a return code not equal to 5, and less than 10 as follows:

rccondsucc "((RC=3) OR ((RC>=5) AND (RC<10)))"

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.

after [workstation#]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:

- 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 workstation must be in the same domain as the parent job workstation.
• If the recovery job workstation is a fault-tolerant agent, it must have a value of on for fullstatus.

`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.

The following table summarizes all possible combinations of recovery options and actions. The table is based on the following criteria from a job stream called sked1:
• Job stream sked1 has two jobs, job1 and job2.
• If selected for job1, the recovery job is jobr.
• job2 is dependent on job1 and will not start until job1 is complete.

<table>
<thead>
<tr>
<th>Recovery prompt</th>
<th>Stop</th>
<th>Continue</th>
<th>Rerun</th>
</tr>
</thead>
<tbody>
<tr>
<td>Recovery job:</td>
<td>Intervention is</td>
<td>Run job2.</td>
<td>Rerun job1. If job1</td>
</tr>
<tr>
<td>No</td>
<td>required.</td>
<td></td>
<td>abends, issue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>scheduler prompt. If reply is yes, repeat above. If job1 is successful, run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rerun job1. If job1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>abends, issue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>scheduler prompt. If reply is yes, repeat above. If job1 is successful, run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rerun job1. If job1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>abends, intervention</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is required. If it is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successful, run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run jobr. Run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run. If jobr abends,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>intervention is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>required. If it is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successful, rerun</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>job1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If job1 abends, issue</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>scheduler prompt. If reply is yes, repeat above. If job1 is successful, run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Rerun job1. If job1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>abends, intervention</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>is required. If it is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successful, rerun</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>job1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If job1 abends, intervention is required. If jobr is successful, rerun job1. If job1 abends, repeat above. If job1 is successful, run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run jobr. Run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run. If jobr abends,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>intervention is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>required. If it is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successful, run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue recovery prompt. If</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reply is yes, run jobr.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run jobr. Run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run. If jobr abends,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>intervention is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>required. If it is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successful, run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue recovery prompt. If</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reply is yes, run jobr.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run jobr. Run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run. If jobr abends,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>intervention is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>required. If it is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successful, run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Issue recovery prompt. If</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>reply is yes, run jobr.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run jobr. Run job2.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Run. If jobr abends,</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>intervention is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>required. If it is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>successful, run</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>job2.</td>
</tr>
</tbody>
</table>

Notes:
1. "Intervention is required" means that job2 is not released from its dependency on job1, and therefore must be released by the operator.
2. The continue recovery option overrides the abend state, which may cause the schedule containing the abended job to be marked as successful. This will prevent the schedule from being carried forward to the next day.

3. If you select the Rerun option without supplying a recovery prompt, the scheduler generates its own prompt.

4. To reference a recovery job in Conman, you must use the name of the original job (job1 in the scenario above, not jobr). Recovery jobs are run only one per abend.

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.
- Ensure that the caret characters are not preceded by a backslash in the string. If it occurs, move that backslash in the definition of the parameter between carets.

For example instead of entering the following parameters definition:

```
$PARM
MYDIR "scripts"
job01 scriptname "c:\pippo\home\MYDIR\test.cmd"
```

you must enter:

```
$PARM
MYDIR \"scripts\"
job01 scriptname "c:\pippo\home\MYDIR\test.cmd"
```

In the example below a parameter named mis is used in the streamlogon value. For additional examples, see ["Parameter definitions" on page 50].

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
```

See Also
For the equivalent Job Scheduling Console task, see "Creating a Job Definition" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
User definitions

The user names used as the streamlogon value for Windows job definitions must have user definitions. This is not required for users who run jobs on other platforms. Each user definition has the following format and arguments:

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 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 domain of the user and user is the name of the user.

    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 user names are case-sensitive. Also, the user must be able to log on to the workstation on which the scheduler will launch jobs, and must have the right to Log on as batch.

    If the name is not unique in Windows, 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.

Using the Tivoli Workload Scheduler user and streamlogon definitions: On Windows, user definitions are specified through composer in the form [wkstation#]username. The workstation name is optional; its absence indicating all the workstations running on Widows in the Tivoli Workload Scheduler network.

When you define or submit a job with composer, you must specify both a workstation and a valid user logon for the workstation. The logon in these cases is
just a *username*—a valid user name for Windows—without the complement of the workstation name. For example, in the following job definition:

```
$JOB
wkstation job01 dcommand "dir"
streamlogon username
```

the value for `streamlogon` is `username` and not `wkstation#username`.

However, when you use the `altpass` command, remember that you must use the user definition in the form

```
wkstation#username
```

For this command, you can omit the workstation name only in the case you are changing the password of the workstation from where you are running the command.

**Trusted domain user:** If the scheduler must launch jobs for a trusted domain user, give special attention to defining the user accounts. Assuming the 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
```

**See Also**

For the equivalent Job Scheduling Console task, see "Creating a Windows User" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
Calendar definitions

Calendars are lists of dates that you can use to schedule job streams. Calendar definitions are entered using the composer modify command. When you enter the command, composer copies the complete list of calendar definitions into an edit file and starts an editor where you can modify the list. Each calendar definition has the following format and arguments:

**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. It can contain alphanumeric characters as long as it starts with a letter. It can contain the following characters: comma (,), period (.), dash (-), plus (+), single quote ('), and equal (=). It cannot contain double quotes (") other than the enclosing ones, colon (:), semi-colon (;), and ampersand (&).

- **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
```

**See Also**

For the equivalent Job Scheduling Console task, see "Creating a Calendar" in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
Parameter definitions

Parameters are values which substitute variables in job and job stream definitions when as the new production plan is created. Parameter definitions are entered using the composer modify command. When you enter the command, composer copies the complete list of parameter definitions into an edit file and starts an editor where you can modify the list. Each parameter definition has the following format and arguments:

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.

You can use multiple parameters in a single variable. When you use a parameter enclose it in carets (^), and enclose the entire string in quotation marks. Ensure that the caret characters are not preceded by a backslash in the string. If it occurs, move that backslash in the definition of the parameter between carets. For example instead of entering the following parameters definition:

$PARM
  MYDIR "scripts"
  job01 scriptname "c:\pippo\home\^MYDIR^\test.cmd"

you must enter:

$PARM
  MYDIR \"scripts\"
  job01 scriptname "c:\pippo\home\MYDIR\test.cmd"

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

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

The glpath and gllogon parameters are used in the gljob2 job of the glsched job stream:

schedule glsched on weekdays
  gljob2
    scriptname "/usr/gl^glpath^"
streamlogon "^gllogon^"
opens "^glpath^/datafile"
prompt ":^glpath^ started by ^gllogon^"
end

See Also
For the equivalent Job Scheduling Console task, see "Creating a Parameter" in the
IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
Prompt definitions

You can use prompts as dependencies in jobs and job streams. A prompt is defined by a unique name associated with a textual message and must be answered affirmatively for the dependent job or job stream to launch. Prompt definitions are entered using the composer modify command. When you enter the command, composer copies the complete list of prompt definitions into an edit file and starts an editor where you can modify the list.

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 scheduler database and can be used by any job or job stream.

Note: Predefined or global prompt definitions are reset each time the Jnextday command is run.

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 50 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?"

See Also
For the equivalent Job Scheduling Console task, see "Creating a Prompt" in the
IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
Resource definitions

Resources represent physical or logical scheduling resources that can be used as dependencies for jobs and job streams. Resource definitions are entered using the composer modify command. When you enter the command, composer copies the complete list of resource definitions into an edit file and starts an editor where you can modify the list. Each resource definition has the following format and arguments:

**Synopsis**

```
$resource
wkstation#resourcename units ["description"]
```

**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"
```

**See Also**

For the equivalent Job Scheduling Console task, see "Creating a Distributed Resource" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
## The composer command line program

The Composer command line program manages scheduling objects in the scheduler 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
  ```

- Runs `print` and `version` commands, and quits:
  
  ```
  composer "p parms&v"
  ```

- Runs `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

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:

**Windows variables:**

- **$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.

---

Chapter 3. Composer reference 55
UNIX variables: The \( \texttt{offline} \) 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:

\( > \text{file} \) Redirects output to a file and overwrites the contents of the file. If the file does not exist, it is created.

\( >> \text{file} \) Redirects output to a file and appends the output to the end of the file. If the file does not exist, it is created.

\( \text{1 command} \)
Pipes output to a system command or process. The system command is run whether or not output is generated.

\( \text{11 command} \)
Pipes output to a system command or process. The system command is not run if there is no output.

The default is \( \text{1 lp -CONLIST} \) 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.

The composer editor
Several Composer commands automatically open a text editor. You can select which editor you want Composer to use.

Windows: On Windows, the default editor is the MS-DOS editor ‘edit’. This editor, however, follows the 8.3 naming convention which can be a limitation when composer uses it to modify database objects. If the name of the object is longer than 8.3 characters, the Composer MODIFY command will run into an error. You are recommended to use another editor, such as Notepad. To change the editor, set the variable \texttt{EDITOR} to the name of the new editor before running Composer.

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 \texttt{VISUAL} is set, it defines the editor, otherwise the variable \texttt{EDITOR} defines the editor. If neither of the variables is set, a \texttt{vi} editor is opened.

Selecting the composer command prompt on UNIX
The composer command prompt is defined in the \texttt{TWShome/localopts} file. The default command prompt is a dash (\texttt{-}). To select a different prompt, edit the composer prompt option in the \texttt{localopts} file and change the dash. The prompt can be up to ten characters long, not including the required trailing pound sign (\#):

```bash
#-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-
# Custom format attributes
#
date format = 1 # The possible values are 0-ymd, 1-mdy, 2-dmy, 3-NLS.
composer prompt = -
```
conman prompt = \%
switch sym prompt = <n>\%
#---------------------------------------------------------------

Command syntax

Composer commands consist of the following elements:

\textit{commandname} \textit{selection} \textit{arguments}

where:

\textit{commandname}

Specifies the command name.

\textit{selection}

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

\textit{arguments}

Specifies the command arguments.

Wildcards

The following wildcard characters are permitted in some Composer commands:

@ Replaces one or more alphanumeric characters.

? Replaces one alphanumeric character.

% Replaces one numeric character.

Delimiters and special characters

The following characters have special meanings in Composer commands.

<table>
<thead>
<tr>
<th>Character</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Command delimiter. See &quot;Running composer&quot; on page 55.</td>
</tr>
<tr>
<td>;</td>
<td>Argument delimiter. For example: \textit{;info;offline}</td>
</tr>
<tr>
<td>=</td>
<td>Value delimiter. For example: \textit{sched=sked5}</td>
</tr>
<tr>
<td>:!</td>
<td>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: \textit{!ls} or \textit{:ls}</td>
</tr>
<tr>
<td>&lt;&lt; &gt;&gt;</td>
<td>Comment brackets. Comments can be placed on a single line anywhere in a job stream. For example: \textit{schedule foo &lt;&lt;comment&gt;&gt; on everyday}</td>
</tr>
<tr>
<td>*</td>
<td>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: \texttt{!<em>comment} or \texttt{print</em>!comment}</td>
</tr>
<tr>
<td>&gt;</td>
<td>Redirects command output to a file and overwrites the contents of the file. If the file does not exist, it is created. For example: \texttt{display parms &gt; parmlist}</td>
</tr>
<tr>
<td>Character</td>
<td>Description</td>
</tr>
<tr>
<td>-----------</td>
<td>-------------</td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>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: <code>display parms &gt;&gt; parmlist</code></td>
</tr>
<tr>
<td></td>
<td>Pipes command output to a system command or process. The system command is run whether or not output is generated. For example: `display parms</td>
</tr>
<tr>
<td></td>
<td>Pipes command output to a system command or process. The system command is not run if there is no output. For example: `display parms</td>
</tr>
</tbody>
</table>

---

**Command descriptions**

The following pages describe Composer’s commands.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>add</td>
<td>Adds scheduling objects.</td>
<td>“add” on page 60</td>
</tr>
<tr>
<td>build</td>
<td>Builds or rebuilds a IBM Tivoli Workload Scheduler file.</td>
<td>“build” on page 61</td>
</tr>
<tr>
<td>continue</td>
<td>Ignores the next error.</td>
<td>“continue” on page 62</td>
</tr>
<tr>
<td>create</td>
<td>Creates a file for editing.</td>
<td>“create” on page 63</td>
</tr>
<tr>
<td>delete</td>
<td>Deletes scheduling objects.</td>
<td>“delete” on page 65</td>
</tr>
<tr>
<td>display</td>
<td>Displays scheduling objects.</td>
<td>“display, list, print” on page 67</td>
</tr>
<tr>
<td>edit</td>
<td>Edits a file.</td>
<td>“edit” on page 70</td>
</tr>
<tr>
<td>exit</td>
<td>Terminates Composer.</td>
<td>“exit” on page 71</td>
</tr>
<tr>
<td>list</td>
<td>Lists scheduling objects.</td>
<td>“display, list, print” on page 67</td>
</tr>
<tr>
<td>modify</td>
<td>Modifies scheduling objects.</td>
<td>“modify” on page 72</td>
</tr>
<tr>
<td>new</td>
<td>Edits and adds scheduling objects.</td>
<td>“new” on page 74</td>
</tr>
<tr>
<td>print</td>
<td>Prints scheduling objects.</td>
<td>“display, list, print” on page 67</td>
</tr>
<tr>
<td>redo</td>
<td>Edits the previous command.</td>
<td>“redo” on page 75</td>
</tr>
<tr>
<td>replace</td>
<td>Replaces scheduling objects.</td>
<td>“replace” on page 77</td>
</tr>
<tr>
<td>validate</td>
<td>Validates a file.</td>
<td>“validate” on page 79</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------------------</td>
<td>----------------------------------------------------------</td>
<td>----------------------------</td>
</tr>
<tr>
<td>version</td>
<td>Displays Composer’s command line program banner.</td>
<td>“version” on page 80</td>
</tr>
<tr>
<td>system command</td>
<td>Passes a system command to the system.</td>
<td>“system command” on page 78</td>
</tr>
</tbody>
</table>

You can type command names and keywords in either uppercase or lowercase. You can also 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
Specifications 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
To add the jobs from the file myjobs, run the following command:
add myjobs

To add the job streams from the file mysked, run the following command:
a mysked

To add the workstations, workstation classes, and domains from the file cpus.src, run the following command:
a cpus.src

To add the user definitions from the file users_nt, run the following command:
a users_nt

See Also
For the equivalent Job Scheduling Console task, see "Monitoring Tasks" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
**build**

Builds or rebuilds scheduler database files. You must have **build** access to the file.

If you have implemented fix pack 8.2-TWS-FP04, you can use this command while IBM Tivoli Workload Scheduler is running. If you have not implemented this or a later fix pack, it is advisable to stop all IBM Tivoli Workload Scheduler processes before running this command.

**Synopsis**

```bash
build filename
```

**Arguments**

```text
database file name
```

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**

To rebuild the files containing job stream definitions, run the following command:

```bash
build mastsked
```

To rebuild the file containing calendars, run the following command:

```bash
build calendars
```

To rebuild the file containing workstations, run the following command:

```bash
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**

If you want the Composer to continue with the `print` command if the `delete` command results in an error, run the following command:

```
composer "continue&delete cpu=site4&print cpu=0"
```
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 jobs 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 job streams 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 users 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.

**Note:** You should create regularly a back-up copy of the objects stored in the database.

**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**
To create a file containing all calendars, run the following command:

```plaintext
create caltemp from calendars
```

To create a file containing all job streams, run the following command:

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

**See Also**
For the equivalent Job Scheduling Console task, see "Monitoring Tasks" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.*
**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 workstations, workstation classes, or domains.
  
  `wkstation`
  
  The workstation name. Wildcards are permitted.

  `wkstationclass`
  
  The workstation class name. Wildcards are permitted.

  `domain`
  
  The domain name. Wildcards are permitted.

- **jobs** Deletes jobs.
  
  `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 job streams.
  
  `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 users.
  
  `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**

To delete **job3** that is launched on workstation **site3**, run the following command:

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

To delete all workstations with names starting with **ux**, run the following command:

```
delete cpu=ux
```
To delete all job streams with names starting with *test* on all workstations, run the following command:

de sched=@#test@

**See Also**

For the equivalent Job Scheduling Console task, see "Monitoring Tasks" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
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 MAESTROL. See “Offline output” on page 55 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 name 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**
To display all calendars, run the following command:
```bash
display calendars
```

To print all job streams that are launched on workstation **site2**, run the following command:
```bash
di sched=site2@;offline
```

**See Also**
For the equivalent Job Scheduling Console task, see "Working with Object Lists" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.*
**edit**

Edits a file.

**Synopsis**

```
edit filename
```

**Usage Notes**

An editor is started and the specified file is opened for editing. See "The composer editor" on page 56 for more information.

**Examples**

To open the file `mytemp` for editing, run the following command:

```
edit mytemp
```

To open the file `resfile` for editing, run the following command:

```
ed resfile
```

**See Also**

For the equivalent Job Scheduling Console task, see "Monitoring Tasks" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide*. 
exit

Exits the Composer command line program.

**Synopsis**

`exit`

**Usage Notes**

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

**Examples**

To exit the Composer command line program, run the following command:

`exit`

or:

`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
To modify all calendars, run the following command:
modif calendars

To modify job stream sked9 that is launched on workstation site1, run the following command:
mod sched=site1#sked9

See Also
For the equivalent Job Scheduling Console task, see "Monitoring Tasks" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
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, use the `modify` command on page 72.

**Examples**

To create a temporary file, edit the file, and then add the contents of the file to the database, run the following command:

```
new
```

or:

```
n
```
redo
Edits and reruns the previous command.

Synopsis
redo

Context
When you run the *redo* command, Composer displays the previous command, so that it can be edited and rerun. 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.

>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 *ds*.

*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

To insert a character, run the following command:

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

To replace a character, run the following command:
redo

display site1#sa0

r2

display site2#sa0
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 60.

Examples

To replace the jobs from the file myjobs, run the following command:

replace myjobs

To replace the job streams from the file mysked, run the following command:

rep mysked

To replace all resources with those contained in the file myres, run the following command:

rep myres
**system command**

Runs 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**

To run a *ps* command on UNIX, run the following command:

```
ps -ef
```

To run a *dir* command on Windows, run the following command:

```
dir \bin
```
validate

Validates a file containing scheduling objects.

Synopsis

validate filename[syntax]

Arguments

filename

Specifies the name of a file that contains calendars, 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

To check the syntax of a file containing workstation definitions, run the following command:

validate mycpus;syntax

To revalidate all job streams, run the following command:

create allskeds from sched=@#@
validate allskeds

You can do this to verify the integrity of references to other scheduling objects after changes have been made.
**version**

Displays the Composer’s command line program banner.

**Synopsis**

`version`

**Examples**

To display the Composer’s command line program banner, run the following command:

```
version
```

or:

```
v
```
Chapter 4. The scheduling language

This chapter describes how to create a job stream, based on the scheduling objects defined in the database, using the Composer command line program. Job streams created using the graphical user interface do not reference the keywords listed in this chapter, but all job streams in the scheduler are saved using the same scheduling language syntax and keywords. This chapter contains information about the following:

- Syntax for Job Streams
- Keyword Descriptions

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 [cpu#]sched
    [freedays Calendar_Name [ -sa ] [ -su ]]
    on [ date | day | calendar | request ][ ... ] [ fdignore | fdnext | fdprev ]
    on [ date | day | calendar | request ][ ... ] [ fdignore | fdnext | fdprev ]
    [...]
    [ deadline time [ timezone | tz tzname ][ +n day[s] ][ ... ]
    [ except [ date | day | calendar ][ ... ] [ fdignore | fdnext | fdprev ]
    [...]
    [ at time [ timezone | tz tzname ][ +n day[s] ][ ... ]
    [ carryforward ]
    [ keysched ]
    [ limit number ]
    [ needs resource ]
    [ opens file ]
    [ priority number ]
    [ prompt name | text ]
    [ until time [ timezone | tz tzname ][ +n day[s] ] [ onuntil action ]
    ]:

job-statement
    [ at time [ timezone | tz tzname ][ +n day[s] ] ] [...]
    [ confirmed ]
    [ deadline time [ timezone | tz tzname ][ +n day[s] ] [...]
    [ every rate ]
    [ follows job \ jstream ]
    [ keyjob ]
    [ needs resource ]
    [ opens file ]
    [ priority number ]
    [ prompt name | text ]
    [ until time [ timezone | tz tzname ][ +n day[s] ] [ onuntil action ]
    [ job-statement ] ... ]
end
```
Do not attempt to write job streams on a single line, since they will be rejected by the `schedule` command after having been apparently accepted by composer. For instance, writing the following schedule in one line:

```
SCHEDULE cbdbu01#NN_CBCID1001YS31 ON CBZ0001 PRIORITY 10 FOLLOWs
  cbdbu02#NN_CBCND10011501:cbdbu01#JN_CBCID1001YS1GA1 PRIORITY 10 OPENS
  */st01/st0lf/in/CUBCIFYS1GA1001.dat.ok
```

will result in an error being returned when the schedule is processed by `schedule`. For this example, you should do one of the following:

- **Use composer** to:
  1. Create a filename from `schedule=cbdbu01#NN_CBCID1001YS31` (or all the schedules that are defined in this way).
  2. Run composer `add filename`

- **Split the schedule in the following way**:

```
SCHEDULE cbdbu01#NN_CBCID1001YS31 ON CBZ0001 PRIORITY 10 FOLLOWs
  cbdbu02#NN_CBCND10011501:cbdbu01#JN_CBCID1001YS1GA1 PRIORITY 10 OPENS
  */st01/st0lf/in/CUBCIFYS1GA1001.dat.ok
```

### Keywords

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

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>at</code></td>
<td>Defines the time of day that job stream or job execution begins.</td>
<td>“at” on page 84</td>
</tr>
<tr>
<td><code>carryforward</code></td>
<td>Carries the job stream forward if it is not completed.</td>
<td>“carryforward” on page 85</td>
</tr>
<tr>
<td><code>comments</code></td>
<td>Includes comments in a job stream definition</td>
<td>“comments” on page 86</td>
</tr>
<tr>
<td><code>confirmed</code></td>
<td>Specifies that the completion of this job requires confirmation.</td>
<td>“confirmed” on page 87</td>
</tr>
<tr>
<td><code>deadline</code></td>
<td>Specifies the time within which a job or job stream should complete.</td>
<td>“deadline” on page 88</td>
</tr>
<tr>
<td><code>end</code></td>
<td>Marks the end of a job stream.</td>
<td>“end” on page 89</td>
</tr>
<tr>
<td><code>every</code></td>
<td>Launches the job repeatedly at a specified rate.</td>
<td>“every” on page 90</td>
</tr>
<tr>
<td><code>except</code></td>
<td>Specifies dates that are exceptions to the dates the job stream is selected for execution.</td>
<td>“except” on page 91</td>
</tr>
<tr>
<td><code>follows</code></td>
<td>Specifies not to launch this job or job stream until other jobs and job streams have completed successfully.</td>
<td>“follows” on page 93</td>
</tr>
<tr>
<td><code>freedays</code></td>
<td>Specifies a freeday calendar for calculating workdays for the job stream. It can also set Saturdays and Sundays as workdays.</td>
<td>“freedays” on page 94</td>
</tr>
<tr>
<td><code>job statement</code></td>
<td>Defines a job and its dependencies.</td>
<td>“job statement” on page 96</td>
</tr>
<tr>
<td><code>keyjob</code></td>
<td>Marks a job as key or critical in both the database and daily plan for monitoring by applications, such as IBM Tivoli Business Systems Manager or Tivoli Enterprise Console®.</td>
<td>“keyjob” on page 101</td>
</tr>
<tr>
<td>Keyword</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>keysched</td>
<td>Marks a job stream as key or critical in both the database and daily plan for monitoring by applications, such as IBM Tivoli Business Systems Manager or Tivoli Enterprise Console.</td>
<td>“keysched” on page 102</td>
</tr>
<tr>
<td>limit</td>
<td>Sets a limit on the number of jobs that can be launched concurrently from the job stream.</td>
<td>“limit” on page 103</td>
</tr>
<tr>
<td>needs</td>
<td>Defines the number of units of a resource required by the job or job stream before it can be launched.</td>
<td>“needs” on page 104</td>
</tr>
<tr>
<td>on</td>
<td>Defines the dates on which the job stream is selected for execution.</td>
<td>“on” on page 105</td>
</tr>
<tr>
<td>opens</td>
<td>Defines files that must be accessible before the job or job stream is launched.</td>
<td>“opens” on page 108</td>
</tr>
<tr>
<td>priority</td>
<td>Defines the priority for a job or job stream.</td>
<td>“priority” on page 111</td>
</tr>
<tr>
<td>prompt</td>
<td>Defines prompts that must be replied to before the job or job stream is launched.</td>
<td>“prompt” on page 111</td>
</tr>
<tr>
<td>schedule</td>
<td>Assigns a name to the job stream.</td>
<td>“schedule” on page 112</td>
</tr>
<tr>
<td>until</td>
<td>Defines a time of day after which the job or job stream is not launched.</td>
<td>“until” on page 113</td>
</tr>
</tbody>
</table>

### 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.

**Note:** The only dependency that is checked immediately before running a job is NEEDS. An OPENS dependency is automatically considered resolved before the job runs.

### Case sensitivity

With the exception of path names, user names, and UNIX commands, which are case-sensitive, you can use either upper or lower case characters when writing your schedules.

### 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

\texttt{at } \texttt{time} \texttt{[timezone|tz tzname][+n day[s]] [...]} 

Arguments

\textit{time} \hfill Specifies a time of day. Possible values can range from \texttt{0000} to \texttt{2359}.

\textit{tzname} \hfill Specifies the time zone to be used when computing the start time. See \texttt{Appendix B, “Managing time zones,” on page 319} 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 \textit{at} time and an \texttt{until} time are specified, the time zones must be the same.

\textit{n} \hfill Specifies an offset in days from the scheduled start date and time.

Usage Notes

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

The time value in the \textit{at} option is considered as follows:

\begin{itemize}
  \item If the time value is less than the current time, it is taken as for the following day.
  \item If the time value is greater than the current time, it is taken as for the current day.
  \item If you specify a time value greater than \texttt{2400}, the value is divided by \texttt{2400} to obtain the number of days. If you specify days, these are add to the value obtained by dividing by \texttt{2400}.
\end{itemize}

Examples

The following examples assume that the IBM Tivoli Workload Scheduler 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.

  \begin{verbatim}
  schedule sked7 on tu at 0300:
  job1
  job2
  end
  \end{verbatim}

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

  \begin{verbatim}
  sfran#schedule sked8 on fr at 1000 + 1 day :
  job1
  job2 at 1400 tz est
  nycity#job3 at 1600
  end
  \end{verbatim}
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 definition.

**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 double angle brackets.

**Examples**

The following example includes both types of comments:

****************************************
* The weekly cleanup jobs
****************************************
*  
  schedule wkend on fr at 1830
  carryforward
  ;
  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 145 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.

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

Specifies the time within which a job or job stream must complete. Jobs or job streams that have not yet started or that are still running when the deadline time has expired, are considered *late* in the plan. When a job (or job stream) is late, the following actions are performed:

- Job is shown as late in Conman and Job Scheduling Console.
- An event is sent to the Tivoli Enterprise Console and the IBM Tivoli Business Systems Manager.
- A message is issued to the stdlist and console logs.

**Synopsis**

```
deadline time [timezone 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 deadline. See Appendix B, “Managing time zones,” on page 319 for time zone names. The default is the time zone of the workstation on which the job or job stream is launched.
- `n` Specifies an offset in days from the scheduled deadline time.

**Examples**

The following example launches job stream `sked7` everyday and job `jobc` to start running at 14:30 and to be completed by 16:00.

```
schedule sked7 on everyday :
  jobc at 1430 deadline 1600
end
```
end

Marks the end of a job stream definition.

**Synopsis**

end

**Examples**

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

Defines the repetition rate for a job. The job is launched repeatedly at the specified rate. If the job has a dependency that is not satisfied, the iteration is started when the dependency is satisfied.

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.)

Usage Notes
If an every job abends, the iteration continues. If the every option is used with the AT dependency, jobs are launched at specific times. If one rerun is delayed (for a dependency or for any other reason) IBM Tivoli Workload Scheduler will realign to the AT time. In this case one or two iterations might not respect the EVERY rate. If the EVERY option is used without the AT dependency, the rerun jobs will be scheduled respecting the EVERY rate specified, starting from the time when the job actually started. IF AND ONLY IF the every option is used with the AT dependency can there be any interactions that do not respect the every rate. For all other cases the every rate is always respected.

Examples
The following example runs the job testjob every hour:
testjob every 100

The following example runs 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

The job will run at 1800, 1815, 1830, and so on. If the job is submitted adhoc at 1833, the reruns will be at 1833,1834,1845 and so on.

The following example does not start the job testjob2 iteration until the job testjob1 has completed successfully:
testjob2 every 15 follows testjob1
except

Defines the dates that are exceptions to the on dates of a job stream. See “on” on page 105 for more information.

Synopsis

except {date | day | calendar} [fdignore | fdnext | fdprev][,...]
[except {date | day | calendar}] [fdignore | fdnext | fdprev][,...]

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.

workdays Can be one of the following:
• If you specified a freedays calendar, workdays are everyday excluding saturday and sunday (unless you specified -sa or -su along with the freedays keyword) and excluding all the dates of the specified freedays calendar.
• If you did not specify a freedays calendar, workdays are everyday excluding saturday and sunday and excluding all the dates of the holidays calendar.

freedays The days marked in the freedays calendar, if you specified one.

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 Stands for every day of the week.

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

workdays Stands for every day of the week, except for Saturdays and Sundays (unless otherwise specified with the freedays keyword) and for the dates marked either in a designated freedays calendar or in the holidays calendar.

freeday rule Specifies a rule that must be applied when the date selected for exclusion falls on a freeday. Can be one of the following:
Usage Notes
You can define multiple instances of the except keyword for the same job stream. Each instance is equivalent to a run cycle to which you can associate a freeday rule.

Multiple except instances must be consecutive within the job stream definition.

Each instance of the keyword can contain any of the values allowed by the except syntax.

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

Select job stream sked4 to run on Mondays, Tuesdays, and 2 weekdays prior to each date listed in the MTHEND calendar. If the run date is a freeday, run the job stream on the nearest following workday. Do not run the job stream on Wednesdays.
schedule sked4
on mo
on tu, MTHEND -2 weekdays fdnext
except we

Select job stream testskd2 to run every weekday except for the days listed in MTHEND. If a date in MTHEND falls on a freeday, exclude the nearest workday before it. In this example, the freedays are Saturdays, Sundays, and all the dates listed in the default holidays calendar.
schedule testskd2
on weekdays
except MTHEND fdprev

fdignore
Do not exclude the date.

fdnext Exclude the nearest workday after the freeday.

fdprev Exclude the nearest workday before the freeday.
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:

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

Use the following syntax for jobs:

```plaintext
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.@
```
**freedays**

Enables you to specify the name of a freedays calendar (see the User’s Guide for a description of freedays calendars) that lists the days when the job stream should not run. Tivoli Workload Scheduler uses this calendar as the base calendar for calculating *workdays* for the job stream.

The keyword affects only the scheduling of the job streams for which it is specified.

**Synopsis**

`freedays Calendar_Name [-sa] [-su]`

**Arguments**

*Calendar_Name*

The name of the calendar that must be used as the freedays calendar for the job stream. If *Calendar_Name* is not in the database, Tivoli Workload Scheduler issues a warning message when you save the job stream. If *Calendar_Name* is not in the database when scheduler runs, Tivoli Workload Scheduler issues an error message and uses the default calendar *holidays* in its place. Do not use the names of weekdays for the calendar names.

- `-sa` Count Saturdays as *workdays*.
- `-su` Count Sundays as *workdays*.

**Usage Notes**

If you do specify a freedays calendar in the job stream definition, then the *workdays* keyword takes on the following value: *workdays* = everyday excluding *saturday* and *sunday* (unless user specified `-sa` or `-su` along with *freedays*) and excluding all the dates of *Calendar_Name*

If you do not specify *freedays* in the job stream definition, then: *workdays* = everyday excluding *saturday* and *sunday* and all the dates of the *holidays* calendar.

By default, *saturday* and *sunday* are considered as freedays unless you specify the contrary by adding `-sa` and/or `-su` after *Calendar_Name*.

**Examples**

Select job stream *sked2* to run on 01/01/2001 and on all workdays as long as they are not listed in the freedays calendar named GERMHOL.

```
schedule sked2
freedays GERMHOL
on 01/01/2001, workdays
```

Select job stream *sked3* to run two workdays before each date in the PAYCAL calendar. Workdays are everyday from Monday to Saturday as long as they are not listed in the freedays calendar named USAHOL.

```
schedule sked3
freedays USAHOL -sa
on PAYCAL -2 workdays
```

Select job stream *sked3* on the dates listed in the APDATES calendar. If the selected date is a freeday, do not run the job stream. In this example, Sundays and all the dates listed in the GERMHOL calendar are to be considered as freedays. All days from Monday to Saturday, except for the specific dates listed in GERMHOL, are workdays.
Select job stream **testske3** to run every weekday except 5/15/2002 and 5/23/2002. If 5/23/2002 is a freeday, do not exclude it. In this example, Saturdays, Sundays, and all the dates listed in GERMHOL are to be considered as freedays.

All days from Monday to Friday, except for the specific dates listed in GERMHOL, are workdays.

Select job stream **testske4** to run every day except two weekdays prior to every date listed in the MONTHEND calendar. If the date to be excluded is a freeday, do not exclude it, but exclude the nearest following workday. In this example, freedays are all the dates listed in USAHOL, while workdays are all the days from Monday to Sunday that are not in USAHOL.
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 filename | docommand “commandline”][streamlogon username]
[interactive]
[rccondsucc "Success Condition"]
[recovery {stop | continue | rerun}
  [after [wkstation#]jobname]
  [abendprompt “text”] ]
[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. It can contain up to 40 characters.

Note: Do not use the word `recovery` as the job name. This is reserved.

`description`

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

`scriptname filename`

Specifies the name of the file the job runs. Use `scriptname` for UNIX and Windows jobs. For an executable file, enter the file name and any options and arguments. The length of `filename` plus the length of `Success Condition` (of the `rccondsucc` keyword) must not exceed 4095 characters. You can also use IBM Tivoli Workload Scheduler parameters. See “Using Parameters in Job Definitions” on page 99 for more information.

For Windows 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 runs. Enter a valid command and any options and arguments enclosed in quotes ("""). The length of `command` plus
the length of *Success Condition* (of the **rcondsucc** keyword) must not exceed 4095 characters. A command is run directly and, unlike *scriptname*, the configuration script, **jobmanrc**, is not run. Otherwise, the command is treated as a job, and all job rules apply. You can also enter IBM Tivoli Workload Scheduler parameters. See "Using Parameters in Job Definitions" on page 99 for more information.

On Windows-based fault-tolerant agents, jobs will fail if the path of the script/program that is to run contains a blank. For example, the statement

```
CMD.EXE /C "C:\PROGRAM FILES\CCR\CCR_CONSOLIDATE.BAT"
```

will fail, producing the following message:

The name specified is not recognized as an internal or external command, operable program or batch file.

because blanks are not managed in this context.

**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 IBM Tivoli Workload Scheduler parameters. See "Using Parameters in Job Definitions" on page 99.

For Windows jobs, the user must also have a user definition. See "User definitions" on page 47 for user requirements.

**interactive**

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

**rcondsucc** "Success Condition"

An expression which determines the return code (RC) required to consider a job successful. The success condition maximum length must be 256 characters. This expression can contain a combination of comparison and boolean expressions:

**Comparison expression**

Specifies the job return codes. The syntax is:

```
(RC operator operand)
```

**RC** The RC keyword.

**operator**

Comparison operator. It can have the following values:

<table>
<thead>
<tr>
<th>Example</th>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>RC&lt;a</td>
<td>&lt;</td>
<td>Less than</td>
</tr>
<tr>
<td>RC&lt;=a</td>
<td>&lt;=</td>
<td>Less than or equal to</td>
</tr>
<tr>
<td>RC&gt;</td>
<td>&gt;</td>
<td>Greater than</td>
</tr>
<tr>
<td>RC&gt;=a</td>
<td>&gt;=</td>
<td>Greater than or equal to</td>
</tr>
<tr>
<td>RC=</td>
<td>=</td>
<td>Equal to</td>
</tr>
<tr>
<td>RC!=a</td>
<td>!=</td>
<td>Not equal to</td>
</tr>
<tr>
<td>RC&lt;&gt;a</td>
<td>&lt;&gt;</td>
<td>Not equal to</td>
</tr>
</tbody>
</table>
operand
An integer between -2147483647 and 2147483647.

For example, you can define a successful job as a job that ends with a return code less than or equal to 3 as follows:
\[ rccondsucc \ "(RC <= 3)" \]

Boolean expression
Specifies a logical combination of comparison expressions. The syntax is:
\[ comparison_expression \ operator \ comparison_expression \]

comparison_expression
The expression is evaluated from left to right. You can use parentheses to assign a priority to the expression evaluation.

operator
Logical operator. It can have the following values:

<table>
<thead>
<tr>
<th>Example</th>
<th>Operator</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>expr_a and expr_b</td>
<td>And</td>
<td>TRUE if both expr_a and expr_b are TRUE.</td>
</tr>
<tr>
<td>expr_a or expr_b</td>
<td>Or</td>
<td>TRUE if either expr_a or expr_b is TRUE.</td>
</tr>
<tr>
<td>Not expr_a</td>
<td>Not</td>
<td>TRUE if expr_a is not TRUE.</td>
</tr>
</tbody>
</table>

For example, you can define a successful job as a job that ends with a return code less than or equal to 3 or with a return code not equal to 5, and less than 10 as follows:
\[ rccondsucc \ "((RC<3) OR ((RC>5) AND (RC<10)))" \]

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.

after [workstation#]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:

- 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.

`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 50.

**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 42), 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 bkp on fr at 20:00 :
cpu1#jbk1
cpu2#jbk2
    needs 1 tape
cpu3#jbk3
    follows jbkl
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
keyjob

The keyjob keyword is used to mark a job as key or critical in both the database and daily plan and for monitoring by applications, such as Tivoli Business Systems Manager. See the IBM Tivoli Workload Scheduler Planning and Installation Guide for information about enabling the key flag mechanism.

Examples
The following example

SCHEDULE cpu1#sched1
ON everyday
KEYSCHED
AT 0100
cpu1#myjob1 KEYJOB
END
keysched

The keysched keyword is used to mark a job stream as key or critical in both the database and daily plan and for monitoring by applications, such as Tivoli Business Systems Manager. See the IBM Tivoli Workload Scheduler Planning and Installation Guide for information about enabling the key flag mechanism.

Synopsis
keysched

Examples
The following example:
SCHEDULE cpu1#sched1
ON everyday
KEYSCHED
AT 0100
cpu1#myjob1 KEYJOB
END
limit

Limits the number of jobs that can run 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 run 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.

**Note:** The number of jobs and schedules using a resource at any one time cannot exceed 32.

\(wkstation\) Specifies the name of the workstation on which the resource is locally 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.

Examples

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

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

The \texttt{jlimit} resource has been defined with two available units. The following example allows no more than two jobs to run concurrently in job stream \texttt{sked4}:

```
schedule sked4 on mo, we, fr:
    joba needs 1 jlimit
    jobb needs 1 jlimit
    jobc needs 2 jlimit \texttt{\<runs alone>}
    jobd needs 1 jlimit
end
```
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 91 for more information.

**Synopsis**

```
on [date | day | calendar | request][,...] [fdignore | fdnext | fdprev][,...]  
on [date | day | calendar | request][,...] [fdignore | fdnext | fdprev][,...]  
```

**Arguments**

- **date** Specifies a date in the format, \textit{mm/dd/yy}.
- **day** A day of the week. You can specify one or more of the following:
  - \texttt{mo} Monday
  - \texttt{tu} Tuesday
  - \texttt{we} Wednesday
  - \texttt{th} Thursday
  - \texttt{fr} Friday
  - \texttt{sa} Saturday
  - \texttt{su} Sunday
- **weekdays** Everyday except Saturday and Sunday
- **everyday** Every day of the week
- **workdays** Can be one of the following:
  - If you specified a freedays calendar, workdays are \textit{everyday} excluding \textit{saturday} and \textit{sunday} (unless you specified \texttt{-sa} or \texttt{-su} along with the freedays keyword) and excluding all the dates of the specified freedays calendar.
  - If you did not specify a freedays calendar, workdays are \textit{everyday} excluding \textit{saturday} and \textit{sunday} and excluding all the dates of the \textit{holidays} calendar.
- **freedays** The days marked in the freedays calendar, if you specified one.
- **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:
  - \textit{n} The number of days, weekdays, or workdays.
  - **days** Stands for every day of the week.
  - **weekdays** Stands for every day of the week, except saturday and sunday.
  - **workdays** Stands for every day of the week, except for Saturdays and Sundays (unless otherwise specified with the freedays keyword) and for the dates marked either in a designated freedays calendar or in the holidays calendar.
- **request** Selects the job stream only when requested. This is used for job streams...
that are selected by name rather than date. To prevent a scheduled job stream from being selected for **Jnextday**, change its definition to **ON REQUEST**.

**Note:** When attempting to run a job stream that contains "on request" times, consider that:
- "On request" always takes precedence over "at".
- "On request" never takes precedence over "on".

**freeday rule**
Specifies a rule that must be applied when the selected date falls on a freeday. Can be one of the following:

**fdignore**
Do not run the job stream.

**fdnext** Run the job stream on the nearest workday after the freeday.

**fdprev** Run the job stream on the nearest workday before the freeday.

**Usage Notes**
You can define multiple instances of the **on** keyword for the same job stream. Each instance is equivalent to a run cycle to which you can associate a freeday rule.

Multiple **on** instances must be consecutive within the job stream definition.

Each instance of the keyword can contain any of the values allowed by the **on** syntax.

You must specify the **on** keyword at least once in the definition of a job stream.

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

```bash
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:

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

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

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

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

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

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

```bash
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

Select job stream **sked1** to run all Mondays, Fridays, and on 29/12/2001. If Mondays and 29/12/2001 are freedays, run the job stream on the nearest following workday. If Fridays are freedays, run the job stream on the nearest preceding day. In this example, the freedays are Saturdays, Sundays, and all the dates listed in the default HOLIDAYS calendar. Workdays are all days from Monday to Friday as long as they are not listed in HOLIDAYS.

```
schedule sked1
  on mo, 29/12/2001 fdnext
  on fr fdprev
```
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 IBM Tivoli Workload Scheduler parameters as part or all of the file name string. If you use a parameter, it must be enclosed in carets (^).

The file name must be fully qualified for all workstation types with the exception of extended agents (XAs), where this is not a requirement.

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

On Windows, 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 its size is greater than zero.
- `w %p`  True if the file exists and is writable.

On both UNIX and Windows, 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 the path of the file and the qualifiers cannot exceed 120 characters, and the name of the file 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 \texttt{job6}:

\texttt{job6} opens "/usr/lib/cron/FIFO"(-p \%p)

The following example checks to see that file \texttt{d:\work\john\execit1} on workstation \texttt{dev3} exists and is not empty, before running job \texttt{job2}:

\texttt{job2} opens \texttt{dev3#d:\work\john\execit1"(notempty)}

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

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

**Security for test(1) Commands:** On UNIX, a special security feature prevents unauthorized use of other commands in the qualifier. For example, the file below contains a command in the qualifier:

\texttt{/users/xpr/hp3000/send2(-n \"ls /users/xpr/hp3000/m*\" -o -r \%p)}

If the qualifier contains another command, the following checks are made:

1. The Local Option jm no root must be set to no.

2. In the security file, the user documenting the schedule or adding the Open Files dependency with a Conman adddep command, must have submit access to a job with the following attributes:

   \begin{itemize}
   \item \texttt{name=cmdstest.fileeq}
   \item \texttt{logon=root}
   \item \texttt{jcl=the path of the opens files}
   \item \texttt{cpu=the CPU on which the opens files reside}
   \end{itemize}

Note that \texttt{cmdstest} and \texttt{fileeq} do not exist.
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
priority 50
: job1 priority 15
job2 priority 10
end

schedule sked2 on tu
priority 10
: joba priority 60
jobb priority 50
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 backslash 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 backslash 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, dashes, and underscores. It can contain up to 16 characters.

- **on**
  Specifies when, or how often, the job stream is selected for execution. See “on” on page 105 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]] [onuntil action]

Arguments

- **time**: Specifies the time of day. The possible values are 0000 through 2359.
- **timezone tz tzname**: Specifies the time zone to be used when computing the time. See Appendix B, “Managing time zones,” on page 319 for time zone names. The default is the time zone of the workstation on which the job or job stream is launched.
- **+n day[s]**: Specifies an offset, in days, from the scheduled date and time.

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.

- **onuntil action**: Specifies the action to be taken on a job or job stream whose until time has expired, but the job or job stream has not yet started. The following are the possible values of the action parameter:
  - **suppr**: The final job stream state is HOLD if the job stream contains at least one every job. Otherwise the final state is calculated using the normal rules and the jobs with the option onuntil suppr are considered in SUCC state when the until time occurs, even if their dependencies have actually not been released.
  - **cont**: The job or job stream runs when all necessary conditions are met and a notification message is written to the log when the until time elapses.
  - **canc**: A job or job stream is cancelled when the until time specified expires. Any job or job stream that was dependent on the completion of a job or job stream that was cancelled, will run because the dependency no longer exists.

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 **sked1** at 5:00 p.m., when its "until" time expires:
```
schedule sked1 until 1700 onuntil cont
```

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

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
```

```
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 to 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
```

The following example launches job stream `sked8` on weekdays at 4:00 p.m. and should complete running by 5 p.m. If the job stream is not completed by 5 p.m., it is considered a late job stream. The jobs are to be launched as follows: job1 runs at 4 p.m., or at the latest, 4:20 p.m., at which time, if job1 has not yet started, a notification message is written to the log and it starts running. Job 2 runs at 4:30 p.m. or at the latest 4:50 p.m., at which time, if job2 has not yet started, it is cancelled.

```
schedule sked8 on weekdays at 1600 deadline 1700 :
    job1 at 1600 until 1620 on until cont
    job2 at 1630 until 1650 on until canc
end
```

The following example launches job stream `sked01`. When the `until` event occurs, the job stream `sked02` is run because the job stream `sked01` is placed in SUCC state. The job stream `sked03`, instead, is not run because it has a punctual time dependency on job `job01` and this dependency has not been released.

```
SCHEDULE sked01 on everyday:
    job01 until 2035 on until suppr
end

SCHEDULE sked02 on everyday follows sked01.0 :
    job02
end

SCHEDULE sked03 on everyday follows sked01.JTEST :
    job03
END
```
Chapter 5. Conman reference

The IBM Tivoli Workload Scheduler production plan environment is managed with the Conman command line 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 runs the sj and sp commands, and then quits:

conman"sj&sp"

In the following example, Conman runs 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.

Running system commands

When a system command is entered using a pipe or a system command prefix (: or !), it is run 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 MAESTRO_LINES and MAESTRO_COLUMNS. If either is not set, the standard shell variables, LINES and COLUMN, are used. The variables can be set as follows:

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

$MAESTRO_COLUMNS
Specifies the number of characters per line. The default is 80.

$MAESTRO_OUTPUT_STYLE
Specifies 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 long names are truncated to eight characters followed by a plus sign (+).

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:

$MAESTRO_LP
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 run whether or not output is generated.

| | command Pipes output to a system command or process. The system command is not run if there is no output.

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

Selecting the conman command prompt

The conman command prompt is, by default, a percent sign (%). This is defined in the TWS/home/localopts file. The default command prompt is a dash (-). To select a different prompt, edit the conman prompt option in the localopts file and change the dash. The prompt can be up to ten characters long, not including the required trailing pound sign (#).

#---------------------------------------#
# Custom format attributes             #
# date format = 1  # The possible values are 0-ymd, 1-mdy,
# 2-dmy, 3-NLS.  #
# composer prompt = -                    #
# conman prompt = %                      #
# switch sym prompt = %<>%               #
#---------------------------------------#

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 alphanumeric character.

%  Replaces one numeric character.

### Delimiters and special characters

The following characters have special meanings in Conman commands:

<table>
<thead>
<tr>
<th>Char.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>&amp;</td>
<td>Command delimiter. See “Running conman” on page 115.</td>
</tr>
<tr>
<td>+</td>
<td>A delimiter used to select objects for commands. It adds an attribute the object must have. For example: <code>sked1.0+priority=0</code></td>
</tr>
<tr>
<td>~</td>
<td>A delimiter used to select objects for commands. It adds an attribute the object must not have. For example: <code>sked1.0~priority=0</code></td>
</tr>
<tr>
<td>;</td>
<td>Argument delimiter. For example: <code>;info;offline</code></td>
</tr>
<tr>
<td>,</td>
<td>Repetition and range delimiter. For example: <code>state=hold,sked,pend</code></td>
</tr>
<tr>
<td>=</td>
<td>Value delimiter. For example: <code>state=hold</code></td>
</tr>
<tr>
<td>:!</td>
<td>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: <code>!!ls</code> or <code>:ls</code></td>
</tr>
<tr>
<td>&lt;&lt;&lt;</td>
<td>Comment brackets. For example: <code>sj @&lt;&lt;comment&gt;&gt;</code></td>
</tr>
<tr>
<td>*</td>
<td>Comment prefix. The prefix must be the first character on a command line or following a command delimiter. For example: <code>*comment</code> or <code>sj &amp; *comment</code></td>
</tr>
<tr>
<td>&gt;</td>
<td>Redirects command output to a file and overwrites the contents of the file. If the file does not exist, it is created. For example: <code>sj&gt; joblist</code></td>
</tr>
<tr>
<td>&gt;&gt;</td>
<td>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: <code>sj &gt;&gt; joblist</code></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### List of commands

The following table lists the Conman 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.

<table>
<thead>
<tr>
<th>Command</th>
<th>Short Form</th>
<th>Description</th>
<th>Type¹</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>adddep</td>
<td>adj</td>
<td>Adds job or job stream dependencies.</td>
<td>M,F</td>
<td>135, 137</td>
</tr>
<tr>
<td>altpass</td>
<td></td>
<td>Alters a User object definition password.</td>
<td>M,F</td>
<td>139</td>
</tr>
<tr>
<td>altpri</td>
<td>ap</td>
<td>Alters job or job stream priorities.</td>
<td>M,F</td>
<td>140</td>
</tr>
<tr>
<td>cancel</td>
<td>cj</td>
<td>Cancels a job or a job stream.</td>
<td>M,F</td>
<td>148, 143</td>
</tr>
<tr>
<td>confirm</td>
<td></td>
<td>Confirms job completion.</td>
<td>M,F</td>
<td>145</td>
</tr>
<tr>
<td>console</td>
<td></td>
<td>Assigns the IBM Tivoli Workload Scheduler console.</td>
<td>M,F,A</td>
<td>146</td>
</tr>
<tr>
<td>continue</td>
<td></td>
<td>Ignores the next error.</td>
<td>M,F,A</td>
<td>147</td>
</tr>
<tr>
<td>deldep</td>
<td>ddj</td>
<td>Deletes job or job stream dependencies.</td>
<td>M,F</td>
<td>165, 150</td>
</tr>
<tr>
<td>display</td>
<td>df</td>
<td>Displays files, jobs, and job streams.</td>
<td>M,F,A²</td>
<td>152</td>
</tr>
<tr>
<td>display</td>
<td>dj</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>display</td>
<td>ds</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>exit</td>
<td></td>
<td>Terminates Conman.</td>
<td>M,F,A</td>
<td>153</td>
</tr>
<tr>
<td>fence</td>
<td></td>
<td>Sets IBM Tivoli Workload Scheduler job fence.</td>
<td>M,F,A</td>
<td>154</td>
</tr>
<tr>
<td>help⁵</td>
<td></td>
<td>Displays command information.</td>
<td>M,F,A</td>
<td>155</td>
</tr>
<tr>
<td>kill</td>
<td></td>
<td>Stops an executing job.</td>
<td>M,F</td>
<td>156</td>
</tr>
<tr>
<td>limit</td>
<td>lc</td>
<td>Changes a workstation or job stream job limit.</td>
<td>M,F,A³</td>
<td>157, 158</td>
</tr>
<tr>
<td>link</td>
<td>lk</td>
<td>Opens workstation links.</td>
<td>M,F,A</td>
<td>159</td>
</tr>
<tr>
<td>listsym</td>
<td></td>
<td>Displays a list of Symphony log files.</td>
<td>M,F</td>
<td>161</td>
</tr>
<tr>
<td>recall</td>
<td>rc</td>
<td>Displays prompt messages.</td>
<td>M,F</td>
<td>162</td>
</tr>
<tr>
<td>redo</td>
<td></td>
<td>Edits the previous command.</td>
<td>M,F,A</td>
<td>163</td>
</tr>
<tr>
<td>release</td>
<td>rj</td>
<td>Releases job or job stream dependencies.</td>
<td>M,F</td>
<td>141, 167</td>
</tr>
<tr>
<td>reply</td>
<td></td>
<td>Replies to prompt message.</td>
<td>M,F</td>
<td>169</td>
</tr>
<tr>
<td>rerun</td>
<td>rr</td>
<td>Reruns a job.</td>
<td>M,F</td>
<td>170</td>
</tr>
<tr>
<td>resource</td>
<td></td>
<td>Changes the number of resource units.</td>
<td>M,F</td>
<td>173</td>
</tr>
<tr>
<td>setsym</td>
<td></td>
<td>Selects a Symphony log file.</td>
<td>M,F</td>
<td>174</td>
</tr>
<tr>
<td>showcpus</td>
<td>sc</td>
<td>Displays workstation and link information.</td>
<td>M,F,A</td>
<td>175</td>
</tr>
<tr>
<td>showdomain</td>
<td></td>
<td>Displays domain information.</td>
<td>M,F,A</td>
<td>179</td>
</tr>
<tr>
<td>showfiles</td>
<td>sf</td>
<td>Displays information about files.</td>
<td>M,F</td>
<td>180</td>
</tr>
<tr>
<td>showjobs</td>
<td>sj</td>
<td>Displays information about jobs.</td>
<td>M,F</td>
<td>182</td>
</tr>
<tr>
<td>showprompts</td>
<td>sp</td>
<td>Displays information about prompts.</td>
<td>M,F</td>
<td>190</td>
</tr>
<tr>
<td>Command</td>
<td>Short Form</td>
<td>Description</td>
<td>Type¹</td>
<td>Page</td>
</tr>
<tr>
<td>------------------</td>
<td>------------</td>
<td>-------------------------------------------</td>
<td>-------</td>
<td>------</td>
</tr>
<tr>
<td>showresources</td>
<td>sr</td>
<td>Displays information about resources.</td>
<td>M,F</td>
<td>192</td>
</tr>
<tr>
<td>showschedules</td>
<td>ss</td>
<td>Displays information about job streams.</td>
<td>M,F</td>
<td>194</td>
</tr>
<tr>
<td>shutdown</td>
<td></td>
<td>Stops IBM Tivoli Workload Scheduler production processes.</td>
<td>M,F,A</td>
<td>197</td>
</tr>
<tr>
<td>start</td>
<td></td>
<td>Starts IBM Tivoli Workload Scheduler production processes.</td>
<td>M,F,A</td>
<td>198</td>
</tr>
<tr>
<td>status</td>
<td></td>
<td>Displays IBM Tivoli Workload Scheduler production status.</td>
<td>M,F,A</td>
<td>200</td>
</tr>
<tr>
<td>stop</td>
<td></td>
<td>Stops IBM Tivoli Workload Scheduler production processes.</td>
<td>M,F,A</td>
<td>201</td>
</tr>
<tr>
<td>stop ;progressive</td>
<td></td>
<td>Stops IBM Tivoli Workload Scheduler production processes hierarchically.</td>
<td></td>
<td>203</td>
</tr>
<tr>
<td>submit</td>
<td>sbd l</td>
<td>Submits a command, file, job, or job stream.</td>
<td>M,F,A</td>
<td>204, 206, 208, 210</td>
</tr>
<tr>
<td></td>
<td>sbf l</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbj l</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>sbs</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>switchmgr</td>
<td></td>
<td>Switches the domain manager.</td>
<td>M,F</td>
<td>212</td>
</tr>
<tr>
<td>sys-command</td>
<td></td>
<td>Sends a command to the system.</td>
<td>M,F,A</td>
<td>213</td>
</tr>
<tr>
<td>tellop</td>
<td>to</td>
<td>Sends a message to the console.</td>
<td>M,F,A</td>
<td>214</td>
</tr>
<tr>
<td>unlink</td>
<td></td>
<td>Closes workstation links.</td>
<td>M,F,A</td>
<td>215</td>
</tr>
<tr>
<td>version</td>
<td>v</td>
<td>Displays Conman’s command line program banner.</td>
<td>M,F,A</td>
<td>217</td>
</tr>
</tbody>
</table>

1. Workstation types: domain managers (M), fault-tolerant agents (F), standard agents (A).
2. You can display only files on a standard agent.
3. You can change the limit of only jobs launched on a standard agent workstation.
4. You can use `submit job (sbf)` 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 supported Windows platforms.

**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

[workstation#] [jobstream.job | jobnumber] [+ | ~jobqualifier[...]]

or:

netagent::[workstation#] jobstream.job

Arguments

workstation
  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.”

netagent
  Specifies the name of the scheduler network agent that interfaces with the
  remote scheduler network containing the target job. The two colons (::) are
  a required delimiter. Wildcard characters are permitted. For more
  information refer to Chapter 9, “The Network Agent reference,” on page 289.

Job Qualifiers

Job qualifiers specify attributes of jobs to be acted on by a command. They are
prefixed by a + or a ~. A + means that jobs with the attribute qualify for the
command. A ~ means that jobs with the attribute are excluded from the
commanded.

Job qualifier keywords can be abbreviated to as few leading characters as needed
to distinguish one from another.

at[=time | lowtime | hightime | lowtime,hightime [absolute | abs]]
  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]

  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 Appendix B, “Managing time zones,” on page 319 for valid names.

time conforms to the following rules:

- When hhmm is earlier than the current time, the start time is tomorrow; when hhmm is later than the current time, the start time is today.
- When hhmm is greater than 2400, it is divided by 2400. Of the result, the whole part represents the number of + days, while the decimal part represents the time.

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.

absolute | abs

Specifies the scheduled start time for the current day. This keyword can be used only with timezone | tz tzname and hhmm. It conforms to the following rules:

- When hhmm is earlier than the current time, the start time is immediately.
- When hhmm is later than the current time, the start time is the time specified by hhmm of the current day.
- When hhmm exceeds 2400, it is divided by 2400 to obtain the calculated day and time. Of the division result, the whole part represents the calculated day (number of + days), while the decimal part represents the calculated time:
  - If the calculated time is earlier than the current time, the start time is one day before the calculated day at the calculated time.
  - If the calculated time is later than the current time, the start time is the calculated day and time.

For example, if the current time is 1200 and a job is submitted at hhmm=3500 (2400 + 1100), with the abs keyword specified, the job is launched at 1100 of the following day. Instead, without specifying the abs keyword, the job is launched at 1100, two days later.

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.

deadline[=time | lowtime, | hightime | lowtime|hightime]

Specifies the time within which a job must complete.

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

hhmm The hour and minute.
+$n$ days
An offset in days from the scheduled deadline time.

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

timezone | tz tzname
Specifies the time zone to be used when computing the deadline. See Appendix B, “Managing time zones,” on page 319 for time zone names. The default is the time zone of the workstation on which the job or job stream is launched.

lowtime
Specifies the lower limit of a time range, expressed in the same format as $time$. Jobs are selected that have scheduled deadline 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 deadline on or before this time.

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 hours and minutes ($hhmm$).

lowrate Specifies the lower limit of a rate range, expressed in the same format as $rate$. Jobs are selected that have repetition 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 repetition 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 Appendix B, “Managing time zones,” on page 319 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[;nocheck];[wait=time]]

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

netagent

Specifies the name of the scheduler network agent that interfaces with the remote scheduler network containing the prerequisite job. Wildcard characters are permitted. For more information refer to Chapter 9, “The Network Agent reference,” on page 289.

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. Do not specify the job name using wildcard characters for a follows dependency.

nocheck

Is valid only for the sbd, sbj, and sbs commands. Conman does not check for the existence of prerequisite jobs in the symphony. When Batchman processes the submit command, it adds the arguments of the submit command to the symphony (such as the job to be submitted) including the dependencies on the prerequisite jobs that exist in the symphony. For those prerequisite jobs that do not exist in the symphony file, Batchman prints a warning in the stdlist.

wait=time

Is valid only for the sbd, sbj, and sbs commands. For the time interval specified, conman checks every second for the existence of the prerequisite jobs in the symphony file. Conman checks until the prerequisite jobs are found or the time interval specified for the wait keyword expires. When all prerequisite jobs are found, Batchman processes the submit command. If even one prerequisite job is not found in the symphony and the time interval expires, Conman does not process the submit command and returns an error message. The maximum number of seconds is 1200.

If you specify nocheck together with wait, when the time interval expires, Batchman processes the submit command even if one prerequisite job does not exist in the symphony. The job to be submitted and the prerequisite jobs that exist in the symphony are added to the symphony.

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#]resource=]name

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. A file dependency occurs when a job or job stream is dependent on the existence of one or more files before it can begin execution.

**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. A file dependency occurs when a job or job stream is dependent on the existence of one or more files before it can begin execution.

**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 Appendix B,
“Managing time zones,” on page 319 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.
abep 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.

wait The job is in the wait state. (Extended agent only)

\[until=time \mid lowtime, \mid hightime \mid lowtime,hightime \mid \text{absolute} \mid \text{abs}]\]

Selects or excludes jobs based on their scheduled end time.

time Specifies the scheduled end time expressed as follows:

\[hhmm[+n \text{ days} \mid \text{date}] [\text{timezone} \mid \text{tz} \mid \text{tzname}]\]

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 \mid tz \mid \text{tzname}

The name of the time zone of the job. See Appendix B, “Managing time zones,” on page 319 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.

absolute \mid \text{abs}

Specifies the scheduled start time for the current day. This keyword can be used only with timezone \mid \text{tz} \mid \text{tzname} and hhmm. It conforms to the following rules:

- When hhmm is earlier than the current time, the start time is immediately.
- When hhmm is later than the current time, the start time is the time specified by hhmm of the current day.
- When hhmm exceeds 2400, it is divided by 2400 to obtain the calculated day and time. Of the division result, the whole part represents the calculated day (number of + days), while the decimal part represents the calculated time:
  - If the calculated time is earlier than the current time, the start time is one day before the calculated day at the calculated time.
  - If the calculated time is later than the current time, the start time is the calculated day and time.
For example, if the current time is 1200 and a job is submitted at hhmm=3500 (2400 + 1100), with the abs keyword specified, the job is launched at 1100 of the following day. Instead, without specifying the abs keyword, the job is launched at 1100, two days later.

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

[workstation] jobstream [+ | ~]jobstreamqual[...]]

Arguments

workstation
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 [absolute | abs]]
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 Appendix B, “Managing time zones,” on page 319 for valid names.

absolute | abs
Specifies the scheduled start time for the current day. This keyword can be used only with timezone | tz tzname and hhmm. It conforms to the following rules:

• When hhmm is earlier than the current time, the start time is immediately.
• When hhmm is later than the current time, the start time is the time specified by hhmm of the current day.
• When \textit{hhmm} exceeds 2400, it is divided by 2400 to obtain the calculated day and time. Of the division result, the whole part represents the calculated day (number of + days), while the decimal part represents the calculated time:
  – If the calculated time is earlier than the current time, the start time is one day before the calculated day at the calculated time.
  – If the calculated time is later than the current time, the start time is the calculated day and time.

For example, if the current time is 1200 and a job is submitted at \textit{hhmm}=3500 (2400 + 1100), with the \textit{abs} keyword specified, the job is launched at 1100 of the following day. Instead, without specifying the \textit{abs} keyword, the job is launched at 1100, two days later.

\textit{lowtime}

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

\textit{hightime}

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

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

\textit{carriedforward}

Selects or excludes job streams that were carried forward.

\textit{carryforward}

Selects or excludes job streams that were scheduled using the \textit{carryforward} keyword.

\textit{finished[=time] lowtime, hightime lowtime, hightime]}

Selects or excludes job streams based on whether or not they have finished.

\textit{time}

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

\textit{hhmm [date] [timezone | tz tzname]}

\textit{hhmm} The hour and minute.

\textit{date} The next occurrence of \textit{hhmm} on date, expressed as \textit{mm/dd/yy}.

\textit{timezone | tz tzname}

The name of the time zone of the job stream. See \textbf{Appendix B, “Managing time zones,” on page 319} for valid names.

\textit{lowtime}

Specifies the lower limit of a time range, expressed in the same format as \textit{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(nocheck][;wait=time]]
  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 scheduler network containing the prerequisite job or job
  stream. Wildcard characters are permitted. For more information
  about network agents, refer to Chapter 9, “The Network Agent
  Reference,” on page 289.

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.

nocheck
  Is valid only for the sbd, sbj, and sbs commands. Conman does not
  check for the existence of prerequisite jobs in the symphony. When
  Batchman processes the submit command, it adds the arguments of
  the submit command to the symphony (such as the job to be
  submitted) including the dependencies on the prerequisite jobs that
  exist in the symphony. For those prerequisite jobs that do not exist
  in the symphony file, Batchman prints a warning in the stdlist.

wait=time
  Is valid only for the sbd, sbj, and sbs commands. For the time
  interval specified, conman checks every second for the existence of
  the prerequisite jobs in the symphony file. Conman checks until
  the prerequisite jobs are found or the time interval specified for the
  wait keyword expires. When all prerequisite jobs are found,
  Batchman processes the submit command. If even one prerequisite
  job is not found in the symphony and the time interval expires,
  Conman does not process the submit command and returns an
  error message. The maximum number of seconds is 1200.

  If you specify nocheck together with wait, when the time interval
  expires, Batchman processes the submit command even if one
  prerequisite job does not exist in the symphony. The job to be
  submitted and the prerequisite jobs that exist in the symphony are
  added to the symphony.

  If follows is used alone, job streams are selected or excluded if they have
  any follows dependency.
**limit**\[=\text{limit} \mid \text{lowlimit} \mid \text{highlimit} \mid \text{lowlimit,highlimit}\]

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**\[=\text{wkstation\#} \mid \text{resourcename}\]

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**\[=\text{wkstation\#} \mid \text{filename\{(qualifier\)}}\]

Selects or excludes job streams based on whether or not they have a file dependency. A file dependency occurs when a job or job stream is dependent on the existence of one or more files before it can begin execution.

- **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. A file dependency occurs when a job or job stream is dependent on the existence of one or more files before it can begin execution.

**priority**\[=\text{pri} \mid \text{lowpri} \mid \text{highpri} \mid \text{lowpri,highpri}\]

Selects or excludes job streams based on their priorities.

- **pri** Specifies the priority value. You can enter 0 through 99, hi or go.
Specifies the lower limit of a priority range. Job streams are selected with priorities equal to or greater than this value.

highpri Specifications 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 name]

hhmm The hour and minute.

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

timezone tz name The name of the time zone of the job stream. See Appendix B, “Managing time zones,” on page 319 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.

add The schedule has just been submitted.

exec The job stream is executing.

hold The job stream is awaiting dependency resolution.
The job stream is ready to launch, and all dependencies resolved.

Execution is interrupted. No jobs are launched without operator intervention.

The job stream completed successfully.

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 Appendix B, “Managing time zones,” on page 319 for valid names.

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.

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.

Specifies the scheduled start time for the current day. This keyword can be used only with $timezone\ |\ tz\ \tzname$ and $hhmm$. It conforms to the following rules:

• When $hhmm$ is earlier than the current time, the start time is immediately.

• When $hhmm$ is later than the current time, the start time is the time specified by $hhmm$ of the current day.

• When $hhmm$ exceeds 2400, it is divided by 2400 to obtain the calculated day and time. Of the division result, the whole part represents the calculated day (number of $+\ days$), while the decimal part represents the calculated time:
  – If the calculated time is earlier than the current time, the start time is one day before the calculated day at the calculated time.
  – If the calculated time is later than the current time, the start time is the calculated day and time.

For example, if the current time is 1200 and a job is submitted at $hhmm=3500$ (2400 + 1100), with the abs keyword specified, the job is launched at 1100 of the following day. Instead, without specifying the abs keyword, the job is launched at 1100, two days later.
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 job streams, and the term cpu refers to workstations.

**Conman command processing**

When using various commands, you should be aware of the way in which conman commands are processed.

When you change the dependencies of a job or a schedule in the symphony, conman stops the dependencies from being inserted by preventing the relevant command being forwarded to Batchman. However, it is possible that it might appear that such a change has not completed successfully, despite the fact that it has done so.

The reason for this is because conman does not write directly to the symphony, but asks Batchman to do so instead. Batchman receives a sequence of events as they occur on each system in the network, and in the same sequence as they occur. As Batchman processes these events, the symphony states change around the scheduler network. If a Batchman should be busy for any reason, conman commands are queued and the symphony is not updated until such time as the Batchman reads the commands from the mailbox and processes them. In this way, a conman command could perform different actions, depending when it is processed.

In addition, when Batchman processes the event, the operator is not notified. As a result, you could delete a dependency and it might appear that it had not been deleted because Batchman was busy. If you run the command again, the deletion might have been successful, even though a message is displayed saying that the command has been forwarded to Batchman.
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**

```bash
adj jobselect[dependency;...] [noask]
```

**Arguments**

- `jobselect`
  
  See “Selecting jobs in commands” on page 120.

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

  - `at=hhmm(timezone | tz tzname)[+n days | mm/dd/yy] | [absolute | abs]`
  
  - `confirmed`
  
  - `deadline=time [timezone | tz tzname][+n days | mm/dd/yy]`
  
  - `every=rate`
  
  - `follows=[netagent::][wkstation#][stream::job | @] | job[,...]`
  
  - `needs=[num] [wkstation#][resource[,...]]`
  
  - `opens=[wkstation#]["filename"[(qualifier)][,...] priority=[pri | hi | go]`
  
  - `prompt="[ | !]text" | promptname[,...]`
  
  - `until time [timezone | tz tzname][+n days] | [absolute | abs] [onuntil action]`

- `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 `TWS/home/mozart` directory on the master domain manager.

You cannot use this command to add a resource or a prompt as dependencies unless they are already referenced by a job or a schedule in the Symphony file.

**Examples**

To add a resource dependency to job `job3` in job stream `sked9`, run the following command:

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

To add a file dependency, and an `until` time to job `job6` in job stream `sked2`, run the following command:

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

For the equivalent Job Scheduling Console task, see "Creating Dependencies between Jobs" and "Adding External Dependencies to a Job Stream" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
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

adddep sched

Arguments

jstreamselect

See “Selecting job streams in commands” on page 128.

dependency

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

at=hhmm[timezone | tz tzname][-n days | mm/dd/yy | [absolute | abs]
carryforward
deadline=me time [timezone | tz tzname][-n day[s | mm/dd/yy]
follows=[netagent::][workstation#][jstream[job | @] | job[,...]
limit=limit
needs=[num] [workstation#][resource[,...]
opens=[workstation#]["filename"[(qualifier)[,...]] priority=[priority | hi | go]
prompt="[[: | !]text] promptname[,...]
until [time | timezone | tz tzname][-n day[s | [absolute | abs]] [onuntil action]

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 value for limit, the value defaults to 0.
- 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 TWShome/mozart directory on the master domain manager.
- You cannot use this command to add a resource or a prompt as dependencies unless they are already referenced by a job or a schedule in the Symphony file.

Examples

To add a prompt dependency to job stream sked3, run the following command:

adddep sched=sked3;prompt=msg103

To add a follows dependency and a job limit to job stream sked4, run the following command:

ads sked4;follows=sked3;limit=2
See Also
For the equivalent Job Scheduling Console task, see "Creating Dependencies within a Job Stream" and "Creating Dependencies between Job Streams" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
**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 47.

**Examples**

To change the password of user **jim** on workstation **mis5** to **giraffe**, run the following command:

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

To change the password of user **jim** on workstation **mis5** to **giraffe** without displaying the password, run the following command:

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

**See Also**

For the equivalent Job Scheduling Console task, see "Changing Windows User Passwords" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.*
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 120.

jstreamselect

See “Selecting job streams in commands” on page 128.

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

To change the priority of the balance job in job stream glmonth, run the following command:

altpri glmonth.balance;55

To change the priority of all jobs in job stream mis5, run the following command:

ap mis5.0;25

To change the priority of job stream glmonth, run the following command:

altpri glmonth;10

To change the priority of job stream mis5, run the following command:

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 120.
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 run. 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 292.

**Examples**

To cancel job **report** in job stream **apwkly** on workstation **site3**, run the following command:

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

To cancel job **setup** in job stream **mis5**, if it is not in the **abend** state, run the following command:

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

To cancel job **job3** in job stream **sked3** only after its dependencies are resolved, run the following command:

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

**See Also**

For the equivalent Job Scheduling Console task, see "Deleting a Job from a Job Stream" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
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 128.
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 marked [Cancel Pend], 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

To cancel job stream sked1 on workstation site2, run the following command:
cancel site2#sked1

To cancel job stream mis2 if it is in the stuck state, run the following command:
cs mis2+state=stuck
To cancel job stream `sked3` only after its dependencies are resolved, run the
following command:

```
cs sked3;pend
```

**See Also**

For the equivalent Job Scheduling Console task, see "Deleting Job Streams" in the
*IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.*
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 120.
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 87. For more information about external jobs, see “Internetwork dependencies” on page 292.

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

<table>
<thead>
<tr>
<th>Initial Job State</th>
<th>State after confirm ;succ</th>
<th>State after confirm ;abend</th>
</tr>
</thead>
<tbody>
<tr>
<td>ready</td>
<td>no affect</td>
<td>no affect</td>
</tr>
<tr>
<td>hold</td>
<td>no affect</td>
<td>no affect</td>
</tr>
<tr>
<td>exec</td>
<td>succp</td>
<td>abenp</td>
</tr>
<tr>
<td>abenp</td>
<td>succp</td>
<td>no affect</td>
</tr>
<tr>
<td>succp</td>
<td>no affect</td>
<td>no affect</td>
</tr>
<tr>
<td>pend</td>
<td>succ</td>
<td>abend</td>
</tr>
<tr>
<td>done</td>
<td>succ</td>
<td>abend</td>
</tr>
<tr>
<td>succ</td>
<td>no affect</td>
<td>no affect</td>
</tr>
<tr>
<td>abend</td>
<td>succ</td>
<td>no affect</td>
</tr>
<tr>
<td>fail</td>
<td>no affect</td>
<td>no affect</td>
</tr>
<tr>
<td>skel</td>
<td>no affect</td>
<td>no affect</td>
</tr>
<tr>
<td>any external job</td>
<td>succ</td>
<td>abend</td>
</tr>
</tbody>
</table>

Examples
To issue a succ confirmation for job job3 in job stream midstly, run the following command:
confirm midstly.job3;succ

To issue an abend confirmation to job number 234, run the following command:
conf 234;abend
**console**

Assigns the scheduler console and sets the message level.

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

**Synopsis**

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

**Arguments**

- **sess** Sends scheduler console messages and prompts to standard output.
- **sys** Stops sending scheduler console messages and prompts to standard output. This occurs automatically when you exit Conman.
- **msglevel** The level of 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, scheduler control processes write console messages and prompts to standard list files. On UNIX, you can also have them sent to the **syslog** daemon.

**Examples**

To begin writing console messages and prompts to standard output and change the message level to 1, run the following command:

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

To stop writing console messages and prompts to standard output and change the message level to 4, run the following command:

```
cons sys;1=4
```

To display the current state of the console, run the following command:

```
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
To make Conman continue with the rerun command even if the cancel command fails, run the following command:

```
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 120.

- `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[=time | lowtime | hightime | lowtime,hightime]`
  - `confirmed`
  - `deadline[=time[timezone | tz tzname][+n days | mm/dd/yy]]`
  - `every`
  - `follows[=[netagent:[wkstation#]|jstream|job | @] | job[,...]]`
  - `needs[=[num] [wkstation#|resource[,...]]`
  - `opens[=[wkstation#]"filename"[(qualifier)][,...]]`
  - `priority`
  - `prompt[="[[: | !]text” | promptname[,...]]`
  - `until[=time [timezone | tz tzname][+n day[s] [onuntil action]]`

- `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.

In certain circumstances, when you have submitted a `deldep` command, the command might have succeeded even though it is again forwarded to Batchman. For more information, see "Conman command processing" on page 134.

**Examples**

To delete a resource dependency from job `job3` in job stream `sked5`, run the following command:

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

To delete all `follows` dependencies from job `job4` in job stream `sked3`, run the following command:

```
ddj sked3.job4;follows
```
See Also

For the equivalent Job Scheduling Console task, see "Managing Jobs" in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
**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 120.

- **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[=time | lowtime | hightime | lowtime,hightime]`
  - `carryforward`
  - `deadline[=time[timezone | tz tzname][+n days | mm/dd/yy]]`
  - `follows[=netagent::[wkstation#]jstream[job | @] | job[...]]`
  - `limit`
  - `needs[=num] [wkstation#]resource[,...]`
  - `opens[=wkstation#]"filename"[(qualifier)][,...]`
  - `priority`
  - `prompt[="[: | !]text" | promptname[, ...]]`
  - `until[=time [timezone | tz tzname][+n days] [onuntil action]]`

- **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.

In certain circumstances, when you have submitted a **deldep** command, the command might have succeeded even though it is again forwarded to Batchman. For more information, see “Conman command processing” on page 134.

**Examples**

To delete a resource dependency from job stream **sked5**, run the following command:

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

To delete all **follows** dependencies from job stream **sked3**, run the following command:

```
dds sked3;follows
```
See Also
For the equivalent Job Scheduling Console task, see "Managing Job Streams" in the
IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
**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 120. 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 128. The scheduler 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 116.

**Examples**

To display the file `c:\maestro\jclfiles\arjob3`, run the following command:

```
df c:\apps\maestro\jclfiles\arjob3
```

To display the script file for job `job6` in job stream `sked3` offline, run the following command:

```
dj sked3.job6;off
```

To display the job stream definition for job stream `sked9`, run the following command:

```
ds sked9
```

**See Also**

For the equivalent Job Scheduling Console task, see "Managing Jobs" and "Managing Job Streams" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
**exit**

Exits the Conman command line program.

**Synopsis**

```
e
```

**Usage Notes**

When you are in help mode on UNIX, this command returns Conman to command-input mode.

**Examples**

To exit the Conman command line program, run the following command:

```
exit
```

or

```
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 IBM Tivoli 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
To change the job fence on workstation site4, run the following command:

fence site4;20

To change the job fence on the workstation on which you are running Conman, run the following command:

f ;40

To prevent all jobs from being launched by the scheduler on workstation tx3, run the following command:

f tx3;go

To change the job fence to zero on the workstation on which you are running Conman, run the following command:

f ;system

See Also
For the equivalent Job Scheduling Console task, see "Viewing and Modifying Workstation Properties" in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
help

Displays help information about commands. Not available on Windows.

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

To display a list of all Conman commands, run the following command:

help commands

To display information about the fence command, run the following command:

help fence

To display information about the altpri job and altpri sched commands, run the following command:

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 120.

`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 run by a 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**

To kill the job report in job stream apwkly on workstation site3, run the following command:

```
kill site3#apwkly.report
```

To kill job number 456, run the following command:

```
k 456
```
limit cpu

Changes the limit of jobs launched on a standard agent workstation. You must have limit access to the workstation.

Synopsis

lc workstation;limit[numask]

Arguments

workstation  Specifies the name of the standard agent 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 IBM Tivoli 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.

The 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 occurring.

Examples

To change the job limit on workstation site3, run the following command:

limit cpu=site3;25

To change the job limit on the workstation on which you are running Conman, run the following command:

lc ;12

To change the job limit on workstation rx12, run the following command:

lc rx12;6

See Also

For the equivalent Job Scheduling Console task, see "Viewing and Modifying Workstation Properties" in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
**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 128.
- `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**

To change the job limit on all job streams that include `sales` in their name, run the following command:

`limit sched=sales0;4`

To change the job limit on job stream `apwkly`, run the following command:

`ls apwkly;6`

**See Also**

For the equivalent Job Scheduling Console task, see “Managing Job Streams” in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
link

Opens communication links between workstations. In a 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

link [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 the 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 33.

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 provided that the workstation is a domain manager or host. A peer agent in the local domain cannot be linked.
- 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 run by users in various locations in the network.
DM are domain managers and A are agents.

<table>
<thead>
<tr>
<th>Command</th>
<th>Links Opened by User1</th>
<th>Links Opened by User2</th>
<th>Links Opened by User3</th>
</tr>
</thead>
<tbody>
<tr>
<td>link @!@</td>
<td>All links are opened.</td>
<td>DM1-DM2</td>
<td>DM2-A21</td>
</tr>
<tr>
<td></td>
<td>DM1-A11</td>
<td>DM2-A21</td>
<td>DM2-A21</td>
</tr>
<tr>
<td></td>
<td>DM1-A12</td>
<td>DM2-A22</td>
<td>DM2-A22</td>
</tr>
<tr>
<td></td>
<td>DM1-DM2</td>
<td>DM2-DM4</td>
<td>DM2-DM4</td>
</tr>
<tr>
<td></td>
<td>DM1-DM3</td>
<td>DM4-A41</td>
<td>DM4-A42</td>
</tr>
<tr>
<td>link @</td>
<td>DM1-A11</td>
<td>DM1-DM2</td>
<td>DM2-A21</td>
</tr>
<tr>
<td></td>
<td>DM1-A12</td>
<td>DM2-A21</td>
<td>DM2-A21</td>
</tr>
<tr>
<td></td>
<td>DM1-DM2</td>
<td>DM2-A22</td>
<td>DM2-A22</td>
</tr>
<tr>
<td></td>
<td>DM1-DM3</td>
<td>DM2-DM4</td>
<td>DM2-DM4</td>
</tr>
<tr>
<td>link DOMAIN3!@</td>
<td>DM3-A31</td>
<td>Not allowed.</td>
<td>Not allowed.</td>
</tr>
<tr>
<td></td>
<td>DM3-A32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>link DOMAIN4!@</td>
<td>DM4-A41</td>
<td>Not allowed.</td>
<td>Not allowed.</td>
</tr>
<tr>
<td></td>
<td>DM4-A42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>link DM2</td>
<td>DM1-DM2</td>
<td>Not applicable.</td>
<td>DM2-A21</td>
</tr>
<tr>
<td>link A42</td>
<td>DM4-A42</td>
<td>DM4-A42</td>
<td>Not allowed.</td>
</tr>
<tr>
<td>link A31</td>
<td>DM3-A31</td>
<td>Not allowed.</td>
<td>Not allowed.</td>
</tr>
</tbody>
</table>
listsym

Lists the production plan (Symphony) log files.

You must have display access to the Symphony file.

Synopsis

listsym [;offline]

Arguments

display Sends the output of the command to the Conman output device. For information about this device, see "Offline output" on page 116.

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 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

To list the production plan log files, run the following command:

listsym

To list the production plan log files to Conman’s output device, run the following command:

listsym ;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 116.

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

To display pending prompts issued on the workstation on which you are running Conman, run the following command:

recall

or:

rc

To display pending prompts on workstation site3, run the following command:

rc site3

To display pending prompts on all workstations and have the output sent to Conman’s offline device, run the following command:

rc @;offline
redo

Edits and reruns the previous command.

**Synopsis**

redo

**Context**

When you run the `redo` command, Conman displays the previous command, so that it can be edited and rerun. 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** Inserts text before the character above the `i`.
- **r** Replaces one or more characters with text, beginning with the character above the `r`. Replace is implied if no other directive is entered.
- **>** 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** Replaces characters at the end of the line with text.

**Directive Examples**

- **ddd** Deletes the three characters above the `ds`.
- **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**

To insert a character, run the following command:

```
redo
setsm 4
iy
setsym 4
```

To replace a character, run the following command:
redo
setsym 4
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 120.

dependency

The type of dependency. You can specify one of the following. You can use wildcard characters in workstation, jstream, job, resource, filename, and promptname.

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

confirmed

deadline[=time[timezone | tz tzname][+n days | mm/dd/yy]]

every

follows[=netagent::[workstation#|jstream|job | @] | job[,...]]

needs[=num [workstation#|resource[,...]]]

opens[=[workstation#]|filename|(qualifier)][,...]]

priority

prompt[="[; | !|text" | promptname[,...]]

until[=time [timezone | tz tzname][+n day[s] | onuntil action]]

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.

When you release a job from a priority dependency, the job reverts to its original scheduled priority.

Examples

To release job job3 in job stream ap from all of its dependencies, run the following command:

release job=ap.job3

To release job job2 in job stream skedr from all of its opens dependencies, run the following command:

rj skedr.job2;opens
To release all jobs on workstation **site4** from their dependencies on a prompt named **glprmt**, run the following command:

```
rj site4@.0;prompt=glprmt
```

**See Also**

For the equivalent Job Scheduling Console task, see "Releasing a Job Instance from Dependencies" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
**release sched**

Releases job streams from dependencies.

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

**Synopsis**

```bash
rs jstreamselect[[dependency][...]][noask]
```

**Arguments**

- `jstreamselect`  
  See “Selecting job streams in commands” on page 128.
- `dependency`  
  The type of dependency. Specify one of the following. You can use wildcard characters in `wkstation`, `jstream`, `job`, `resource`, `filename`, and `promptname`.

- `at[=time | lowtime | hightime | lowtime,hightime]`
- `carryforward`
- `deadline[=time[timezone | tz tzname][+n days | mm/dd/yy]]`
- `follows[=netagent::[wkstation#]jstream{.job | @} | job[,...]]`
- `limit`
- `needs[=num[ wkstation#]resource[,...]]`
- `opens[=wkstation#]"filename"[(qualifier)][,...]]`
- `priority`
- `prompt[=": | !|text" | promptname[,...]]`
- `until[=time [timezone | tz tzname][+n days] [onuntil action]]`

- `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.

When you release a schedule from a **priority** dependency, the job stream reverts to its original priority.

In certain circumstances, when you have submitted a **deldep** command, the command might have succeeded even though it is again forwarded to batchman. For more information, see “Conman command processing” on page 134.

**Examples**

To release job stream **ap** from all of its dependencies, run the following command:

```bash
release sched=ap
```
To release job stream `sked5` from all of its `opens` dependencies, run the following command:

```
rs sked5;opens
```

To release all job streams on workstation `site3` from their dependencies on job stream `main#sked23`, run the following command:

```
rs site3#@;follows=main#sked23
```

**See Also**

For the equivalent Job Scheduling Console task, see "Releasing a Job Stream Instance from Dependencies" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.*
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

To reply Y to the global prompt arprmt, run the following command:
reply arprmt;y

To reply N to message number 24 on workstation site4, run the following command:
rep site4#24;n
rerun

Reruns a job.

You must have rerun access to the job.

Synopsis

rerun [from=wkstat#job] [at=time] [pri=pri] [noask]

Arguments

jobselect

Specifies the name of one or more jobs. Wildcard characters are permitted.

from=wkstat#job

Specifies the name of a job defined in the scheduler 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 the scheduler 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 Appendix B, “Managing time zones,” on page 319 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 run, 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. For more information, refer to the *IBM Tivoli Workload Scheduler Planning and Installation Guide*.

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 to run 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 run.

```bash
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” on page 237.

**Examples**

To rerun job **job4** in job stream **sked1** on workstation **main**, run the following command:

```
rerun main#sked1.job4
```
To rerun job job5 in job stream sked2 using the job definition for job jobx where the job’s at time is set to 6:30 p.m. and its priority is set to 25, run the following command:

```
rr sked2.job5;from=jobx;at=1830;pri=25
```

To rerun job job3 in job stream sked4 using the job name jstep2, run the following command:

```
rr sked4.job3;step=jstep2
```

**See Also**

For the equivalent Job Scheduling Console task, see "Rerunning a Job Instance" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
resource
Changes the number of total units of a resource.

You must have resource access to the resource.

Synopsis
resource [workstation#]resource;num;[noask]

Arguments
workstation 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
To change the number of units of resource tapes to 5, run the following command:
resource tapes;5

To change the number of units of resource jobslots on workstation site2 to 23, run the following command:
res site2#jobslots;23

See Also
For the equivalent Job Scheduling Console task, see "Managing Resources in the Database" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
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**

```bash
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**

To select production plan archive file 5, run the following command:

```bash
setsym 5
```

To select the current production plan (Symphony), run the following command:

```bash
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 in which the command is run.
- `wkstation` Specifies the name of a workstation. The default is the workstation where the command is run. When no domain and no workstation are specified, the output can be the following:
  - The following command displays all the workstations that are in the domain of the workstation where the command was run, plus all the connected domain managers if the workstation is a domain manager.
    ```
    conman "sc"
    ```
  - The following command displays all the workstations that are in the domain of the workstation where the command was run, without the connected domain managers.
    ```
    conman "sc @"
    ```

- `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 116.

**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
- OTHER

Workstation types are as follows:
- MASTER
- MANAGER
- FTA
- S-AGENT
- X-AGENT

**LIMIT**
The scheduler job limit.

**FENCE**
The scheduler job fence.
DATE TIME
The date and time the scheduler started executing the current production plan (Symphony).

STATE
The state of the workstation’s links. Up to five characters are displayed as follows:

<table>
<thead>
<tr>
<th>L</th>
<th>T</th>
<th>H</th>
<th>X</th>
<th>I</th>
<th>J</th>
</tr>
</thead>
</table>
L  | The primary link is open (linked) to its domain/upper manager. |
T  | The link is TCP/IP. |
H  | The workstation is linked through its host. |
X  | The workstation is linked as an extended agent (x-agent). |
I  | The jobman program has completed startup initialization. |
J  | The jobman program is running. |
W  | The workstation is linked by TCP/IP. |
F  | The workstation is fully linked through primary and all secondary connections. |

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 the IBM Tivoli Workload Scheduler 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] [T]
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.
T  The link is defined as TCP/IP.

ADDR
The TCP port number for the workstation.

NODE
The node name of the workstation.

Examples
1. To display information about the workstation on which you are running
   Conman in the info format, run the following command:
   showcpus ;info
2. To display link information offline for all workstations, run the following command:
   sc 0@1;link;off
3. To display information about the workstation, run the following command:
   showcpus
   If you run this command in an environment when the primary connection of
   the workstation with its domain or upper manager is not active, you receive
   the following output:

   MASTER  360  *WNT MASTER  10  0  12/04/03  13:48  I J  MASTERDM
   FTA1  360  WNT FTA  10  0  12/04/03  13:48  FTI JW  MASTERDM
   FTA2  360  WNT FTA  10  0  12/04/03  13:48  FTI JW  MASTERDM
   FTA3  360  WNT MANAGER  10  0  12/04/03  13:48  LTI JW  DOMAIN1
   FTA4  360  WNT FTA  10  0  12/04/03  13:48  F I J  DOMAIN1
   FTA5  360  WNT FTA  10  0  12/04/03  13:48  F I J  DOMAIN1
   S A1  360  WNT S-AGENT  10  0  12/04/03  13:48  F I J  DOMAIN1
   XA_FTA4  360  OTHR X-AGENT  10  0  12/04/03  13:48  L I J  DOMAIN1
   FTA6  360  WNT MANAGER  10  0  12/04/03  13:48  F I J  DOMAIN2
   FTA7  360  WNT FTA  10  0  12/04/03  13:49  F I J  DOMAIN2

   If you run this command in an environment when the primary connection of
   the workstation with its domain or upper manager is active and at least one
   secondary connection is not active, you receive the following output:

   MASTER  360  *WNT MASTER  10  0  12/04/03  13:48  I J  MASTERDM
   FTA1  360  WNT FTA  10  0  12/04/03  13:48  FTI JW  MASTERDM
   FTA2  360  WNT FTA  10  0  12/04/03  13:48  FTI JW  MASTERDM
   FTA3  360  WNT MANAGER  10  0  12/04/03  13:48  FTI JW  DOMAIN1
   FTA4  360  WNT FTA  10  0  12/04/03  13:48  F I J  DOMAIN1
   FTA5  360  WNT FTA  10  0  12/04/03  13:48  F I J  DOMAIN1
   S A1  360  WNT S-AGENT  10  0  12/04/03  13:48  F I J  DOMAIN1
   XA_FTA4  360  OTHR X-AGENT  10  0  12/04/03  13:48  F I J  DOMAIN1
   FTA6  360  WNT MANAGER  10  0  12/04/03  13:48  F I J  DOMAIN2
   FTA7  360  WNT FTA  10  0  12/04/03  13:49  F I J  DOMAIN2

   If you run this command in an environment when the primary connection of
   the workstation with its domain or upper manager and all secondary
   connections are active, you receive the following output:

   MASTER  360  *WNT MASTER  10  0  12/04/03  13:48  I J  MASTERDM
   FTA1  360  WNT FTA  10  0  12/04/03  13:48  FTI JW  MASTERDM
   FTA2  360  WNT FTA  10  0  12/04/03  13:48  FTI JW  MASTERDM
   FTA3  360  WNT MANAGER  10  0  12/04/03  13:48  FTI JW  DOMAIN1
   FTA4  360  WNT FTA  10  0  12/04/03  13:48  F I J  DOMAIN1
   FTA5  360  WNT FTA  10  0  12/04/03  13:48  F I J  DOMAIN1
   S A1  360  WNT S-AGENT  10  0  12/04/03  13:48  F I J  DOMAIN1
   XA_FTA4  360  OTHR X-AGENT  10  0  12/04/03  13:48  F I J  DOMAIN1
   FTA6  360  WNT MANAGER  10  0  12/04/03  13:48  F I J  DOMAIN2
   FTA7  360  WNT FTA  10  0  12/04/03  13:49  F I J  DOMAIN2
See Also

For the equivalent Job Scheduling Console task, see "Working with Object Lists" in the IBM Tivoli Workload Scheduler Job Scheduling Console User's Guide.
**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 116.

**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**

To display information about the domain `masterdm`, run the following command:

```
showdomain masterdm
```

To display the member workstations in all domains in the info format, run the following command:

```
showdomain @;info
```

**See Also**

For the equivalent Job Scheduling Console task, see "Displaying and Modifying Domain Properties" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
showfiles
Displays information about file dependencies. A file dependency occurs when a job or job stream is dependent on the existence of one or more files before it can begin running.

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 116.

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**

To display the status of a file dependency for `d:\apps\mis\lib\data4`, run the following command:

```
showfiles d:\apps\mis\lib\data4
```

To display `offline` the status of all file dependencies on all workstations in the `deps` format, run the following command:

```
sf @@@;deps;offline
```
showjobs
Displays information about jobs.

Synopsis
sj [jobselect] [;keys | info | step | logon | retcod][;short | single][;offline]
sj [jobselect] [;deps][;keys | info | logon][;short | single][;offline]
sj [jobselect | wkstation# | jobnumber.hhmm][;stdlist][;keys][;short | single][;offline]

Arguments
jobselect See “Selecting jobs in commands” on page 120.
wkstation The name of the workstation on which the job runs. Wildcard characters are permitted.
jobnumber The job number.
hhmm The started time of the job. Use this, together with the stdlist and single arguments, to display a specific instance of the job.
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.
retcod Displays the return code for the job.
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 116.

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.

182 IBM Tivoli Workload Scheduler Reference Guide
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 as follows:

abend The job terminated with a non-zero exit code.
abep 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.
wait The job is in the wait state. (Extended agent)

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 scheduler 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 a deadline time, the time preceded by an angle bracket (<) is displayed.
• 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 #Jnumnn.
• Jobs submitted on UNIX using the IBM Tivoli Workload Scheduler 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.
• [Script] applies to end-to-end networks only; it means that this job has a centralized script and that Tivoli Workload Scheduler for z/OS has not yet downloaded it to the agent.

**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.

- **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.

Return code  The return code of the job.

Job#  The process identification number displayed as #Jnnnnn.

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.

Return code  The return code of the job.

Job#  The process identification number displayed as #Jnnnnn.

Logon  The user name under which the job runs.

stdlist format:  A standard list file is created automatically by Jobmon on Windows or Jobman on UNIX, for each job Jobmon and Jobman launch. You can display the contents of the standard list files using Conman. A standard list file contains:

• Header and trailer banners.
• Echoed commands.
• The stdout output of the job.
• The stderr output of the job.

To specify a particular date format to be used in the standard list files, change the IBM Tivoli Workload Scheduler date format before creating the standard list files. You do this by modifying the date locale format.

Depending on your environment, change the date locale format by performing the steps listed below:

• On UNIX, set the LANG variable in the environment when Netman starts. If the LANG variable is not set, the operating system locale is set by default to "C".
• On Windows, perform the following steps:
  1. Go to Control Panel→Regional Options and set your locale (location).
  2. Right-click on "My Computer", go to Properties, click on “Advanced”, go to Environment Variables and set the LANG variable as a system variable.
  3. Shut down and restart the system.
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 standard **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**

To display the status of all jobs in all **acctg** job streams on workstation **site3**, run the following command:

```
showjobs site3#acctg@.@
```

To display the status of all jobs on the workstation on which you are running Conman, in the **logon** format, run the following command:

```
sj ;logon
```

To display the status of all jobs in the **hold** state on all workstations, in the **deps** format, run the following command:

```
sj @@@.@+state=hold;deps
```

To display all the standard list files for the job **sendmail** in the job stream **mail** on workstation **WN1**, running in a Windows environment, run the following command:

```
sj WN1#mail.sendmail;stdlist
```

The output is the following:

```
=================================================================
= JOB : WN1#MAIL.SENDMAIL
= USER : xpress
= JCLFILE : \users\xpress\sendxpmail
= Job Number : 8027
= Fri Jun 5 12:17:10 1997
=================================================================

... <stdout, stderr, and echoed commands>
...

=================================================================
= Exit Status : 0
= System Time (Seconds) : 0 Elapsed Time (Minutes) : 0
= User Time (Seconds) : 0
= Fri Jun 5 12:17:12 1997
=================================================================
```

where:
Exit Status
Is the status of the job when it completed.

Elapsed Time
Is the elapsed time for the job.

System Time
Is the time the kernel system spent for the job.

User Time
Is the time the system user spent for the job.

Note: The System Time and User Time fields are used only in UNIX. Their values in Windows are always set to 0. This is because, in Windows, the joblnch.exe process runs in a very short time, which can be considered null.

The following example displays the status of the job dbseload with a return code of 7 and a state of SUCCESSFUL:

```sh
$ conman sj workstation#DAILY_DB_LOAD
TWS for UNIX (AIX)/CONMAN 8.2 (1.36.1.7)
Licensed Materials Property of IBM
5698-WKB
(C) Copyright IBM Corp 1998,2001
US Government User Restricted Rights
Use, duplication or disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
Installed for user ''
Locale LANG set to "en_US"
Schedule (Exp) 09/30/03 (#126) on MASTER. Batchman LIVES. Limit: 10,
Fence: 0, Audit Level: 1
sj workstation#DAILY_DB_LOAD
(Est) (Est)
CPU Schedule Job State Pr Start
Elapse Dependencies Return Code
WORKSTATION #DAILY_DB_LOAD **************************************** SUCC 10 22:11
00:04
DATASPLT SUCC 10 22:11
00:01 #J17922 0
DATAMRGE ABEND 10 22:12
00:01 #J17924 1
CHCKMRGE SUCC 10 22:12
00:01 #J17926 0
DATACLNS SUCC 10 22:12
00:01 #J17932 0
DATARMRG SUCC 10 22:13
00:01 #J18704 0
DBSELOAD SUCC 10 22:13
00:01 #J18706 7
DATAREPT SUCC 10 22:13
00:01 #J18712 0
DATARTRN SUCC 10 22:14
00:01 #J18714 0
$ 
```

There is also a new arguments retcod, that when used in conjunction with the keys argument, gives the return code for a specified job, as shown in the following example:

```sh
$ conman sj workstation#daily_db_load.dbseload;keys;retcod
```

License and copyright notice follow.
Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
Installed for user ' ''
Locale LANG set to "en_US"
Schedule (Exp) 10/16/03 (#150) on MASTER. Batchman LIVES. Limit: 10, Fence: 0,
Audit Level: 1
sj workstation#daily_db_load.dbload;keys;retcod
B$

The retcod feature when integrated into a script can become quite powerful.

**See Also**

For the equivalent Job Scheduling Console task, see "Working with Object Lists" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.*
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 116.

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**

To display the status of all prompts issued on the workstation on which you are running Conman, run the following command:

```
showprompts
```

To display the status of all mis prompts that have been issued, in the deps format, run the following command:

```
sp mis0;asked;deps
```

To display the status of prompt number 34 on workstation main, run the following command:

```
sp main#34
```

**See Also**

For the equivalent Job Scheduling Console task, see "Displaying and Modifying Prompt Properties" in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
**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 [116].

**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**

To display information about all resources on the workstation on which you are running Conman, run the following command:

```
showresources
```

To display information about the `dbase` resource on workstation `main` in the `deps` format, run the following command:

```
sr main#dbase;deps
```

**See Also**

For the equivalent Job Scheduling Console task, see "Displaying Resource Properties" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.*
**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 128.
- `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 116.

**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 scheduler 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].
- For a deadline time, the time preceded by an angle bracket (<) is displayed.
- 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 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 **showsschedules** format.

**Examples**

To display the status of all job streams in the **hold** state on the workstation on which you are running Conman, run the following command:

```
showsschedules @+state=hold
```

To display the status of all **corp** job streams on workstation **site2** in the **deps;info** format, run the following command:

```
ss site2#corp@;deps;info
```

To display **offline** the status of all job streams in the **abend** state on all workstations, run the following command:

```
ss @@@state=abend;off
```

To display the status of all job streams on all workstations, run the following command:

```
%ss @@@
```

After running the command, you receive a similar output to:

```
CPU Schedule State Pr Start Elapse #OK Lim
TWS820A #SCHED-1 ABEND 10 09/03 0:00 100 0
TWS820A #SCHED-2 READY 10 100 0
TWS820A #SCHED-3 ABEND 10 09/03 0:02 100 0
TWS820A #SCHED-4 EXEC 10 09/03 100 0
TWS820A #SCHED-5 READY 10 100 0
TWS820A #SCHED-6 READY 10 100 0
TWS820A #SCHED-7 SUCC 10 09/03 0:02 2 2
TWS820A #SCHED-8 READY 10 ( 0:01) 1 0
```

**See Also**

For the equivalent Job Scheduling Console task, see "Displaying a Job Stream" in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
shutdown

Unconditionally stops all of scheduler 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, run the StartUp command. For information about the StartUp command, see “StartUp” on page 255. To restart the entire process tree, run a Conman start command.

You must run a Conman unlink @ command before executing a shutdown command.

Examples
To shut down production on the workstation on which you are running Conman, run the following command:

unlink @
shutdown

To shut down production on the workstation on which you are running Conman and wait for all processes to stop, run the following command:

unlink@;noask
shut ;wait
start

Starts scheduler production processes.

You must have start access to the workstation.

Synopsis

start [domain!]wkstation[mgr][noask][demgr]

Arguments

domain Specify the name of the domain in which workstations are started. 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 Specify 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 212 for more information.

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

demgr This option prevents the opening of external connections during the transition time between when an agent starts as an old domain manager, and when the switchmgr command is run, depriving the agent of the domain manager function. This option is run automatically, but until the old domain manager has processed the switchmgr event (in the case, for example, of delayed restart or restart after repairing a damaged agent), the demgr option must be used to start the old domain manager from the local command line. For more details on this option, see the IBM Tivoli Workload Scheduler Planning and Installation Guide.

Usage Notes

The start command is used at the start of each day to restart the 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 run 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 workstations that are hosted by that agent.

Examples
The illustration and table below show the workstations started by start commands run by users in various locations in the network. DM<n> are domain managers and Ann are agents.

![Diagram](image)

Figure 4. Started workstations in network

<table>
<thead>
<tr>
<th>Command</th>
<th>Started by User1</th>
<th>Started by User2</th>
<th>Started by User3</th>
</tr>
</thead>
<tbody>
<tr>
<td>start @!@</td>
<td>All workstations are started.</td>
<td>DM2 A21 A22 DM4 A41 A42</td>
<td>A21</td>
</tr>
<tr>
<td>start @</td>
<td>DM1 A11 A12</td>
<td>DM2 A21 A22</td>
<td>A21</td>
</tr>
<tr>
<td>start DOMAIN3@</td>
<td>DM3 A31 A32</td>
<td>Not allowed.</td>
<td>Not allowed</td>
</tr>
<tr>
<td>start DOMAIN4@</td>
<td>DM4 A41 A42</td>
<td>DM4 A41 A42</td>
<td>Not allowed</td>
</tr>
<tr>
<td>start DM2</td>
<td>DM2</td>
<td>DM2</td>
<td>Not allowed</td>
</tr>
<tr>
<td>start A42</td>
<td>A42</td>
<td>A42</td>
<td>Not allowed</td>
</tr>
<tr>
<td>start A31</td>
<td>A31</td>
<td>Not allowed.</td>
<td>Not allowed</td>
</tr>
</tbody>
</table>
status
Displays the Conman banner and the IBM Tivoli Workload Scheduler 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. The Def or Exp information can appear. 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. With IBM Tivoli Workload Scheduler, Version 8.2, databases and plans are always expanded, but this information appears for backward compatibility.

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
TWS for WINDOWS NT/CONMAN 8.2 (1.36.1.7)
Licensed Materials Property of IBM
5698-WKB
(C) Copyright IBM Corp 1998,2001
US Government User Restricted Rights
Use, duplication or disclosure restricted by GSA ADP Schedule Contract
with IBM Corp.
Schedule (Exp) 05/19/03 (#17) on DEMOCPU. Batchman down.
Limit: 30, Fence: 0, Audit Level: 1
**stop**

Stops scheduler 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.

When you issue a `stop @` command on a domain manager, a local `conman stop` command runs on the remote CPUs. The command starts running on the lowest stations in the network hierarchy, then finally runs on the domain manager. However, the symphony files are not updated before the CPUs go down. Therefore, if you issue a `conman sc@!@` command form any CPU, the resulting information might be an accurate picture of the states of the CPUs, even of the domain manager.
Examples
The illustration and table below show the workstations stopped by different stop commands run by users in different locations in the network. DMn are domain managers and Ann are agents.

![Diagram showing workstations stopped by different users and locations]

**Figure 5. Stopped workstations in network**

<table>
<thead>
<tr>
<th>Command</th>
<th>Stopped by: User1</th>
<th>Stopped by User2</th>
<th>Stopped by User3</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop @!@</td>
<td>All workstations are stopped.</td>
<td>DM2 A21 A22 A41 A42</td>
<td>DM2 A21 A22</td>
</tr>
<tr>
<td>stop @</td>
<td>DM1 A11 A12</td>
<td>DM2 A21 A22 A41 A42</td>
<td>DM2 A21 A22</td>
</tr>
<tr>
<td>stop DOMAIN3!@</td>
<td>DM3 A31 A32</td>
<td>Not allowed.</td>
<td>Not allowed.</td>
</tr>
<tr>
<td>stop DOMAIN4!@</td>
<td>DM4 A41 A42</td>
<td>DM4 A41 A42</td>
<td>Not allowed.</td>
</tr>
<tr>
<td>stop DM2</td>
<td>DM2</td>
<td>DM2</td>
<td>DM2</td>
</tr>
<tr>
<td>stop A42</td>
<td>A42</td>
<td>A42</td>
<td>Not allowed.</td>
</tr>
<tr>
<td>stop A31</td>
<td>A31</td>
<td>Not allowed.</td>
<td>Not allowed.</td>
</tr>
</tbody>
</table>
**stop ;progressive**

Stops scheduler production processes hierarchically when you have defined at least one workstation as behind firewall in an IBM Tivoli Workload Scheduler Version 8.2 network. Similar to the stop @@ command, but more effective in improving plan execution performance. The command does not run from the domain in which the command was initially issued for each subordinate domain, but runs at each hierarchical level.

You must have stop access to the workstation.

**Synopsis**

*stop ;progressive*

**Usage Notes**

When you issue the command on a domain manager, all workstations in that domain are stopped and then the domain manager itself is stopped and the command continues to run on any subordinate domains. The command continues to run in this hierarchical manner, the domain manager stops workstations in the same domain, stops itself, and then continues to run on subordinate domains.

**Examples**

The illustration [Figure 5 on page 202](#) and the table below show the workstations stopped by issuing the *stop ;progressive* command on DM2.

<table>
<thead>
<tr>
<th>Command</th>
<th>Stopped by DM2</th>
<th>Stopped by DM4</th>
</tr>
</thead>
<tbody>
<tr>
<td>stop ;progressive</td>
<td>A21 A22 DM2</td>
<td>A41 A42 DM4</td>
</tr>
</tbody>
</table>
submit docommand

Submits a command to be launched as a 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 up to the first six alphanumeric characters (in upper case) of the command, depending on the number of characters in the command, followed by a ten digit random number. If there are blanks in the command, the name is constructed using up to the first six alphanumeric characters before the blank. For example, if the command is "rm apfile", the generated name will be similar to RM0123456789. If the command is longer than six alphanumeric characters such as, "wlsinst", the generated name will be wlsins0396578515.

If you do not include alias the first time you submit the command, a job name is constructed using up to 255 characters of the command name. If you submit a command a second time from the same workstation, the alias keyword is mandatory and must be unique for each command submission.

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 tzone] [+n days | mm/dd/yy] | [absolute | abs]
confirmed

deadline=time [timezone | tz tzone][+n days | mm/dd/yy]

every=rate

follows=[netagent:][wkstation#]jstream[.job | @] | job[;nocheck][;wait=time][,...]

interactive
logon=user
needs=\[num\] \[wkstation\#\]resource[...]
opens=\[wkstation\#\]"filename"\[(\text{qualifier})\][...]
priority=\[	ext{pri} \mid \text{hi} \mid \text{go}\]
prompt=\[\text{: | !}]text \mid \text{promptname}[...]
recovery=\text{stop} \mid \text{continue} \mid \text{rerun}
after=\[wkstation\#\]jobname
abendprompt =\"text\"
until=\text{time} \text{[timezone | tz tzname]}\[+n \text{ day[s]} \mid \text{absolute} \mid \text{abs}] \text{[onuntil action]}

Usage Notes
If you do not specify a \textit{workstation} with \textbf{follows}, \textbf{needs}, or \textbf{opens}, the default is the workstation of the job. If the job is run on a fault-tolerant agent, and you want to include a \textbf{prompt} dependency or a \textbf{recoveryprompt}, the mozart directory \texttt{(TWShome/mozart)} on the master domain manager must be accessible, either mounted or shared.

Examples
To submit an \texttt{rm} command into the job stream \texttt{JOBS} with a \texttt{follows} dependency, run the following command:
\begin{verbatim}
submit docommand="rm apfile";follows sked3
\end{verbatim}

To submit a \texttt{sort} command with the alias \texttt{sortit} and place the job in the job stream \texttt{reports} with an \texttt{at} time of 5:30 p.m., run the following command:
\begin{verbatim}
sbd "sort < file1 > file2";alias=sortit;into=reports;at=1730
\end{verbatim}

To submit \texttt{chmod} commands on all workstations with names beginning with \texttt{site}, run the following command:
\begin{verbatim}
sbd site@\\"chmod 444 file2";alias
\end{verbatim}
submit file

Submits a file to be launched as a 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 up to the first six alphanumeric characters (in upper case) of the file name, depending on the number of characters in the file name, followed by a ten digit random number. For example, if the file name is jclttx5, the generated name will be similar to jCLLTX0123456789.

If you do not include alias, a filename is constructed using up to 255 alphanumeric characters of the file’s base name, in upper case.

In either of the above cases, if the file name does not start with a letter, you are prompted to use alias=.name.

If you submit a file a second time from the same workstation, the alias keyword is mandatory and must be unique for each file submission.

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] | [absolute | abs]

confirmed

deadline=time[timezone | tz **tzname**][+n **days** | mm/dd/yy]

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**[...]


**rccondsucc** "Success Condition"

**recovery**=stop | continue | rerun

**after** [workstation#] jobname

**abendprompt** “text”

**until** **time** [timezone|tz tzname][+n day[s] | [absolute | abs]] [onuntil action]

**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 run on a fault-tolerant agent, and you want to include a **prompt** dependency or a **recoveryprompt**, the mozart directory (**TWShome/mozart**) on the master domain manager must be accessible, either mounted or shared.

**Examples**
To submit a file into the job stream **jobs** (the job name is **myjcl**), run the following command:

```
submit file=d:\jobs\lib\daily\myjcl
```

To submit a file, with a job name of **misjob4**, into the job stream **missked**, run the following command:

```
sbf /usr/lib/mis/jcl4;alias=misjob4;into=missked ;needs=2 slots
```

The job needs two units of the **slots** resource.

To submit all files that have names beginning with **back** into the job stream **bkup**, run the following command:

```
sbf "/usr/lib/backup/back*";into=bkup
```
**submit job**

Submits a job to be launched by the 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 scheduler databases on the master domain manager.

**Synopsis**

```
submit [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 jcrtx5, 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 workstation, 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] | [absolute | abs]
confirmed
```

```
deadline=time [timezone | tz tzname] [+n days | mm/dd/yy]
every=rate
follows=[netagent:]wkstation#]jstream[,job | @] | job[,nocheck][,wait=time][,...]
needs=[num] wkstation#]resource[,...]
opens=[[wkstation#]"filename" [qualifier]][,...] priority=[pri | hi | go]
prompt="[c | !]text" | promptname[,...]
```
rccondsucc “Success Condition”

recovery=stop | continue | rerun

after [workstation#]jobname

abendprompt “text”

until time [timezone | tz tzname]+n day[s] | [absolute | abs] [onuntil action]

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 run on a fault-tolerant agent, and you want to include a prompt dependency or a recoveryprompt, the mozart directory (TWShome/mozart) on the master domain manager must be accessible, either mounted or shared.

Examples

To submit the test jobs into the job stream JOBS, run the following command:

submit job=test

To submit a job with an alias of rptx4 and place the job in the job stream reports with an at time of 5:30 p.m., run the following command:

sbj rjob4;alias=rptx4;into=reports;at=1730

To submit job txjob3 on all workstations whose names begin with site, run the following command:

sbj site@#txjob3;alias

See Also

For the equivalent Job Scheduling Console task, see “Submitting Jobs” in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
submit sched

Submits a job stream to be launched by the 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 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=hh:mm [timezone | tz tzname] [+n days | mm/dd/yy] | [absolute | abs]

carryforward

deadline=time [timezone | tz tzname] [+n days | mm/dd/yy]

follows=[netagent::][wkstation#]jstream1,job | @] | job[,nocheck][;wait=time][,...]

limit=joblimit

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

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

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

until time [timezone | tz tzname] [+n days] | [absolute | abs] [onuntil action]

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 run on a fault-tolerant agent, and you want to include a **prompt** dependency or a **recoveryprompt**, the mozart directory (`TWS/home/mozart`) on the master domain manager must be accessible, either mounted or shared.

**Examples**

To submit the **adhoc** job stream on workstation **site1** and flags it as a **carryforward** job stream, run the following command:

```bash
submit sched=site1#adhoc;carryforward
```

To submit job stream **fox4** with a job limit of 2, a priority of 23, and an **until** time of midnight, run the following command:

```bash
sbs fox4;limit=2;pri=23;until=0000
```

To submit job stream **sched3** on all workstations with names that start with **site**, run the following command:

```bash
sbs site@#sched3
```

**See Also**

For the equivalent Job Scheduling Console task, see “Plan Tasks for Distributed” in the *IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide*. 
**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.

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 run and the old domain manager will automatically resume domain management responsibilities.

**Examples**

To switch the domain manager to workstation `orca` in the `masterdm` domain, run the following command:

```
switchmgr masterdm,orca
```

To switch the domain manager to workstation `ruby` in the `bldg2` domain, run the following command:

```
switchmgr bldg2,ruby
```
system

runs 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

To run a ps command on UNIX, run the following command:

ps -ef

to run a dir command on Windows, run the following command:

dir \bin
**telop**

Sends a message to the scheduler console.

**Synopsis**

`tellop` 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 146](#).

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**

To send a message, run the following command:

```
tellop TWS will be stopped at 4:30 for 15 minutes.
```

To prompt for text before sending a message, run the following command:

```
tellop TWS will be stopped at 4:30 for 15 minutes.
```

```
tellop>*********************************
tellop>* TWS will be stopped at *
tellop>* 4:30 for 15 minutes. *
tellop>*********************************
tellop>//
```
unlink

Closes communication 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. It is not necessary to specify the domain name of a workstation in the master domain. Wildcard characters are permitted.

**Note:** You must always specify the domain name when unlinking a workstation not in the master domain.

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 provided that the workstation is either a domain manager or host. A peer agent in the same domain cannot be unlinked.

For additional information see "[link](page-159)" on page 159.

**Examples**

The illustration and table below show the links closed by **unlink** commands run by users in various locations in the network.
DM*n* are domain managers and A*n* are agents.

<table>
<thead>
<tr>
<th>Command</th>
<th>Closed by User1</th>
<th>Closed by User2</th>
<th>Closed by User3</th>
</tr>
</thead>
<tbody>
<tr>
<td>unlink @!@</td>
<td>All links are closed.</td>
<td>DM1-DM2</td>
<td>DM2-A21</td>
</tr>
<tr>
<td>unlink @</td>
<td>DM1-A11</td>
<td>DM1-A12</td>
<td>DM2-A21</td>
</tr>
<tr>
<td></td>
<td>DM1-DM2</td>
<td>DM2-A21</td>
<td>DM2-A22</td>
</tr>
<tr>
<td></td>
<td>DM1-DM3</td>
<td>DM2-A22</td>
<td>DM2-DM4</td>
</tr>
<tr>
<td></td>
<td>DM3-A31</td>
<td>DM4-A41</td>
<td>DM4-A42</td>
</tr>
<tr>
<td>unlink DOMAIN3!@</td>
<td>DM3-A31</td>
<td>Not allowed.</td>
<td>Not allowed.</td>
</tr>
<tr>
<td></td>
<td>DM3-A32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>unlink DOMAIN4!@</td>
<td>DM4-A41</td>
<td>DM4-A41</td>
<td>Not allowed.</td>
</tr>
<tr>
<td></td>
<td>DM4-A42</td>
<td>DM4-A42</td>
<td>Not allowed.</td>
</tr>
<tr>
<td>unlink DM2</td>
<td>DM1-DM2</td>
<td>Not applicable.</td>
<td>DM2-A21</td>
</tr>
<tr>
<td>unlink A42</td>
<td>DM4-A42</td>
<td>DM4-A42</td>
<td>Not allowed.</td>
</tr>
<tr>
<td>unlink A31</td>
<td>DM3-A31</td>
<td>Not allowed.</td>
<td>Not allowed.</td>
</tr>
</tbody>
</table>

Figure 6. Unlinked network workstations
version
Displays Conman’s command line program banner.

Synopsis
version

Examples
To display Conman’s command line program banner, run the following command:
`%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
Chapter 6. Utility commands

This chapter describes IBM Tivoli Workload Scheduler utility commands. These commands are tools to help you manage the scheduler. The commands, with the exception of StartUp and version, are installed in TWS_home/bin directory. StartUp is installed in the TWS_home directory, and version is installed in the TWS_home/version directory.

Command descriptions

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>at</td>
<td>For UNIX only. Submits a job to be run at a specific time.</td>
</tr>
<tr>
<td>batch</td>
<td>For UNIX only. Submits a job to be run as soon as possible.</td>
</tr>
<tr>
<td>caxtract</td>
<td>Extracts information about calendars.</td>
</tr>
<tr>
<td>cpuinfo</td>
<td>Returns information from a workstation definition.</td>
</tr>
<tr>
<td>datecalc</td>
<td>Converts date and time to a desired format.</td>
</tr>
<tr>
<td>dbexpand</td>
<td>Expands scheduler databases.</td>
</tr>
<tr>
<td>delete</td>
<td>Removes script files and standard list files by name.</td>
</tr>
<tr>
<td>evtsize</td>
<td>Defines the maximum size of event message files.</td>
</tr>
<tr>
<td>jbxtract</td>
<td>Extracts information about jobs.</td>
</tr>
<tr>
<td>jobinfo</td>
<td>Returns information about a job.</td>
</tr>
<tr>
<td>jobstdl</td>
<td>Returns the pathnames of standard list files.</td>
</tr>
<tr>
<td>listproc</td>
<td>For Windows only. Lists processes. This command is unsupported.</td>
</tr>
<tr>
<td>killproc</td>
<td>For Windows only. Kills processes. This command is unsupported.</td>
</tr>
<tr>
<td>maestro</td>
<td>Returns the scheduler home directory.</td>
</tr>
<tr>
<td>makecal</td>
<td>Creates custom calendars.</td>
</tr>
<tr>
<td>metronome.pl</td>
<td>Takes a snapshot of IBM Tivoli Workload Scheduler and generates an html report to aid in solving problems.</td>
</tr>
<tr>
<td>morestdl</td>
<td>Displays the contents of standard list files.</td>
</tr>
<tr>
<td>parms</td>
<td>Displays, changes, and adds parameters.</td>
</tr>
<tr>
<td>paxtract</td>
<td>Extracts information about parameters.</td>
</tr>
<tr>
<td>prxtract</td>
<td>Extracts information about prompts.</td>
</tr>
<tr>
<td>r11xtr</td>
<td>Extracts information about job streams.</td>
</tr>
<tr>
<td>release</td>
<td>Releases units of a resource.</td>
</tr>
<tr>
<td>reextract</td>
<td>Extracts information about resources.</td>
</tr>
<tr>
<td>rmstdlist</td>
<td>Removes standard list files based on age.</td>
</tr>
<tr>
<td>showexec</td>
<td>For UNIX only. Displays information about executing jobs.</td>
</tr>
<tr>
<td>StartUp</td>
<td>Starts the Netman process.</td>
</tr>
<tr>
<td>version</td>
<td>For UNIX only. Displays version information.</td>
</tr>
<tr>
<td>xrxtrct</td>
<td>Extracts information about cross-references.</td>
</tr>
<tr>
<td>wmaeutil</td>
<td>Extracts information about cross-references.</td>
</tr>
</tbody>
</table>
at and batch commands

For UNIX only. Submits ad hoc commands and jobs to be launched by IBM Tivoli Workload Scheduler.

See at.allow and at.deny below for information about the availability to users.

Synopsis

at -v | -u

at -s jstream | -q queue time-spec

batch -v | -u

batch [-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. 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 222 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 222 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 “Examples” on page 222.

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 the IBM Tivoli Workload Scheduler **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 IBM Tivoli Workload Scheduler Planning and Installation Guide.

Once submitted, jobs are launched in the same manner as other scheduled jobs. Each job runs in the submitting user 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 by scheduler 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
```

On tier 2 platforms only, when you install the scheduler through the **customize** script, the following links are created by default:

```
/usr/bin/mat —> TWShome/bin/at
/usr/bin/mbatch —> TWShome/bin/batch
```

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. On tier 2 platforms, if the commands are run 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 the scheduler 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:** The scheduler 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 the scheduler 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.

If the time value is less than the current time, the value is regarded as for the following day. If the time value is greater than the current time, the value is regarded as for the current day. If the time value is greater than 2400, the value is divided by 2400 to obtain the number of days. If you specify the time value in days, this is added to the value obtained by dividing by 2400.

Examples
To submit a job into job stream `sched8` to be launched as soon as possible, run the following command:
```
batch -s sched8
command <Return>
...<Control d>
```

To submit a job into job stream `k` to be launched at 9:45 p.m., run the following command:
```
at -qk 21h45
command <Return>
...<Control d>
```

To submit a job to be launched two hours from the time at which the command was entered, run the following command:
```
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 as the user’s group. Otherwise, it is submitted into a job stream named `at`.

To submit a job into job stream `sked-mis` to be launched at 5:30 p.m., run the following command:
```
at -s sked-mis 17h30
command <Return>
...<Control d>
```

The following command is the same as the previous command, except that the job’s commands are read from a file:
```
at -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

Extracts information about calendars from the scheduler 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Max Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>calendar name</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>calendar description</td>
<td>64</td>
</tr>
</tbody>
</table>

Examples

To extract information about all calendar definitions and direct the output to the file cainfo, run the following command:

caxtract -o cainfo
cpuinfo

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: UNIX, WNT,, and 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 scheduler 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:
To print the os type for workstation oak, run the following command:

```
>cpuinfo oak os_type
UNIX
```

To print the node and port for workstation oak, run the following command:

```
>cpuinfo oak node port
oak.tivoli.com 31111
```

To print all information for workstation oak, run the following command:

```
>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/Pacific
VERSION:6.1
INFO:SunOS 5.3 Generic 1016 sun4m
```
datecalc

Resolves date expressions and returns dates in desired formats.

Synopsis
datecalc -v | -u

datecalc base-date [offset] [pic format][freedays Calendar_Name [-sa] [-su]]
datecalc -t time [base-date] [offset] [pic format]
datecalc yyyymmddhhmtt [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 scheduler 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. This may not be the same
as the system date. When used inside jobs within a job stream that
is not a carried forward job stream, it returns the date when the job
should run, which could be different from the production date of
the job stream if the job has an AT® dependency specified. When
used inside jobs within a carried forward job stream, it returns the
date when the job should have run, which could be different from
the production date of the carried forward job stream if the job has
an AT dependency specified. If the AT dependency is used with
the following syntax: at=hhmm + n days, the n days are not added
to the variable TIVOLI_JOB_DATE and therefore, the datecalc
command does not report these days.

-t time [base-date]
Specify time in one of the following formats:

now | noon | midnight | [h][m]:[s] [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[ms]  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.

yyyymmddhhmmttt
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:
+ l > Specifies an offset to a later date or time. Use + (Plus) on Windows. Use > (Greater than) on UNIX. Be sure to escape the angle bracket ("\">").

- l < Specifies an offset to an earlier date or time. Use - (Minus) on Windows. Use < (Less than) on UNIX. 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. Use ^ (carat) on UNIX (this character must be escaped (\) in the Bourne shell). Use / (slash) on Windows.
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) environment variables. If the NLS variables are not defined, the native language defaults to C.

freedays
Specifies the name of a freedays calendar Calendar_Name that is to replace holidays in the evaluation of workdays.

In this case, workdays is evaluated as everyday excluding saturday, sunday and all the dates listed in Calendar_Name.

By default, saturday and sunday are not regarded as workdays, unless you explicitly state the opposite by adding -sa and/or -su after Calendar_Name.

You can also specify holidays as the name of the freedays calendar.

Examples
To return the next date, from today, on the monthend calendar, run the following command:

```bash
>datecalc today next monthend
```

In the following examples, the current system date is Friday, April 9, 1999.

```bash
>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.

```bash
>datecalc -t now \> 4 hours pic hh:tt
14:24
```
**dbexpand**

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 IBM 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**

```bash
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 the scheduler. 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**

To expand the databases and back up the current files in the `/usr/lib/maestro/temp` directory, run the following command:

```bash
dbexpand -n -b /usr/lib/maestro/temp
```

If the directory already exists, its contents are overwritten.

To expand the databases and back up the current files in the `c:\programs\wsched\temp` directory, run the following command:

```bash
dbexpand -b c:\\programs\\wsched\\temp
```

A prompt is displayed if the directory already exists.
delete

Removes files. This command is intended to remove standard list files. The users maestro and root on UNIX, and Administrator on Windows 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
To remove all the standard list files for 4/11/04, run the following command:
delete d:\win32app\maestro\stdlist\2004.4.11\@

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 standard list file. For more information about jobmanrc refer to the IBM Tivoli Workload Scheduler Planning and Installation Guide.
evtsize

Defines the size of the scheduler 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 maestro or root on UNIX, or Administrator on Windows to run evtsize. Be sure to stop the IBM Tivoli Workload Scheduler engine before running this command.

Synopsis

```
evtsize -v | -u

evtsize filename size

evtsize -show filename
```

Arguments

- `-v` Displays the command version and exits.
- `-u` Displays command usage information and exits.
- `-show filename` Queries the current queue length of the specified file.

`filename` The name of the event file. Specify one of the following:

- `Courier.msg`
- `Intercom.msg`
- `Mailbox.msg`
- `pobox/workstation.msg`

`size` The maximum size of the event file in bytes. When first built by the scheduler, the maximum size is set to 10 MB.

Examples

To set the maximum size of the `Intercom` file to 20 MB, run the following command:

```
evtsize Intercom.msg 20000000
```

To set the maximum size of the `pobox` file for workstation `chicago` to 15 MB, run the following command:

```
evtsize pobox\chicago.msg 15000000
```

The following command:

```
evtsize -show Intercom.msg
```

returns the following output:

```
TWS for Windows NT/EVTSIZE 8.2 (1.2.2.2)
Licensed Materials Property of IBM
5698-WKB
(C) Copyright IBM Corp 1998,2001
US Government User Restricted Rights
Use, duplication or disclosure restricted by GSA ADP
Schedule Contract with IBM Corp.
AWS11140703 Queue size current 880, maximum 10000000 bytes
(read 48, write 928)

where:
```
880 Is the size of the current queue of the Intercom.msg file
10000000 Is the maximum size of the Intercom.msg file
read 48 Is the pointer position to read records
write 928 Is the pointer position to write records
jbxtract

Extracts information about jobs from the scheduler 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 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, long names are truncated to eight characters and a plus sign. For example: A1234567+.

Each job record contains tab-delimited, variable length fields. The fields are described in the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Max Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>job name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>job script file name</td>
<td>4096</td>
</tr>
<tr>
<td>4</td>
<td>job description</td>
<td>65</td>
</tr>
<tr>
<td>5</td>
<td>recovery job name</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>recovery option (0=stop, 1=rerun, 2=continue)</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>recovery prompt text</td>
<td>64</td>
</tr>
<tr>
<td>8</td>
<td>auto-documentation flag (0=disabled, 1=enabled)</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>job login user name</td>
<td>36</td>
</tr>
<tr>
<td>10</td>
<td>job creator user name</td>
<td>36</td>
</tr>
<tr>
<td>11</td>
<td>number of successful runs</td>
<td>5</td>
</tr>
<tr>
<td>12</td>
<td>number of abended runs</td>
<td>5</td>
</tr>
<tr>
<td>13</td>
<td>total elapsed time of all job runs</td>
<td>8</td>
</tr>
<tr>
<td>14</td>
<td>total cpu time of all job runs</td>
<td>8</td>
</tr>
<tr>
<td>15</td>
<td>average elapsed time</td>
<td>8</td>
</tr>
<tr>
<td>16</td>
<td>last run date (yymmd)</td>
<td>8</td>
</tr>
<tr>
<td>17</td>
<td>last run time (hhmm)</td>
<td>8</td>
</tr>
<tr>
<td>18</td>
<td>last cpu seconds</td>
<td>8</td>
</tr>
<tr>
<td>19</td>
<td>last elapsed time</td>
<td>8</td>
</tr>
<tr>
<td>20</td>
<td>maximum cpu seconds</td>
<td>8</td>
</tr>
<tr>
<td>Field</td>
<td>Description</td>
<td>Max Length (bytes)</td>
</tr>
<tr>
<td>-------</td>
<td>------------------------------</td>
<td>--------------------</td>
</tr>
<tr>
<td>21</td>
<td>maximum elapsed time</td>
<td>8</td>
</tr>
<tr>
<td>22</td>
<td>maximum run date (yymmdd)</td>
<td>8</td>
</tr>
<tr>
<td>23</td>
<td>minimum cpu seconds</td>
<td>8</td>
</tr>
<tr>
<td>24</td>
<td>minimum elapsed time</td>
<td>8</td>
</tr>
<tr>
<td>25</td>
<td>minimum run date (yymmdd)</td>
<td>8</td>
</tr>
</tbody>
</table>

**Examples**

To extract information about job **myjob** on workstation **main** and direct the output to the file **jinfo**, run the following command:

```
jbxtract -j myjob -c main -o jinfo
```
**jobinfo**

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 scheduler `at` or `batch` command. UNIX only.
- `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.
- `rstrt_retcode` If the current job is a recovery job, returns the return code of the parent 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

1. 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:

```bash
# Determine partial (1) or full (2):
if [ "\"maestro\"/bin/jobinfo job_name" = "PARTBACK" ]
then
  bkup=1
else
  bkup=2
fi
```

2. To display the return code of the parent job, if the current job is a recovery job, run the following command:

```bash
$ jobinfo rstrt_retcode
```

The first job (parent job) has been defined in the script recovery.sh while the second job (recovery job) gets enabled only if the first job abends.

When combined with a return code condition, `jobinfo rstrt_retcode` can be used to direct the recovery job to take different actions depending on the parent job's return code. A recovery job is shown in the example below:

```bash
$JOBS
MASTER#DBSELOAD DOCOMMAND "/usr/local/tws/maestro/scripts/populate.sh"
STREAMLOGON "^TWSUSER"
DESCRIPTION "populate database manual"
RECOVERY RERUN AFTER MASTER#RECOVERY
RCONDSSUC "(RC = 0) OR ((RC > 4) AND (RC < 11))"
```

Note that the job is defined with the recovery action RERUN. This enables the recovery job to take some corrective action, before the parent job attempts to run again.

The recovery job itself is defined as shown in the example below:

```bash
$JOBS
MASTER#RECOVERY DOCOMMAND "^TWSHOME^/scripts/recovery.sh"
STREAMLOGON "^TWSUSER"
DESCRIPTION "populate database recovery manual"
RECOVERY STOP
```
jobstdl

Returns the names of standard list files. This command must be run by the user for which IBM Tivoli Workload Scheduler was installed. If you use this command without any parameters, ensure that you are logged on as an IBM Tivoli Workload Scheduler user.

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.
-jbnum 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 a space.

Examples

To return the path names of all standard list files for the current day, run the following command:

jobstdl

To return the path name of the standard list for the first run of job mailxhg1.getmail on the current day, run the following command:

jobstdl -first -name mailxhg1.getmail

To return the path name of the standard list for the second run of job mailxhg1.getmail on the current day, run the following command:

jobstdl -num 2 -name mailxhg1.getmail

To return the path names of the standard list files for all runs of job mailxhg1.getmail from three days ago, run the following command:

jobstdl -day 3 -name mailxhg1.getmail
To return the path name of the standard list for the last run of job
mailxhg1.getmail from four days ago, run the following command:

```
jobstdl -day 4 -last -name mailxhg1.getmail
```

To return the path name of the standard list for job number 455, run the following command:

```
jobstdl 455
```

To print the contents of the standard list file for job number 455, run the following command:

```
\[cd \text{"maestro"}/bin
lp -p 6 \text{"jobstdl 455"}
\]
```
maestro

Returns the path name of the scheduler home directory, referred to as TWS\textit{home}.

\textbf{Synopsis}

\begin{verbatim}
maestro [-v | -u]
\end{verbatim}

\textbf{Arguments}

-\texttt{v} \quad Displays the command version and exits.
-\texttt{u} \quad Displays command usage information and exits.

\textbf{Examples}

To display the scheduler home directory, run the following command:

\begin{verbatim}
$ maestro
/usr/lib/maestro
\end{verbatim}

To change the directory to the scheduler home directory, run the following command:

\begin{verbatim}
$ cd `maestro`
\end{verbatim}
**makecal**

Creates a custom calendar. On UNIX, the Korn shell is required to run 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][-freedays Calendar_Name [-sa] [-su]]
```

**Arguments**

- **-v** Displays the command version and exits.
- **-u** Displays command usage information and exits.
- **-c name** Specifies a name for the calendar. IBM Tivoli Workload Scheduler keywords (such as *Freedays* or *Schedule*) cannot be used as calendar names. The name can contain up to eight alphanumeric characters and must start with a letter. Do not use the names of weekdays for the calendar names. 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.

**Note:** Using this argument results in the new calendar including also the last workday of the month that precedes the date of creation of the calendar.
- **-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* on page 227 for more information about date formats.
-w n  Specifies the workday after the nth 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.

-freedays  Specifies the name of a freedays calendar Calendar_Name that is to replace holidays in the evaluation of workdays.

In this case, workdays is evaluated as everyday excluding saturday, sunday and all the dates listed in Calendar_Name.

By default, saturday and sunday are not regarded as workdays, unless you explicitly state the opposite by adding -sa and/or -su after Calendar_Name.

You can also specify holidays as the name of the freedays calendar.

This keyword affects the processing of makecal with options -I, -p, and -w.

Examples

To make a two-year calendar with the last day of every month selected, run the following command:

```
makecal -e -i24
```

To make a calendar with 30 days that starts on May 30, 1999, and has every third day selected, run the following command:

```
makecal -r 3 -s "30 MAY 1999" -i30
```
**metronome.pl**

A PERL script that takes a snapshot of IBM Tivoli Workload Scheduler and generates an html report. When a user encounters a problem with the product, this report can aid IBM Tivoli Workload Scheduler customer support.

**Usage Notes**

Refer to *IBM Tivoli Workload Scheduler Troubleshooting and Error Messages* for more information about the metronome utility.
morestdl

Displays the contents of standard list files. This command must be run by the user for which IBM Tivoli Workload Scheduler was installed. If you use this command without any parameters, ensure that you are logged as an IBM Tivoli Workload Scheduler user.

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
To display the standard list file for the first run of job mailxhg1.getmail on the current day, run the following command:
morestdl -first -name mailxhg1.getmail

To display the standard list file for the second run of job mailxhg1.getmail on the current day, run the following command:
morestdl -num 2 -name mailxhg1.getmail

To display the standard list files for all runs of job mailxhg1.getmail from three days ago, run the following command:
morestdl -day 3 -name mailxhg1.getmail

To display the standard list file for the last run of job mailxhg1.getmail from four days ago, run the following command:
morestdl -day 4 -last -name mailxhg1.getmail

To print the standard list file for job number 455, run the following command:
morestdl 455 | lp -p 6
**parms**

Returns the current value of a parameter, changes the value of a parameter, or adds a new parameter.

**Synopsis**

```bash
parms [-v | -u]
```

```bash
parms name
```

```bash
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**

To return the value of **myparm**, run the following command:

```bash
parms myparm
```

To change the value of **myparm**, run the following command:

```bash
parms -c myparm "item 123"
```

To create a new parameter named **hisparm**, run the following command:

```bash
parms -c hisparm "item 789"
```

To change the value of **myparm** and add **herparm**, run the following command:

```bash
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**

Extracts information about parameters from the scheduler 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Max Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>parameter name</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>parameter value</td>
<td>64</td>
</tr>
</tbody>
</table>

**Examples**

To extract information about all parameter definitions and direct the output to the file `painfo`, run the following command:

```
paxtract -o painfo
```
prxtract

Extracts information about prompts from the scheduler 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Max Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>prompt name</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>prompt value</td>
<td>200</td>
</tr>
</tbody>
</table>

Examples
To extract information about all prompt definitions and direct the output to the file prinfo, run the following command:

prxtract -o prinfo
r11xtr

Extracts information about job streams from the scheduler 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 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`, long names are truncated to eight characters and a plus sign. For example: `A1234567+`.

Each job stream record contains tab-delimited, variable length fields. The fields are described in the following table.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Max Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>job stream name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>job stream date (yymmdd)</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>estimated cpu seconds</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>multiple workstation flag (* means some jobs run on other workstations)</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>number of jobs</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>day of week (Su, Mo, Tu, We, Th, Fr, Sa)</td>
<td>2</td>
</tr>
</tbody>
</table>

**Examples**

To extract information about job streams on June 1999 for workstation `main`, run the following command:

```
r11xtr -m 0699 -c main
```

To extract information about job streams on June of this year for all workstations, and direct the output to file `r11out`, run the following command:

```
r11xtr -m 06 -o r11out
```
release

Releases units of a resource at the job stream or job level.

Synopsis
release -v | -u

release [-s] [workstation#]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.

workstation#
      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:

```bash
# Release 5 units of discio:
`maestro`/bin/release discio

# Release 3 units of discio:
`maestro`/bin/release discio
```

...
rextract

Extracts information about resources from the scheduler 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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Max Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>workstation name</td>
<td>8/16</td>
</tr>
<tr>
<td>2</td>
<td>resource name</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>total resource units</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>resource description</td>
<td>72</td>
</tr>
</tbody>
</table>

**Examples**
To extract information about all resource definitions and direct the output to the file **reinfo**, run the following command:
rextract -o reinfo
rmstdlist

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

To display the names of standard list file directories that are more than 14 days old, run the following command:

rmstdlist -p 14

To remove all standard list files (and their directories) that are more than 7 days old, run the following command:

rmstdlist 7
showexec
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 runs.
Schedule The name of the job stream in which the job runs.
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 run.

info format:
CPU The workstation on which the job runs.
Schedule The name of the job stream in which the job runs.
Job The job name.
Job# The job number.
JCL The file name of the job.

Examples
To display executing jobs in the standard format, run the following command:
showexec

To display executing jobs in the info format, run the following command:
showexec -info
StartUp

Starts the scheduler 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, 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 198 for more information.

Examples
To display the command name and version, run the following command:
StartUp -v

To start the Netman process, run the following command:
StartUp
version
Displays 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
IBM Tivoli Workload Scheduler file information is contained in the version.info file. This file is placed in the TWS/home/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
To display information about all files, run the following command:

```
version/version -a -f version/version.info
```

To display information about the file `customize`, run the following command:

```
cd version
./version customize
```

To display information about the file `customize`, when `version.info` is located in `/apps/maestro`, run the following command:

```
cd version
./version -f /apps/maestro/version.info customize
```
wmaeutil

Used to stop the connector server for the plan, database, and engine. The makesec command will not run successfully on Windows platforms until the connectors are stopped.

Synopsis

UNIX:

wmaeutil.sh instance_name [-stop DB | PL | EG | "*" ] [-version DB | PL | EG | "*" ] [-dbinfo DB | PL] [-sethome] [-gethome] [ALL -stop]

Windows:

wmaeutil.cmd instance_name [-stop DB | PL | EG | "*" ] [-version DB | PL | EG | "*" ] [-dbinfo DB | PL] [-sethome] [-gethome] [ALL -stop]

Arguments

instance_name
The name of the scheduler instance. This refers to the instance name you entered during installation of the scheduler 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 scheduler database and plan to which this connector is linked is expanded or non-expanded. With IBM Tivoli Workload Scheduler, Version 8.2, databases and plans are always expanded, but this option exists for backward compatibility.

-sethome
This option is used to set TWSHome attribute of the scheduler objects (Engine, Database, and Plan) in Tivoli’s object database. This attribute value links connectors for the specified object instance to the core scheduler product. It takes fully qualified name of the scheduler 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 the 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 scheduler connector instances connected to the current scheduler installation, that is, it stops the connector servers for all instances whose TWSHome attribute matches the home directory of the scheduler current installation.
Usage Notes

Set environment variables: Before wmaeutil can be run successfully, you must:
1. Set the Tivoli Management Framework environment:
   - On Windows: c:\>%SystemRoot%\system32\drivers\etc\Tivoli\setup_env.cmd
   - On UNIX: $ . /etc/Tivoli/setup_env.sh
2. Set the IBM Tivoli Workload Scheduler environment:
   - On Windows: TWShome\tws_env.cmd

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 platforms until the connectors are stopped. You should also stop the connectors when using the makesec command on UNIX.

IBM Tivoli Workload Scheduler instance name: If you do not remember the instance name that was entered at installation, 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 Windows)
2. Run the wlookup command to get the scheduler instance name:
   - wlookup -ar MaestroEngine
   - maestro2 1697429415.1.596@Maestro::Engine#
   - where maestro2 is the scheduler instance name.

Examples
To stop the connectors for the database, plan, and engine for an instance called maestro, run the following command:
- wmaeutil.sh maestro ALL -stop

To stop the connectors for the database for an instance called tws, run the following command:
- wmaeutil.sh tws ALL -stop DB

To stop the connector versions for the database, plan, and engine for an instance called maestro2, run the following command:
- wmaeutil.sh maestro2 -version *
xrxtrct

Extracts information about cross-references from the scheduler 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 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, long names are truncated to eight characters and a plus sign. For example: A1234567+.

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>03</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>job name</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>job stream name</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>not used</td>
<td>240</td>
</tr>
<tr>
<td>6</td>
<td>dependent job stream name</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>dependent job stream name</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>dependent job stream name</td>
<td>16</td>
</tr>
<tr>
<td>9</td>
<td>dependent job name</td>
<td>40</td>
</tr>
<tr>
<td>10</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>12</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>13</td>
<td>end-of-record (null)</td>
<td>1</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>08</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>job name</td>
<td>120</td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>02</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>job stream name</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>not used</td>
<td>248</td>
</tr>
<tr>
<td>5</td>
<td>dependent job stream workstation name</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>dependent job stream name</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>end-of-record (null)</td>
<td>1</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>07</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>file name</td>
<td>256</td>
</tr>
<tr>
<td>4</td>
<td>dependent job stream workstation name</td>
<td>16</td>
</tr>
<tr>
<td>5</td>
<td>dependent job stream name</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>dependent job workstation name</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>dependent job name</td>
<td>40</td>
</tr>
<tr>
<td>8</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>9</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>not used</td>
<td>8</td>
</tr>
</tbody>
</table>
**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>04</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>job name</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>not used</td>
<td>248</td>
</tr>
<tr>
<td>5</td>
<td>job stream workstation name</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>job stream name</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>end-of-record (null)</td>
<td>1</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>05</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>prompt name or text</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>not used</td>
<td>236</td>
</tr>
<tr>
<td>5</td>
<td>dependent job stream workstation name</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>dependent job stream name</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>dependent job workstation name</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>dependent job name</td>
<td>40</td>
</tr>
<tr>
<td>9</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>end-of-record (null)</td>
<td>1</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>06</td>
<td>2</td>
</tr>
</tbody>
</table>
Field Description Length (bytes)
2 workstation name 16
3 resource name 8
4 not used 248
5 dependent job stream workstation name 16
6 dependent job stream name 16
7 dependent job workstation name 16
8 dependent job name 40
9 units allocated 6
10 not used 6
11 not used 8
12 end-of-record (null) 1

**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>job stream name</td>
<td>16</td>
</tr>
<tr>
<td>4</td>
<td>not used</td>
<td>248</td>
</tr>
<tr>
<td>5</td>
<td>workstation name (same as 2 above)</td>
<td>16</td>
</tr>
<tr>
<td>6</td>
<td>job stream name (same as 3 above)</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>8</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>10</td>
<td>not used</td>
<td>6</td>
</tr>
<tr>
<td>11</td>
<td>not used</td>
<td>8</td>
</tr>
<tr>
<td>12</td>
<td>end-of-record (null)</td>
<td>1</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
<th>Length (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>01</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>3</td>
<td>ON/EXCEPT name or date</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>except flag (*=EXCEPT)</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>not used</td>
<td>247</td>
</tr>
<tr>
<td>6</td>
<td>workstation name</td>
<td>16</td>
</tr>
<tr>
<td>7</td>
<td>job stream name</td>
<td>16</td>
</tr>
<tr>
<td>8</td>
<td>not used</td>
<td>8</td>
</tr>
</tbody>
</table>

Chapter 6. Utility commands  263
### Examples

To extract information about all cross-references, run the following command:

```bash
xrxtrct
```
Unsupported commands

The following unsupported utility commands provide functions on Windows that are similar to UNIX `ps` and `kill` commands. They can be used if similar Windows 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`.

**Note:** When run by the Administrator, `killproc` is capable of killing system processes.
Chapter 7. The report commands

This chapter describes the report commands that allow you to get summary or detailed information about the previous or next production day.

Report commands

IBM Tivoli Workload Scheduler report commands are listed in the following table.

<table>
<thead>
<tr>
<th>Command</th>
<th>Report</th>
</tr>
</thead>
<tbody>
<tr>
<td>rep1</td>
<td>Report 01 - Job Details Listing</td>
</tr>
<tr>
<td>rep2</td>
<td>Report 02 - Prompt Listing</td>
</tr>
<tr>
<td>rep3</td>
<td>Report 03 - Calendar Listing</td>
</tr>
<tr>
<td>rep4a</td>
<td>Report 04A - Parameter Listing</td>
</tr>
<tr>
<td>rep4b</td>
<td>Report 04B - Resource Listing</td>
</tr>
<tr>
<td>rep7</td>
<td>Report 07 - Job History Listing</td>
</tr>
<tr>
<td>rep8</td>
<td>Report 08 - Job Histogram</td>
</tr>
<tr>
<td>rep11</td>
<td>Report 11 - Planned Production Schedule</td>
</tr>
<tr>
<td>reptr</td>
<td>Report 09A - Planned Production Summary</td>
</tr>
<tr>
<td></td>
<td>Report 09B - Planned Production Detail</td>
</tr>
<tr>
<td></td>
<td>Report 09D - Planned Production Detail (Long Names)</td>
</tr>
<tr>
<td></td>
<td>Report 10A - Actual Production Summary</td>
</tr>
<tr>
<td></td>
<td>Report 10B - Actual Production Detail</td>
</tr>
<tr>
<td>xref</td>
<td>Report 12 - Cross Reference Report</td>
</tr>
</tbody>
</table>

Command output

The output of the report commands is controlled by the following variables:

MAESTROLP

Specifies the destination of the output of a command. The default is stdout. You can set it to any of the following:

filename

Write the output to a file.

> filename

UNIX only. Redirect output to a file, overwriting the contents of the file. If the file does not exist it is created.

>> filename

UNIX only. Redirect output to a file, appending to the end of file. If the file does not exist it is created.

| command |

UNIX only. Pipe output to a system command or process. The system command is always run.

| | command |

UNIX only. Pipe output to a system command or process. The system command is not run if there is no output.
MAESTRO_OUTPUT_STYLE
Specifies the output style for long object names. Set the variable to LONG to use full length (long) fields for object names.

If it is not set or it is set to anything other than LONG, long names are truncated to eight characters and a plus sign. For example: A1234567+.

Changing the date format
In IBM Tivoli Workload Scheduler, the date format affects all commands that accept a date as an input option (except the datecalc command), and the headers in all reports. The default date format is mm/dd/yy. To select a different format, edit the date format local option. The values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>To select</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>yy/mm/dd</td>
</tr>
<tr>
<td>1</td>
<td>mm/dd/yy</td>
</tr>
<tr>
<td>2</td>
<td>dd/mm/yy</td>
</tr>
<tr>
<td>3</td>
<td>Native language support variables.</td>
</tr>
</tbody>
</table>

See the IBM Tivoli Workload Scheduler Planning and Installation Guide for details on modifying local variables.
**rep1 - rep4b commands**

These commands print the following reports:

Report 01 - Job Details Listing

Report 02 - Prompt Messages Listing

Report 03 - User Calendar Listing

Report 04A - User Parameters Listing

Report 04B - Maestro Resource Listing

**Synopsis**

rep\[x\] [-v | -u]

**Arguments**

\(x\) A number corresponding to the report. The numbers are: 1, 2, 3, 4a, or 4b.

-\(u\) Display the command version and exit.

-\(v\) Display command usage information and exit.

**Examples**

Print Report 03, User Calendar Listing:

rep3

Display usage information for the rep2 command:

rep2 -u

On UNIX, print two copies of report 04A, User Parameters Listing, on printer lp2:

MAESTROLP="| lp -d lp2 -n 2"

export MAESTROLP
rep4a
rep7 command

This command prints Report 07-Job History Listing.

Synopsis
rep7 -v | -u

rep7 [-c wkstat] [-s jstream] [-j job] [-f date -t date] [-l]

Arguments
-u Display the command version and exit.
-v Display command usage information and exit.
-c wkstat
  Specifies the name of the workstation on which the jobs run. The default is all workstations.
-s jstream
  Specifies the name of the job stream in which the jobs run. The default is all job streams.
-j job Specifies the name of the job. The default is all jobs.
-f date Specifies to print job history from this date forward. Enter the date as yyyyymmdd. The default is the earliest available date.
-t date Specifies to print job history up to this date. Enter the date as yyyyymmdd. The default is the most recent date.
-l Limits the summary line information to the jobs which fall in the date range specified by the -f or -t options. Using this option causes the order of output to be reversed: the job summary line will be printed after the individual job run lines. This option is valid only if you also specify at least one of the -f or -t options.

Examples
Print all job history for workstation ux3:
rep7 -c ux3

Print all job history for all jobs in job stream sked25:
rep7 -s sked25

Print job history for all jobs in job stream mysked on workstation x15 between 1/21/99 and 1/25/99:
rep7 -c x15 -s mysked -f 19990121 -t 19990125
**rep8 command**

This command prints Report 08-Job Histogram.

**Synopsis**

```
rep8 -v | -u
rep8 [-f date -b time -t date -e time] [-i file] [-p ]
rep8 [-b time -e time] [-i file] [-p ]
```

**Arguments**

- **-u** Display the command version and exit.
- **-v** Display command usage information and exit.
- **-f date** Specifies to print job history from this date forward. Enter the date as `yyyymmdd`. The default is today’s date.
- **-b time** Specifies to print job history from this time forward. Enter the time as `hhmm`. The default is the scheduler start of day time.
- **-t date** Specifies to print job history up to this date. Enter the date as `yyyymmdd`. The default is the most recent date.
- **-e time** Specifies to print job history up to this time. Enter the time as `hhmm`. The default is the scheduler start of day time.
- **-i file** Specifies the name of the log file from which job history is extracted. Note that log files are stored in the `schedlog` directory. The default is the current plan (Symphony file).

**Note:** Ensure that the time range specified by the `[-f date -b time -t date -e time]` arguments is within the date and time range defined in the `-i file` log file argument.

- **-p** Specifies to insert a page break after each run date.

**Examples**

Print a job histogram which includes all information in the current plan (Symphony file):

```
rep8
```

Print a job histogram beginning at 6:00 a.m. on 1/25/99, and ending at 5:59 a.m. on 1/26/99:

```
rep8 -f 19990125 -b 0600 -t 19990126 -e 0559 -i schedlog/M199801260601
```

Print a job histogram, from the current plan (Symphony file), beginning at 6:00 am, and ending at 10:00 pm:

```
rep8 -b 0600 -e 2200
```
**rep11 command**

This command prints Report 11-Planned Production Schedule.

**Synopsis**

`rep11 -v | -u`

`rep11 [-m mmm[yy]] [-c wkstat [...] [-o file]`

**Arguments**

|-u Display the command version and exit.
|-v Display command usage information and exit.
|-m mmm[yy] Specifies the months to be reported. Enter the month number as mm. The default is the current month. You can also enter a year as yy. The default is the current year or next year if you specify a month earlier than the current month.
|-c wkstat Specifies workstations to be reported. The default is all workstations.
|-o file Specifies the output file. The default is the file defined by the MAESTROLP variable. If MAESTROLP is not set, the default is stdout.

**Examples**

Report on June, July, and August of 1999 for workstations main, site1 and sagent1:

```
rep11 -m 0699 0799 0899 -c main site1 sagent1
```

Report on June, July, and August of this year for all workstations, and direct output to the file r11out:

```
rep11 -m 06 07 08 -o r11out
```

Report on this month and year for workstation site2:

```
rep11 -c site2
```
reptr command

This command prints the following reports:
- Report 09A - Planned Production Summary
- Report 09B - Planned Production Detail
- Report 10A - Actual Production Summary
- Report 10B - Actual Production Detail

Synopsis
reptr [-v | -u]
reptr -pre [-{summary | detail}] [symfile]
reptr -post [-{summary | detail}] [logfile]

Arguments
-u Display the command version and exit.
-v Display command usage information and exit.
-pre Specifies to print the pre-production reports (09A, 09B).
-post Specifies to print the post-production reports (10A, 10B).
-summary Specifies to print the summary reports (09A, 10A). If -summary and -detail are omitted, both sets of reports are printed.
-detail Specifies to print the detail reports (09B, 10B). If -summary and -detail are omitted, both sets of reports are printed.
symfile Specifies the name of the plan file from which reports will be printed. The default is Symnew in the current directory.
logfile Specifies the name of the log file from which the reports will be printed. Note that plan log files are stored in the schedlog directory. The default is the current plan (Symphony file).

If the command is run with no options, all pre and post reports are printed.

Examples
Print the pre-production detail report from the Symnew file:
reptr -pre -detail

Print the pre-production summary report from the file mysym:
reptr -pre -summary mysym

Print the post-production summary report from the log file M199903170935:
reptr -post -summary schedlog/M199903170935

Print all pre and post-production reports.
reptr

The pre-production reports are based on information read from the Symnew file. The post-production reports are based on information read from the Symphony file.
**xref command**

This command prints Report 12-Cross Reference Report.

**Synopsis**

xref [-V|U]

xref [-cpu wkstat] [-depends | -files | -jobs | -prompts | -resource | -schedules | -when [...]]

**Arguments**

-U Display the command version and exit.

-V Display command usage information and exit.

-cpu wkstat
  Specifies to print the report for the named workstation. The @ wildcard is permitted, in which case, information from all qualified workstations is included. The default is all workstations.

-depends
  Specifies to print a report showing the job streams and jobs that are successors of each job.

-files
  Specifies to print a report showing the job streams and jobs that are dependent on each file.

-jobs
  Specifies to print a report showing the job streams in which each job is run.

-prompts
  Specifies to print a report showing the job streams and jobs that are dependent on each prompt.

-resource
  Specifies to print a report showing the job streams and jobs that are dependent on each resource.

-schedules
  Specifies to print a report showing the job streams and jobs that are successors of each job stream.

-when
  Specifies to print a report showing job stream Include and Exclude dates.

If the command is run with no options, all workstations and all options are selected.

**Examples**

Print a report for all workstations, showing all cross-reference information:

xref

Print a report for all workstations. Include cross-reference information about all successor dependencies:

xref -cpu @ -depends -schedules
Sample reports

Report 01 — Job Details Listing:

TWS for UNIX (SOLARIS)/REPORT1 8.2 (1.8)
IBM Page 1
Report 01 Job Details
Listing 06/11/03
Job: SUN001 #ACC
Description:
JCL File: ^ACCHOME^
Logon: ^ACCLOGIN^ Creator: maestro
Recovery Job :
Recovery Type : STOP
Recovery Prompt :
Composer Autodoc : Yes
Total Runs : 0 - 0 Successful, 0 Aborted
Elapsed(mins) CPU(secs)
Total 00:00:00 0
Normal 00:00:00
Last Run 00:00:00 0 (On at 00:00)
Maximum 00:00:00 0 (On )
Minimum 00:00:00 0 (On )

Report 02 — Prompt Listing:

TWS for UNIX (SOLARIS)/REPORT2 8.2 (1.6) IBM
Page 1
Report 02 Prompt Message Listing
06/11/03
Prompt Message
ACCRES Reply YES when ready to run acc103 and acc104.
ACCUSERS Have all ACC users logged out?
CALLNO 555-0911
CALLOPER Call ^PERSONONCALL^ at ^CALLNO^ to ensure all time cards have been processed.
HRSRES Reply YES when ready to run hrs103 and hrs104.
MISRES Reply YES when ready to run mis103 and mis104.
PERSONONCALL Lou Armstrong
Total number of prompts on file: 7

Report 03 — Calendar Listing:

TWS for UNIX (SOLARIS)/REPORT3 8.2 (1.7) IBM
Report 03 User Calendar Listing 06/22/03
Calendar Type: PAYDAYS Description: Company Paydays
Oct 2003 Nov 2003
Sun Mon Tue Wed Thu Fri Sat Sun Mon Tue Wed Thu Fri
... . . . . . . . . . . .
12 . . . . . . . . . .
... . . . . . . . . . .
... . . . . . . . . . .27 28

Report 04A — Parameter Listing:

TWS for UNIX (SOLARIS)/REPORT4A 8.2 (1.6) IBM
Page 1
Report 4A User Parameter Listing
06/11/03
Parameter Name Contents
ACCHOME /usr/local/Tivoli/maestro/scripts
ACCLOGIN maestro
BADEXIT 99
GOODEXIT 0
LONG 2000
SCRPATH /usr/local/Tivoli/maestro/scripts
SHORT 20
Number of Parameters on file: 7

Report 04B — Resource Listing:
TWS for UNIX (SOLARIS)/REPORT4B 8.2 (1.6) IBM
Page 1
Report 4B TWS Resources Listing
06/11/03
Resource Number
CPU Name Avail Description
SUN001 #ACCDB 1 Database
SUN001 #ACCODB 5 Maestro Class Resource
SUN001 #ACCONE 1 Maestro Class Resource
SUN001 #ACCPRT 2 Printers
SUN001 #ACCTAPE 1 Tape Drive
SUN001 #HRSODB 5 Maestro Class Resource
SUN001 #HRSONE 1 Maestro Class Resource
SUN001 #MISODB 5 Maestro Class Resource
SUN001 #MISONE 1 Maestro Class Resource
Number of Resources on file: 9

Report 07 — Job History Listing:
TWS for UNIX (SOLARIS)/REPORT7 8.2 (1.10) IBM
Page 1
Report 07 Job History
Listing 06/11/03
Date Time Schedule Name Elapsed CPU
Status
Job:SUN001 #ACCJOB01 Runs: Aborted 1 Successful 1 Elapsed
Time: Normal 2 Min 2 Max 2
06/10/03 09:36 SUN001 #ACCW01 2 1
SU
06/10/03 10:32 SUN001 #ACCI1 2 1
AB
Job:SUN001 #ACCJOB02 Runs: Aborted 1 Successful 1 Elapsed
Time: Normal 2 Min 1 Max 2
06/10/03 09:44 SUN001 #ACCW01 2 1
SU
06/10/03 10:31 SUN001 #ACCI1 1 1
AB

Report 08 — Job Histogram:
TWS for UNIX (SOLARIS)/REPORT8 8.2 (1.7) IBM
Page 1
Report 08 Job Histogram 06/10/03 12:41 - 06/11/03 12:40
06/11/03
Interval Per Column: 15 minutes
2 0 0 0 0 0 0 0 1 1
CPU is SUN001 3 0 2 3 5 6 8 9 1 2
1 4 1 4 1 4 1 4 1 4
Job Name Stat 1 1 1 1 1 1 1 1 1 1 0
06/11/03
TESTRET+,ACCJOB03SU .
ACCI02 ,ACCC0B14AB .
HRSTW02 .HRSJOB14AB .
MISW02 .MISJ0B14AB .
JOBS .ACCENV SU .

Report 11 — Planned Production Schedule:
TWS for UNIX/REPII 8.2
AWSBJA001I Licensed Materials Property of IBM
5698-WKB
Report 12 — Cross Reference Report:

TWS for UNIX (SOLARIS)/CROSSREF 8.2 (1.5)
IBM Page 1
Report 12 Cross Reference Report for Job
Dependencies. 06/13/03
CPU: SUN001
Job Name Dependencies
ACCR01 #ACCJOB01 ACCR01 .ACCJOB02
ACCD01 #ACCJOB01 ACC01 .ACCJOB04
ACCR01 #ACCJOB02 ACCR01 .ACCJOB03
ACCR01 #ACCJOB03 ACCR01 .ACCJOB04
ACCR01 #ACCJOB03 ACCW01 .ACCJOB01
ACCR01 #ACCJOB04 ACCW01 .ACCJOB02
ACCR01 #ACCJOB05 ACCR01 .ACCJOB06
ACCD02 #ACCJOB11 ACCW02 .ACCJOB12
ACCR02 #ACCJOB12 ACCW02 .ACCJOB13
ACCD02 #ACCJOB14 ACCW02 .ACCJOB15

Chapter 7. The report commands 277
Chapter 8. The Extended Agent reference

This chapter describes the Extended Agent Interface and provides information for programmers who want to create custom access methods. Extended Agents are used to extend the job scheduling functions of IBM Tivoli Workload Scheduler to other systems and applications like the following: Local or Remote UNIX Systems, Peoplesoft, SAP R/3, z/OS, OPC, Oracle CCM, and VMS. This chapter contains information about the following:

- What are Extended Agents?
- Access Method Interface
- Method Execution
- Troubleshooting

What are Extended Agents?

Extended agents are used to extend the job scheduling functions of IBM Tivoli Workload Scheduler to other systems and applications.

An extended agent is defined as a workstation that has a host and an access method. The host is any other workstation, except another extended agent. The access method is an IBM-supplied or user-supplied script or program that is run 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 runs 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 an IBM Tivoli Workload Scheduler physical workstation, either a master, domain manager, FTA, or standard agent. 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.

Access method interface

The interface between IBM Tivoli Workload Scheduler and an access method consists of information passed to the method on the command line, and of messages returned to the scheduler in stdout.

Method command line syntax

The 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: TWShome/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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Options</th>
<th>Task String</th>
</tr>
</thead>
<tbody>
<tr>
<td>-t</td>
<td>-c -n -p -r -s -d -l -o -j -q -w -S</td>
<td></td>
</tr>
<tr>
<td>LJ</td>
<td>X X X X</td>
<td>X X X X X X X X X</td>
</tr>
<tr>
<td>MJ</td>
<td>X X X X</td>
<td>X X X X</td>
</tr>
<tr>
<td>CF</td>
<td>X X X</td>
<td></td>
</tr>
<tr>
<td>GS</td>
<td>X X X X</td>
<td>X X</td>
</tr>
</tbody>
</table>

-c xagent,host,master
Specifies the 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 the 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 (ymmd) 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 the 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 the scheduler waits to get a reply on an external job before sending a SIGTERM signal to the access method. The default is 300.

-S new name
  Specifies that the job is rerun using this name in place of the original job name. Within a job script, you can use the jobinfo command to return the job name and run the script differently for each iteration. See the description of the conman rerun command in the IBM Tivoli Workload Scheduler Reference Guide for more information.

--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 the 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 the 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 the scheduler in a %CJ response to a previous GS task.
Method response messages

Methods return information to IBM Tivoli 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]

%RC rc

%UT [errormessage]

where:

CJ Changes the job state.

state The state to which the job is changed. All 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 the scheduler will include in any MJ task associated with the job. See 281

jobid A string of up to 64 characters that the scheduler will include in any GS task associated with the job. See 281

JS [cputime] Indicates successful completion of a job and provides its elapsed run time in seconds.

RC rc rc is a number that is interpreted by IBM Tivoli Workload Scheduler as the return code of the extended agent job. The return code is taken into account only if a return code condition was specified in the definition of the extended agent job. If this is not the case, it is ignored and the successful completion of the extended agent job is indicated by the presence of message %JS [cputime]. Likewise, if the method does not send the %RC message, then the successful completion of the extended agent job is indicated by the presence of message %JS [cputime].

UT [errormessage] Indicates that the requested task is not supported by the method. Displays a string of up to 255 characters that the 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. The scheduler reads the file, if it exists, before executing a method. If the file is modified after the scheduler is started, the changes take effect when it is stopped and restarted.
The file can contain scheduler options and any other method information. The options recognized by the 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 it is the user name of the account in which the scheduler was installed.

**GSuser**=username

Specifies the login to use for the GS tasks. The default is root for Unix, and for Windows it is the user name of the account in which the scheduler was installed.

**GStimeout**=seconds

Specifies the amount of time, in seconds, the 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 computer, these users must be defined as 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 path name of an options file for a method named netmth is TWS\home\methods\netmth.opts.

---

**Method execution**

The following topics describe the interchange between IBM Tivoli Workload Scheduler and an access method.

**Launch job (LJ) task**

The LJ task instructs the extended agent method to launch a job on an external system or application. Before running the method, IBM Tivoli 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, it is set to %SYSTEM\%SYSTEM32.

TZ  The time zone.

If the method cannot be run, 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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Method Response</th>
<th>IBM Tivoli Workload Scheduler Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>LJ and MJ</td>
<td>%CJ state [mjstring]</td>
<td>Sets job state to state. Includes mjstring in any subsequent MJ task.</td>
</tr>
<tr>
<td></td>
<td>%JS [cputime]</td>
<td>Sets job state to succ.</td>
</tr>
<tr>
<td></td>
<td>Exit code=non-zero</td>
<td>Sets job state to abend.</td>
</tr>
<tr>
<td></td>
<td>%UT [errormessage] and Exit code=2</td>
<td>Sets job state to abend and displays errormessage.</td>
</tr>
</tbody>
</table>

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 the scheduler determines that the LJ task terminated unexpectedly. The 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 283 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 the 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 scheduler 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 method to check the availability of a file on the external system. Before running the method, the 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, the user name of the account in which the scheduler was installed is used. If the method cannot be run, 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 run.

Once it is running, the method runs 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.

<table>
<thead>
<tr>
<th>Task</th>
<th>Method Response</th>
<th>IBM Tivoli Workload Scheduler Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>CF</td>
<td>Exit code=0</td>
<td>Set file state to YES.</td>
</tr>
<tr>
<td></td>
<td>Exit code=non-zero</td>
<td>Set file state to NO.</td>
</tr>
<tr>
<td></td>
<td>%UT [errmsg] and Exit code=2</td>
<td>Set file state to NO.</td>
</tr>
</tbody>
</table>

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 GSusan 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, the user name of the account in which the scheduler was installed is used. If the method cannot be run, the dependent job or job stream is not allowed to run. 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-run 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 the scheduler will wait for a response before killing the method. See “Method options file” on page 282 for more information.

Method responses are summarized in the following table:

<table>
<thead>
<tr>
<th>Task</th>
<th>Method Response</th>
<th>IBM Tivoli Workload Scheduler Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>GS</td>
<td>%CJ state [jobid]</td>
<td>Sets job state to state and includes jobid in any subsequent GS task.</td>
</tr>
<tr>
<td></td>
<td>%UT [errmsg] and Exit code=2</td>
<td>Job state is unchanged.</td>
</tr>
</tbody>
</table>

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" on page 225 for complete command information.

Troubleshooting

The following topics are provided to help troubleshoot and debug extended 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 scheduler jobs, messages are written to the scheduler standard list file. For information regarding a problem of any kind, check these files.

Method not executable

If an access method cannot be run, 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 the 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, #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, #jobnumber for wkstation using methodname.
"first 64 characters of the offending message"
Chapter 9. The Network Agent reference

IBM Tivoli Workload Scheduler internetwork dependencies permit jobs and job streams in a local scheduler network to use jobs and job streams in a remote network as follows dependencies. This chapter describes how to use internetwork dependencies. This chapter contains information about the following:

- Overview
- Configuring a Network Agent Workstation
- Internetwork Dependencies

Overview

A network agent is a logical workstation definition that allows follows dependencies between a local scheduler network and a remote scheduler 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 scheduler network. This type of dependency is called an internetwork dependency.

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, netmeth. 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 netmeth.opts.

The access method is invoked by the scheduler 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. The scheduler 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 specify an internetwork dependency, you must create a scheduler 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.

Using the Job Scheduling Console, define a Network Agent workstation in the scheduler database as follows:

1. In the Actions List pane of the Job Scheduling Console, open the New Workstation switch.
2. Click the engine name. 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 16 alphanumeric characters. The name can include a dash and underscore, but must begin with a letter, such as MasterB.

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 the port number used by the master domain manager in the remote network. In the example that follows, the port number used by MasterA.

6. In the Operating System field, select 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 Ignore check box blank.

12. In the Access Method field, enter netmeth.

13. In the Host field, specify the node name of the host workstation (the MasterB hostname in the example) or select one from the list provided by clicking the Workstations button.

14. Click OK. The Network Agent workstation is saved in the MasterB (local network) scheduler database.

The following example shows how to define a network agent workstation for a remote network, Network A, that allows local network, Network B, use jobs and job streams in the remote network as follows dependencies.

Remote Network A

\[ \rightarrow \]

Local Network B

MasterA

\[ \rightarrow \]

MasterB

Network Agent

*Figure 7. Local and remote networks*

The following steps show how to define a network agent workstation named NetAgent that uses the netmth access method.
1. Define MasterA cpu on the MasterA database. The user of this cpu is tws_masterA.
2. Specify the TCP port number for MasterA as 12345.
4. Define MasterB cpu on the MasterB database. The user of this cpu is tws_masterB.
5. Specify the node for MasterB as MasterB.rome.tivoli.com.

The network agent workstation definition is saved in the Master B database.

**Network Agent command line example**

The following example shows the command line workstation definition for the NetAgent network agent in [Figure 7 on page 290](#).

```
CPUNAME NETAGENT
DESCRIPTION "NETWORK AGENT"
OS OTHER
NODE MASTERA.ROME.TIVOLI.COM
TCPADDR 12345

(The NetAgent cpu listens on the MasterA port.)
FOR maestro
HOST MASTERB
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 the scheduler.

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 computer, this user must be defined in the scheduler.
- `seconds` The amount of time, in seconds, the scheduler 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 on MasterB for method netmth is as follows:

```
GSuser=tws_masterA
GStimeout=600
```
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 a job stream

To add an internetwork dependency to a job stream:
1. Open the Graph view, by right-clicking a job stream and selecting Open from the pop-up menu.
2. In the Graph view, click the Add Dependency on Internetwork button in the toolbar. The Internetwork Dependency window is displayed.
3. Click the Find (...) button and use the Find Workstation window to select the name of the network agent.
4. In the Dependency field, specify either the freeform dependency, or the job or job stream predecessor in the format workstation#jobstream.job. The maximum length of this field is:
   • 120 for freeform characters
   • 16 for the workstation
   • 16 for the job stream
   • 40 for the job
5. Click OK to close the window. If you are adding a new internetwork dependency, a new internetwork dependency button is added in the Graph view.
6. Save the job stream.

Using the command line

Internetwork dependencies can be included in job streams composed with the command line Composer. For example, to create a follows dependency to a job and job stream using the follows keyword complete the following steps:

• Job stream internetwork dependency
  1. Define a job stream called schedA on the MasterA database.
  2. Define a job stream called schedB on the MasterB database with a dependency on schedA.

Use the composer command line to define the job stream internetwork dependency as follows:

```
schedule schedB
  on everyday
    follows NetAgt::MasterA#schedA
;
end
```
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 in the plan. The dependencies are specified as they are for other follows dependencies. Refer to "Creating an internetwork dependency to a job stream" on page 292 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:

ENNnmmss

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.

**Taking action on external jobs**
You can take three actions on remote jobs in an EXTERNAL schedule: Cancel, Rerun, and Confirm.

**Note:** None of these commands have any effect on the remote job or schedule on the remote network. They simply manipulate the dependency for the local network.

**Cancel**
Cancels the EXTERNAL job, releasing the dependency for all local jobs and schedules. The status of the dependency ceases to be checked.

**Rerun**
Instructs Conman to restart checking the state of the EXTERNAL job. The job state is set to EXTRN immediately after a Rerun is performed.

Rerun is useful for EXTERNAL jobs in the ERROR state. For example, if an EXTERNAL job cannot be launched because the network access method does not grant execute permission, the job enters the ERROR state and its status ceases to be checked. After you correct the permissions, the method can start but Conman will not start checking the EXTERNAL job state until you perform a Rerun.

**Confirm SUCC / ABEND**
Sets the status of the EXTERNAL to SUCC or ABEND, releasing the dependency for all local jobs and schedules. The status of the dependency ceases to be checked.

**Taking action on internetwork dependencies for jobs and schedules**
Internetwork dependencies are listed in the Dependencies column of the SHOWJOBS and SHOWSCHEDULES windows in the following format:

```
net::net_dep
```

where:

<table>
<thead>
<tr>
<th>net</th>
<th>The CPU name of the network agent. The two colons (:) are a required delimiter. Wildcards are valid.</th>
</tr>
</thead>
<tbody>
<tr>
<td>net_dep</td>
<td>The internetwork dependency in the following format:</td>
</tr>
<tr>
<td></td>
<td>[cpu#] sched[job]</td>
</tr>
<tr>
<td></td>
<td>If no CPU is specified, the default is the scheduler CPU to which the network agent is connected. This is determined by the Node and TCP Address fields of the network agent CPU definition. Wildcards are valid.</td>
</tr>
</tbody>
</table>
The Release, Add Dependency, and Delete Dependency actions work the same for internetwork dependencies as they do for other dependencies.

**Conman command line specification**

Conman commands that can specify internetwork dependencies are listed below with an example based on:

- A local scheduler CPU called local1
- A schedule defined for local1 called sched1
- A job in local1#sched1 called job1
- A scheduler network agent called netagt
- A scheduler CPU in the netagt remote network called remote1
- A schedule defined for remote1 called rsched
- A job in remote1#rsched called rjob

**adddep job:** Add a remote job as a Follows dependency to a job:
\[ \text{adj local1#sched1.job1;follows=netagt::remote1#rsched.rjob} \]

**adddep sched:** Add a remote schedule as a Follows dependency to a schedule:
\[ \text{ads local1#sched1;follows=netagt::remote1#rsched} \]

**cancel job:** Cancel all EXTERNAL jobs for a network agent (the two commands are equivalent):
\[ \text{cj netagt#EXTERNAL.0} \]
\[ \text{cj netagt::0} \]

**confirm:** Confirm that an EXTERNAL job has completed successfully:
\[ \text{confirm netagt::remote1#rsched.rjob;succ} \]

**deldep job:** Delete a remote job dependency from a job:
\[ \text{ddj local1#sched1.job1;follows=netagt::remote1#rsched.rjob} \]

**deldep sched:** Delete a remote job dependency from a schedule:
\[ \text{dds local1#sched1;follows=netagt::remote1#rsched.rjob} \]

**release job:** Release a job from an internetwork dependency:
\[ \text{rj local1#sched1.job1;follows=netagt::remote1#rsched.rjob} \]

**release sched:** Release a schedule from an internetwork dependency:
\[ \text{rs local1#sched1;follows=netagt::remote1#rsched.rjob} \]

**rerun:** Rerun an EXTERNAL job (the two commands are equivalent):
\[ \text{rr netagt#EXTERNAL.rjob} \]
\[ \text{rr netagt::remote1#rsched.rjob} \]

**showjobs/info:** Display all the remote dependencies for a network agent with their original names and their scheduler-generated names:
\[ \text{sj netagt#EXTERNAL.0;info} \]

**submit:** Submit a `rm` command into the JOBS schedule with a remote schedule as Follows dependency:
\[ \text{sbd "rm apfile";follows=netagt::remote1#rsched} \]
See Also

For the equivalent Job Scheduling Console task, see "Creating Job Streams" in the IBM Tivoli Workload Scheduler Job Scheduling Console User’s Guide.
Chapter 10. Setting User Security Definitions

IBM Tivoli Workload Scheduler programs and commands determine a user’s capabilities by comparing the user’s name with the user definitions in the Security file. This chapter explains how to write user definitions and manage the Security file.

Centralizing security

The centralised security global option enables the Tivoli Workload Scheduler administrator to choose between the following two security models:

Centralized security

The security files of all the workstations in the Tivoli Workload Scheduler network can be created and modified only on the master domain manager, and the Tivoli Workload Scheduler administrator is responsible for their production, maintenance, and distribution.

Note: Centralized security is not supported if you are working with IBM Tivoli Workload Scheduler non-expanded databases.

Local Security

The security file of each workstation can be managed by its root user or administrator. The local user can run the makesec command to create or update the file.

Set the centralized security global option to ON for the IBM Tivoli Workload Scheduler master to enable the centralized security mechanism. After the global option is active, the Jnextday utility reloads security-related information in the symphony file. The symphony file also records the option as being turned on. After the master domain manager distributes the symphony file across the network, each workstation has a symphony file containing the same security information.

The centralized security mechanism is activated every time a link is established, as well as when an IBM Tivoli Workload Scheduler command is run.

In a network with centralized security, two workstations will not be able to establish a connection if one has centralized security turned off in its symphony file or if their security file information does not match. However, a workstation must always accept incoming connections from its domain manager, even if the security file information sent from the domain manager does not match the information in the workstation’s symphony file. This is enforced to allow the administrator to change the security file on the master without having to redistribute it to the entire network before running Jnextday.

Every time a command is run on the workstation of a network with centralized security, either with conman or from the Job Scheduling Console, the security routines verify that the security information in the symphony file matches the information in the local security file. If it does not, no (or reduced) access is granted to IBM Tivoli Workload Scheduler objects and commands and a security violation message is logged.
Centralized security does not apply to IBM Tivoli Workload Scheduler operations for which the symphony is not required. Commands that do not require the symphony file to run use the local security file. For example, the command `parms`, used to modify or display the local parameters database continues to work according to the local security file, even if centralized security is active and the local security file differs from the centralized security rules.

If a workstation’s security file is deleted and recreated, its security information will not match the information in the security file of the master (that is also recorded in the symphony file). In addition, a mechanism associated with the creation process of the symphony file ensures prevention from tampering with the file.

### Working with the security file

Each workstation in an IBM Tivoli Workload Scheduler network (domain managers, fault-tolerant agents, and standard agents) has its own Security file. You can maintain a file on each workstation, or you can create a security file on the master domain manager and copy it to each domain manager, fault-tolerant agent, and standard agent. Ensure that all IBM Tivoli Workload Scheduler users are assigned the required authorization in the security file.

A template file named `TWS_home/config/Security.conf` is provided with the software. During installation, a copy of the template is installed as `TWS_home/config/Security.conf`, and a compiled, operational copy is installed as `TWS_home/Security.conf`.

### Creating the security file

To create user definitions, edit the template file `TWS_home/Security.conf`. Do not modify the original template in `TWS_home/config/Security.conf`. Use the `makesec` command to compile and install a new operational security file. After it is installed, you can make further modifications by using the `dumpsec` command. See [“dumpsec” on page 312](#) and [“makesec” on page 313](#).

### Modifying the security file

To modify the security file, perform the following steps:

1. Stop the connectors. See [“Stopping connectors to implement changes.”](#)
2. Run `dumpsec` to download the security file. See [“dumpsec” on page 312](#)
3. Modify the syntax of the security file. See [“Security file syntax” on page 299](#)
4. Run `makesec` to upload the security file. See [“makesec” on page 313](#)
5. Stop and start `conman` or `composer` to implement the changes to the security file. See [Chapter 5, “Conman reference,” on page 115](#) or [Chapter 3, “Composer reference,” on page 31](#)

### Stopping connectors to implement changes

On Windows, you must stop the connector before running the `makesec` command. On UNIX, you can stop it either before or after running `makesec`.

Use the `wmaeutil` command to stop the connector.

### Running `wmaeutil`

To run the `wmaeutil` command:

1. Set the Tivoli environment:
   - From a UNIX command line:
     - For ksh:
– For csh:
  
  ```bash
  source /etc/Tivoli/setup_env.sh
  ```

  - From a Windows command line:
    ```bash
    %SYSTEMROOT%\system32\drivers\etc\Tivoli\setup_env.cmd
    ```

2. Enter the following command:

   - On UNIX
     ```bash
     wmaeutil.sh ALL -stop
     ```
   
   - On Windows
     ```bash
     wmaeutil.cmd ALL -stop
     ```

---

## Security file syntax

The Security file contains one or more user definitions.

### User definitions

A user definition defines a set of users, the objects they can access, and the actions they can perform.

### Synopsis

```bash
[ # comment]

user def-name user-attributes

begin [ * comment]

object-type [object-attributes] access[=action[,...]]

[object-type ... ]

[end]
```

### Parameters

```bash
[ # | *] comment
```

- All text following a pound sign or an asterisk and at least one space is treated as a comment. Comments are not copied into the operational Security file installed by the `makesec` command.

- `def-name`
  
  Specifies the name of the user definition. The name can contain up to 36 alphanumeric characters and must start with an alphabetic character.

- `user-attributes`
  
  Specifies one or more attributes that identify the set of users to whom the definition applies. Specify user attributes as follows:

  ```bash
  user-attribute[|+ | ~]user-attribute[,...]
  ```

  Use a plus sign (+) to add an attribute the user or users must have. Use a tilde (~) to add an attribute the user or users must not have. A `user-attribute` is defined as:

  ```bash
  cpu=wkstation | $framework | @ [,...]
  ```

  where:
Specifies the workstation on which the user is logged in. Wildcard characters are permitted. The following IBM Tivoli Workload Scheduler variables can be used:

$master
The user is logged in on the IBM Tivoli Workload Scheduler master domain manager.

$sremotes
The user is logged in on any IBM Tivoli Workload Scheduler standard agent.

$sslaves
The user is logged in on any IBM Tivoli Workload Scheduler fault-tolerant agent.

$thiscpu
The user is logged in on the IBM Tivoli Workload Scheduler workstation on which the secured program is running.

For Job Scheduling Console users, use $framework.

$framework
Specifies the workstation from which the user is running the Job Scheduling Console.

@ Specifies that the user is accessing IBM Tivoli Workload Scheduler with the Job Scheduling Console or is logged in on any IBM Tivoli Workload Scheduler workstation.

group=groupname[,...]
For UNIX users only. Do not use this argument for Job Scheduling Console users. Specifies the UNIX group in which the user is a member. Wildcard characters are permitted.

logon=username | tme-admin | @ [...]
where:

username
Specifies the name with which the user is logged in on an IBM Tivoli Workload Scheduler workstation. Wildcard characters are permitted. The cpu= attribute must be set to a workstation name or @.

tme-admin
Specifies the name of the Tivoli administrators group in which the user is a member. If the name contains spaces, it must be enclosed in double quotation marks. Wildcards are permitted. The cpu= attribute must be set to $framework or @.

@ Specifies that the user is logged in with any name or is a member of any Tivoli administrators group.

object-type
Specifies the type of object the user is given permission to access. The object types are as follows:

calendar User calendars.
cpu Workstations and domains.
You can include multiple object types in a user definition. Omitting an object type prevents access to all objects of that type.

**object-attributes**

Table 11 lists object attributes according to the object type.

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Name</th>
<th>cpu</th>
<th>jcl</th>
<th>logon</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calendar</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>cpu</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>file</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>job</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>parameter</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>prompt</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>resource</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>schedule</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>userobj</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
</tbody>
</table>

Specifies one or more attributes that identify a set of objects of the specified type. If no attributes are specified, all objects of the specified type are accessible. Specify object attributes as follows:

```
object-attribute[ [+ | ~ ] object-attribute[...]]
```

Use a plus sign (+) to add an attribute that objects must have. Use a tilde (~) to add an attribute that objects must not have. An object-attribute can be any of the following:

**name=name[,....]**

Specifies one or more names for the object type. Wildcard characters are permitted. Multiple names must be separated by commas. If this attribute is not specified, all defined objects can be accessed. The following values are relevant to the file object type:

- **calendars**: Contains calendars.
- **cpudata**: Contains workstations and domains.
- **jobs**: Contains jobs.
- **mastsked**: Contains job streams.
- **parameters**: Contains parameters.
- **prodsked**: Contains the production schedule.
- **prompts**: Contains prompts.
- **resources**: Contains resources.
- **security**: The Security file.
- **Symphony**: Contains the production plan.
cpu=workstation[...]

Specifies one or more workstation or domain names. Wildcard characters are permitted. Multiple names must be separated by commas. If omitted, all workstation qualify. The following IBM Tivoli Workload Scheduler variables are permitted: $master, $remotes, $slaves, and $thiscpu. See "The variables supplied with the product" on page 305 for more information.

jcl="path" 1 "cmd"

Specifies the command or the path name of a job object’s executable file. The command or path must be enclosed in quotation marks ("’). Wildcard characters are permitted. If omitted, all job files and commands qualify.

logon=username[...]

Specifies the user names. Wildcard characters are permitted. Multiple names must be separated by commas. If omitted, all user names qualify. For the job type object, the following variables are permitted: $jclowner, $owner, and $user. See "The variables supplied with the product" on page 305 for more information.

action

Specifies the actions users can perform. Multiple actions must be separated by commas. If none are specified, no actions are permitted. Entering access=@ gives users the ability to perform all actions. Table 12 and Table 13 list all the possible action types according to the object type.

Table 12. Actions

<table>
<thead>
<tr>
<th>Action</th>
<th>add</th>
<th>adddep</th>
<th>altpass</th>
<th>altpri</th>
<th>build</th>
<th>cancel</th>
<th>confirm</th>
<th>console</th>
<th>deldep</th>
<th>delete</th>
<th>display</th>
<th>fence</th>
<th>kill</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calendar</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td></td>
</tr>
<tr>
<td>cpu</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td>file</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>job</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>parameter</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>prompt</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>resource</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>schedule</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>useobj</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Table 13. Actions (continued)

<table>
<thead>
<tr>
<th>Action</th>
<th>limit</th>
<th>link</th>
<th>list</th>
<th>modify</th>
<th>release</th>
<th>reply</th>
<th>rerun</th>
<th>shut down</th>
<th>start</th>
<th>stop</th>
<th>submit</th>
<th>unlink</th>
<th>use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Object</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>calendar</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td></td>
</tr>
<tr>
<td>cpu</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>file</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>job</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>parameter</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>prompt</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>resource</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>schedule</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
</tr>
<tr>
<td>useobj</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>no</td>
</tr>
</tbody>
</table>

Notes:

1. The object type file is used to manage security on the CLI and is only valid for the CLI.
2. For the **file** object type, the **clean** action is equivalent to the **build** action.

3. To permit a user to switch the domain manager function to a workstation, the user must have both **start** and **stop** access to the workstation.

   - **add** Add and save new calendars in the database.
   - **adddep** Add dependencies to jobs in the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.
   - **altpass** Alter user passwords in the database.
   - **altpri** Alter the priority of jobs in the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.
   - **build** Build IBM Tivoli Workload Scheduler’s database files. This action is only available from the command line. The key **clean** that appears in the security file for this object is equivalent to the **build** key.
   - **cancel** Cancel jobs in the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.
   - **confirm** Confirm the completion of jobs in the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.
   - **console** View and alter the IBM Tivoli Workload Scheduler console.
   - **deldep** Delete dependencies from jobs in the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.
   - **delete** Delete calendars from the database.
   - **display** Display calendars in the database.
   - **fence** Alter workstation job fences in the production plan.
   - **kill** Kill jobs in the production plan.
   - **limit** Alter workstation job limits in the production plan.
   - **link** Open workstation links.
list

Allows the user to display workstations and domains in the plan when running a Job Scheduling Console query or a conman show command.

modify

Modify calendars in the database.

release

Release jobs from their dependencies in the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.

reply

Reply to job prompts in the production plan.

rerun

Rerun jobs in the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.

shutdown

Shut down IBM Tivoli Workload Scheduler processing. This action is only available in the command line.

start

Start IBM Tivoli Workload Scheduler processing.

stop

Stop IBM Tivoli Workload Scheduler processing.

submit

Submit jobs into the production plan. This action is not available on workstations connected to IBM Tivoli Workload Scheduler for z/OS for end-to-end scheduling.

unlink

Close workstation links.

use

Use calendars to schedule job streams.

Order of user qualification

In qualifying users to access IBM Tivoli Workload Scheduler objects, a user’s actual attributes are compared to user definitions in the order the definitions are entered in the Security file. The first definition that matches the user determines the user’s capabilities. For this reason, it is important to order user definitions from most specific to least specific. For example:

#Incorrect:

#First User Definition in the Security File
USER First
CPU=@+LOGON=TWSUser
Begin
...
End

#Second User Definition in the Security File
USER Second
CPU=@+LOGON=TWSDomain/TWSUser
Begin
...
End

#Correct:
#First User Definition in the Security File
USER First
CPU=@+LOGON=TWSDomain/TWSUser
Begin
...
End

#Second User Definition in the Security File
USER Second
CPU=@+LOGON=TWSUser
Begin
...
End

See “Sample security file” on page 308 for more information.

**Order of object qualification**

In a user definition, you can use multiple statements for a single object type to assign different access capabilities to different sets of objects. Because the first matching statement is used, the order of object statements is important. They must be ordered from most specific to least specific. For example:

#Incorrect:
job name=ar@ access=display
job name=@ access=@
#Correct:
job name=ar@ access=@
job name=@ access=display

See “Sample security file” on page 308 for more information.

**The variables supplied with the product**

The variables supplied with the product that can be used in object attributes are as follows:

- **$jclgroup**
  The group name of a job’s executable file.

- **$jclowner**
  The owner of a job’s executable file.

- **$master**
  The IBM Tivoli Workload Scheduler master domain manager.

- **$owner**
  The creator of a job stream and its jobs.

- **$remotes**
  All standard agent workstations.

- **$slaves**
  All fault-tolerant agent workstations.

- **$thiscpu**
  The workstation on which the user is running the IBM Tivoli Workload Scheduler command or program.

- **$user**
  The user running the IBM Tivoli Workload Scheduler command or program.

The variables **$jclgroup** and **$jclowner** can only be verified if the user is running an IBM Tivoli Workload Scheduler program on the workstation where the job’s executable file resides. If the program is being run on a different workstation, the user is denied access.
**Wildcard characters**

Where noted in the syntax descriptions, the following wildcard characters are permitted:

- `?` Replaces one alphanumeric character.
- `%` Replaces one numeric character.
- `@` Replaces zero or more alphanumeric characters.

**The superuser on UNIX**

If a Security file does not exist, no users other than root can access IBM Tivoli Workload Scheduler objects, and the root user has unrestricted access to all objects and can run all IBM Tivoli Workload Scheduler programs and commands. To control root, create a Security file with a user definition for the root user. In the Security file for a network, you can make a distinction between local root users and the root user on the master domain manager. For example, you can restrict local users to performing operations affecting only their login workstations and permit the master root user to perform operations that affect any workstation in the network. See “Sample Security File” for more information.

**Access capabilities**

The following table lists the available access keywords for each object type, and the capabilities given to users in the Tivoli Workload Scheduler command line.

<table>
<thead>
<tr>
<th>Object Type</th>
<th>Access Keyword</th>
<th>CLI User Capability</th>
</tr>
</thead>
<tbody>
<tr>
<td>calendar</td>
<td>add</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>composer display, create</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td>na (see file modify)</td>
</tr>
<tr>
<td></td>
<td>use</td>
<td>composer - use calendar in schedules</td>
</tr>
<tr>
<td>cpu (Includes domains)</td>
<td>add</td>
<td>composer add, new</td>
</tr>
<tr>
<td></td>
<td>console</td>
<td>conman console</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>composer delete</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>composer display, create</td>
</tr>
<tr>
<td></td>
<td>list</td>
<td>conman showcpus</td>
</tr>
<tr>
<td></td>
<td>fence</td>
<td>conman fence</td>
</tr>
<tr>
<td></td>
<td>limit</td>
<td>conman limit</td>
</tr>
<tr>
<td></td>
<td>link</td>
<td>conman link</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td>composer modify, replace</td>
</tr>
<tr>
<td></td>
<td>shutdown</td>
<td>conman shutdown</td>
</tr>
<tr>
<td></td>
<td>start</td>
<td>conman start</td>
</tr>
<tr>
<td></td>
<td>stop</td>
<td>conman stop</td>
</tr>
<tr>
<td></td>
<td>unlink</td>
<td>conman unlink</td>
</tr>
<tr>
<td>file</td>
<td>build</td>
<td>composer build</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>Access Security with dumpsec</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td>Access Security with makesec and composer modify calendars, parameters, prompts, and resources</td>
</tr>
<tr>
<td>Object Type</td>
<td>Access Keyword</td>
<td>CLI User Capability</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>job</td>
<td>add</td>
<td>composer add, new</td>
</tr>
<tr>
<td></td>
<td>adddep</td>
<td>conman adddep</td>
</tr>
<tr>
<td></td>
<td>altpri</td>
<td>conman altpri</td>
</tr>
<tr>
<td></td>
<td>cancel</td>
<td>conman cancel</td>
</tr>
<tr>
<td></td>
<td>confirm</td>
<td>conman confirm</td>
</tr>
<tr>
<td></td>
<td>deldep</td>
<td>conman deldep</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>composer delete</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>conman display</td>
</tr>
<tr>
<td></td>
<td></td>
<td>composer display</td>
</tr>
<tr>
<td></td>
<td>list</td>
<td>conman showjobs</td>
</tr>
<tr>
<td></td>
<td>kill</td>
<td>conman kill</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td>composer modify, replace</td>
</tr>
<tr>
<td></td>
<td>release</td>
<td>conman release</td>
</tr>
<tr>
<td></td>
<td>reply</td>
<td>conman reply</td>
</tr>
<tr>
<td></td>
<td>rerun</td>
<td>conman rerun</td>
</tr>
<tr>
<td></td>
<td>submit</td>
<td>conman submit (docommand, file, job)</td>
</tr>
<tr>
<td></td>
<td>use</td>
<td>composer - use job in schedule</td>
</tr>
<tr>
<td>parameter</td>
<td>add</td>
<td>parms to add parameters</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>composer display, create and parms to display parameters</td>
</tr>
<tr>
<td></td>
<td></td>
<td>na (see also file modify)</td>
</tr>
<tr>
<td>prompt</td>
<td>add</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>composer display, create conman recall</td>
</tr>
<tr>
<td></td>
<td></td>
<td>conman showprompts</td>
</tr>
<tr>
<td></td>
<td>list</td>
<td>na (see file modify)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>conman reply</td>
</tr>
<tr>
<td></td>
<td>use</td>
<td>composer - use prompt in job stream, and conman - add dependencies</td>
</tr>
<tr>
<td>resource</td>
<td>add</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>na</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>composer display, create</td>
</tr>
<tr>
<td></td>
<td>list</td>
<td>conman showresources</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td>na (see file modify)</td>
</tr>
<tr>
<td></td>
<td>resource</td>
<td>conman resource</td>
</tr>
<tr>
<td></td>
<td>use</td>
<td>composer - use resource in job stream, and conman - add dependencies</td>
</tr>
<tr>
<td>Object Type</td>
<td>Access Keyword</td>
<td>CLI User Capability</td>
</tr>
<tr>
<td>-------------</td>
<td>----------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>schedule</td>
<td>add</td>
<td>composer add, new</td>
</tr>
<tr>
<td></td>
<td>adddep</td>
<td>conman adddep</td>
</tr>
<tr>
<td></td>
<td>altpri</td>
<td>conman altpri</td>
</tr>
<tr>
<td></td>
<td>cancel</td>
<td>conman cancel</td>
</tr>
<tr>
<td></td>
<td>deldep</td>
<td>conman deldep</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>composer delete</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>composer display, create conman display</td>
</tr>
<tr>
<td></td>
<td>list</td>
<td>conman showschedules</td>
</tr>
<tr>
<td></td>
<td>limit</td>
<td>conman limit</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td>composer modify, replace</td>
</tr>
<tr>
<td></td>
<td>release</td>
<td>conman release</td>
</tr>
<tr>
<td></td>
<td>reply</td>
<td>conman reply</td>
</tr>
<tr>
<td></td>
<td>submit</td>
<td>conman submit sched</td>
</tr>
<tr>
<td>userjob</td>
<td>add</td>
<td>composer add, new</td>
</tr>
<tr>
<td></td>
<td>delete</td>
<td>composer delete</td>
</tr>
<tr>
<td></td>
<td>display</td>
<td>composer display (password not displayed)</td>
</tr>
<tr>
<td></td>
<td>modify</td>
<td>composer modify</td>
</tr>
<tr>
<td></td>
<td>altpass</td>
<td>conman altpass</td>
</tr>
</tbody>
</table>

Sample security file

The following is a sample Security file. A description of the file follows the listing.

`# Sample Security File

# (1) APPLIES TO MAESTRO OR ROOT USERS LOGGED IN ON THE
#     MASTER DOMAIN MANAGER OR FRAMEWORK.
user mastersm cpu=$master,$framework + logon=maestro,root,Root_london-region
begin
# OBJECT     ATTRIBUTES       ACCESS CAPABILITIES
# ---------  ----------  ----------------------
job          access=0
schedule     access=0
resource     access=0
prompt       access=0
file         access=0
calendar    access=0
cpu          access=0
parameter    name=0 ~ name=r0 access=0
userobj      cpu=0 + logon=0 access=0
end

# (2) APPLIES TO MAESTRO OR ROOT USERS LOGGED IN ON ANY
#     WORKSTATION OTHER THAN THE MASTER DOMAIN MANAGER.`
user sm logon=maestro,root
begin
# OBJECT       ATTRIBUTES       ACCESS CAPABILITIES
# ---------     -----------       ----------------------
job            cpu=$thiscpu       access=@
job            cpu=$thiscpu       access=@
schedule      cpu=$thiscpu       access=@
resource       cpu=$thiscpu       access=@
prompt         access=@
file           access=@
calendar      access=@
cpu            cpu=$thiscpu       access=@
parameter     cpu=$thiscpu ~ name=r@ access=@
end
###########################################################
# (3) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON THE
# MASTER DOMAIN MANAGER OR FRAMEWORK.
user masterop cpu=$master,$fw + group=sys
begin
# OBJECT       ATTRIBUTES       ACCESS CAPABILITIES
# ---------     -----------       ----------------------
job            cpu=@
                + logon=TWSDomain\TWSUser access=@
job            cpu=@
                + logon=root          access=adddep,altpri,cancel,
                                            confirm,deldep,release,
                                            reply,rerun,submit,use
job            cpu=@
                + logon=$user,$jclowner
                ~ logon=root          access=add,adddep,altpri,
                                            cancel,confirm,
                                            deldep,release,reply,
                                            rerun,submit,use
schedule       cpu=$thiscpu       access=@
schedule       cpu=@
                access=adddep,altpri,cancel,
                                        deldep,limit,release,
                                        submit
resource       access=add,display,
                resource,use
prompt         access=add,display,reply,use
file           access=build
calendar      access;display,use
cpu            cpu=@
parameter     name=0 ~ name=r@ access=@
end
###########################################################
# (4) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON ANY
# WORKSTATION OTHER THAN THE MASTER DOMAIN MANAGER
user op group=sys
begin
# OBJECT       ATTRIBUTES       ACCESS CAPABILITIES
# ---------     -----------       ----------------------
job            cpu=$thiscpu       access=@
job            cpu=$thiscpu       access=@
job            cpu=$thiscpu       access=@
job            cpu=$thiscpu ~ logon=root access=adddep,altpri,cancel,
                                            confirm,deldep,release,
                                            reply,rerun,submit,use
job            cpu=$thiscpu ~ logon=root access=adddep,altpri,cancel,
                                            confirm,deldep,release,
                                            reply,rerun,submit,use
schedule       cpu=$thiscpu       access=@
resource       access=add,display,resource,use
prompt         access=add,display,reply,use
file           access=build
calendar      access=use
### (5) Applies to Users Logged Into the MIS Group on Any Workstation or Framework.

```plaintext
user misusers group=mis
begin
# OBJECT ATTRIBUTES ACCESS CAPABILITIES
# ---------- ----------- ----------------------
job        cpu=$thiscpu + logon=$user access=@
job        cpu=$thiscpu + logon=$jclowner ~ logon=root access=submit,use
schedule   cpu=$thiscpu access=add,submit,
            # modify,display
parameter  name=r@ access=@
parameter  name=@ access=display
end
```

### (6) Applies to All Other Users Logged In on Any Workstation.

```plaintext
user default logon=@
begin
# OBJECT ATTRIBUTES ACCESS CAPABILITIES
# ---------- ----------- ----------------------
job        cpu=$thiscpu + logon=$user access=@
job        cpu=$thiscpu + logon=$jclowner ~ logon=root access=submit,use
schedule   cpu=$thiscpu access=add,submit,
            # modify,display
parameter  name=u@ access=@
parameter  name=@ ~ name=r@ access=display
end
```

Note that the order of definitions is from most to least specific. Because of the order, **maestro** and **root** users are matched first, followed by users in the **sys** group, and then users in the **mis** group. All other users are matched with the last definition, which is the least specific.

#### # (1) Applies to Maestro or Root Users Logged In on the Master Domain Manager or Framework.

```plaintext
cpu=$master,$fw + logon=maestro,root,Root_london-region
```

This user definition applies to GUI and CLI access for **maestro** and **root** users logged into a master domain manager. It also gives GUI access to users listed in the **Root_london-region** Tivoli administrators group. They are given unrestricted access to all objects, except parameters that have names beginning with **r**. Access to the **r** parameters is given only to users in the **mis** group.

#### # (2) Applies to Maestro or Root Users Logged In on Any Workstation Other Than the Master Domain Manager.

```plaintext
logon=maestro,root
```

This user definition applies to **maestro** and **root** users to whom definition (1) does not apply, which are those who are logged in on any workstation other than the
master domain manager or a Framework computer. They are given unrestricted
access to all objects on their login workstation. Note that prompts, files, and
calendars are global in nature and are not associated with a workstation.

# (3) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON THE
MASTER DOMAIN MANAGER OR FRAMEWORK. user masterop
cpu=\$master,\$fw + group=sys

This user definition applies to users logged into the sys group on the master
domain manager or a Framework computer. They are given a unique set of access
capabilities. Multiple object statements are used to give these users specific types
of access to different sets of objects. For example, there are three job statements:

• The first job statement permits unrestricted access to jobs that run on any
  workstation (@) under the user’s name ($user).

• The second job statement permits specific types of access to jobs that run on any
  workstation and that run as root.

• The third job statement permits specific types of access to jobs that run on any
  workstation. The jobs must run under the user’s name ($user) or under the
  name of the owner of the job file ($jclowner). Jobs that run as root are excluded.

# (4) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON ANY
WORKSTATION OTHER THAN THE MASTER DOMAIN MANAGER. user op
group=sys

This user definition applies to sys group users to whom definition (3) does not
apply, which are those who are logged in on any workstation other than the
master domain manager or a Framework computer. They are given a set of access
capabilities similar to those in definition (3). The exception is that access is
restricted to objects on the user’s login workstation ($thiscpu).

# (5) APPLIES TO USERS LOGGED INTO THE MIS GROUP ON ANY
WORKSTATION OR FRAMEWORK. user misusers group=mis

This user definition applies to users logged into the mis group on any workstation
or a Framework computer. They are given a limited set of access capabilities.
Resources, prompts, files, calendars, and workstations are omitted, which prevents
access to these objects. These users are given unrestricted access to parameters with
names that begin with r, but can only display other parameters.

# (6) APPLIES TO ALL OTHER USERS LOGGED IN ON ANY
WORKSTATION.

user default logon=@

This user definition gives a set of default capabilities to users other than those
covered by the preceding definitions (1 to 5). These users are given unrestricted
access to parameters with names that begin with u, but can only display other
parameters. No access is permitted to parameters with names that begin with r.
dumpsec

Decompiles the Security file and sends the output to stdout.

The user must have display access to the Security file.

Synopsis

dumpsec -v | -u
dumpsec security-file

Description

If no arguments are specified, the operational Security file (TWS/home/Security.conf) is dumped. To create an editable copy of a Security file, redirect the output of the command to another file, as shown in the examples below.

Arguments

- -v Displays command version information only.
- -u Displays command usage information only.

security-file

    Specifies the name of the Security file to dump.

Examples

The following command displays the command version:
dumpsec -v

The following command dumps the operational Security file to stdout:
dumpsec

The following command dumps the operational Security file to a file named mysec:
dumpsec > mysec

The following command dumps a Security file named sectemp to stdout:
dumpsec sectemp
makesec

Compiles user definitions and installs the security file. Changes to the security file will be recognized when one of the following programs is stopped and restarted:

Conman
Simply exit and run again.

Composer
Simply exit and run again.

IBM Tivoli Workload Scheduler connectors
Stop the connectors by running the wmaeutil command. The connectors will automatically be restarted with the first query refresh on the Job Scheduling Console.

To use the makesec command, you must have the modify access type to the security file.

Synopsis

makesec -v | -u

makesec [-verify] in-file

Description

The makesec command compiles the specified file and installs it as the operational Security file (.JTWS/home/Security). If the -verify argument is specified, the file is checked for correct syntax, but it is not compiled and installed.

Arguments

-v Displays command version information only.
-u Displays command usage information only.
-verify Checks the syntax of the user definitions in in-file only. The file is not installed as the Security file. (Syntax checking is performed automatically when the Security file is installed.)

in-file Specifies the name of a file or set of files containing user definitions. A file name expansion pattern is permitted.

Examples

The following command displays the command version:

makesec -v

The following command creates an editable copy of the operational Security file in a file named tempsec; modifies the user definitions with a text editor; then compiles tempsec and replaces the operational Security file:

dumpsec > tempsec
edit tempsec

Here you make any required modifications to the tempsec file. When you have finished modifying the tempsec file, run the makesec command to load the security file into IBM Tivoli Workload Scheduler:

makesec tempsec
The following command compiles user definitions from the file set `userdef*` and replaces the operational Security file:

```
makesec userdef*
```
Appendix A. Support information

This section describes the following options for obtaining support for IBM products:

- “Searching knowledge bases”
- “Obtaining fixes” on page 316
- “Contacting IBM Software Support” on page 316

Searching knowledge bases

If you have a problem with your IBM software, you want it resolved quickly. Begin by searching the available knowledge bases to determine whether the resolution to your problem is already documented.

Search the information center on your local system or network

IBM provides extensive product documentation that can be installed on your local computer or on an intranet server. The documentation is supplied on the publications CD available with the product, can be downloaded from IBM as described in “Accessing publications online” on page xvi, or ordered in hardcopy from IBM as described in “Ordering publications” on page xvi.

Open the pdf versions of documents and use the built-in search facilities of Adobe Reader to find the information you require.

Search the information center at the IBM support Web site

The IBM software support Web site has many documents available online, one or more of which may provide the information you require:

2. Under Products A - Z, select your product name: select "I" for IBM and then scroll down to the product entries that commence "IBM Tivoli Workload Scheduler". These open product-specific support sites.
3. Under Self help and Learn, choose from the list of different types of product support documentation:
   - Manuals
   - Redbooks
   - White papers
   - Readme files and other documentation

To access some documents you need to register (indicated by a key icon beside the document title). To register, select the document you wish to look at, and when asked to sign in follow the links to register yourself. There is also a FAQ available on the advantages of registering.

Search the Internet

If you cannot find an answer to your question in the information center, search the Internet for other information that might help you resolve your problem.
Obtaining fixes

A product fix might be available to resolve your problem. You can determine what fixes are available for your IBM software product by checking the product support Web site:

2. Under Products A - Z, select your product name: select "I" for IBM and then scroll down to the product entries that commence "IBM Tivoli Workload Scheduler". These open product-specific support sites.
3. Under Self help, follow the link to Search all Downloads, where you will find a list of fixes, fix packs, and other service updates for your product.
4. Click the name of a fix to read the description and optionally download the fix.

To receive weekly e-mail notifications about fixes and other news about IBM products, follow these steps:

1. From the support page for any IBM product, click My support in the panel on the left of the page.
2. If you have already registered, skip to the next step. If you have not registered, click register in the upper-right corner of the support page to establish your user ID and password.
3. Sign in to My support.
4. On the My support page, select the Edit profile tab and click Subscribe to email. Select a product family and check the appropriate boxes for the type of information you want.
5. Click Update.
6. For e-mail notification for other product groups, repeat Steps 4 and 5.

For more information about types of fixes, see the Software Support Handbook [http://techsupport.services.ibm.com/guides/handbook.html].

Contacting IBM Software Support

IBM Software Support provides assistance with product defects.

Before contacting IBM Software Support, your company must have an active IBM software maintenance contract, and you must be authorized to submit problems to IBM. The type of software maintenance contract that you need depends on the type of product you have:

- For IBM distributed software products (including, but not limited to, Tivoli, Lotus, and Rational products, as well as DB2 and WebSphere products that run on Windows or UNIX operating systems), enroll in Passport Advantage in one of the following ways:
  - **Online**: Go to the Passport Advantage Web page [http://www.lotus.com/services/passport.nsf/WebDocs/Passport_Advantage_Home] and click How to Enroll
  - **By phone**: For the phone number to call in your country, go to the IBM Software Support Web site [http://techsupport.services.ibm.com/guides/contacts.html] and click the name of your geographic region.

- For IBM eServer software products (including, but not limited to, DB2 and WebSphere products that run in zSeries, pSeries, and iSeries environments), you can purchase a software maintenance agreement by working directly with an

If you are not sure what type of software maintenance contract you need, call 1-800-IBMSERV (1-800-426-7378) in the United States or, from other countries, go to the contacts page of the IBM Software Support Handbook on the Web [http://techsupport.services.ibm.com/guides/contacts.html](http://techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region for phone numbers of people who provide support for your location.

Follow the steps in this topic to contact IBM Software Support:
1. **Determine the business impact of your problem.**
2. **Describe your problem and gather background information.**
3. **Submit your problem to IBM Software Support.**

### Determine the business impact of your problem
When you report a problem to IBM, you are asked to supply a severity level. Therefore, you need to understand and assess the business impact of the problem you are reporting. Use the following criteria:

<table>
<thead>
<tr>
<th>Severity 1</th>
<th>Critical business impact: You are unable to use the program, resulting in a critical impact on operations. This condition requires an immediate solution.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Severity 2</td>
<td>Significant business impact: The program is usable but is severely limited.</td>
</tr>
<tr>
<td>Severity 3</td>
<td>Some business impact: The program is usable with less significant features (not critical to operations) unavailable.</td>
</tr>
<tr>
<td>Severity 4</td>
<td>Minimal business impact: The problem causes little impact on operations, or a reasonable circumvention to the problem has been implemented.</td>
</tr>
</tbody>
</table>

### Describe your problem and gather background information
When explaining a problem to IBM, be as specific as possible. Include all relevant background information so that IBM Software Support specialists can help you solve the problem efficiently. To save time, know the answers to these questions:

- What software versions were you running when the problem occurred?
- Do you have logs, traces, and messages that are related to the problem symptoms? IBM Software Support is likely to ask for this information.
- Can the problem be re-created? If so, what steps led to the failure?
- Have any changes been made to the system? (For example, hardware, operating system, networking software, and so on.)
- Are you currently using a workaround for this problem? If so, please be prepared to explain it when you report the problem.

### Submit your problem to IBM Software Support
You can submit your problem in one of two ways:

- **Online:** Go to the “Submit and track problems” page on the IBM Software Support site [http://www.ibm.com/software/support/probsub.html](http://www.ibm.com/software/support/probsub.html). Enter your information into the appropriate problem submission tool.
• **By phone:** For the phone number to call in your country, go to the contacts page of the IBM Software Support Handbook on the Web [http://techsupport.services.ibm.com/guides/contacts.html](http://techsupport.services.ibm.com/guides/contacts.html) and click the name of your geographic region.

If the problem you submit is for a software defect or for missing or inaccurate documentation, IBM Software Support creates an Authorized Program Analysis Report (APAR). The APAR describes the problem in detail. Whenever possible, IBM Software Support provides a workaround for you to implement until the APAR is resolved and a fix is delivered. IBM publishes resolved APARs on the IBM product support Web pages daily, so that other users who experience the same problem can benefit from the same resolutions.

For more information about problem resolution, see [Searching knowledge bases](http://techsupport.services.ibm.com/guides/contacts.html) and [Obtaining fixes](http://techsupport.services.ibm.com/guides/contacts.html).
Appendix B. Managing time zones

IBM Tivoli Workload Scheduler supports different time zones. Enabling time zones provides you with the ability to manage your workload on a global level. Time zone implementation also allows for easy scheduling across multiple time zones and for jobs that need to run in the "dead zone." The dead zone is the difference between the IBM Tivoli Workload Scheduler start of day time on the master domain manager and the time on the fault-tolerant agent in another time zone. For example, if an eastern master with an IBM Tivoli Workload Scheduler start of day of 6 a.m. initializes a western agent with a 3-hour time zone difference, the dead zone for this fault-tolerant agent is between the hours of 3 a.m. and 6 a.m.

Previously, special handling was required to run jobs in this time period. Now by enabling the time zone feature and specifying a time zone with a start time on a job or job stream, IBM Tivoli Workload Scheduler runs them when you expect it to.

Figure 8 shows the start of day time and the dead zone for an eastern master with an IBM Tivoli Workload Scheduler start of day of 6 a.m. and a western agent with a 3-hour time zone difference.

![Diagram](image)

Figure 8. Dead zones

Activating the Time Zone feature

Activate the time zone feature on IBM Tivoli Workload Scheduler by performing the following steps:

1. Set to yes the Timezone enable option in the TWSHOME/mozart/globalopts file on the Master Domain Manager. The default value is no, time zone not enabled.
2. Set a specific time zone for every workstation in the IBM Tivoli Workload Scheduler network by modifying the workstation properties using the Job Scheduling Console.

If you do not specify a time zone for a workstation, this workstation uses the master domain manager time zone.
For more details on how to enable the time zone feature, refer to the *IBM Tivoli Workload Scheduler Planning and Installation Guide.*

**Running a job stream not time zone–enabled when the master domain manager is ahead of the fault-tolerant agent**

This section shows the job stream definition statement to run a job stream every weekday morning on a Pacific Standard Time (PST) fault-tolerant agent connected to an Eastern Standard Time (EST) master domain manager with a start of day of 0600 a.m., if you do not enable the time zone feature. To run the job stream every weekday morning you must schedule it to run Sunday through Thursday, and specify the carryforward and the at keywords as follows:

```
Schedule FTA1# PST_SCHEDULE1
  On SU, weekdays except FR
  CARRYFORWARD
  AT 0330
  .
  .
  Job1
  Job2
  END
```

If you do not specify the carryforward keyword, the job would never run because the fault-tolerant agent would be initialized at 0300 a.m. every day.

**Running a job stream time zone–enabled when the master is ahead of the fault-tolerant agent**

This section shows the job stream definition statement to run a job stream every weekday morning on a PST fault-tolerant agent connected to an EST master domain manager with a start of day of 0600 a.m., if you enable the time zone feature. To run a job stream every weekday morning you must schedule it to run every weekday by specifying the at keyword as follows:

```
Schedule FTA1# PST_SCHEDULE2
  On weekdays
  AT 0330
  .
  .
  Job1
  Job2
  END
```

When the EST master domain manager initializes the PST fault-tolerant agent at 0300 a.m., the fault-tolerant agent starts the job stream the same day at 0300 a.m.

**Running a job stream not time zone–enabled when the master domain manager is behind the fault-tolerant agent.**

This section shows the job stream definition statement to run a job stream every weekday morning on an EST fault-tolerant agent having a PST master domain manager with a 0600 a.m. start of day, if you do not enable the time zone feature. To run the job stream every weekday morning, you must schedule it to run Sunday through Thursday, and specify the carryforward keyword and +1DAY for the at keyword as shown in the following job stream definition statement:

```
Schedule FTA1#EST_SCHEDULE1
  On SU, weekdays except FR
  AT 0800 + 1 DAY CARRYFORWARD
  .
```
You need to specify the carryforward keyword because otherwise the job stream is not selected to be included in the plan that runs on the following day. Without the +1DAY specification, the job stream would launch immediately after initialization from the western master at 0900 a.m.

Running a job stream time zone–enabled when the master domain manager is behind the fault-tolerant agent.

This section shows the job stream definition statement to run a job stream every weekday morning on an EST fault-tolerant agent having a PST master with a start of day of 0600 a.m., if you enable the time zone feature. To run the job stream every weekday morning, specify the following job stream definition statement:

```
Schedule FTA1#EST_SCHEDULE2
On SU, weekdays except FR
AT 0800
.
job1
job2
END
```

When the eastern fault-tolerant agent is initialized at 0900 a.m., it runs the job stream at 0800 a.m. the next day.

Submitting ad hoc a job stream specifying an at dependency

This section shows how to submit ad hoc a job stream specifying an at dependency. If you defined a job stream without any time dependencies, and you want to submit it specifying a time dependency, specify the at dependency using the Conman program on the master domain manager. For example, if you defined the SCHED1 job stream without the at dependency as follows:

```
Schedule FTA1#SCHED1
on request
.
job1
END
```

To submit the SCHED1 job stream with a PST at dependency, run the following Conman command on the master domain manager:

```
sbs FTA1#SCHED1 ;at=0400  T2  PST
```

The master domain manager converts the submission time for example, 2:00 p.m. ECT to the time zone (PST) you specified in the command. The master domain manager then compares the value specified in the at keyword with the resulting value:

- If the at dependency value is later than or the same as the resulting value, the schedule runs on the same day.
- If the at dependency value is earlier than or the same as the resulting value, the schedule runs on the following day.

The date obtained and the time specified in the command (4:00 a.m. PST) are now converted by the master domain manager from the time zone (PST) specified in the command into the time zone specified on the FTA1 workstation.
Submitting a job stream specifying an at dependency that occurs during daylight saving time

This section shows how to submit a job stream specifying an at dependency that occurs during daylight saving time. Consider a master domain manager defined in time zone ECT (GMT+1:00). The date is April, 3 2005. On this date, time zone ECT is already on daylight saving time and therefore GMT+2:00. You submit a job stream with at dependency related to time zone PST (GMT-8:00):

Schedule MDM\#SCHED1
AT 0230 TZ PST
. .
Job1
Job2
END

In the time zone PST on the April, 3 2005 at 2:00 a.m. there is a change from standard time to daylight saving time (DST), therefore the time is moved ahead to 3:00 a.m. and the time zone changes from GMT-8:00 to GMT-7:00. The master domain manager converts the AT 0230 TZ PST time dependency to the time zone of the fault-tolerant agent where the job stream must run, which in this case is the time zone of the master domain manager itself (ECT). All time dependencies between AT 0200 TZ PST and AT 0259 TZ PST are translated to ECT 1159 because all minutes in the time zone PST match with only one minute in the time zone ECT. Outside this range of time dependencies, any dependency is converted following the standard conversion rules. For example, a time dependency of AT 0300 TZ PST is converted to ECT 1200. A time dependency of AT 0301 TZ PST is converted to ECT 1201, and so on.

Time zone list

This section lists the supported time zones together with their description.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Relative to GMT</th>
</tr>
</thead>
<tbody>
<tr>
<td>GMT</td>
<td>Greenwich Mean Time</td>
<td>GMT</td>
</tr>
<tr>
<td>UTC</td>
<td>Universal Coordinated Time</td>
<td>GMT</td>
</tr>
<tr>
<td>ECT</td>
<td>European Central Time</td>
<td>GMT+1:00</td>
</tr>
<tr>
<td>EET</td>
<td>Eastern European Time</td>
<td>GMT+2:00</td>
</tr>
<tr>
<td>ART</td>
<td>(Arabic) Egypt Standard Time</td>
<td>GMT+2:00</td>
</tr>
<tr>
<td>EAT</td>
<td>Eastern African Time</td>
<td>GMT+3:00</td>
</tr>
<tr>
<td>MET</td>
<td>Middle East Time</td>
<td>GMT+3:30</td>
</tr>
<tr>
<td>NET</td>
<td>Near East Time</td>
<td>GMT+4:00</td>
</tr>
<tr>
<td>PLT</td>
<td>Pakistan Lahore Time</td>
<td>GMT+5:00</td>
</tr>
<tr>
<td>IST</td>
<td>India Standard Time</td>
<td>GMT+5:30</td>
</tr>
<tr>
<td>BST</td>
<td>Bangladesh Standard Time</td>
<td>GMT+6:00</td>
</tr>
<tr>
<td>VST</td>
<td>Vietnam Standard Time</td>
<td>GMT+7:00</td>
</tr>
<tr>
<td>CTT</td>
<td>China Taiwan Time</td>
<td>GMT+8:00</td>
</tr>
<tr>
<td>JST</td>
<td>Japan Standard Time</td>
<td>GMT+9:00</td>
</tr>
<tr>
<td>ACT</td>
<td>Australia Central Time</td>
<td>GMT+9:30</td>
</tr>
<tr>
<td>AET</td>
<td>Australia Eastern Time</td>
<td>GMT+10:00</td>
</tr>
<tr>
<td>SST</td>
<td>Solomon Standard Time</td>
<td>GMT+11:00</td>
</tr>
<tr>
<td>Name</td>
<td>Descriptionishing</td>
<td>Relative to GMT</td>
</tr>
<tr>
<td>------</td>
<td>-------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>NST</td>
<td>New Zealand Standard Time</td>
<td>GMT+12:00</td>
</tr>
<tr>
<td>MIT</td>
<td>Midway Islands Time</td>
<td>GMT+11:00</td>
</tr>
<tr>
<td>HST</td>
<td>Hawaii Standard Time</td>
<td>GMT+10:00</td>
</tr>
<tr>
<td>AST</td>
<td>Alaska Standard Time</td>
<td>GMT+9:00</td>
</tr>
<tr>
<td>PST</td>
<td>Pacific Standard Time</td>
<td>GMT+8:00</td>
</tr>
<tr>
<td>PNT</td>
<td>Phoenix Standard Time</td>
<td>GMT+7:00</td>
</tr>
<tr>
<td>MST</td>
<td>Mountain Standard Time</td>
<td>GMT+7:00</td>
</tr>
<tr>
<td>CST</td>
<td>Central Standard Time</td>
<td>GMT+6:00</td>
</tr>
<tr>
<td>EST</td>
<td>Eastern Standard Time</td>
<td>GMT+5:00</td>
</tr>
<tr>
<td>IET</td>
<td>Indiana Eastern Standard Time</td>
<td>GMT+5:00</td>
</tr>
<tr>
<td>PRT</td>
<td>Puerto Rico and US Virgin Islands Time</td>
<td>GMT+4:00</td>
</tr>
<tr>
<td>CNT</td>
<td>Canada Newfoundland Time</td>
<td>GMT+3:30</td>
</tr>
<tr>
<td>AGT</td>
<td>Argentina Standard Time</td>
<td>GMT+3:00</td>
</tr>
<tr>
<td>BET</td>
<td>Brazil Eastern Time</td>
<td>GMT+3:00</td>
</tr>
<tr>
<td>CAT</td>
<td>Central African Time</td>
<td>GMT+1:00</td>
</tr>
</tbody>
</table>
Appendix C. The auditing feature

An auditing option is available to track changes to the database and the plan:

- For the database, all user modifications are logged. However, the delta of the modifications, or before image and after image, will not be logged. If an object is opened and saved, the action will be logged even if no modification has been done.
- For the plan, all user modifications to the plan are logged. Actions are logged whether they are successful or not.

The auditing logs are created in the following directories:

TWShome/audit/plan
TWShome/audit/database

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

**Note:** For modify commands, two entries are made in the log for resources, calendars, parameters and prompts. The modify command is displayed in the log as the delete and add commands.

Enabling the audit feature

The auditing option is enabled by two entries in the `globalopts` file:

```plaintext
plan audit level = 0|1
database audit level = 0|1
```

A value of 1 enables auditing and a value of 0 disables auditing. IBM Tivoli Workload Scheduler currently defaults to 0, or auditing disabled. If these options are not present in the `globalopts` file, auditing is disabled. Auditing is disabled by default on installation of the product.

To initiate database auditing, you must shut down IBM Tivoli Workload Scheduler completely and use the `wmaeutil` command to stop the connector instance. When you restart IBM Tivoli Workload Scheduler and the connector instance, the database audit log is initiated. Plan auditing takes effect when `Jnextday` is run.

Auditing log format

The audit log formats are basically the same for the plan and database. The log consists of a header portion, an action ID, and data sections that vary with the action type. The data is in clear text format and each data item is separated by a vertical bar ( | ).

The log file entries will be in the following format:
The log files contain the following information:

**Log Type**
This field displays an eight character value indicating the source of the log record. The following log types are supported:

- **HEADER**
  The log file header
- **CONMAN**
  conman command text
- **FILEAID**
  Command that opens a file
- **PLAN**
  Plan action
- **STAGEMAN**
  stageman run
- **RELEASE**
  release command text
- **DATABASE**
  Database action
- **PARMS**
  Parameter command text
- **MAKESEC**
  makesec run
- **DBEXPAND**
  dbexpand run

**GMT Date**
This field displays the GMT date the action was performed. The format is **yyyyymmdd** where **yyyy** is the year, **mm** is the month, and **dd** is the day.

**GMT Time**
This field displays the GMT time the action was performed. The format is **hhmmss** where **hh** is the hour, **mm** is the minutes, and **ss** is the seconds.

**Local Date**
This field displays the local date the action was performed. The local date is defined by the time zone option of the workstation. The format is **yyyyymmdd** where **yyyy** is the year, **mm** is the month, and **dd** is the day.

**Local Time**
This field displays the local time the action was performed. The local time is defined by the time zone option of the workstation. The format is **hhmmss** where **hh** is the hour, **mm** is the minutes, and **ss** is the seconds.

**Object Type**
This field displays the type of the object that was affected by an action. The object type is one of the following:

- **DATABASE**
  Database definition
DBWKSTN
   Database workstation definition
DBWKCLS
   Database workstation class definition
DBDOMAIN
   Database domain definition
DBUSER
   Database user definition
DBJBSTRM
   Database job stream definition
DBJOB
   Database job definition
DBCAL
   Database calendar definition
DBPROMPT
   Database prompt definition
DBPARM
   Database parameter definition
DBRES
   Database resource definition
DBSEC
   Database security
PLAN
   Plan
PLWKSTN
   Plan workstation
PLDOMAIN
   Plan domain
PLJBSTRM
   Plan job stream
PLJOB
   Plan job
PLPROMPT
   Plan prompt
PLRES
   Plan resource
PLFILE
   Plan file

**Action Type**

This field displays what action was taken against the object. The appropriate values for this field are dependent on the action being taken.

For the database, the Action Type can be ADD, DELETE, MODIFY, EXPAND, or INSTALL. IBM Tivoli Workload Scheduler will record ADD, DELETE and MODIFY actions for workstation, workstation classes, domains, users, jobs, job streams, calendars, prompts, resources and parameters in the database. The Action Type field also records the installation of a new security file. When `makesec` is run IBM Tivoli
Workload Scheduler will record it as INSTALL action for a Security definition object. When dbexpand is run it will be recorded as a EXPAND action for DATABASE object. LIST and DISPLAY actions for objects are not logged. For fileaid IBM Tivoli Workload Scheduler will only log the commands that result in the opening of a file. For parameters, the command line with arguments is logged.

Workstation Name
This field displays the IBM Tivoli Workload Scheduler workstation from which the user is performing the action.

User ID
This field displays the logon user who performed the particular action. On Win32 platforms it will be the fully qualified domain name domain\user.

Framework User
This field displays the Tivoli Framework recognized user ID. This is the login ID of the Job Scheduling Console user.

Object Name
This field displays the fully qualified name of the object. The format of this field will depend on the object type as shown here:

DATABASE
   N/A

DBWKSTN
   workstation

DBWKCLS
   workstation_class

DBDOMAIN
   domain

DBUSER
   [workstation#]user

DBJBSTRM
   workstation#jobstream

DBJOB
   workstation#job

DBCAL
   calendar

DBPROMPT
   prompt

DBPARM
   workstation#parameter

DBRES
   workstation#resource

DBSEC
   N/A

PLAN
   N/A

PLWKSTN
   workstation
Action Dependent Data
This field displays the action-specific data fields. The format of this data is dependent on the Action Type field.

Audit log header

Each log file will start with a header record that contains information about when the log was created and whether it is a plan or database log.

The contents of the header file entry is as follows:

Log Type
HEADER

GMT Date
The GMT date that the log file was created.

GMT Time
The GMT time that the log file was created.

Local Date
The local date that the log file was created.

Local Time
The local time that the log file was created.

Workstation Name
The IBM Tivoli Workload Scheduler workstation name for which this file was created. Each workstation in the IBM Tivoli Workload Scheduler network creates its own log.

User ID
The IBM Tivoli Workload Scheduler user ID that created the log file.

Object Type
This field reads DATABASE for a database log file and PLAN for a plan log file.

Object Name
N/A

Action Type
N/A

Action Dependent Data
This field displays the version of the file.
Sample audit log entries

Below are some sample log file entries:

<table>
<thead>
<tr>
<th>HEADER</th>
<th>19991202</th>
<th>201200</th>
<th>19991202</th>
<th>131200</th>
<th>DATABASE</th>
<th>DATABASE</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIVERS\pyasa</td>
<td></td>
<td></td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATABASE</th>
<th>19991202</th>
<th>224504</th>
<th>19991202</th>
<th>154504</th>
<th>DBWKSTN</th>
<th>ADD</th>
</tr>
</thead>
<tbody>
<tr>
<td>GANGES</td>
<td>RIVERS\pyasa</td>
<td></td>
<td></td>
<td>JAMUNA</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>DATABASE</th>
<th>19991203</th>
<th>001400</th>
<th>19991202</th>
<th>171400</th>
<th>DBJOB</th>
<th>MODIFY</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIVERS\tairak</td>
<td></td>
<td></td>
<td></td>
<td>NARMADA\dubo</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Notices

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

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

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

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

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

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

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

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

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.
IBM may use or distribute any of the information you supply in any way it
believes appropriate without incurring any obligation to you.

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

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

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

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

Information concerning non-IBM products was obtained from the suppliers of
those products, their published announcements or other publicly available sources.
IBM has not tested those products and cannot confirm the accuracy of
performance, compatibility or any other claims related to non-IBM products.
Questions on the capabilities of non-IBM products should be addressed to the
suppliers of those products.

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

**Trademarks**

IBM, the IBM logo, z/OS, Tivoli, the Tivoli logo are trademarks or registered
trademarks of International Business Machines Corporation in the United States,
other countries, or both.

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

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

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

Batchman. Batchman is a process started at the beginning of each Tivoli Workload Scheduler 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 run 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 Tivoli Workload Scheduler 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 dbxexpand 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.

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 run. 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 run 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.

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 Tivoli Workload Scheduler network. See also “Local options”.

Host. A Workload Scheduler workstation required by extended agents. It can be any Tivoli Workload Scheduler workstation except another extended agent.

Inclusionary Run Cycle. A run cycle that specifies the days a job stream is scheduled to run. Exclusionary run cycles take precedence 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 Tivoli Workload Scheduler engine. Internal status is unique to Tivoli Workload Scheduler. See also Status.

Internet (INET) dependencies. A dependency between jobs or job streams in separate Tivoli Workload Scheduler networks. See also “Network agent”.

Internet (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 Internet 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 Tivoli Workload Scheduler.

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 Tivoli Workload Scheduler 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 run 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 Tivoli Workload Scheduler 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 Tivoli Workload Scheduler, the plan is created every 24 hours and consists of all the jobs, job streams, and dependency objects that are scheduled to run 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 run according to their time restrictions and other dependencies. Any jobs or job streams that do not run successfully are rolled over into the next day’s plan.

Planned Start Time. The time that Tivoli Workload Scheduler 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. Tivoli Workload Scheduler 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 run.

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 Tivoli Workload Scheduler 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 Tivoli Workload Scheduler 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 run. 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 Tivoli Workload Scheduler 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 run 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 run. 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 Tivoli Workload Scheduler server, groups of default lists, and groups of user created lists.

**W**

**Weekly Run Cycle.** A run cycle that specifies the days of the week that a job stream is run. For example, a job stream can be specified to run 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 alphanumeric character.
- % Replaces one numeric character.
- * Replaces zero or more alphanumeric characters in the Tivoli Job Scheduling console.
- @ Replaces zero or more alphanumeric characters in the Tivoli Workload Scheduler command line.

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 run. They are defined in the Tivoli Workload Scheduler database as a unique object. A workstation definition is required for every computer that runs 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 run on a workstation class. This makes replication of a job or job stream across many workstations easy.

**X**

**X-agent.** See “Extended agent”.

**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.
Index

A
absolute keyword 122, 127, 128, 133
adddep job command 135
adddep sched command 137
altpri command 139
altpass command 140
APARs
IY46485 16
IY46918 185
IY48317 279
IY49121.1 306, 307, 308
IY49333 57, 118, 306
IY52215 186, 187
IY52475 90
IY52493 233
IY53050 97
IY53064 298, 313
IY53459 186
IY53755 82
IY57906 303
IY58702 25
at command 219
at keyword 15, 83
auditing
  enabling 325
  header format 329
  log files 325
  log format 325
  sample log entries 330

B
batch command 219
batchman 8
behindfirewall 35
books xii
  see publications xvi

C
calendar definition 49
cancel job command 142
cancel sched command 143
carryforward keyword 15, 85
caxtract command 224
centralized security 297
command prompt 56
command prompt, conman 117
command syntax, conman 117
commands
  add 60
  adddep job 135
  adddep sched 137
  altpri 140
  build 61
  cancel job 141
  cancel sched 143
  compiler 8
  confirm 145
commands (continued)
  console 146
  continue 62, 147
  create 63
  deldep job 148
  deldep sched 150
  delete 65
  display 67, 152
  dumpsec 298, 312
  edit 70
  exit 71, 153
  fence 154
  help 155
  kill 156
  limit cpu 157
  limit sched 158
  link 159
  list 67
  listsym 161
  logman 8
  makesec 298, 313
  modify 72
  new 74
  print 67
  recall 162
  redo 75, 163
  release job 165
  release sched 167
  rep1 269
  rep1 272
  rep2 269
  rep3 269
  rep4a 269
  rep4b 269
  rep7 270
  rep8 271
  replace 77
  reply 169
  repr 273
  rerun 170
  resource 173
  scheddur 8
  setsym 174
  showcpus 175
  showdomain 179
  showfiles 180
  showjobs 182
  showprompts 190
  showresources 192
  showschedules 194
  shutdown 197
  stageman 8
  start 198
  status 200
  stop 201
  stop :progressive 203
  submit docommand 204
  submit file 206
  submit job 208
  submit sched 210
  switchmgr 212
commands (continued)
  system command 213
  tellop 214
  unlink 215
  validate 79
  version 80, 217
  xref 274
  comments keyword 86
  compiler command 8
  composer 7
  composer command descriptions 58
  composer command syntax 57
  composer control characters 55
  composer editor 56
  composer program 55
  composer reference 31
  confirm command 145
  confirmed keyword 87
  conman 7
  conman command list 118
  conman command prompt 117
  conman command syntax 117
  conman reference 115
  conman, running 115
  conman, starting 298
  console command 146
  continue command 147
  control characters 55
  control characters, conman 115
  conventions
    typeface xvii
    cpuinfo command 226
    cpuname 33
    creating an internetwork dependency 292
    creating the security file 298
    customer support
      see Software Support 316
D
database entries 31
database object editor 56
datecalc command 227
dbexpand command 231
dead zones 319
deadline 15
deadline keyword 88
Defects
  155861 42
  169932 175
  170660 108
deldep command 148
deldep sched command 150
delete command 232
delimiters 57
delimiters, conman 118
dependency
  file 125, 131, 180
  internetwork 289, 292
directory names, notation xviii
jobinfo command 152
domain definition 40
dumpsec command 8, 298, 312

e-mail contact xvi
editor, database object 56
education
   see Tivoli technical training xvii

enabling auditing 325
dependent command 89

environment variables, notation xviii
every keyword 90
evtsize command 234
except keyword 91
exit command 153
extended agent
   overview 279
   reference 279
   workstation definition 279

feedback about publications xvi
fence command 154
file dependency 125, 131, 180
final job stream
   adding 13
   customizing 13
firewall
   setting 35
fixes, obtaining 316
follow word 93
freedays keyword 94

globalopts file
   auditing feature 325

help command 155

information centers, searching to find software problem resolution 315
Internet, searching to find software problem resolution 315, 316
internetwork dependency 289, 292

jobinfo command 235
jnextday command 13
   running 13
job definition 42
job launch 8
job scheduling console 7
job statement keyword 96
job states 126
jobinfo command 237

job 8
jobmanrc 17
jobmon 8
jobstld command 239

keyjob keyword 101
keysched keyword 102
keywords
   absolute 122, 127, 128, 133
deadline 88
keywords, scheduling language 82
kill command 156
killproc command 265
knowledge bases, searching to find software problem resolution 315

late status 88
latest start time 15
library xii
limit cpu command 157
limit keyword 103
limit sched command 158
link command 159
listproc command 265
listsym command 161
local security 297
logman command 8

maestro command 241
mailman 8
makecal command 242
makesec 298
makesec command 8, 298, 313
managing security 298
managing time zone 319
manuels xii

   see publications xvi
metronome.pl command 244
morestld command 245

needs keyword 104
netman 8
netmth access method 289
network agent
   internetwork dependency 292
   netmth 289
options file 291
overview 289
workstation 289
workstation command line example 291
notation
   environment variables xviii
   path names xviii
   typeface xviii

offline output 116
offline output for UNIX 56
on keyword 15, 105
online publications
   accessing xvi
onuntil 113
onuntil keyword 113
opens keyword 108
order of objects, security file 305
order of users, security file 304
ordering publications xvi
os_type 33

parameter definition 50
parms command 246
path names, notation xviii
paxtract command 247
priority keyword 110
problem determination
   describing problem for IBM Software Support 317
determining business impact for IBM Software Support 317
   submitting problem to IBM Software Support 317
process tree on UNIX 9
process tree on Windows 11
product library xii
production cycle 7
automating 13
progressive 203
prompt definition 52
prompt keyword 111
paxtract command 248
publications xii
   accessing online xvi
   ordering xvi

r11xtract command 249
rcondsacc 43, 97
recall command 162
redo command 163
refreshing the connectors 325
release command 250
release job command 165
release sched command 167
rep1 command 269
rep11 command 272
rep2 command 269
rep3 command 269
rep4a command 269
rep4b command 269
rep7 command 270
rep8 command 271
reply command 169
reptr command 273
rerun command 170
reserved words 31
resource command 173
resource definition 54
rextract command 252
Index

S
sample log entries, auditing 330
sample reports 275
sample security file 308
schedule keyword 112
scheduling language 81
scheduling language keywords 82
scheduling objects 31
schedule command 8
security
creating the security file 298
dumpsec 299
file syntax 299
local 297
makesec 298
managing 298
order of objects 305
order of user definitions 304
sample file 308
template file 298
UNIX superuser 306
user definitions 299
variables 305
security file
updinating 298
security level
enabled 35
force 35
on 35
selecting job streams in commands 128
selecting jobs in commands 120
setsym command 174
setup_env 299
Sfinal
adding 13
showcpus command 175
showdomains command 179
showexec command 254
showfiles command 180
showjobs command 182
showprompts command 191
showresources command 192
showschedules command 194
shutdown command 197
Software Support
contacting 316
describing problem for IBM Software Support 317
determining business impact for IBM Software Support 317
submitting problem to IBM Software Support 317
special characters, composer 57
special characters, conman 118
SSL communication
enabled 35
force 35
on 35
stageman command 8
start command 198
start time 15
start, latest 15
startup command 253
states, jobs 126
status
late 88
status command 200
stop :progressive command 203
stop command 201
stop scheduler processes 16, 203
submit docommand 204
submit file command 206
submit job command 208
submit sched command 210
success condition 43, 97
support Web site, searching to find software problem resolution 315
switchmgr command 212
syntax, scheduling language 81
syntax, security file 299
system command, conman 213
system commands, conman 115
T
telop command 214
template file, security 298
terminal output 55, 116
time zone 121
Tivoli software information center xvi
Tivoli technical training xvii
training, Tivoli technical xvii
typeface conventions xvii
U
UNIX superuser, security 306
unlink command 215
unsupported commands 265
until keyword 113
user definition 47
user definitions, security 298
user interfaces 7
user prompts 116
utility commands 219
V
variables, notation for xviii
variables, security file 305
version command 217, 256
W
wildcard characters 57, 117
wmaeutil 298
wmaeutil command 28, 258, 325
workstation class definition 39
workstation definition 33
writer 8
X
xref command 274
xxrtrct command 260
Program Number: 5698-WSH

Printed in USA