Workload Scheduler for z/OS
Version 8.6

Managing the Workload

IBM
Managing the Workload
# Contents

<table>
<thead>
<tr>
<th>Figures</th>
<th>xiii</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tables</td>
<td>xix</td>
</tr>
</tbody>
</table>

## About this publication
- What is new in this release: xxi
- What is new in this publication: xxi
- Who should read this publication: xxi

## Publications
- Using LookAt to look up message explanations: xxi
- Tivoli technical training: xxi
- Support information: xxi

## Conventions used in this publication
- How to read the syntax diagrams: xxiv

## Part 1. Planning and scheduling

### Chapter 1. What is the scheduler
- How the scheduler works: 7
- How does the Scheduler keep track of jobs?: 9
- Can you define a dependency for the job?: 9
- How does the scheduler help with job preparation?: 10
- How can the scheduler help you recover jobs?: 10
- Can the scheduler help with online systems?: 10
- Can you run jobs outside the scheduler?: 11
- Finding out about the scheduler: 11

### Chapter 2. A user scenario
- Implementing a payroll system: 13
- Designing the payroll applications: 15
- Creating workstations: 17
- Controlling general workstations: 18
- Using servers: 19
- Creating special resources: 20
- Creating the default calendar: 21
- Preparing the payroll jobs: 22
- Creating the groups, applications, and operations: 24
- Creating plans: 30
- What happens if jobs or started tasks fail?: 32
- Coping with more complex problems: 33
- Summary of what you need to do: 34

### Chapter 3. Using the scheduler panels
- The scheduler ISPF panels: 35
  - Setting options: 36
  - Common panel commands and facilities: 42
  - Concatenating options: 42
  - Quick return command: 42
  - Primary commands: 43
  - Specifying list criteria: 43
  - Specifying panel views: 44
  - Using generic search arguments: 46
- Sorting list output: 46
- Locating data strings in list output: 47
- Graphically displaying lists: 48
- PF Key assignment: 48
- Modeling your environment: 49
- The graphical user interface: 50
- Utilities: 50

### Chapter 4. Creating workstations
- What types of workstation are there?: 53
- Computer workstations: 53
- Job computer workstations: 54
- Started-task computer workstations: 54
- Virtual workstations: 55
- Fault-tolerant workstations: 55
- Tivoli Workload Scheduler for z/OS agent workstations: 55
- Dynamic workstations: 55
- Alternate workstations: 56
- Printer workstations: 56
- General workstations: 56
- WTO general workstations: 57
- WAIT general workstations: 57
- Remote engine workstations: 58
- Guidelines for creating workstations: 58
- Which workstation types are needed?: 59
- How many workstations of each type?: 59
- Using virtual workstations: 60
- Dummy workstations: 61
- Creating a workstation: 62
- Specifying workstation attributes and options: 65
- Specifying workstation reporting attributes: 65
- Specifying fault-tolerant or Tivoli Workload Scheduler for z/OS agent workstations: 67
- Specifying dynamic workstations: 69
- Specifying remote engine workstations: 71
- Specifying workstation parallel servers: 71
- Specifying splittable workstations: 72
- Specifying workstation destinations: 73
- Specifying the default workstation transport time: 73
- Specifying the default duration: 74
- Specifying the Automation option: 74
- Specifying workstation availability: 74
- Specifying workstation open intervals: 75
- Closing workstations: 76
- Specifying workstation fixed resources: 78
- Controlling workstations: 79

### Chapter 5. Creating special resources
- Understanding special resources: 81
- Example using data sets: 84
- Example using tape drives: 84
- Example using communication lines: 85
- How are special resources used?: 85
Chapter 6. Creating calendars and periods

Chapter 7. Creating applications and groups
Restart and cleanup of operations .... 161
Specifying extended information .... 161
Specifying system automation information ... 162
Creating user fields to specify additional information .... 164
Specifying remote job information in shadow jobs .... 164
Associating job statements with operations .... 165
Job statements and computer workstation operations .... 165
JCL and job setup operations .... 166
Using print operations .... 167
Using WTO operations .... 167
Started-task operations .... 167
Scheduling the closedown of online systems and started tasks .... 168
Creating time-dependent operations .... 168
Creating job descriptions .... 169
Using the Job Description panel .... 173
Creating run cycles for job descriptions .... 173
Specifying additional operation details for job descriptions .... 173
How job descriptions affect the current and long-term plans .... 174
Listing applications .... 174
Creating a list of applications .... 174
Performing tasks from the List of Applications panel .... 176
Browsing an application .... 178
Browsing run cycle information .... 180
Browsing application operations .... 181
Chapter 8. Defining applications in batch .... 185
Mass update or batch loader? .... 185
Maintaining a sequential data set .... 185
Changing application descriptions in batch .... 185
Changing group definitions and operator instructions .... 186
Making changes conditional on the original values .... 186
Searching applications for values in specified fields .... 186
How to run the mass update utility .... 186
What is the batch loader? .... 189
Input .... 189
Output .... 190
Validity checking .... 190
Description of the databases .... 190
AD database .... 190
OI Database .... 192
How to code batch-loader control statements .... 192
Structure of AD control statements .... 192
Structure of OI control statements .... 192
Example of control statement sequence .... 193
Batch-loader security considerations .... 194
How to run the batch-loader program .... 194
Coding the JCL .... 194
Batch-loader reports .... 196
Batch-loader sample .... 196
Batch-loader control statements .... 200
Syntax and construction .... 200
Defaults .... 201
ADCNC .... 203
ADCNs .... 205
ADDEP .... 208
ADOP .... 211
ADOP ExtN .... 219
ADOP SAI .... 220
ADRE .... 222
ADR ULE .... 224
ADR U N .... 228
ADSR .... 231
ADSTART .... 233
ADUSF .... 237
OIT .... 238
O I S TART .... 239
OPTIONS .... 242
Chapter 9. Overview of the long-term and current plans ........ 245
Long-term plan .... 247
Current plan .... 249
Removing data from NCP by running daily planning .... 250
Resolving pending occurrences .... 250
Tail plans .... 250
Pending occurrences .... 251
Creating the plans for the first time .... 251
Chapter 10. Producing and modifying the long-term plan .... 255
Running long-term plan batch jobs .... 256
The long-term plan data sets .... 256
Long-term plan messages .... 258
How a long-term plan is created .... 257
Creating the long-term plan .... 258
Extending the long-term plan .... 259
Modifying the long-term plan in batch .... 259
Creating reports about the long-term plan .... 260
Generated comments in long-term plan reports .... 261
Creating a trial long-term plan .... 261
Displaying the status of the long-term plan .... 262
Setting default successor and predecessor workstations .... 263
Amending applications in the long-term plan online .... 264
Modifying occurrences in the long-term plan .... 264
Modifying dependencies in the long-term plan .... 265
Chapter 11. Producing the current plan .... 267
Information used by daily planning .... 267
Creating and extending the current plan .... 268
Extending the current plan .... 270
Recreating the current plan .... 271
Creating a trial current plan .... 272
For end-to-end scheduling with fault tolerance capabilities .... 272
Renewing the symphony file .... 273
Producing reports using daily planning .... 273
Step 3. Define a shadow job on the remote engine workstation. 408
Step 4. Add a dependency on the shadow job 409
Daily plan consistency checks on shadow jobs and remote engine workstations 409
Monitoring a cross dependency resolution in the current plan 410
How the shadow job status changes until the bind is established 411
How a distributed shadow job is bound with a remote job instance 412
How a z/OS shadow job is bound with a remote job instance 415
Monitoring the shadow job status changes after the bind is established 418
Modifying shadow job instances in the current plan 421
Shadow job status transition during the remote job recovery or rerun 422
Modifying remote engine workstations in the current plan 423
Contacting the remote engine in a failover scenario 423

Chapter 21. Running event-driven workload automation 425

Business scenario 425
The event-driven process 425
Data set triggering 425
Defining event rules 427
Producing the configuration files 427
Deploying configuration files 428
Effects on special resource availability 429
HFS or ZFS file triggering 429
Syntax 429
Arguments 430
Example 431
Adding occurrences by event-triggered tracking 432
Triggering event types 432
Activating ETT 433
Defining your ETT criteria 433
Automatically adding an occurrence to the current plan 435
How you can use ETT to automate tasks 437
Using ETT occurrence-related variables 437
Combining data set triggering and ETT 440

Chapter 22. Job tailoring 441

Variable substitution 441
Associating variable tables with applications 442
Invoking and avoiding variable substitution 444
Coding variables in JCL 445
Ampersand, percent, and question mark variables 446
Ampersand variables 446
Percent variables 447
Simple variables 447
Compound variables 447
Question mark variables 448
Supplied variables 449
Occurrence-related supplied variables 449

Chapter 23. Job scheduling and WLM 493

Integrating with the service class 495
Environment 495
When a job should be set as critical 495
Selecting WLM assistance policies 495
Integrating with the scheduling environment 497
Associating a scheduling environment with a job 497
Submitting prevention and automatic tailoring 498
Pre-existing definition in the JCL JOB card 498
Checking for scheduling environment availability 498
Restrictions for using job card comments on the right 499
Chapter 27. Handling operations that end in error .......... 593
Displaying the ended-in-error list for action .......... 594
Selecting an ended-in-error list layout .......... 594
Creating your own ended-in-error list layout .......... 594
Getting rerun or recovery instructions .......... 596
Completing an ended-in-error operation .......... 596
Modifying a job that has failed .......... 597
Restarting ended-in-error operations .......... 597
Restarting ended-in-error operations managing cleanup action .......... 597
Taking action at the occurrence level .......... 599
Handling operations with the OSEQ error code .......... 599
Restarting an operation from a certain step .......... 600
Using cleanup options .......... 601
Specifying automatic restart for operations that fail .......... 602

Chapter 28. Monitoring special resources .......... 605
Understanding special resources .......... 605
Example using data sets .......... 607
Example using tape drives .......... 608
Example using communication lines .......... 608
How does the scheduler use special resources? .......... 609
Using the special resource monitor .......... 610
Understanding availability intervals .......... 610
Accessing the special resource monitor .......... 611
Looking at the operations waiting for a resource .......... 613
Modifying a special resource .......... 615

Chapter 29. Using Tivoli Business Systems Manager .......... 619
Enabling monitoring by Tivoli Business Systems Manager .......... 619
Scheduler start options .......... 619
Identifying jobs for monitoring .......... 620
Setting monitors from the ISPF panels .......... 620
Setting monitors from the programming interface .......... 621
How monitoring works .......... 621
Discovery of the scheduler .......... 622

Chapter 30. Using IBM Tivoli Monitoring .......... 623
Enabling monitoring by IBM Tivoli Monitoring .......... 624
Controller Configuration options .......... 624
Tivoli Enterprise Portal Configuration options .......... 625
How monitoring works .......... 626
Discovery of monitored objects .......... 627
Identifying and displaying alerts .......... 627
Identifying jobs for monitoring .......... 628
Setting monitors from the ISPF panels .......... 628
Setting monitors from the programming interface .......... 630
Setting monitors from the graphical user interfaces .......... 630

Chapter 31. Reporting with Tivoli Workload Scheduler for z/OS .......... 631
Archiving historical data .......... 632
Historical reports .......... 633
Setting up the environment .......... 635
Setting up the database .......... 637
Creating the database .......... 637
Migrating the database to the latest version .......... 639
Encrypting the password in DBOPT statement .......... 639
Setting up RACF authorization .......... 640
Running batch reports from the command line interface .......... 640
A sample business scenario .......... 640
Setting up for command line reporting .......... 641
Running batch reports .......... 642
Examples .......... 643
Logs and traces for batch reports .......... 644
Keeping historical data in the database .......... 645

Appendix A. TSO commands .......... 647
Important considerations for using the TSO commands .......... 647
BACKUP .......... 648
BULKDISC .......... 651
JSUACT .......... 653
OPINFO .......... 655
OPSTAT .......... 659
SRSTAT .......... 664
WSSTAT .......... 669

Appendix B. Batch programs .......... 673
DD statements referenced by batch jobs .......... 673
DD statements used by EQQBATCH .......... 673
DD statements describing the data .......... 674
Security and batch programs .......... 674
Batch programs and JCL examples .......... 675
EQQADCOP - Print applications: calculate and print application run days .......... 676
SYIN requirements .......... 676
JCL example .......... 676
EQQADDEP - Print applications: cross-reference external dependencies .......... 677
SYIN requirements .......... 677
JCL example .......... 677
EQQADMUP - Perform mass update of application Descriptions .......... 677
SYIN requirements .......... 677
JCL example .......... 677
EQQAXR00 - Print applications: cross-reference different items .......... 681
SYIN requirements .......... 682
JCL example .......... 682
## Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>A rolling long-term plan</td>
<td>8</td>
</tr>
<tr>
<td>2.</td>
<td>Extending the current plan</td>
<td>8</td>
</tr>
<tr>
<td>3.</td>
<td>Example of payroll jobs at paymore incorporated</td>
<td>14</td>
</tr>
<tr>
<td>4.</td>
<td>EQQWCGEP - Creating general information about a workstation.</td>
<td>18</td>
</tr>
<tr>
<td>5.</td>
<td>Specifying the payroll database as a resource</td>
<td>21</td>
</tr>
<tr>
<td>6.</td>
<td>Changing the default calendar</td>
<td>22</td>
</tr>
<tr>
<td>7.</td>
<td>Creating a variable</td>
<td>23</td>
</tr>
<tr>
<td>8.</td>
<td>Creating the PAYDAILY application</td>
<td>25</td>
</tr>
<tr>
<td>9.</td>
<td>Creating the run cycle for PAYDAILY</td>
<td>25</td>
</tr>
<tr>
<td>10.</td>
<td>Creating the rule for PAYDAILY</td>
<td>26</td>
</tr>
<tr>
<td>11.</td>
<td>Creating operations in PAYDAILY</td>
<td>27</td>
</tr>
<tr>
<td>12.</td>
<td>Specifying the predecessors for PAYDAILY operations</td>
<td>28</td>
</tr>
<tr>
<td>13.</td>
<td>Specifying the payroll database as a resource</td>
<td>28</td>
</tr>
<tr>
<td>14.</td>
<td>Specifying that the WTO is time-dependent</td>
<td>30</td>
</tr>
<tr>
<td>15.</td>
<td>Creating the long-term plan</td>
<td>31</td>
</tr>
<tr>
<td>16.</td>
<td>Listing occurrences in the long-term plan</td>
<td>31</td>
</tr>
<tr>
<td>17.</td>
<td>Creating the current plan</td>
<td>32</td>
</tr>
<tr>
<td>18.</td>
<td>EQQOPCAP - Main menu</td>
<td>35</td>
</tr>
<tr>
<td>19.</td>
<td>EQQXOPTP - Defining parameters and options</td>
<td>36</td>
</tr>
<tr>
<td>20.</td>
<td>EQQXDATP - Setting date and time format</td>
<td>37</td>
</tr>
<tr>
<td>21.</td>
<td>EQQXCOLP - Setting color and highlight attributes</td>
<td>38</td>
</tr>
<tr>
<td>22.</td>
<td>EQQXAIOIP - Setting AD/OI consistency check</td>
<td>40</td>
</tr>
<tr>
<td>23.</td>
<td>EQQXJCLP - Setting JCL edit tool information</td>
<td>41</td>
</tr>
<tr>
<td>24.</td>
<td>EQQXPSTL - Setting the panel style</td>
<td>42</td>
</tr>
<tr>
<td>25.</td>
<td>EQQSOIPF - Selecting operations</td>
<td>44</td>
</tr>
<tr>
<td>26.</td>
<td>EQQMOPRV - Choose View in the menu bar</td>
<td>45</td>
</tr>
<tr>
<td>27.</td>
<td>EQQNALSL - Choose View in the menu bar to change the panel view</td>
<td>46</td>
</tr>
<tr>
<td>28.</td>
<td>EQQXSRTL - Sorting a list</td>
<td>47</td>
</tr>
<tr>
<td>29.</td>
<td>ISPOPT3B - PF key definitions and labels</td>
<td>49</td>
</tr>
<tr>
<td>30.</td>
<td>EQQXSUBP - Generating JCL for a batch job</td>
<td>50</td>
</tr>
<tr>
<td>31.</td>
<td>Example of virtual workstation matching a JES2 MAS</td>
<td>60</td>
</tr>
<tr>
<td>32.</td>
<td>Dependent operations with complex dependencies</td>
<td>61</td>
</tr>
<tr>
<td>33.</td>
<td>Using a dummy operation to simplify complex dependencies</td>
<td>62</td>
</tr>
<tr>
<td>34.</td>
<td>EQQWMLSL - List of workstation descriptions</td>
<td>63</td>
</tr>
<tr>
<td>35.</td>
<td>EQQWCGEP - Creating general information about a workstation.</td>
<td>63</td>
</tr>
<tr>
<td>36.</td>
<td>EQQWDES - Modifying virtual workstation destination</td>
<td>64</td>
</tr>
<tr>
<td>37.</td>
<td>EQQWAV - Availability of a workstation</td>
<td>64</td>
</tr>
<tr>
<td>38.</td>
<td>EQQWMTAL - All open time interval</td>
<td>65</td>
</tr>
<tr>
<td>39.</td>
<td>Computer workstation with server usage set to B or C, controlling job submission</td>
<td>72</td>
</tr>
<tr>
<td>40.</td>
<td>EQQWAVL - Availability of a workstation</td>
<td>75</td>
</tr>
<tr>
<td>41.</td>
<td>EQQWMTOL - Open time intervals for one day</td>
<td>75</td>
</tr>
<tr>
<td>42.</td>
<td>EQQWMTL - All open time intervals</td>
<td>76</td>
</tr>
<tr>
<td>43.</td>
<td>EQQMACL - Modifying all workstations closed</td>
<td>77</td>
</tr>
<tr>
<td>44.</td>
<td>EQWRE - Resources for a workstation</td>
<td>78</td>
</tr>
<tr>
<td>45.</td>
<td>EQQDBSP - Maintaining TWSz databases</td>
<td>89</td>
</tr>
<tr>
<td>46.</td>
<td>EQQDTOP - Maintaining special resources</td>
<td>89</td>
</tr>
<tr>
<td>47.</td>
<td>EQQDLSL - List of special resources</td>
<td>89</td>
</tr>
<tr>
<td>48.</td>
<td>EQQDCRP - Creating a special resource</td>
<td>90</td>
</tr>
<tr>
<td>49.</td>
<td>EQQDWML - Modifying connected workstations for a special resource</td>
<td>92</td>
</tr>
<tr>
<td>50.</td>
<td>EQQDIML - Modifying intervals for a special resource</td>
<td>92</td>
</tr>
<tr>
<td>51.</td>
<td>EQQDWML - Modifying connected workstations for a special resource</td>
<td>93</td>
</tr>
<tr>
<td>52.</td>
<td>EQTCC - Creating a calendar</td>
<td>104</td>
</tr>
<tr>
<td>53.</td>
<td>EQTPERL - List of calendar periods</td>
<td>107</td>
</tr>
<tr>
<td>54.</td>
<td>EQTCRPL - Creating a calendar period</td>
<td>107</td>
</tr>
<tr>
<td>55.</td>
<td>A 5-day work-days-only cyclic period, MYWEEK</td>
<td>112</td>
</tr>
<tr>
<td>56.</td>
<td>How Tivoli Workload Scheduler for z/OS counts offsets with work-days-only cyclic periods</td>
<td>112</td>
</tr>
<tr>
<td>57.</td>
<td>Results when the work-days-only cyclic period origin is a free day</td>
<td>113</td>
</tr>
<tr>
<td>58.</td>
<td>EQQASUBP - Maintaining application descriptions</td>
<td>120</td>
</tr>
<tr>
<td>59.</td>
<td>EQACGPP - Creating an application</td>
<td>120</td>
</tr>
<tr>
<td>60.</td>
<td>EQAMRPL - Run cycles</td>
<td>125</td>
</tr>
<tr>
<td>61.</td>
<td>EQRULEP - Modifying a rule</td>
<td>127</td>
</tr>
<tr>
<td>62.</td>
<td>EQRLSL - List of generated dates</td>
<td>128</td>
</tr>
<tr>
<td>63.</td>
<td>EQARNL - Run days</td>
<td>130</td>
</tr>
<tr>
<td>64.</td>
<td>The effect of the free-day rule</td>
<td>131</td>
</tr>
<tr>
<td>65.</td>
<td>Negative run cycles</td>
<td>132</td>
</tr>
<tr>
<td>66.</td>
<td>Using in-effect dates to switch between run cycles</td>
<td>133</td>
</tr>
<tr>
<td>67.</td>
<td>EQAMREP - Every options</td>
<td>134</td>
</tr>
<tr>
<td>68.</td>
<td>EQAMOPL - Operations</td>
<td>138</td>
</tr>
<tr>
<td>69.</td>
<td>EQAMOSL - Operations</td>
<td>140</td>
</tr>
<tr>
<td>70.</td>
<td>EQAMSDP - Operation details</td>
<td>144</td>
</tr>
<tr>
<td>71.</td>
<td>EQAMPPDL - Predecessors</td>
<td>144</td>
</tr>
<tr>
<td>72.</td>
<td>EQAMCCL - Conditions list</td>
<td>145</td>
</tr>
<tr>
<td>73.</td>
<td>EQAMCCP - Condition dependencies definitions</td>
<td>145</td>
</tr>
<tr>
<td>74.</td>
<td>EQAMSRL - Special resources</td>
<td>148</td>
</tr>
<tr>
<td>75.</td>
<td>EQAMWRP - Work station resources and servers</td>
<td>150</td>
</tr>
<tr>
<td>76.</td>
<td>EQAMBP - Job, WTO, and print options</td>
<td>151</td>
</tr>
<tr>
<td>77.</td>
<td>EQAMFBP - Feedback options</td>
<td>157</td>
</tr>
<tr>
<td>78.</td>
<td>EQAMTP - Time specifications</td>
<td>158</td>
</tr>
<tr>
<td>79.</td>
<td>EQALSML - List of operator instructions</td>
<td>159</td>
</tr>
<tr>
<td>80.</td>
<td>EQKCRTE - Creating an operator instruction</td>
<td>160</td>
</tr>
<tr>
<td>81.</td>
<td>EQAQIDP - Confirm the deletion of OI</td>
<td>160</td>
</tr>
<tr>
<td>82.</td>
<td>EQAMARL - Restart and cleanup of operation details</td>
<td>161</td>
</tr>
<tr>
<td>83.</td>
<td>EQQAMXDP - Operation extended info</td>
<td>162</td>
</tr>
<tr>
<td>Figure</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>--------</td>
<td>-----------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>306</td>
<td>Long-term plan report for applications, sorted by owner</td>
<td>717</td>
</tr>
<tr>
<td>307</td>
<td>Long-term plan report - Total duration per workstation</td>
<td>718</td>
</tr>
<tr>
<td>308</td>
<td>Long-term plan report - Grand total workload for period</td>
<td>719</td>
</tr>
<tr>
<td>309</td>
<td>Daily planning reports - General information</td>
<td>720</td>
</tr>
<tr>
<td>310</td>
<td>Daily planning reports - Daily operating plan</td>
<td>721</td>
</tr>
<tr>
<td>311</td>
<td>Daily planning reports - Plan for workstation</td>
<td>722</td>
</tr>
<tr>
<td>312</td>
<td>Daily planning reports - Workstation utilization (parallel operations)</td>
<td>723</td>
</tr>
<tr>
<td>313</td>
<td>Daily planning reports - Workstation utilization (resource 1)</td>
<td>724</td>
</tr>
<tr>
<td>314</td>
<td>Daily planning reports - Planned resource utilization</td>
<td>724</td>
</tr>
<tr>
<td>315</td>
<td>Daily planning reports - Summary of completed applications</td>
<td>726</td>
</tr>
<tr>
<td>316</td>
<td>Daily planning reports - Completed applications</td>
<td>726</td>
</tr>
<tr>
<td>317</td>
<td>Daily planning reports - Error statistics on completed applications</td>
<td>727</td>
</tr>
<tr>
<td>318</td>
<td>Daily planning reports - Operations in error</td>
<td>728</td>
</tr>
<tr>
<td>319</td>
<td>Daily planning reports - Missed feedback report</td>
<td>728</td>
</tr>
<tr>
<td>320</td>
<td>Daily planning reports - Actual resource utilization</td>
<td>729</td>
</tr>
<tr>
<td>Table</td>
<td>Description</td>
<td>Page</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>------</td>
</tr>
<tr>
<td>1</td>
<td>Grouping of payroll applications</td>
<td>16</td>
</tr>
<tr>
<td>2</td>
<td>Creating workstations for paymore</td>
<td>19</td>
</tr>
<tr>
<td>3</td>
<td>Primary commands for the panels</td>
<td>43</td>
</tr>
<tr>
<td>4</td>
<td>Settings for fault-tolerant workstations</td>
<td>67</td>
</tr>
<tr>
<td>5</td>
<td>Settings for Tivoli Workload Scheduler for z/OS agent workstations</td>
<td>68</td>
</tr>
<tr>
<td>6</td>
<td>Settings for dynamic workstations</td>
<td>69</td>
</tr>
<tr>
<td>7</td>
<td>Settings for remote engine workstations</td>
<td>71</td>
</tr>
<tr>
<td>8</td>
<td>How resource quantity and availability can be changed</td>
<td>88</td>
</tr>
<tr>
<td>9</td>
<td>Where values are taken from for each interval</td>
<td>93</td>
</tr>
<tr>
<td>10</td>
<td>Differences between applications and group definitions</td>
<td>118</td>
</tr>
<tr>
<td>11</td>
<td>How to use the RUN and OPER commands and the GROUP DEFINITION field</td>
<td>121</td>
</tr>
<tr>
<td>12</td>
<td>Examples of rules</td>
<td>124</td>
</tr>
<tr>
<td>13</td>
<td>Effect of the input arrival time and the free-day rule</td>
<td>129</td>
</tr>
<tr>
<td>14</td>
<td>Period definitions required for offset examples</td>
<td>136</td>
</tr>
<tr>
<td>15</td>
<td>Examples of smoothing factors</td>
<td>157</td>
</tr>
<tr>
<td>16</td>
<td>Examples of limits for feedback</td>
<td>158</td>
</tr>
<tr>
<td>17</td>
<td>Comparison of batch loader (BL) and mass update (MU)</td>
<td>185</td>
</tr>
<tr>
<td>18</td>
<td>Deciding whether to update the active subsystem</td>
<td>193</td>
</tr>
<tr>
<td>19</td>
<td>Determining the best restart step</td>
<td>324</td>
</tr>
<tr>
<td>20</td>
<td>Cleanup actions for data set dispositions in a job</td>
<td>335</td>
</tr>
<tr>
<td>21</td>
<td>Symbols that mark the end of variables</td>
<td>445</td>
</tr>
<tr>
<td>22</td>
<td>Occurrence-related supplied variables</td>
<td>449</td>
</tr>
<tr>
<td>23</td>
<td>Operation-related supplied variables</td>
<td>451</td>
</tr>
<tr>
<td>24</td>
<td>Date-related supplied variables</td>
<td>452</td>
</tr>
<tr>
<td>25</td>
<td>Dynamic-format date-related supplied variables</td>
<td>453</td>
</tr>
<tr>
<td>26</td>
<td>Specifying the SETUP option</td>
<td>456</td>
</tr>
<tr>
<td>27</td>
<td>Dynamic-format substitution results</td>
<td>473</td>
</tr>
<tr>
<td>28</td>
<td>Date formats allowed in the SETVAR directive</td>
<td>478</td>
</tr>
<tr>
<td>29</td>
<td>Day-in-year formats allowed in the SETVAR directive</td>
<td>478</td>
</tr>
<tr>
<td>30</td>
<td>Month formats allowed in the SETVAR directive</td>
<td>478</td>
</tr>
<tr>
<td>31</td>
<td>Time formats allowed in the SETVAR directive</td>
<td>478</td>
</tr>
<tr>
<td>32</td>
<td>Advantages and disadvantages of assistance policies</td>
<td>496</td>
</tr>
<tr>
<td>33</td>
<td>JESPLEX parameters for the trackers</td>
<td>506</td>
</tr>
<tr>
<td>34</td>
<td>The MAS resources and their associated JESplex</td>
<td>507</td>
</tr>
<tr>
<td>35</td>
<td>The MAS resources and their associated JESplex</td>
<td>507</td>
</tr>
<tr>
<td>36</td>
<td>Using the MODIFYING CURRENT PLAN panel</td>
<td>548</td>
</tr>
<tr>
<td>37</td>
<td>Codes for the ability to restart a step</td>
<td>600</td>
</tr>
<tr>
<td>38</td>
<td>How attributes are preserved across intervals</td>
<td>610</td>
</tr>
<tr>
<td>39</td>
<td>Supported report output formats</td>
<td>632</td>
</tr>
<tr>
<td>40</td>
<td>Summary of historical reports</td>
<td>633</td>
</tr>
<tr>
<td>41</td>
<td>Batch programs</td>
<td>675</td>
</tr>
<tr>
<td>42</td>
<td>Keyword and owner combinations</td>
<td>741</td>
</tr>
<tr>
<td>43</td>
<td>Fields available for display in ready and error lists</td>
<td>749</td>
</tr>
<tr>
<td>44</td>
<td>Valid number of occurrences for a language element</td>
<td>753</td>
</tr>
<tr>
<td>45</td>
<td>SMF events</td>
<td>754</td>
</tr>
<tr>
<td>46</td>
<td>Parameters of ReadCompleted and ModificationCompleted event types</td>
<td>755</td>
</tr>
<tr>
<td>47</td>
<td>Parameters of SpecialResourceEvent action type</td>
<td>757</td>
</tr>
</tbody>
</table>
About this publication

IBM® Tivoli® Workload Scheduler for z/OS® Managing the Workload was created by combining what used to be two separate publications:

- **Planning and Scheduling the Workload**
  Describes organizing when your workload runs, and the dependencies between parts of the workload. The terms *planning* and *scheduling* are used interchangeably and describe these activities. [Part 1, “Planning and scheduling,” on page 1](#) of *Managing the Workload* contains this information.

- **Controlling and Monitoring the Workload**
  Describes managing the workload when it becomes part of a plan and is run with real dates and times. [Part 2, “Controlling and monitoring,” on page 517](#) of *Managing the Workload* contains this information.

The term *scheduler*, when used in this publication, refers to IBM Tivoli Workload Scheduler for z/OS. The term DB2®, when used in this publication, also includes references to DATABASE 2 and DB2 Universal Database.

What is new in this release

For information about the new and changed functions in this release, see *Tivoli Workload Automation: Overview*.

What is new in this publication

This section lists what has changed in this publication since version 8.5.1. Except for editorial changes, changed or added text is marked in the left margin with a vertical bar. The following product enhancements are described in this publication:

- Use cross dependencies to integrate the workload running on different engines, which can be Tivoli Workload Scheduler for z/OS engines (controller) or Tivoli Workload Scheduler engines (master domain manager and backup master domain manager). For more information see [Chapter 20, “Defining and managing cross dependencies,” on page 405](#).

- Several enhancements to the usability of ISPF panels have been made. A new panel style can be applied to the AD application to enable you to list and browse a single AD, and also for the CP operation, to list and browse a single operation in the plan. The new panel style gives you a quick, at-a-glance scrollable view of the AD application descriptions and CP operations. A further enhancement related to color-coded fields enable you to quickly distinguish the status of applications and operations. The color code can also be applied to distinguish applications from groups of applications. See [setting color options and panel style](#) in “Setting options” on page 36, as well as “Listing applications” on page 174 and “Listing and browsing operations” on page 570. See also how colors are used to distinguish different operation status codes in “Operation status codes” on page 743.

- You can apply variable substitution also to job types with advanced options (jobs that run tasks within the areas of File Transfer, Web Services, Database, Java, and more). See “Variable substitution for job types with advanced options” on page 491 for details.

- Job logs can be configured to be automatically retrieved when a job ends in error. The automatic job log retrieval can be customized also for z-centric jobs.
What is new in this publication

that end in error. See “What you need for job log retrieval” on page 304. You can also configure to automatically retrieve the restart information for a restart and cleanup action. For more information about the keyword settings, see the JOBLOGRETRIEVAL and RESTARTINFORETRIEVAL keywords in the HTTPOPTS and RCLOPTS statements in Customization and Tuning.

• Run batch reports for historical data from a command-line interface. See “Running batch reports from the command line interface” on page 640.

• Send generated reports by email. Reports generated by running batch programs can optionally be sent to other users by email. Simply customize the file, SENDREPORT.MAC, to include the email address, the subject, a message, the name of the report, and the name of the output file generated. See “Sending generated reports by email” on page 703.

Who should read this publication

This publication is intended for those involved in planning, scheduling, monitoring, or managing the work in the production department of a computer installation. You do not need to know any programming languages to use this publication. However, you must know the work that you will automate using Tivoli Workload Scheduler for z/OS, how that work is broken down into jobs and what the dependencies between them are.

To make the most of Tivoli Workload Scheduler for z/OS, work together with the system programmer who installs and customizes Tivoli Workload Scheduler for z/OS. Work closely with the job preparation and operations staff that control the work with Tivoli Workload Scheduler for z/OS; their tasks will change considerably when Tivoli Workload Scheduler for z/OS controls your work, so consider their new tasks, which are described in Managing the Workload.

Publications

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

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

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

Using LookAt to look up message explanations

LookAt is an online facility that lets you look up explanations for most of the IBM messages you encounter, as well as for some system abends (an abnormal end of a task) and codes. Using LookAt to find information is faster than a conventional search because in most cases LookAt goes directly to the message explanation.

You can use LookAt from the following locations to find IBM message explanations for z/OS elements and features, z/VM®, VSE/ESA, and Clusters for AIX® and Linux:

• The Internet. You can access IBM message explanations directly from the LookAt website at http://www.ibm.com/eserver/zseries/zos/bkserv/lookat/
Using LookAt

- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX System Services running OMVS).

- Your Microsoft Windows workstation. You can install code to access IBM message explanations on the IBM Online Library z/OS Software Products Collection Kit (SK3T-4270), using LookAt from a Microsoft Windows DOS command line.

- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt website.

You can obtain code to install LookAt on your host system or Microsoft Windows workstation from a disk on your IBM Online Library z/OS Software Products Collection Kit (SK3T-4270), or from the LookAt website (click Download, and select the platform, release, collection, and location that suit your needs). More information is available in the LOOKAT.ME files available during the download process.

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.


Tivoli technical training

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

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

Support information

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

Online

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

IBM Support Assistant

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

Troubleshooting Guide

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

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

Conventions used in this publication

The publication uses several typeface conventions for special terms and actions. Technical changes to the text are indicated by a vertical line to the left of the change. These conventions have the following meanings:

<table>
<thead>
<tr>
<th>Information type</th>
<th>Style convention</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Commands</td>
<td>All capital letters</td>
<td>CREATE</td>
</tr>
<tr>
<td>References in the text to fields on panels</td>
<td>All capital letters</td>
<td>QUANTITY</td>
</tr>
<tr>
<td>Input you should type in panel fields</td>
<td>Monospace</td>
<td>MYAPPLICATION</td>
</tr>
<tr>
<td>First time new term introduced, publication titles</td>
<td>Italic</td>
<td>Application</td>
</tr>
</tbody>
</table>

How to read the syntax diagrams

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

```
+---SRSTAT——resource name——+
   | SUBSYS(subsystem name)     |
   | OPCA                        |
   | AVAIL(KEEP             NO               )  |
   |                       RESET              |
   | QUANTITY(amount            KEEP             )  |
   |                       RESET              |
   | CREATE(YES               NO              )  |
   |                       RESET              |
   | TRACE(trace level)         |
```

The symbols have these meanings:

- The statement begins here.
- The statement is continued on the next line.
How to read the syntax diagrams

The statement is continued from a previous line.

The statement ends here.

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

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

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

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

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

These are the conventions used in the diagrams:
• Required items appear on the horizontal line (main path):

  >>>STATEMENT-required item

• Optional items appear below the main path:

  >>>STATEMENT

  optional item

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

  >>>STATEMENT

  repeatable item

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

    >>>STATEMENT

    required choice 1

    required choice 2

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

    >>>STATEMENT

    optional choice 1

    optional choice 2

  – A repeat arrow above a stack indicates that you can make more than one choice from the stacked items:
How to read the syntax diagrams

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

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

**option 1**

- default

**option 2**

- default

IBM Tivoli Workload Scheduler for z/OS: Managing the Workload
Part 1. Planning and scheduling

Chapter 1. What is the scheduler? ............................................. 7
  How the scheduler works .................................................. 7
  How does the Scheduler keep track of jobs? ......................... 9
  Can you define a dependency for the job? ......................... 9
  How does the scheduler help with job preparation? ............... 10
  How can the scheduler help you recover jobs? ................... 10
  Can the scheduler help with online systems? ...................... 10
  Can you run jobs outside the scheduler? ......................... 11
Finding out about the scheduler ......................................... 11

Chapter 2. A user scenario .................................................. 13
  Implementing a payroll system ......................................... 13
  Designing the payroll applications .................................. 15
  Creating workstations .................................................... 17
  Controlling general workstations .................................... 18
  Using servers ............................................................... 19
  Creating special resources ............................................. 20
  Creating the default calendar ........................................ 21
  Preparing the payroll jobs ............................................. 22
  Creating the groups, applications, and operations ................ 24
  Creating plans ............................................................. 30
  What happens if jobs or started tasks fail? ....................... 32
  Coping with more complex problems ................................ 33
Summary of what you need to do ........................................ 34

Chapter 3. Using the scheduler panels ................................. 35
  The scheduler ISPF panels ............................................. 35
  Setting options ........................................................... 36
  Common panel commands and facilities ................................ 42
  Concatenating options .................................................. 42
  Quick return command .................................................. 42
  Primary commands ....................................................... 43
  Specifying list criteria ................................................ 43
  Specifying panel views ................................................. 44
  Using generic search arguments ..................................... 46
  Sorting list output ....................................................... 46
  Locating data strings in list output ................................ 47
  Graphically displaying lists ......................................... 48
  PF Key assignment ...................................................... 48
Modeling your environment .............................................. 49
The graphical user interface ............................................. 50
Utilities ............................................................................. 50

Chapter 4. Creating workstations ........................................ 53
  What types of workstation are there? ................................ 53
  Computer workstations .................................................. 53
  Job computer workstations ............................................ 54
  Started-task computer workstations ................................ 54
  Virtual workstations ..................................................... 55
  Fault-tolerant workstations .......................................... 55
  Tivoli Workload Scheduler for z/OS agent workstations ....... 55
  Dynamic workstations ................................................... 55
  Alternate workstations .................................................. 56
  Printer workstations ..................................................... 56
  General workstations .................................................... 56
  Job setup general workstations ...................................... 57
  WTO general workstations ............................................ 57
  WAIT general workstations ........................................... 57
  Remote engine workstations ......................................... 58
  Guidelines for creating workstations ............................... 58
  Which workstation types are needed? ............................... 59
  How many workstations of each type? ............................. 59
  Using virtual workstations ......................................... 60
  Dummy workstations .................................................... 61
  Creating a workstation .................................................. 62
  Specifying workstation attributes and options ................. 65
  Specifying workstation reporting attributes .................... 65
  Specifying fault-tolerant or Tivoli Workload Scheduler for z/OS agent workstations ........................................ 67
  Specifying dynamic workstations .................................. 69
  Specifying remote engine workstations ............................ 71
  Specifying workstation parallel servers ........................... 71
  Specifying splittable workstations .................................. 72
  Specifying workstation destinations ............................... 73
  Specifying the default workstation transport time ............. 73
  Specifying the default duration ..................................... 74
  Specifying the Automation option ................................... 74
  Specifying workstation availability ................................ 74
  Specifying workstation open intervals ............................ 75
  Closing workstations ................................................... 76
  Specifying workstation fixed resources ........................... 78
  Controlling workstations ............................................. 79

Chapter 5. Creating special resources ................................ 81
  Understanding special resources ..................................... 81
  Example using data sets ............................................... 84
  Example using tape drives ............................................. 84
  Example using communication lines ................................. 85
  How are special resources used? ..................................... 85
  Updating the special resource database ............................ 86
  Creating special resources ............................................ 87
  Setting the availability of a resource ............................... 94
  Using NetView to set the availability of a resource ............ 94
  Using RODM to change the availability of a resource ........ 95
  Using On Complete to change the availability of a resource .. 95
  Using Max Usage Limit to change the availability of a resource ....................................................... 96
  Using SRSTAT LIFESPAN to change the availability of a resource ..................................................... 96
  Using event-triggered resource handling to change the availability of a resource ................................... 98
  Creating a resource dynamically ...................................... 98
  Creating missing resources during batch planning ............... 98
  Creating missing resources during the life of the plan .......... 98

© Copyright IBM Corp. 1991, 2011
How job descriptions affect the current and long-term plans .......................... 174
Listing applications ................................................. 174
Creating a list of applications ...................................... 174
Performing tasks from the List of Applications panel .......... 176
Browsing an application ............................................. 178
Browsing run cycle information ..................................... 180
Browsing application operations .................................... 181

Chapter 8. Defining applications in batch ........................................ 185
Mass update or batch loader? ........................................ 185
Maintaining a sequential data set .................................... 185
Changing application descriptions in batch ....................... 185
Changing group definitions and operator instructions ............ 186
Making changes conditional on the original values ............... 186
Searching applications for values in specified fields .......... 186
How to run the mass update utility ................................. 186
What is the batch loader? ............................................. 189
Input ................................................................. 189
Output .............................................................. 190
Validity checking .................................................... 190
Description of the databases ........................................ 190
AD database ......................................................... 190
OI Database .......................................................... 192
How to code batch-loader control statements ..................... 192
Structure of AD control statements .................................. 192
Structure of OI control statements .................................. 192
Example of control statement sequence ............................. 193
Batch-loader security considerations ................................. 194
How to run the batch-loader program ............................... 194
Coding the JCL ....................................................... 194
Batch-loader reports .................................................. 196
Batch-loader sample ................................................... 196
Batch-loader control statements ...................................... 200
Syntax and construction ............................................. 200
Defaults ............................................................ 201
ADNCN ............................................................... 203
ADCN ............................................................... 205
ADDEP ............................................................... 208
ADOP ............................................................... 211
ADOPEXTN ......................................................... 219
ADOPSAI .......................................................... 220
ADRE ............................................................... 222
ADRULE ............................................................. 224
ADRUN .............................................................. 228
ADSR ............................................................... 231
ADSTART .......................................................... 233
ADUSF .............................................................. 237
OIT ................................................................. 238
OISTART .......................................................... 239
OPTIONS ........................................................... 242

Chapter 9. Overview of the long-term and current plans ...................... 245
Long-term plan ......................................................... 247
Current plan .......................................................... 249

Chapter 10. Producing and modifying the long-term plan ....................... 255
Running long-term plan batch jobs .................................. 256
The long-term plan data sets ......................................... 256
Long-term plan messages ............................................ 256
How a long-term plan is created ..................................... 257
Creating the long-term plan .......................................... 258
Extending the long-term plan ........................................ 259
Modifying the long-term plan in batch .............................. 259
Creating reports about the long-term plan ........................ 260
Generating comments in long-term plan reports .................. 261
Creating a trial long-term plan ....................................... 261
Displaying the status of the long-term plan ....................... 262
Setting default successor and predecessor workstations ......... 263
Amending applications in the long-term plan online ............. 264
Modifying occurrences in the long-term plan ........................ 264
Modifying dependencies in the long-term plan ..................... 265

Chapter 11. Producing the current plan ......................................... 267
Information used by daily planning .................................... 267
Creating and extending the current plan ............................ 268
Extending the current plan .......................................... 270
Recreating the current plan .......................................... 271
Creating a trial current plan ......................................... 272
For end-to-end scheduling with fault tolerance capabilities .... 272
Renewing the symphony file ......................................... 273
Producing reports using daily planning .............................. 273
Plan reports ........................................................ 273
Management reports ................................................. 274
Current period ....................................................... 274
Previous period ..................................................... 274
Using the track log .................................................. 276
Resolving dependencies between operations ....................... 276
Analyzing problems reported by daily planning ................... 277
For end-to-end scheduling with fault tolerance capabilities .... 278
Modifying the current plan .......................................... 278
Querying the current plan .......................................... 279
Current plan reference information .................................. 279
Organization and integrity of the current plan ...................... 280
Current plan during normal processing ............................. 281
The current-plan backup process .................................... 283
Creating and activating the new current plan ....................... 283
Problems with current plan batch jobs .............................. 286
Recovery actions .................................................... 287
Post-recovery actions ................................................. 287

Chapter 12. Using service and optional functions ............................. 289
Activating and deactivating job submission ......................... 289
Adding predecessor recovery occurrences to the current plan .................. 366
Status changes .................................................. 366
Status ................................................................. 366
Extended status .................................................... 366
Security in automatic recovery ......................................... 367
Recovery actions from the Modifying Current Plan panel .......... 367
Message and comment statements .................................. 368
Message statement .................................................. 368
Comment statement .................................................. 368
Logging and failure reporting ........................................ 368
Successful recovery start ........................................... 368
Unsuccessful recovery start ......................................... 369
Case code lists ....................................................... 369
Deactivating and activating automatic recovery ................. 369
Restarting work on different workstations ..................... 370

Chapter 19. Conditioning the processing of operations .......... 373
Using conditional dependencies .................................... 373
Evaluating conditions and conditional successor status ..... 374
How to define a rule in a condition ................................ 374
When the scheduler evaluates the status of a condition dependency .... 375
How the scheduler evaluates the status of a condition dependency .... 375
How the scheduler evaluates the status of a condition .... 376
How the scheduler evaluates the status of the conditional successor operation .... 376
Making an operation dependent on the status or return code of another operation ................... 377
Examples of job conditional dependencies evaluation .......... 377
Job conditional dependencies key concepts and restrictions ........ 381
How to ensure application consistency using FOP ......... 382
Making an operation dependent on another operation step return code ............. 383
How to identify a step .............................................. 384
How to activate step dependency evaluation ...................... 384
How a step dependency is evaluated ................................ 385
How the evaluation is done when no step-end event is received .......... 385
Examples of step-level conditional dependencies evaluation .......... 386
How to condition the status of operations with predecessors in X status ............. 390
How to propagate the X status in a branch ......................... 390
How to set to W the status of an operation with predecessors in X status .......... 391
Handling recovery using conditional dependencies .......... 392
An example of using recovery jobs to implement workflow .......... 393
An example of handling recovery in an end-to-end environment .......... 394
Interactions with other functions .................................. 395
MCP consistency rules .............................................. 395
Rerunning an occurrence ............................................ 396
Setting an occurrence to waiting .................................. 396
Setting an occurrence to complete ................................ 396
Deleting an operation or an occurrence ............................ 397
Monitoring conditions in the current plan ...................... 397
How to look for operations that ended in error ................. 397
How to see if an operation that ended in error prevents job processing from proceeding ........ 398
How to check if conditional successors are prevented from running .......... 398
How to check if conditional successors are not prevented from running ........ 399
How to check if normal successors are prevented from running .......... 399
How to check if an operation is waiting due to conditions .......... 399
How to search for skipped operations in a conditional flow .......... 400
How to display information about conditions ...................... 400
Daily plan batch handling of conditional dependencies .......... 402
Automatic conditional dependencies resolution ................. 402

Chapter 20. Defining and managing cross dependencies .......... 405
An introduction to cross dependencies .......................... 405
Defining a cross dependency in the database ..................... 406
Step 1. Set up destinations ......................................... 407
How to get the correct values to specify in the destination .......... 408
Step 2. Create a remote engine workstation ....................... 408
Step 3. Define a shadow job on the remote engine workstation ....................... 408
Step 4. Add a dependency on the shadow job ...................... 409
Daily plan consistency checks on shadow jobs and remote engine workstations ....................... 409
Monitoring a cross dependency resolution in the current plan .......... 410
How the shadow job status changes until the bind is established .......... 411
How a distributed shadow job is bound with a remote job instance .......... 412
How a z/OS shadow job is bound with a remote job instance .......... 415
Monitoring the shadow job status changes after the bind is established .......... 418
Modifying shadow job instances in the current plan .............. 421
Shadow job status transition during the remote job recovery or rerun .......... 422
Modifying remote engine workstations in the current plan .......... 423
Contacting the remote engine in a failover scenario .......... 423

Chapter 21. Running event-driven workload automation .......... 425
Business scenario .................................................. 425
The event-driven process .......................................... 425
Data set triggering .................................................. 426
Defining event rules ................................................ 427
Producing the configuration files .................................. 427
### Chapter 22. Job tailoring

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deploying configuration files</td>
<td>428</td>
</tr>
<tr>
<td>Effects on special resource availability</td>
<td>429</td>
</tr>
<tr>
<td>HFS or ZFS file triggering</td>
<td>429</td>
</tr>
<tr>
<td>Syntax</td>
<td>429</td>
</tr>
<tr>
<td>Arguments</td>
<td>430</td>
</tr>
<tr>
<td>Example</td>
<td>431</td>
</tr>
<tr>
<td>Adding occurrences by event-triggered tracking</td>
<td>432</td>
</tr>
<tr>
<td>Triggering event types</td>
<td>432</td>
</tr>
<tr>
<td>Activating ETT</td>
<td>433</td>
</tr>
<tr>
<td>Defining your ETT criteria</td>
<td>433</td>
</tr>
<tr>
<td>Automatically adding an occurrence to the current plan</td>
<td>435</td>
</tr>
<tr>
<td>How you can use ETT to automate tasks</td>
<td>437</td>
</tr>
<tr>
<td>Using ETT occurrence-related variables</td>
<td>437</td>
</tr>
<tr>
<td>Combining data set triggering and ETT</td>
<td>440</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 23. Job scheduling and WLM</td>
<td>493</td>
</tr>
<tr>
<td>Integrating with the service class</td>
<td>495</td>
</tr>
<tr>
<td>Environment</td>
<td>495</td>
</tr>
<tr>
<td>When a job should be set as critical</td>
<td>495</td>
</tr>
<tr>
<td>Selecting WLM assistance policies</td>
<td>495</td>
</tr>
<tr>
<td>Integrating with the scheduling environment</td>
<td>497</td>
</tr>
<tr>
<td>Associating a scheduling environment with a job</td>
<td>497</td>
</tr>
<tr>
<td>Submitting prevention and automatic tailoring</td>
<td>498</td>
</tr>
<tr>
<td>Pre-existing definition in the JCL JOB card</td>
<td>498</td>
</tr>
<tr>
<td>Checking for scheduling environment combined with variables</td>
<td>498</td>
</tr>
<tr>
<td>Restrictions for using job card comments on the right</td>
<td>499</td>
</tr>
<tr>
<td>The retry mechanism for operation waiting for SE</td>
<td>499</td>
</tr>
<tr>
<td>Monitoring operations</td>
<td>499</td>
</tr>
<tr>
<td>Considerations for using the ROUTE JCL statement</td>
<td>499</td>
</tr>
<tr>
<td>User actions to activate scheduling environment integration</td>
<td>499</td>
</tr>
<tr>
<td>Supported configuration</td>
<td>500</td>
</tr>
<tr>
<td>Multi-syplex configuration with JESplex</td>
<td>500</td>
</tr>
<tr>
<td>Multi-syplex configuration with JESplex not matching syplex</td>
<td>502</td>
</tr>
<tr>
<td>Performance considerations</td>
<td>507</td>
</tr>
<tr>
<td>The ENF 57 and ENF 41 listener exits</td>
<td>508</td>
</tr>
<tr>
<td>The DPCATCH EQQDPX01 exit</td>
<td>508</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chapter 24. Loop detection and analysis</td>
<td>509</td>
</tr>
<tr>
<td>Detecting a loop condition</td>
<td>509</td>
</tr>
<tr>
<td>&quot;NO ENTRY AND/OR EXIT POINT&quot; loop</td>
<td>510</td>
</tr>
<tr>
<td>&quot;SOME NODES COULD NOT BE CHECKED&quot; loop</td>
<td>510</td>
</tr>
<tr>
<td>Starting the loop analysis</td>
<td>511</td>
</tr>
<tr>
<td>Example of a loop detection and analysis</td>
<td>512</td>
</tr>
<tr>
<td>Suggestions to speed up the loop resolution</td>
<td>515</td>
</tr>
</tbody>
</table>
Chapter 1. What is the scheduler?

This chapter gives a description of IBM Tivoli Workload Scheduler for z/OS and introduces its terminology. Read this chapter if you are new to Tivoli Workload Scheduler for z/OS.

One system in your complex is designated as the controlling system: it runs the controller. From this system, you can automatically plan, control, and monitor your entire production workload. All the systems in your complex must run the tracker. The tracker acts as the communication link between the system that it runs on and the controller.

How the scheduler works

If you do not have an automated planning system such as Tivoli Workload Scheduler for z/OS, you submit jobs on request, or according to run sheets. If they fail, you correct the error and resubmit them, perhaps after running recovery jobs. The jobs depend on many conditions, such as:

- Hardware, such as tape drives.
- Online systems, such as Customer Information Control System (CICS®). Often a system must be shut down before a batch job can run.
- Operating system resources, such as job entry subsystem (JES) initiators, that you need to run a job of the correct class.
- Other jobs. You cannot run the job to print pay slips until the payroll tax deduction program has completed successfully.
- Job parameters that you must change each run.
- The time of day.
- The day of the week or year. Some jobs must be run on Friday. There are sometimes complex rules that specify what you do when the normal day is a holiday.

When you run jobs and started tasks under Tivoli Workload Scheduler for z/OS, these dependencies are defined in Tivoli Workload Scheduler for z/OS databases by someone in your enterprise who is Tivoli Workload Scheduler for z/OS administrator. The administrator defines your workload to Tivoli Workload Scheduler for z/OS like this:

1. Create one or more calendars with the holidays that you take.
2. Define applications, which are sets of jobs and other steps such as job preparation and print processing. Applications can themselves be grouped into application groups.
3. Create a long-term plan (LTP). This lists all occurrences of the applications that will run in a long period of typically a few months and the dependencies among them.
4. Create a current plan (CP). This is a detailed plan, typically for one day, that lists the applications that will run and the operations in each application. An operation can be a computer job, but it can also be any other operation that you want to control with Tivoli Workload Scheduler for z/OS, such as printing and job preparation.
How the scheduler works

You work mostly with the current plan. It is created by a batch job, usually at a fixed time each day. The current plan is really a data set of Tivoli Workload Scheduler for z/OS, which is continually updated by events on the processors, but you can have a printed plan, which is a report that is produced when the current plan is created.

Strictly speaking, the current plan is created only once and the daily planning process is called extending the plan. Extension is a better term than creation, because the old current plan is also part of the new current plan. Look at the long-term plan in Figure 1.

<table>
<thead>
<tr>
<th></th>
<th>2001</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jul</td>
<td>Aug</td>
</tr>
<tr>
<td>Create the long-term plan</td>
<td></td>
</tr>
<tr>
<td>Extend the plan 28 days</td>
<td></td>
</tr>
<tr>
<td>Extend the plan another 28 days</td>
<td></td>
</tr>
</tbody>
</table>

Figure 1. A rolling long-term plan

The current plan should always stretch for some hours or days into the future. Extend the current plan at regular intervals, using the EXTEND option of the DAILY PLANNING menu. You can extend the current plan to a fixed date and time, or you can extend it by a period of hours and minutes.

Figure 2 shows a 48-hour current plan. The initial current plan lasts 48 hours: every morning, the current plan is extended by a further 24 hours.

Figure 2. Extending the current plan
How the scheduler works

Input is taken from both the long-term plan and from the present current plan. The planning performed on Tuesday for the day's work considers the actual situation (both completed and outstanding work) as reflected in the current plan.

The extended current plan always keeps uncompleted application occurrences, but the current plan will usually be about 48 hours long, extended by 24 hours every 24 hours.

How does the Scheduler keep track of jobs?

The scheduler submits jobs to the operating system. The code supplied by the scheduler in the operating system (in the case of a z/OS tracker, this code is in JES and System Management Facilities [SMF] exits) tells Tivoli Workload Scheduler for z/OS when the jobs have finished and also how they have finished (Tivoli Workload Scheduler for z/OS looks at abend and return codes and can also look for certain error messages in the job log that do not always give rise to nonzero return codes).

The scheduler calculates the latest time that a job can be started before it is in danger of missing its deadline. The scheduler continually adjusts its estimate of how long a job takes, taking the actual run times into account.

When a job or other operation is running late, Tivoli Workload Scheduler for z/OS can issue alerts. An alert can be a message to the operator console, but it can also trigger other events.

Can you define a dependency for the job?

When managing the workload in the plan, you can control job processing using dependencies. Dependencies can be of the following types:

Normal

It is a relationship between two jobs stating that one job, named successor, can run only when the other job, named predecessor, is completed. Normal dependencies can be internal or external:

Internal dependency

The relationship links jobs belonging to the same application.

External dependency

The relationship links jobs belonging to different applications.

For detailed information, see “Specifying dependencies” on page 139.

Conditional

It is a relationship between one job, named conditional successor, and one or more jobs or job steps, named conditional predecessors, stating that the conditional successor can run only when a specific combination of conditional predecessors status and return code values occurs.

For detailed information, see Chapter 19, ”Conditioning the processing of operations,” on page 373.

Cross

It is a dependency of a local job from another job running in a different scheduling environment. It indicates that a local job cannot start until the remote job is completed. Cross dependencies helps you to integrate the workload running on different engines. They can be both Tivoli Workload Scheduler for z/OS engines (controller) and Tivoli Workload Scheduler engines (master domain manager).
Can you define a dependency for the job?

For detailed information, see Chapter 20, “Defining and managing cross-depencies,” on page 405.

How does the scheduler help with job preparation?

The scheduler helps in two ways:

- Runtime variables can often be automatically substituted, even if they vary from run to run. Many such variables are related to the date, and Tivoli Workload Scheduler for z/OS can build a string in the format required by a program.
- When jobs need manual job preparation, the administrator specifies a job setup operation as a predecessor for the job operation. The scheduler does not submit the job until you have finished preparing the job statements.

Do not edit and submit the job outside Tivoli Workload Scheduler for z/OS. Instead, use the READY LIST panel to edit the job: Tivoli Workload Scheduler for z/OS submits the job for you when you have prepared the job (and when other dependencies have been met).

How can the scheduler help you recover jobs?

The scheduler supports automatic recovery by having its own job statements that take effect when a job fails: these job statements look like comments to z/OS and JES.

For jobs tracked on a z/OS system, Tivoli Workload Scheduler for z/OS also notices when the catalog has been updated by a job and is able to undo the catalog updates (step by step, if required) to the point before the job ran, for all data sets allocated with job control language (JCL) DD statements. This facility is called restart and cleanup. For example, when a job creates a data set, a rerun often fails because the data set already exists. With cleanup active for the job, Tivoli Workload Scheduler for z/OS uncatalogs and deletes the data set before resubmitting the job.

Can the scheduler help with online systems?

An online system, such as a CICS system, is a job or started task, so it can be started like any other operation defined to Tivoli Workload Scheduler for z/OS.

Many batch applications cannot start until the online system shuts down. One advantage of defining an online system to Tivoli Workload Scheduler for z/OS is that a batch application can be made dependent on the online system and Tivoli Workload Scheduler for z/OS can start the batch application automatically (assuming other dependencies are met) when the online system ends.

The scheduler administrator can specify that Tivoli Workload Scheduler for z/OS issues a message when an operation deadline passes before the operation is complete. This is called a DEADLINE WTO. If a DEADLINE WTO is specified for the operation representing an online system on a z/OS system, Tivoli Workload Scheduler for z/OS sends a write-to-operator (WTO) message to the operator console when the online system must shut down. This message can trigger events in NetView, such as the broadcasting of a “Closing in 5 minutes” message to online users and the initiation of the shutdown transaction 5 minutes later.
Can you run jobs outside the scheduler?

Can you run jobs outside the scheduler?

Jobs fall into four categories:

1. Jobs that are in the current plan and are submitted by Tivoli Workload Scheduler for z/OS.
   These are scheduled jobs, or jobs that you have added to the current plan using the MODIFY CURRENT PLAN (MCP) panel.

2. Jobs that are in the current plan, but are not submitted by Tivoli Workload Scheduler for z/OS.
   These are usually jobs that are generated and submitted by some other subsystem, such as CICS. The scheduler can track these jobs and take account of the resources that they use. It is also possible for other subsystems to submit held jobs and for Tivoli Workload Scheduler for z/OS to release them when all dependencies are met.

3. Jobs that are not submitted by Tivoli Workload Scheduler for z/OS, but trigger events in Tivoli Workload Scheduler for z/OS.
   These jobs are specified with event-triggered tracking (ETT). Refer to “Adding occurrences by event-triggered tracking” on page 432 for more details.

4. Jobs that are completely ignored by Tivoli Workload Scheduler for z/OS controller.

One disadvantage of having jobs outside Tivoli Workload Scheduler for z/OS (categories 3 and 4) is that Tivoli Workload Scheduler for z/OS cannot take account of the resources that they use. The scheduler can schedule and control its jobs to avoid contention for resources (such as tapes, data sets, and JES initiators), but if other jobs use these resources, Tivoli Workload Scheduler for z/OS might submit its jobs when a resource is unavailable.

Finding out about the scheduler

If you are new to Tivoli Workload Scheduler for z/OS, the number of pages in its library can be daunting, but you do not need to read it all. Start with this book, using the index and the table of contents to find help for the task that you have to do.

Do

- Read the reports that are produced when the current plan is extended.
- Use the panels, especially the panels mentioned in this book (READY LIST, QUERY CURRENT PLAN, and MODIFY CURRENT PLAN), to find out more about Tivoli Workload Scheduler for z/OS and your application and workstation definitions.
- Press PF1 for help in the panels when you need additional information.
- Suggest improvements to your administrator if jobs do not run smoothly or if you very often have to make changes through the panels.
- Tell the administrator about any tasks that you often do and you think could be automated by Tivoli Workload Scheduler for z/OS.
- Use Getting Started for a quick introduction to Tivoli Workload Scheduler for z/OS.

Do not

- Run Tivoli Workload Scheduler for z/OS-controlled jobs (or their associated job preparation tasks) outside Tivoli Workload Scheduler for z/OS, except where this is planned.
Finding out about the scheduler

- Attempt to get work going by changing the status of jobs or by using the EXECUTE command. If a job is not being submitted, there must be a reason: use Tivoli Workload Scheduler for z/OS panels to find the reason and resolve the dependency. Changing the status of an operation and using the EXECUTE command are for exceptional occasions.
Chapter 2. A user scenario

This chapter introduces you to Tivoli Workload Scheduler for z/OS by describing the steps that you would take to run a payroll system under the control of Tivoli Workload Scheduler for z/OS.

If you are the scheduling administrator, responsible for automating applications that are run manually, you have to:

1. Design the automation of the work.
   Your work is more than a collection of jobs: you have people who know how to edit JCL and when to run jobs, documentation that describes your jobs and procedures, and people on call who decide what action to take when jobs fail at night.
   Successful design takes into account all procedures, including manual processes. Tivoli Workload Scheduler for z/OS can reduce manual work and can handle the exceptions as well as the routine.

2. Specify your data processing environment to Tivoli Workload Scheduler for z/OS.

3. Create your calendar and any necessary periods. Periods are special units of time such as semesters and tax years.

4. Specify the jobs and started tasks. This is a big step; you must specify:
   a. How jobs are grouped
   b. When jobs will run, using the calendar and periods created earlier
   c. What jobs they depend on
   d. What resources they need, such as data sets, tape drives, and initiators
   e. How the JCL is processed before each job is submitted
   f. What actions Tivoli Workload Scheduler for z/OS must take if each job fails

5. Build the high-level schedule, which is called the long-term plan (LTP). The long-term plan typically lasts a few months, and is extended once a week.

6. Build the low-level schedule, which is called the current plan (CP). The CP typically lasts one day, and is extended a few hours before it expires.

When you finish this chapter, you should understand the main building blocks of Tivoli Workload Scheduler for z/OS and the way they fit together.

You can work through the chapter using your Tivoli Workload Scheduler for z/OS system. By performing the steps in the chapter, you cannot corrupt any programs, though you might want to recreate your databases before you specify your own systems.

Implementing a payroll system

To help you understand Tivoli Workload Scheduler for z/OS and plan the definition of your systems to Tivoli Workload Scheduler for z/OS, this book bases its examples on a fictional corporation, Paymore Incorporated, that has a system running z/OS, and is converting its applications to run under the control of Tivoli Workload Scheduler for z/OS. It first converts the payroll system.
Implementing a payroll system

The payroll analyst describes the process for the Tivoli Workload Scheduler for z/OS scheduler:

1. CICSA runs 24 hours a day, but the payroll transactions are closed before PAYDAILY is started.
2. On each working day, payroll clerks enter information about hours worked, new employees, and so on, into a CICS system, CICSA.
3. Payroll closes the CICSA payroll data set and asks the job preparation team to schedule the daily PAYDAILY job.
4. The weekly job is PAYWEEK, which runs every Thursday or on the closest work day before Thursday if Thursday is a holiday. The weekly job must run after the daily PAYDAILY job.
   PAYWEEK calculates deductions, such as tax and insurance, and updates the bank-transfer data set. This is for people who choose to be paid by monthly bank transfer into their bank accounts. This job must run before the monthly bank-transfer job PAYTRANS, which runs on the third Thursday in the month.
5. When PAYWEEK completes successfully, the job PAYWSLIP runs to print pay slips. It includes these programs:

`PAYBACKP runs after all the day’s payroll jobs have finished`

`PAYQUERY is run on demand`

Figure 3. Example of payroll jobs at paymore incorporated

The payroll analyst describes the process for the Tivoli Workload Scheduler for z/OS scheduler:
Implementing a payroll system

a. PAY14 writes pay slips to a data set, and creates a management report.
b. PAY15 prints the pay slips.

6. After the weekly payroll jobs:
a. Job Preparation decollates the pay slips.
b. Payroll checks the pay slips.
c. The cash office takes the pay slips and prepares pay packets. This depends on the delivery of cash by security van.

7. Monthly jobs, including PAYTRANS, have similar gross-to-net and print programs for monthly-paid employees. These jobs run on the third Thursday of each month, after the weekly payroll run. But if Thursday is a holiday, they run the day before.

8. After the payroll run for the day, the job preparation team runs the backup job PAYBACKP and reopens the CICSA data set.

9. If updates fail, the PAYRECOV job is run before the update job is rerun.

This is a short description of the flow of jobs. When you design the automation of a system such as this, include manual work in the analysis, and the recovery procedure for each job. The operator might have recovery instructions for PAYDAILY like “Page the on-call payroll analyst.” The reason for such instructions is that the recovery process is too complex to automate using standard JCL facilities. When you install Tivoli Workload Scheduler for z/OS, there is a good chance that you can automate recovery, so consider all procedures, even manual ones. One of the benefits of implementing Tivoli Workload Scheduler for z/OS is that analysts have to document procedures, and there is less risk that the vital person is on holiday, or left the corporation six months earlier, when something goes wrong.

Designing the payroll applications

Table 1 on page 16 shows how the payroll jobs can be grouped to run under Tivoli Workload Scheduler for z/OS. Here are other problems to solve:

How do you get PAYDAILY to run after the CICSA data set is closed?

Paymore runs CICS 24 hours a day. It shuts down only for essential maintenance. So PAYDAILY cannot be made dependent on the CICSA started task. Instead, it must be triggered by the CICS transaction that closes the payroll data set in CICSA. There are many ways to do this. Here are some examples:

1. Give PAYDAILY exclusive access to a special resource representing the payroll data set. Include code supplied by Tivoli Workload Scheduler for z/OS in your IEFU83 SMF exit to generate a special resource event when the data set is closed, allowing PAYDAILY to start. Refer to Customization and Tuning for information on data set triggering.

2. Give PAYDAILY exclusive access to a special resource as before. The CICS transaction can call the EQQUSIN subroutine, which can generate a special resource event. Refer to Customization and Tuning for information on the EQQUSIN subroutine. This is the solution adopted in this example.

3. Trigger the CICS transaction with a WTO operation, but use an automatic reporting WTO general workstation (see “Specifying workstation reporting attributes” on page 65) so that the WTO does not complete automatically. The CICS transaction sets the WTO operation to status C (Complete) with the EQQUSIN subroutine. This solution is not used, because the special resource is needed for other reasons.
Designing the payroll applications

How do you automate the running of the CICS transaction?
Use Tivoli Workload Scheduler for z/OS to schedule PAYDAILY at a fixed
time each day. The first operation in the PAYDAILY application is a
write-to-operator (WTO) operation. NetView can intercept the operation
and issue the CICS transaction that closes the data set.

How do you force the weekly and monthly jobs to run after the daily job?
Make PAYWEEK and PAYMONTH dependent on PAYDAILY.

How do you make the backup run after all the payroll jobs?
Make PAYBACKP dependent on all the payroll jobs. On days where no
weekly or monthly job is scheduled, that dependency does not take effect,
so PAYBACKP will run after PAYDAILY.

How do you automate the reopening of the CICSA data set?
Include a final WTO operation in PAYBACKP that triggers, via NetView, a
CICS transaction that reopens the payroll data set in CICS.

How do you handle job recovery?
Consider the different error conditions that must be handled, such as:

- B37 abends arising from a lack of space.
- Invalid data; for example, hours-worked data for an employee who has
left the company. There can be no automatic recovery for this error, but
Paymore avoids the most common errors by validating each transaction
against the database when the payroll clerks enter them into the CICS
system.

You can include operator instructions for each operation in the Tivoli
Workload Scheduler for z/OS OI database in the event that operators have
to take manual action.

How do you stop PAYQUERY running alongside a scheduled job?
PAYQUERY is run on request. In Tivoli Workload Scheduler for z/OS
terms, this means that an occurrence of the PAYQUERY application is
added to the schedule, or current plan, by an operator using the MODIFY
CURRENT PLAN panel. To stop PAYQUERY accidentally running beside a
payroll job, create a resource that represents the payroll database. All
payroll jobs that update the database are given exclusive control of the
resource. If the resource is unavailable, Tivoli Workload Scheduler for
z/OS waits until the allocating operation has freed it.

Table 1. Grouping of payroll applications

<table>
<thead>
<tr>
<th>Groups</th>
<th>Applications</th>
<th>Operation names</th>
<th>Workstations</th>
<th>Operation numbers</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CICSA</td>
<td>PAYDAILY</td>
<td>WTO1 SETP</td>
<td>005 010 020</td>
<td>PAY04</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYDAILY</td>
<td>CPU1</td>
<td></td>
<td>PAY06</td>
</tr>
<tr>
<td>GPAYW</td>
<td>PAYW</td>
<td>PAYWEEK</td>
<td>CPU1</td>
<td>020 030</td>
<td>PAY07</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYWSLIP</td>
<td>PRT1 PAY1</td>
<td>090 095</td>
<td>PAY10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYWSLIP</td>
<td></td>
<td></td>
<td>PAY14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYWSLIP</td>
<td></td>
<td></td>
<td>PAY15</td>
</tr>
<tr>
<td>GPAYM</td>
<td>PAYM1</td>
<td>PAYMONTH</td>
<td>CPU1</td>
<td>040 050</td>
<td>PAYM07</td>
</tr>
<tr>
<td></td>
<td>PAYM2</td>
<td>PAYM1SLIP</td>
<td>PRT1 CPU1</td>
<td>099 040</td>
<td>PAYM10</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYM1SLIP</td>
<td></td>
<td></td>
<td>PAYM16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYM1SLIP</td>
<td></td>
<td></td>
<td>PAYM14</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PAYTRANS</td>
<td></td>
<td></td>
<td>PAYM15</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PAYGIRO</td>
</tr>
</tbody>
</table>
### Table 1. Grouping of payroll applications (continued)

<table>
<thead>
<tr>
<th>Groups</th>
<th>Applications</th>
<th>Operation names</th>
<th>Workstations</th>
<th>Operation numbers</th>
<th>Programs</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYTAXYR</td>
<td>PAYTAXYR</td>
<td></td>
<td>CPU1</td>
<td>015</td>
<td>PAYY10</td>
</tr>
<tr>
<td>PAYQUERY</td>
<td>PAYQUERY</td>
<td></td>
<td>CPU1</td>
<td>050</td>
<td>PAYQ1</td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>PAYBACKP</td>
<td></td>
<td>CPU1</td>
<td>015 030</td>
<td>IDCAMS</td>
</tr>
<tr>
<td>PAYRECOV</td>
<td>PAYRECOV</td>
<td></td>
<td>CPU1</td>
<td>015</td>
<td>IDCAMS</td>
</tr>
</tbody>
</table>

### Creating workstations

Specify your environment to Tivoli Workload Scheduler for z/OS in terms of workstations and resources. A workstation is not necessarily hardware. It is a stage in the processing that is controlled by Tivoli Workload Scheduler for z/OS. The processor complex that runs your controller is a computer workstation, and you will probably create two workstations for it, because there must be separate workstations for jobs and for started tasks.

The job preparation team submits jobs, often adding runtime parameters on request. Often a runtime parameter is just the day’s date. The program cannot simply get the date from the time-of-day clock because sometimes the date must be when the group of jobs starts, which must not change, even if the jobs continue to run past midnight. At other times, the date appears on the pay slips, and must be Friday’s date, regardless of the date when the jobs run. In these cases, Tivoli Workload Scheduler for z/OS is usually able to automatically substitute the correct date in the jobs. Where manual intervention is unavoidable, Tivoli Workload Scheduler for z/OS can control when the job is available for variable substitution and submission. Tivoli Workload Scheduler for z/OS can also keep track of what stage the job is at. You specify manual job preparation work as a workstation.

Another type of workstation is a printer workstation. Tivoli Workload Scheduler for z/OS is notified when a print output group has finished printing. This way Tivoli Workload Scheduler for z/OS can keep track of the status of important print operations and the time they take. You do not have to specify a print operation for every job that prints something. Only specify a print operation when you want to track it.

If there is a manual process that you want to track, specify it as a general workstation. Such workstations do not, of course, have JES and SMF exits to tell Tivoli Workload Scheduler for z/OS what is going on, but you can have a terminal where the operator changes the Tivoli Workload Scheduler for z/OS status of an operation through the ISPF panel. If the status of an operation can be detected by VTAM®, NetView, or by one of your own programs, you can get the program to inform Tivoli Workload Scheduler for z/OS of the new status, and Tivoli Workload Scheduler for z/OS then starts the dependent operations.

Returning to Paymore, how many workstations do you create, and how do you create them? You need a:

- Computer workstation for jobs
- Computer workstation for started tasks, if CICSA is a started task and you want it to run under Tivoli Workload Scheduler for z/OS
- General workstation for job preparation
- Printer workstation
Creating workstations

- General workstation for the payroll office, if you need to keep track of manual operations there, such as the collation of pay slips and preparation of pay packets.

See Chapter 4, “Creating workstations,” on page 53 for a full description of workstations. This section shows how you might create them for Paymore.

Select option 2 from the MAINTAINING WORKSTATION DESCRIPTIONS menu, (or option 1.1.2 from the main menu) to list workstations. The command CREATE displays the panel shown in Figure 4.

![Figure 4](image)

**Figure 4. EQQWCGEP - Creating general information about a workstation**

Figure 4 is one of several ISPF panels that you need to create a workstation. Use the A command to specify when the workstation is available.

Use the R command to specify workstation fixed resources, called R1 and R2. Although these fixed resources are less useful, because you can do more with special resources (see Chapter 5, “Creating special resources,” on page 81). So the Paymore application does not use any workstation fixed resources.

Table 1 on page 16 shows values that you might specify for the workstations. If you have a general workstation to track manual operations in the payroll office, PAY1, create it like the job setup workstation SETP, but with JOB SETUP = N.

**Controlling general workstations**

You control general workstations such as SETP and PAY1 using the READY LIST panel, which you access using option 4 (WORK STATIONS) from the main menu.

Using the READY LIST panel, the job preparation team can see what operations are waiting for setup, initiate setup by setting next status, and complete the setup operation when they finish editing the JCL. The process is similar for other manual operations.
Controlling general workstations

A general workstation is not a location or a device, but rather a logical stage in processing that you normally control with an ISPF session (although you could use a command or program interface to set the status of general workstations).

Using servers

For each workstation, you specify parallel servers on the availability panels. This is another resource, usually representing (for z/OS computer workstations) JES initiators. If CPU1 has 15 servers, Tivoli Workload Scheduler for z/OS counts how many are being used, and does not start an operation (which uses by default one server) if all 15 servers are in use. This is a convenient way of controlling JES initiators on computer workstations.

For other types of workstation, you normally specify usage P (planning only) for parallel servers, because you do not want Tivoli Workload Scheduler for z/OS to refuse to start a setup operation, for example, if an operator requests it. If an operator wants to set up a job, the server (operator) must be available! Tivoli Workload Scheduler for z/OS still uses the count of servers on setup workstations, but only for planning, when it works out in advance whether there could be a backlog of setup operations.

If you do not need to use servers, set the number to 99, as for WTO workstations in this example, and specify N (neither) for the server usage.

Table 2. Creating workstations for paymore

<table>
<thead>
<tr>
<th>Field name</th>
<th>Job setup workstation</th>
<th>Computer workstation</th>
<th>Printer workstation</th>
<th>WTO workstation</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK STATION NAME</td>
<td>SETP</td>
<td>CPU1</td>
<td>PRT1</td>
<td>WTO1</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Used to prepare JCL</td>
<td>Main JES processor</td>
<td>Printer pool</td>
<td>Messages for NetView</td>
</tr>
<tr>
<td>WORK STATION TYPE</td>
<td>G (general)</td>
<td>C (computer)</td>
<td>P (printer)</td>
<td>G (general)</td>
</tr>
<tr>
<td>REPORTING ATTR</td>
<td>S (manual start and completion)</td>
<td>A (automatic)</td>
<td>A (automatic)</td>
<td>C (completion only)</td>
</tr>
<tr>
<td>FT WORK STATION</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
</tr>
<tr>
<td>PRINTOUT ROUTING</td>
<td>SYSPRINT</td>
<td>SYSPRINT</td>
<td>SYSPRINT</td>
<td>SYSPRINT</td>
</tr>
<tr>
<td>SERVER USAGE</td>
<td>P (planning only)</td>
<td>B (both planning and control)</td>
<td>P (planning only)</td>
<td>N (neither)</td>
</tr>
<tr>
<td>SPLITTABLE</td>
<td>Y (yes)</td>
<td>N (no)</td>
<td>Y (yes)</td>
<td>N (no)</td>
</tr>
<tr>
<td>JOB SETUP</td>
<td>Y (yes)</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
</tr>
<tr>
<td>STARTED TASK, STC</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
</tr>
<tr>
<td>WTO</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
<td>Y (yes)</td>
</tr>
<tr>
<td>WAIT</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
<td>N (no)</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>Blank (Tivoli Workload Scheduler for z/OS controlling processor)</td>
<td>Blank (Tivoli Workload Scheduler for z/OS controlling processor)</td>
<td>Blank (Tivoli Workload Scheduler for z/OS controlling processor)</td>
<td>Blank (Tivoli Workload Scheduler for z/OS controlling processor)</td>
</tr>
</tbody>
</table>

These fields are in the CREATING GENERAL INFORMATION ABOUT A WORKSTATION panel (Figure 4 on page 18).
### Creating special resources

The payroll system has several jobs that update the payroll database, and these cannot run together. If you submit them together, z/OS forces one job to wait because of the DISP=OLD disposition for the database (or similar mechanism for VSAM). Letting jobs compete and be locked out in this way ties up z/OS resources, such as initiators, and can result in deadlock.

One way of forcing serialization is to make one job a predecessor of the other, but often it is unimportant which job runs first: the important thing is that they do not run together.

The best way of doing this is to create a special resource, which in this case is the database. See Chapter 5, “Creating special resources,” on page 81 for a full description of special resources. Create a resource to control the payroll database by following these steps:

1. Select option 6 from the MAINTAINING TWSz DATABASES menu (Figure 45 on page 89).
2. Select option 3 (LIST) on the MAINTAINING SPECIAL RESOURCES menu (Figure 46 on page 89). You can select option 2 (CREATE) instead, but option 3 gives you a chance to see the resources already created.

   When you select 3 (LIST), the SPECIFYING SPECIAL RESOURCE LIST CRITERIA panel is displayed, enabling you to filter the resources shown in the list. Enter an * (asterisk) in the SPECIAL RESOURCE and SPECRES GROUP ID fields to list all resources.

3. On the LIST OF SPECIAL RESOURCES panel, enter the CREATE command. The CREATING A SPECIAL RESOURCE panel is shown (Figure 5 on page 21).
Creating special resources

4. Type the values shown. Note especially these values:

   **SPECIAL RESOURCE**
   This name must be exactly the same in all the application descriptions that use the database.

   **USED FOR**
   Type B, because you want Tivoli Workload Scheduler for z/OS to take account of resource availability when creating plans (planning), and to check resource status before starting an operation (control).

   **ON ERROR**
   Type K, because the jobs are to keep the resource allocation even if they fail. This is so that there is no risk of another job taking the database before the operator (or automatic recovery) handles the error.

   **QUANTITY and AVAILABLE**
   This is the default for all time intervals. For this resource, you do not need to specify intervals, so these values always apply, unless altered dynamically; for example, by the EQQUSIN subroutine or the SRSTAT TSO command.

5. Press PF3 (End) to save the resource definition.

For some resources, such as those representing tapes or communication lines, you would normally specify intervals, and, for each interval, the quantity, availability, and connected workstations.

For PAYROLL.DATABASE, the defaults are for all times, and the resource is accessible from all workstations.

**Creating the default calendar**

Paymore does not run its jobs at weekends or on national holidays. Specify the national holidays by creating the default calendar, which is a calendar with an ID of DEFAULT.
Creating the default calendar

Create the default calendar like this:

1. Select option 1.2.2 from the main menu.
2. On the MODIFYING CALENDARS panel, enter the CREATE command. The CREATING A CALENDAR panel is shown, (Figure 6).

| CALENDAR ID | => DEFAULT______ |
| DESCRIPTION | => default calendar____ |
| WORK DAY END TIME | => 06.00 |

Row | Weekday or Comments | Status |
--- | ------------------- | ------ |
'' M | MONDAY_______________ | W |
'' T | TUESDAY_____________ | W |
'' W | WEDNESDAY____________ | W |
'' T | THURSDAY_____________ | W |
'' F | FRIDAY_____________ | W |
'' S | SATURDAY____________ | F |
'' S | SUNDAY_____________ | F |
'' 03/12/25_ Christmas Day | F |
'' 03/03/28_ Good Friday | F |

3. Enter W for the status of the days of the week that are work days, and F for the weekends and holidays. When work would otherwise be planned on a free (F) day, Tivoli Workload Scheduler for z/OS plans the work according to the free-day rule on each application run cycle. See "Selecting a free-day rule" on page 130 for detailed information about the free-day rule.

The status specified for a particular date overrides the specification for the corresponding day of the week.

4. Specify the workday end time, which is the time that Tivoli Workload Scheduler for z/OS considers the day to end for planning purposes. Paymore's night shift, for example, goes home at 06.00, and work is run until 06.00 on holidays.

Remember to maintain your calendar once a year. Make full use of the calendar by using Tivoli Workload Scheduler for z/OS to schedule other things than batch jobs and started tasks. Why not use it to initiate events in NetView?

Preparing the payroll jobs

Consider these problems:

1. The PAYWEEK job can process data from many departments. Each department produces its own transaction data set, which must be concatenated with the head office data set, which is always present.

2. The PAYWEEK job uses the date of the Friday of the week in which the job is run. The job is usually run on Thursday, but if Thursday is a holiday, the job runs on the previous work day.

Both problems can be solved using variable substitution. Follow these steps:

1. Use the JCL VARIABLE TABLE panel (1.9.2 from the main menu) to create a variable table called PAY.
Preparing the payroll jobs

2. Specify a variable DEPT, as shown in Figure 7.

This is an explanation of the marked lines:

1. is a statement that tells Tivoli Workload Scheduler for z/OS to perform variable substitution on the following lines. You need this unless you have VARSUB set to YES on the OPCOPTS initialization statement.

2. tells Tivoli Workload Scheduler for z/OS to use the format MM/DD/CCYY for OCDATE. If the input arrival date is 16 March 2003, for example, Tivoli Workload Scheduler for z/OS substitutes 03/16/2003 for &OCDATE.

3. tests the day of the week. If the input arrival day is Monday (ODAY=1), it includes a SETVAR statement to add 4 calendar days, giving Friday’s date. This is repeated for the other days of the week. OCDATE and ODAY do not have to be in your variable table because they are predefined by Tivoli Workload Scheduler for z/OS.

4. passes Friday’s date to the PAY07 program.
Preparing the payroll jobs

6. tests whether the variable DEPT has its default value N, and if not, it adds the extra line of JCL, which is the extra data set for another department.

8. marks the end of the lines to be included.

4. Put the JCL into the partitioned data set that is allocated to the ddname EQQJBLIB. Tivoli Workload Scheduler for z/OS never updates this JCL. It always makes a copy and stores the modified copy in the job repository (a group of VSAM clusters used cyclically with DD names in the format EQQJSnDS). Tivoli Workload Scheduler for z/OS takes a fresh copy of the JCL from EQQJBLIB for each occurrence, but once the JCL is changed, the changed copy in the job repository is used.

Tivoli Workload Scheduler for z/OS changes JCL:

- When you set up the JCL for an operation. You use the READY LIST to perform setup and complete the operation on the job setup workstation.

- On request from the MODIFY CURRENT PLAN (MCP) panel. This is independent of the job setup operation. If you change the JCL for an occurrence, Tivoli Workload Scheduler for z/OS puts your edited job in the repository, where it will be picked up by any subsequent setup operation using the Ready list.

- On request from the LONG-TERM PLAN panel. This is how you change the JCL for an individual occurrence that is not yet in the current plan, without affecting the JCL for other occurrences of the job.

- When a job or started task completes successfully. Tivoli Workload Scheduler for z/OS removes the JCL for the previous occurrence from the job repository.

- When you specify automatic recovery. Tivoli Workload Scheduler for z/OS checks the JCL for Tivoli Workload Scheduler for z/OS recovery statements. If there are recovery statements, Tivoli Workload Scheduler for z/OS changes the JCL and stores it in the job repository.

See Chapter 22, “Job tailoring,” on page 441 for more details of variable substitution.

Note: PAYWEEK is a z/OS job, but you can use variable substitution for jobs that run on other operating systems. The syntax of the directives is the same. Job setup and variable substitution is always performed on the z/OS system that runs the controller, and the prepared job is then passed to the system where it will run.

Creating the groups, applications, and operations

There are two ways to create applications: using the ISPF panels and using the batch loader program. See Chapter 8, “Defining applications in batch,” on page 185 for a description of the batch loader program and a list of the control statements necessary to specify the Paymore system using the batch loader.

This section shows you the steps involved in creating the Paymore applications using the Tivoli Workload Scheduler for z/OS panels. You can create all applications and groups by using the APPLICATION DESCRIPTION panel, but simple applications, having only one computer operation and optionally a job setup operation and another manual operation, can be created using the JOB DESCRIPTION panel.

First, create the PAYDAILY job, which is part of an application description of the same name, with three operations, as shown in Table 1 on page 16. Follow these steps:
Creating the groups, applications, and operations

1. Select option 1.4 from the main menu to display the APPLICATION DESCRIPTION panel, shown in Figure 58 on page 120.

2. Select option 2 to display the CREATING AN APPLICATION panel, shown in Figure 8.

3. Type the fields as shown and press Enter. See “Standard applications and group definitions” on page 119 for more information about each field.

4. To schedule PAYDAILY, enter the RUN command to specify a run cycle. The RUN CYCLES panel, shown in Figure 9, is displayed.

5. Specify an input arrival time of 12.00. This time has two main purposes:
   a. It identifies the occurrence of the application, differentiating “the midday run of PAYDAILY” from other runs on the same day.
   b. It tells Tivoli Workload Scheduler for z/OS when to try to start the application, if it is time-dependent. Normally, you need not make applications time-dependent, because they run immediately after their
Creating the groups, applications, and operations

predecessors. But PAYDAILY is an exception, because it is the first payroll job of the day, and is also responsible for triggering the closure of the CICSA payroll data set at midday.

6. Specify the **deadline day and time**, which is the latest time that all operations in any occurrence of the application should be completed by. Tivoli Workload Scheduler for z/OS takes various actions if an operation is not started by its deadline time, depending on parameters that you specify. Day 0 and time 16.00 means that the deadline is 16.00 on the same day as the input arrival time.

7. Type **R** for a normal rule.

8. The value of 4 for F day rule (the free-day rule) means that the application is not scheduled on free days in the calendar. This is not so important for PAYDAILY, because avoiding free days is part of the rule definition, but for other rules, such as Last Friday in the Month, it is important because Tivoli Workload Scheduler for z/OS needs to know what to do if Friday is Christmas Day, for example.

9. Specify the in-effect and out-of-effect dates. If you leave these blank, Tivoli Workload Scheduler for z/OS fills them in with today’s date and 71/12/31, 31 December 2071.

10. Specify the JCL variable table that will be used on the days selected by this run cycle. The PAYDAILY JCL can have variables, promptable or nonpromptable. Tivoli Workload Scheduler for z/OS looks for values to substitute into the JCL in a **variable table**.

11. Enter the **S** row command to specify the days that this rule will select. The MODIFYING A RULE panel, shown in [Figure 10](#) is displayed.

12. Select the fields shown so that Tivoli Workload Scheduler for z/OS schedules PAYDAILY every work day, and then press PF3 (End) to return to the RUN CYCLES panel.

13. Press PF3 (End) again to return to the CREATING AN APPLICATION panel.
Creating the groups, applications, and operations

14. Enter the OPER command. The OPERATIONS panel, shown in Figure 11 on page 27 is displayed.

### Figure 11. Creating operations in PAYDAILY

The first operation, the WTO, causes message EQQW775I to be issued at midday with the operation text (PAYX CLOSE data set) as part of the message, which NetView can then use as a CICS transaction.

15. For each operation, type the workstation (Oper ws), operation number (Oper no.), and, if the workstation is a job setup, computer, or printer workstation, the name of the job or started task that the operation represents.

16. Specify the estimated duration of each operation, which Tivoli Workload Scheduler for z/OS uses for planning purposes and also to judge whether an operation might finish late. Tivoli Workload Scheduler for z/OS can use the actual run times of PAYDAILY to adjust this estimate.

17. Specify the job name. For batch and setup operations, Tivoli Workload Scheduler for z/OS uses this to find the JCL in the EQQJBLIB partitioned data set. A setup operation must have the same job name as the successor computer operation. A print operation must have the same job name as its predecessor, because this identifies the spool file that Tivoli Workload Scheduler for z/OS tracks.

18. Enter the PRED command to specify the internal predecessors (the dependencies within this application). The panel shown in Figure 12 on page 28 is displayed.
Creating the groups, applications, and operations

19. Specify the predecessors shown to tell Tivoli Workload Scheduler for z/OS that the setup operation and the WTO must come before the batch PAYDAILY job.

   The batch job, operation 020, also depends on CICSA successfully closing the payroll data set after getting the command from NetView. This dependency, however, is handled using resources. While CICSA has the data set open, it has exclusive use of the resource PAYROLL.DATABASE. When the PAYX transaction successfully closes the data set, it executes the EQUSIN subroutine to release the special resource. Then the PAYDAILY job can run as soon as the job preparation team has finished any manual JCL overrides and has completed the SETP operation.

20. Specify the details of each operation by entering s beside the operation. The OPERATION DETAILS panel is displayed.

21. Specify option 3 (SPECIAL RES) to specify that the batch job must have exclusive use of the PAYROLL.DATABASE resource (see Chapter 5, “Creating special resources,” on page 81 for details about using resources). Fill in the values shown in Figure 13 and press PF3 (End).

22. You never have to select option 2 (WS RES and SERV) for Paymore operations, because they do not use workstation fixed resources, and the default number of parallel servers is one, which is what you need.
Creating the groups, applications, and operations

23. You do not need to specify a time for the operation (option 6), because the default is for the operation to take the input arrival time specified for the occurrence, which is what is needed for PAYDAILY. But you must specify that the WTO operation is time-dependent. From the OPERATION DETAILS menu, select option 4 (AUTOMATIC OPTIONS). The JOB, WTO, AND PRINT OPTIONS panel, shown in Figure 14, is displayed.

Specify y in the TIME DEPENDENT field. Now Tivoli Workload Scheduler for z/OS will not try to start the WTO operation until 12.00.

24. Return to the CREATING AN APPLICATION panel by pressing PF3 (End), and press PF3 (End) twice again to save the new application description in the Tivoli Workload Scheduler for z/OS database. It will now be included in the long-term plan when you create it.

You have now created one application and its operations. Continue in a similar manner for the other applications in Table 1 on page 16, but note especially:

**GPAYW and GPAYM**
These are groups, so you specify run cycles for these, but no operations. Their applications (PAYW, PAYM1, and PAYM2) have operations, but no run cycles.

**Dependencies for PAYBACKP**
PAYBACKP must run after all scheduled payroll updates, so give it external dependencies to all of them. PAYBACKP will therefore depend on PAYTAXYR, for example, but the dependency will not take effect on the 364 days of the year when PAYTAXYR is not scheduled.

**Opening the CICSA payroll data set**
The last operation of PAYBACKP is a WTO that triggers a CICS transaction to reopen the data set.

**Job setup for other operations**
There are no job setup operations other than for PAYDAILY. Normally, you
Creating the groups, applications, and operations

do not need job setup, because most JCL variable substitution can be handled automatically with JCL variable tables, and in these cases Tivoli Workload Scheduler for z/OS edits the JCL and submits it when the job becomes ready.

Tracking print operations
Print operations are specified only for the payslip printing jobs, where it is important for Tivoli Workload Scheduler for z/OS to know if and when JES has finished printing. For these jobs, PAYWSLIP and PAYMSLIP, take care to specify the print form number and SYSOUT class when you specify the operation details. Tivoli Workload Scheduler for z/OS uses a key of job name, form number, and SYSOUT class to track the event, and this key must be unique, or Tivoli Workload Scheduler for z/OS might track the wrong spool data set. See “Options that apply to print operations” on page 154.

Rule for the GPAYW run cycle
EVERY THURSDAY of every YEAR. You can find equivalent rules, such as:
- ONLY THURSDAY in every WEEK
- EVERY THURSDAY in every WEEK
- EVERY THURSDAY in every MONTH

Which one you specify does not matter. If you are unsure about the rule, use the GENDAYS command to check the selected days. Note that EVERY FOURTH DAY in every WEEK is not equivalent, because the day selected depends on whether free days are to be counted among the four days (the free day rule).

Rule for the GPAYM run cycle
ONLY THIRD THURSDAY of every MONTH. For this run cycle, as for GPAYW and PAYTAXYR, you need free-day rule 1, so that the occurrence is scheduled on the previous work day if the Thursday is a holiday.

Rule for the PAYTAXYR run cycle
ONLY THIRD THURSDAY of every JULY.

Rules for the on-demand jobs
Do not specify rules for PAYRECOV, CICSA, and PAYQUERY. You can add these to the long-term or current plan when you need them.

Creating plans

When you have specified your system environment (workstation, calendars, and periods) and your applications, you are ready to create the plans.

Follow these steps to create the plans:
1. Select option 2 (LTP) from the main menu. The MAINTAINING THE LONG-TERM PLAN menu is displayed.
2. Select option 2 (BATCH). The SELECTING THE LONG-TERM PLAN BATCH JOB menu is displayed.
3. Select option 7 (CREATE). The CREATING THE LONG-TERM PLAN panel, shown in Figure 15 on page 31 is displayed.
4. Start the long-term plan a few days later than the current date, so that you have time to check it before it goes into effect.

5. Choose an end date about one month from the start date, and press Enter. The GENERATING JCL FOR A BATCH JOB panel is displayed.

6. Type the batch job parameters, and press Enter.

7. When the batch job finishes, check it for errors. If the long-term plan is created, you can scan the scheduled occurrences most easily online, using option 1 (ONLINE) from the LTP menu. Figure 16 shows the LONG-TERM PLAN OCCURRENCES panel.

8. If things are not what you expect, you can change occurrences using this panel, but it is easier, while you have no current plan, to correct the database and recreate the long-term plan. You cannot recreate the long-term plan once you have a current plan; you have to delete the current plan first with the REFRESH function.

9. If the long-term plan looks good, put the jobs that you need into the EQQJBLIB data set. The member name must be the same as the operation name.

10. Create the current plan. Select option 3 (DAILY PLANNING) from the main menu. The PRODUCING OPC DAILY PLANS menu is displayed.

11. Select option 2 (EXTEND). The EXTENDING CURRENT PLAN PERIOD panel, shown in Figure 17 on page 32, is displayed.
12. Type the values shown, and press Enter.
13. Enter the batch parameters as before to submit the job, and check the output for error messages.

What happens if jobs or started tasks fail?

If you are automating a system, remember the 1% or 0.1% of cases when something goes wrong. Nobody likes a telephone call at 3 a.m., even if there is a terminal next to the bed. Ask your on-call experts what they do when they have to look at failed jobs. Then get Tivoli Workload Scheduler for z/OS to do it for them.

Here are the payroll analyst's notes on PAYDAILY:

1. PAY04 transfers the hours-worked data from the CICSA key-sequenced data set to a sequential data set. If this program fails, the job can be rerun.
2. PAY06 validates the hours-worked data and updates the payroll database if the data is valid. If there are validation errors (return code 4), payroll inspects and corrects the data, and the job is restarted from PAY04. If there are other errors, PAY04 can be rerun after running PAYRECOV.

This is how to change the PAYDAILY JCL so that Tivoli Workload Scheduler for z/OS handles it:
What happens if jobs or started tasks fail?

```plaintext
//PAYDAILY JOB (890122,NOBO), 'SAMPLE'
//*
//* PAYMORE PAYROLL SAMPLE
//* THIS JOB RUNS PAY04 AND PAY06
//** OPC RECOVER ERRSTEP=PAY04
//** OPC RECOVER ERRSTEP=PAY06,STEPCODE=4,RESTART=N
//** OPC RECOVER ERRSTEP=PAY06,ADDAPPL=PAYRECOV
//PAY04 EXEC PGM=PAY04
//STEPLIB DD DSN=XRAYNER.OPC.LOADLIB,DISP=SHR
//PAYIN DD DSN=XRAYNER.CICS.PAYDB,DISP=SHR
//PAYOUT DD DSN=XRAYNER.DAY.TRANS,DISP=SHR
//SYSPRINT DD SYSOUT=* 
//PAY06 EXEC PGM=PAY06,COND=(0,LT)
//STEPLIB DD DSN=XRAYNER.OPC.LOADLIB,DISP=SHR
//PAYIN DD DSN=XRAYNER.DAY.TRANS,DISP=SHR
//PAYOUT DD DSN=XRAYNER.PAYROLL.DATABASE,DISP=SHR
//SYSPRINT DD SYSOUT=* 
```

This is an explanation of the marked lines:

1. tells Tivoli Workload Scheduler for z/OS what to do if PAY04 fails for whatever reason. The default, with no action on the RECOVER statement, is that Tivoli Workload Scheduler for z/OS reruns the job. But before rerunning the job, it turns this RECOVER statement into an Tivoli Workload Scheduler for z/OS comment and saves the JCL in the job repository, so that PAY04 is not continually rerun in a loop if it keeps failing.

2. tells Tivoli Workload Scheduler for z/OS what to do if step PAY06 fails with return code 4. RESTART = N means leave the operation on the error list for the analysts to look at.

3. is the action for any other failure in PAY06. Tivoli Workload Scheduler for z/OS will add the PAYRECOV application to the plan, run it, and make the rerun of PAYDAILY dependent on PAYRECOV (PAYRECOV becomes the predecessor of PAYDAILY).

See Chapter 18, “Automatic recovery of jobs and started tasks,” on page 347 for a full description of automatic recovery.

Coping with more complex problems

Sometimes,

- A job creates a data set, and the NEW disposition gives an error on a rerun. Tivoli Workload Scheduler for z/OS handles problems like this with cleanup, (see Chapter 17, “Restart and cleanup,” on page 313). It can delete data sets that have been created by failing jobs.

- You have to look at the SYSOUT data set of a job before you know how to recover it. For example, PAY06 might issue a message just when it starts to update the database. If that message is there, you know you have to run PAYRECOV. Use the job completion checker, described in Customization and Tuning, to scan SYSOUT for messages and set an error code that you can pass to automatic recovery or to the operator.
Summary of what you need to do

If you are new to Tivoli Workload Scheduler for z/OS, implement it in stages. Specify just one of your applications, and wait for this to run smoothly before implementing the others. Leave the optional functions, such as restart and cleanup and JCL variable substitution, until later.

These are the implementation tasks:
• Design the automation.
• Create the workstations.
• Create the calendar.
• Create special resources.
• Create JCL variable tables and prepare JCL.
• Create groups, application and job descriptions.
• Create a long-term plan.
• Create a current plan.

These are the daily tasks:
• Control Tivoli Workload Scheduler for z/OS.
• Extend the current plan.

This is the weekly or monthly task:
• Extend the long-term plan.

This is the yearly task:
• Maintain the calendar and period definitions.

The following chapters in this book tell you in detail how to describe your data processing activities and how to build a schedule.
Chapter 3. Using the scheduler panels

To perform most operator tasks, you use Tivoli Workload Scheduler for z/OS panels, which run under Interactive System Productivity Facility (ISPF).

This chapter shows you how to get help, how to tailor the program function (PF) keys and ISPF options, and how to use Tivoli Workload Scheduler for z/OS filter panels to reduce the amount of data displayed in lists.

The scheduler ISPF panels

How you invoke the panels on your system depends on your installation standards. The department that installed Tivoli Workload Scheduler for z/OS can tell you how to invoke Tivoli Workload Scheduler for z/OS at your installation. Usually, you select Tivoli Workload Scheduler for z/OS option from the ISPF main menu.

The scheduler administration is performed on the controlling system in a multisystem complex, so you must use the panels on the system in your configuration that is running the controller.

You can reach all the controller panels from Tivoli Workload Scheduler for z/OS main menu. See Figure 18.

Before you can use any panel in Tivoli Workload Scheduler for z/OS, you must have the authority to access it. See your security administrator or system programmer if you do not have the access that you need.
Setting options

Setting options

You do not have to set options every time you use the panels. The options, and many of the parameters that you enter in the panels, are saved when you leave ISPF (though not if the session is not terminated normally) and will be the default next time.

Select option 0 on the main menu to display the DEFINING PARAMETERS AND OPTIONS panel:

EQQXOPTP -------- DEFINING TWSz PARAMETERS AND OPTIONS --------------
Option ==>>
Select one of the following:

0 REINIT - Re-initialize the application profile values
1 SUBSYSTEM NAME - Set or change name ofSubsystem and Server LU
2 DATE - Specify date/time formats and default calendar
3 COLOR - Specify panel color and highlight attributes
4 ISPF OPTIONS - Specify ISPF/PDF options
5 AD/OI CHECKS - Specify AD/OI consistency checks
6 JCL EDIT - Specify JCL edit tool
7 CLEANUP CHECK - Specify check option for Automatic Cleanup type
8 PANELS STYLE - Specify the style for the operation panels

Figure 19. EQQXOPTP - Defining parameters and options

After you specify the options you prefer, they are used throughout Tivoli Workload Scheduler for z/OS. The options are stored in your ISPF profile data set. When you use the panels, the options are retrieved from your profile.

REINIT

Use this option to set Tivoli Workload Scheduler for z/OS profile to the default values defined at installation time. This is done automatically the first time that you use Tivoli Workload Scheduler for z/OS. If you start to use Tivoli Workload Scheduler for z/OS for a new language feature, perform a REINIT.

SUBSYSTEM NAME

Select this option to specify the name of the controller subsystem with which the panels are to communicate. The name must be an alphanumeric string of not more than 4 characters. If the controller is on a different z/OS, the SERVER LU NAME must be given. The logical unit (LU) name can be a fully qualified LU name networkid.luname, 3-17 characters. The LU name is sufficient if the Server is on the same net as the panels.

DATE

Use this option to specify the format of dates and times in Tivoli Workload Scheduler for z/OS and, if required, to set a local time offset. The SETTING DATE AND TIME FORMAT panel is displayed:
Setting options

**DATE-FORMAT**
You can specify dates with these formats:

- **CCYYMMDD** or **YY/MM/DD**, where **CC** is the century, **YY**, **MM**, and **DD** can be in any order.
- **YY/DDD** or **CCYY/DDD**, where **DDD** is the day number in the Julian calendar. **CC**, **YY**, and **DDD** can be in any order.

The delimiter character, shown as a slash (/), is optional and can be any character other than **C**, **Y**, **M**, or **D**. If you specify **CCYYMMDD**, you cannot use delimiters. The date format can be no more than 8 characters. Example panels in this book use the format **YY/MM/DD**.

**TIME-FORMAT**
Similarly, you can specify the time format as **HH.MM** or **MM.HH**. The delimiter character, shown as a period (.), is optional and can be any character other than **H** or **M**. Example panels in this book use the format **HH.MM**.

**DURATION-FORMAT**
You can specify hours (HH), minutes (MM), and seconds (SS) or minutes (MMMM) and seconds (SS). Any character can be specified as a separation character.

**LOCAL TIME OFFSET**
If you are in a different time zone from the controller, you can specify a local time offset. This means that actual start and end times are adjusted to take your local time into consideration. The local time offset is the number of minutes your local time is ahead of or behind controller subsystem time; that is, Tivoli Workload Scheduler for z/OS controlling processor clock time. The local time offset specified on this panel applies only to the ISPF profile that you are using.
Note: All time values stored in controller data sets are specified in controller subsystem time so that you cannot, for example, use the local-time offset option to specify or display workstation open intervals, input arrival times, or run-cycle start times, in your local time. Reports created by Tivoli Workload Scheduler for z/OS batch jobs always have time values expressed in controller subsystem time and not your local time.

TIME OFFSET SIGN
This option is used with LOCAL TIME OFFSET to specify whether your local time is ahead of or behind Tivoli Workload Scheduler for z/OS controller subsystem time. For example, you specify + if your local time is ahead of controller subsystem time.

CALENDAR ID
Specify the default calendar that Tivoli Workload Scheduler for z/OS uses for panel functions such as the LONG TERM PLAN panel, the GENDAYS command for checking run cycles, and substituting job control language (JCL) variables at job setup. For batch functions, Tivoli Workload Scheduler for z/OS uses the calendar specified in the BATCHOPT initialization statement. In both cases, Tivoli Workload Scheduler for z/OS looks for a calendar called DEFAULT if no other calendar is specified. For some functions (ETT and the EQQUSIN subroutine), Tivoli Workload Scheduler for z/OS has no access to BATCHOPT or the panel default, so the scheduler always looks for the calendar DEFAULT if no calendar is specified. If no calendar is specified and there is no calendar called DEFAULT, Tivoli Workload Scheduler for z/OS treats all days as work days.

COLOR
Select this option to display the SETTING COLOR AND HIGHLIGHT ATTRIBUTES panel, where you specify color and highlighting attributes for different panel elements:

<table>
<thead>
<tr>
<th>COLOR</th>
<th>HILITE</th>
<th>PANEL ELEMENT CATEGORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>WHITE</td>
<td>_____</td>
<td>Panel titles and data items</td>
</tr>
<tr>
<td>BLUE</td>
<td>_____</td>
<td>Directional lines and explanatory text</td>
</tr>
<tr>
<td>BLUE</td>
<td>REVERSE</td>
<td>Header text</td>
</tr>
<tr>
<td>WHITE</td>
<td>_____</td>
<td>Option numbers and command text</td>
</tr>
<tr>
<td>BLUE</td>
<td>_____</td>
<td>Normal status (for instance output text)</td>
</tr>
<tr>
<td>WHITE</td>
<td>_____</td>
<td>Important status (for instance output data)</td>
</tr>
<tr>
<td>RED</td>
<td>_____</td>
<td>Command input</td>
</tr>
<tr>
<td>GREEN</td>
<td>_____</td>
<td>Optional input</td>
</tr>
<tr>
<td>RED</td>
<td>_____</td>
<td>Required input</td>
</tr>
<tr>
<td>RED</td>
<td>BLINK</td>
<td>Error flagged input</td>
</tr>
</tbody>
</table>

Valid color specifications are: WHITE, RED, BLUE, GREEN, PINK, YELLOW, and TURQ
Valid highlight specifications are: underscore, reverse, blink, and blank for no highlighting

Figure 21. EQQXCOLP - Setting color and highlight attributes
Setting options

The panels have different elements; for example, the title of the panel and the command input field. For each of these elements, you can specify color and highlighting. If you set a color to blank, the installation default is used.

To test the effect of the color and highlighting attributes, press Enter to re-display the panel with the specified attributes.

All panel elements in Tivoli Workload Scheduler for z/OS are subsequently displayed with the attributes specified on this panel. You can also use the COLOR command at any time.

ISPF OPTIONS
Select this option in the DEFINING OPC PARAMETERS AND OPTIONS panel to change the following:

TERMINAL Specify these terminal characteristics:
- Terminal type
- Number of PF keys
- Input field pad characters: nulls or blanks
- Command delimiter for stacking commands
- Screen format

LOG/LIST Specify the log data set and list data set options, print process options, and job statement information for the system printer.

PF KEYS Specify the number of PF keys and the operations they will perform.

DISPLAY Specify whether the command line is to be placed at:
- The top (ASIS) of the panel (as shown in the examples in this publication).
- The bottom (BOTTOM) of the panel.

LIST Specify the list data set record format (FBA or VBA), logical record length, and line length.

GRAPHIC Specify Graphical Data Display Manager (GDDM®) graphic print attributes.

ENVIRON Use the ENVIRON commands to trace the TPUT, TGET, and PUTLINE buffers, to produce system ABEND dumps when not running in ISPF TEST mode, and to gather terminal status information.

KEYLIST Modify keylist function. You can issue the KEYLIST command from the command line of a panel. If you issue the KEYLIST command from a panel displaying a keylist, that keylist will be marked *** CURRENTLY ACTIVE KEYLIST ***.

DIALOG TEST Specify dialog test options. You can specify that ending Dialog Test (option 7 on the ISPF primary panel) will restore the TEST/TRACE values that were in effect when Dialog Test was called.

COLOR Change the default color specifications for ISPF panels.

CUAATTR Use the Common User Access (CUA) Color/Emphasis Change Utility to change the color, intensity, and highlight values for panel elements on CUA panels.
Setting options

AD/OI CHECKS
Use this option to specify whether you want application description (AD) and operator instructions (OI) consistency checks done each time an Application is created, deleted, or modified.

Consistency checks consist of looking for a match in the AD database for the operator instruction used in the Application. All operator instructions for which no match is found are deleted.

These checks are made immediately after the AD panel action has been completed, either confirming or canceling it.

For example, if you start the creation of a new Application, APPLX, with one operation, 001, select option 7 from the OPERATION DETAILS panel and create an operator instruction, APPLX-001. Finally, press CANCEL instead of PF3 so that the APPLX application is not created; the consistency checks will lead to the deletion of the newly created operator instruction, because an operation with the key APPLX-001 could not be found in the AD database.

When you select option 0.5, the following panel is displayed:

EQQXAOIP ---------- SETTING AD/OI CONSISTENCY CHECK ----------
Command ===>

CONSIDENCY CHECK ===> Y Specify Y to check AD-OI consistency when you create, update, or delete an Application Description.
Specify N to bypass AD-OI consistency checking.
Default is Y.

CONFIRM PANEL ===> Y Specify Y to display a confirmation panel before AD-OI consistency check actions are applied.
Specify N to bypass the confirmation panel.
Default is Y.

Figure 22. EQQXAOIP - Setting AD/OI consistency check

CONSIDENCY CHECK
If Y is specified, consistency checks will be done from the APPLICATION DESCRIPTION panel. Y is the default.

CONFIRM PANEL
This field is meaningful only if Consistency Check has been set to Y.
If Y is specified, a confirmation panel is displayed before deleting any operations selected by the consistency checks. Y is the default.

JCL EDIT
Use this option if you have a tool to edit JCL, and you want to be able to call it from the AD panels.

When you select option 0.6 the following panel is displayed:
A link between AD panels and a user tool for editing JCL is established by using a panel name. When the JCL Edit option is selected from the AD panels, then the panel above is displayed. The panel must exist, or you receive the ISPF error message: Panel not found.

The default value for the panel name is EQQAJCLE, which is the name of a dummy panel.

When you select option 0.7 on the main menu, the SETTING CHECK FOR AUTOMATIC CLEANUP TYPE panel is displayed. Specify Y to check and possibly modify the Cleanup Data Set list, displayed on the MODIFYING CLEANUP ACTIONS panels, even if the Automatic option is specified at operation level.

Specify N to bypass the check. In this case, the CONFIRM RESTART panel is directly displayed when you request a RESTART function with the cleanup type automatic. The default is N.

You can choose to use the advanced panels or the basic panels, which are the default. By using the advanced panels you can:

- Simplify the way in which you modify operations in the current plan.
- Enhance the way you view the list of all operations in the current plan.
- Browse comprehensive information about an operation in a single panel.

Select option 0.8 in the main menu to display the SETTING PANEL STYLE panel. Enter Y for the advanced panels or N for the basic panels (default). You can change the style of panels at any time by changing the value in the SETTING PANEL STYLE panel. Alternatively, you can enter 0.0 from the main menu to reset the profile values to the default settings.

Because the advanced panels offer different functions, there are often different ways to complete some tasks in these panels.
Common panel commands and facilities

The common commands that you use on Tivoli Workload Scheduler for z/OS panels are described in the following sections. Special commands, as they relate to a specific panel, are described in the individual task descriptions.

The following sections also provide some hints and tips for customizing, locating, and displaying Tivoli Workload Scheduler for z/OS information promptly.

Concatenating options
You can concatenate Tivoli Workload Scheduler for z/OS selection options in the standard ISPF manner. For example, entering 4.1.0 on the main menu takes you directly into the workstation ready list that you last displayed. The scheduler lets you enter concatenated options both in row input fields and on the command line.

**Note:** You cannot use the concatenated option to pass through a confirmation panel without displaying it. For example, entering 9.5.Y from the main panel takes you into option 9.5.

You can also use the ISPF command delimiter to concatenate options in Tivoli Workload Scheduler for z/OS. For example, enter this command string:

```
9.2;y;=9.1;y;=4.1.cpu1
```

to start and stop the job submission function and return to the ready list display for the CPU1 workstation. When you use the ISPF command delimiter, you pass through confirmation panels without displaying them. This can make completing or deleting a long list of applications or occurrences much faster.

From some list panels, you can also concatenate row commands. When the row command input field allows more than 1 character to be specified, you can concatenate options to reach faster the data to which you need access. For example, if you are browsing a list of occurrences in the current plan, you can enter s.2 as the row command for an occurrence in the list to display the list of operations in the selected occurrence. Entering s.6 as the row command for any computer operation in the subsequent list displays the job for the selected operation. Concatenating row commands lets you bypass the intervening panels that present the list of available options.

Quick return command
As with all ISPF applications, the END command returns you to the previously displayed panel. For example, if you select option 4.1. cpu1 from the main menu, the first panel displayed is the ready list. When you enter END or press PF3 from the ready list, the next panel displayed is the main menu. If you choose not to concatenate the options, it takes three panel displays to reach the ready list and three to return.
Quick return command

You can use the ISPF quick return command (=) as a fast path through Tivoli Workload Scheduler for z/OS. For example, to return to the ready list from wherever you are in Tivoli Workload Scheduler for z/OS panels, enter =4.1.0 on the command line.

When you are displaying occurrences and operations in the current plan, you can sometimes navigate through an exceptionally long path. In this case, consider using the quick return command even when you want to return to the same area of the panels.

Primary commands

You can use normal ISPF commands freely throughout the panel. Table 3 shows some of the ISPF commands used on the command line of Tivoli Workload Scheduler for z/OS panels.

Table 3. Primary commands for the panels

<table>
<thead>
<tr>
<th>Command</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>END</td>
<td>Save all changes made in the panel and return to the previous panel if no errors were found. END often starts the operation described in the panel. <strong>Note:</strong> If there is an error in an input field in a panel, you cannot leave that panel by pressing the End PF key or by issuing the END command; either correct the error or issue the CANCEL command.</td>
</tr>
<tr>
<td>RETURN</td>
<td>Return to main menu. An END operation is executed for each panel in the sequence leading back to the main menu; all changes in the individual panels are saved.</td>
</tr>
<tr>
<td>ENTER</td>
<td>Verify and save all changes. The panel is then redisplayed (if no error was found), unless you press Enter on a menu or selection criteria panel, in which case the next logical panel is displayed. If an error is discovered, a short error message is issued. You can obtain more information (a long error message) by pressing the Help PF key.</td>
</tr>
<tr>
<td>CANCEL</td>
<td>Return to the previous panel without making any changes.</td>
</tr>
<tr>
<td>UP nn</td>
<td>Scroll upward through the data by the number of lines specified by nn.</td>
</tr>
<tr>
<td>DOWN nn</td>
<td>Scroll downward through the data by the number of lines specified by nn.</td>
</tr>
<tr>
<td>RIGHT</td>
<td>Display the right part of the data. This is available only from panels that have the text LEFT PART in the panel title.</td>
</tr>
<tr>
<td>LEFT</td>
<td>Display the left part of the data. This is available only from panels that have the text RIGHT PART in the panel title.</td>
</tr>
<tr>
<td>HELP</td>
<td>Display help information.</td>
</tr>
<tr>
<td>SORT</td>
<td>Sort information in a list.</td>
</tr>
<tr>
<td>LOCATE lparm</td>
<td>Scroll to the field specified. If the field is not found, the list is displayed starting with the entry before which the specified field would have occurred. If the list is sorted by application name, then lparm is the name of the application; if sorted by job name, lparm is a job name.</td>
</tr>
<tr>
<td>GRAPH</td>
<td>Display a network of dependencies.</td>
</tr>
<tr>
<td>GDDM</td>
<td>Execute GDDM functions on a graphically displayed network.</td>
</tr>
<tr>
<td>ATTR</td>
<td>Set graphic attributes.</td>
</tr>
</tbody>
</table>

Specifying list criteria

Throughout all the Tivoli Workload Scheduler for z/OS panels, you can choose the options that determine which data elements to list. You can use commands to edit the data and to perform other operations on these lists. Depending on which panel...
Specifying list criteria

You use selection criteria to specify the content of lists in Tivoli Workload Scheduler for z/OS panels. An example of a panel where you specify list criteria follows:

```
Specify selection criteria below and press ENTER to create an operation list.

JOBNAME => P* _____ WORK STATION NAME => ____
APPLICATION ID => ___________ OWNER ID => ________
AUTHORITY GROUP => ___________ PRIORITY => __________
GROUP DEFINITION => ___________ STATUS => ____
CLEAN UP TYPE => ___________ CLEAN UP RESULT => __
OP.EXTENDED NAME => ___________
OP. SE NAME => ___________
Input arrival in format YY/MM/DD HH.MM
FROM => ________ _____
TO => ________ _____
Additional Options: ( Y N )
FAST PATH => Y Valid only along with jobname
MANUALLY HELD => _ leave blank to select all
WAITING FOR SE => _ leave blank to select all
STARTED ON WAIT WS => Y leave blank to select all
```

Figure 25. EQQSOPFP - Selecting operations

You can use blanks, complete names, IDs, or generic search arguments in the input fields in this panel to determine the contents of the list.

When you request a list of operations in the current plan, Tivoli Workload Scheduler for z/OS scans sequentially through all operations defined in the plan to determine which operations match the criteria that you specified. If the current plan is large, this search can take a long time. On some selection criteria panels, you can choose the fastpath option to reduce the overhead of the search. When you use a fastpath, Tivoli Workload Scheduler for z/OS searches through operations included in the current plan job name table, which only has operations on automatic reporting workstations (such as print and computer workstations). When it finds a job name that matches, Tivoli Workload Scheduler for z/OS includes all operations with that job name, whether on an automatic workstation or not. So the search in Figure 25 finds all automatic workstation operations with a job name beginning with P and all other operations with the same name.

Specifying panel views

If you chose to use the advanced panels (see "Panels Style" on page 41) instead of the basic panels, the content and layout of some ISPF panels are different. Several of the advanced panels have a menu bar in which there is a View menu. Depending on which view you select, the type and amount of information changes.

When listing and browsing operations in the current plan, from the View menu at the top of the panel (see Figure 26 on page 45), you can choose from the following views:

- **Compact**
  - Is the default view and displays the concise, summarized information.
Specifying panel views

**Full**
Is an extended view that provides more in-depth information about dates and time, dependencies, and operation properties.

**Job Detail**
Is the view of detailed, job-centric data such as JCL execution and completion information, timings, and operational data.

**Graph**
Is the graphical view of the operations, from which you can click an operation that you want to browse or modify.

In addition, the name of the template the panel uses is displayed next to the name of the view. There is a different template for each type of view of each type of advanced panel.

<table>
<thead>
<tr>
<th>Action</th>
<th>View</th>
<th>Help</th>
</tr>
</thead>
</table>
| EQQMOPRV | OPERATIONS IN THE CURRENT PLAN | Scroll ===>
| Command  | PAGE   |      |

**View: Compact (EQQMOPR) Row 4 of 28**

<table>
<thead>
<tr>
<th>Row</th>
<th>Application ID</th>
<th>Operation</th>
<th>Input</th>
<th>Arrival</th>
<th>Dep</th>
<th>Cond</th>
<th>Dep</th>
<th>SXU</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td></td>
<td>WS No. Jobname</td>
<td>Date</td>
<td>Time</td>
<td>Suc</td>
<td>Pre</td>
<td>Suc</td>
<td>Pre</td>
</tr>
<tr>
<td>___</td>
<td>PAYDAILY</td>
<td>CPU3 040 IEFBR1R</td>
<td>11/05/24</td>
<td>18.00</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>___</td>
<td>PAYDAILY</td>
<td>CPU3 060 IEFBR1R</td>
<td>11/05/25</td>
<td>18.00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>C</td>
</tr>
<tr>
<td>___</td>
<td>PAYDAILY</td>
<td>CPU1 010 IEFBR14</td>
<td>11/05/25</td>
<td>20.00</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>R</td>
</tr>
<tr>
<td>___</td>
<td>PYWEEKLY</td>
<td>CPU3 060 IEFBR1R</td>
<td>11/05/25</td>
<td>18.00</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>C</td>
</tr>
</tbody>
</table>

*Figure 26. EQQMOPRV - Choose View in the menu bar to change the panel view*

When you are listing and browsing applications in the application database, from the **View** menu at the top of the panel, as shown in the example in Figure 27 on page 46, you can choose from the following views:

**Compact**
Is the default view and displays the concise, summarized information.

**Full**
Is an extended view that provides more in-depth information about dates and time, dependencies, and operation properties.

**Graph**
Is the graphical view of the operations, from which you can click on an operation you want to browse or modify.

The name of the template the panel is using is displayed next to the name of the view. There is a different template for each type of view of each type of advanced panel.
Using generic search arguments

Many input fields in Tivoli Workload Scheduler for z/OS panel accept generic search arguments. You specify a generic search argument by entering either an asterisk (*) or a percent sign (%) in an input field. You can enter these characters by themselves or in combination with other characters.

Use an asterisk (*) to represent any character string or a null string. The percent sign (%) represents any single character.

If you want to select all applications whose first three letters are PAY, enter these characters in the input field:

APPLICATION ID ===> PAY*________

If you want to select all applications where P is the first letter and Y is the third letter, you enter:

APPLICATION ID ===> P%Y*________

The percent sign in this example results in a search for application identifiers where there is any single character between P and Y.

Some section panels contain the:

TYPE OF MATCH ===>

field. Using this field you can specify the type of match that should be applied for filters, allocating wild characters (*) and %) to be treated as normal characters. If the field is left blank, then standard generic matching is done.

When double-byte character set (DBCS) mode is specified for kanji, the DBCS asterisk (X’425C’) and the DBCS percent sign (X’426C’) are accepted as generic search arguments. These search arguments can be used for application and owner IDs.

Sorting list output

In all list displays, you can enter the SORT command at the command prompt to display a panel where you can specify the order of the list items. If you are using the advanced panels (see “Panels Style” on page 41), from the Operations in the current plan panel you can do either of the following:

Figure 27. EQQNALS - Choose View in the menu bar to change the panel view
Sorting list output

- Click Action > Sort
- On the command line, enter `SORT <column identifier> A|D`
  
  where column identifier is the name of the column on which you want to sort your list in ascending or descending order.

The sort order you request remains in effect for that specific list type until changed.

For example, if you request a list of occurrences in the current plan and then enter `SORT` when the list is presented, the following panel is displayed:

```
EQQXSRTL ---------------------- SORTING A LIST ----------- ROW 1 TO 14 OF 49
Command ===>

Enter/change data in rows
or enter DELETE in the command field to delete the active sort:
Sort order = 1-9, where 1 is the first sort field.
Direction = A for ascending (default) or D for descending.
```

<table>
<thead>
<tr>
<th>Sort Direction</th>
<th>Title of field</th>
<th>Description of field</th>
</tr>
</thead>
<tbody>
<tr>
<td>op.</td>
<td>Operation no.</td>
<td>first critical op.</td>
</tr>
<tr>
<td>time</td>
<td>Actual input arrival time of occ.</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Actual completion time of the occ.</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Actual start time of first critical op.</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Latest start time first critical op.</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Deadline time of first critical op.</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Input arrival time of first critical op</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Planned start time of first critical op</td>
<td></td>
</tr>
<tr>
<td>time</td>
<td>Deadline time of the occurrence</td>
<td></td>
</tr>
<tr>
<td>Adding function</td>
<td>What added occurrence to current plan</td>
<td></td>
</tr>
<tr>
<td>Application</td>
<td>Application ID</td>
<td></td>
</tr>
<tr>
<td>Application text</td>
<td>Verbal description of the application</td>
<td></td>
</tr>
<tr>
<td>Arrived</td>
<td>Actual input arrival date of occ.</td>
<td></td>
</tr>
<tr>
<td>Authgrp</td>
<td>Authority group</td>
<td></td>
</tr>
</tbody>
</table>

Figure 28. EQQXSRTL - Sorting a list

DBCS identifiers in sorted lists appear in binary order.

Locating data strings in list output

You can enter LOCATE at the command prompt in any list display panel to find a particular data string in the primary sort field. The command also supports generic search strings. For example, if you enter LOC ABC* to find any item in the list beginning with ABC, the list scrolls to the specified field. If the field is not found, the list is displayed starting with the entry before which the specified field would have occurred.

If application name is the primary sort field, request the following

`LOCATE applname`

Similarly, if sorted by job name, request the following:

`LOCATE Jobname`

If you need to regularly issue a LOCATE command against a list of items that is not sorted by the item you want to locate, you can change the sort order by entering the SORT command.
This command is useful when working with lists of special resources or event-triggered-tracking criteria.

**Graphically displaying lists**

If you have Graphical Data Display Manager (GDDM) installed and have a terminal capable of displaying graphics, you can display lists of applications, occurrences, and operations graphically. Graphic displays contain the same information as edit or selection lists, only the format is different. On a graphically displayed list, you can see dependency connections that might be difficult to visualize from a conventional list. An entire network of applications can be represented.

To see a graphically displayed list, enter GRAPH or, if you are using the advanced panels (see “Panels Style” on page 41), click View > Graph. The clarity of the graph depends on how much data GDDM is trying to present on your display terminal. For example, if you request a GRAPH of all operations in the current plan, you would probably see an attractive geometric design that resembles a bunch of grapes: there is simply too much data to present graphically on a standard workstation display terminal.

You can use the GDDM command to display the GDDM menu where you can choose the zoom and print functions. With GDDM Version 3, you can print a zoomed area. Before printing, change the colors so that you have a black-and-white image. It is often useful to save the graph in the ADMGDF format, which you can manipulate with other programs.

The attributes of a graphically displayed list can be customized. Enter ATTR at the command prompt from the graph panel. You can specify the color, shape, and line type for different items displayed in the graph. Press PF1 (HELP) for possible values for the attributes.

**PF Key assignment**

The scheduler panel maintains separate program function (PF) keys from your normal ISPF key assignments. Enter KEYS at the command prompt to display or change the current assignment.

You can assign a PF key to a command that you use regularly; for example, to display the ready list. To ensure that the command will be executed correctly regardless of Tivoli Workload Scheduler for z/OS panel it is entered from, assign the PF key like this:

```
PF5 ===> ;=4.1.cpu1
```

where the semicolon (;) is your ISPF command delimiter.

You can assign PF keys separately for each panel. For example, if you regularly use the application description panel, you can assign PF keys to the OPER and RUN commands. The PF keys that you assign for a particular panel are in effect only for that panel; in other parts of the panel, your standard PF keys for Tivoli Workload Scheduler for z/OS are in effect.

It is recommended that you do not alter the key assignments for PF1 (HELP), or PF12 (RETRIEVE). PF12 (RETRIEVE) returns the command you last executed to the command prompt; press PF12 again, and the command before that is returned. A stack of approximately 25 commands is maintained.
In support of the PFShow command, the PF KEY DEFINITIONS AND LABELS panel lets you optionally assign labels to the PF key definitions. A label is used for display in place of the corresponding definition when the user issues the PFShow command.

When you enter the PFShow command from Tivoli Workload Scheduler for z/OS panels, the labels displayed at the bottom of Figure 29 are presented on Tivoli Workload Scheduler for z/OS panels. To remove the display from your panels, enter PFShow OFF.

**Modeling your environment**

To access your environment data that is stored in the scheduler databases, use the MAINTAINING TWSz DATABASES panel.

This publication groups the topics related to each type of data as follows:

- Chapter 4, “Creating workstations,” on page 53
- Chapter 6, “Creating calendars and periods,” on page 103
- Chapter 7, “Creating applications and groups,” on page 115
- “Specifying operator instructions” on page 159
- Chapter 5, “Creating special resources,” on page 81
- “Adding occurrences by event-triggered tracking” on page 432
- “Creating job descriptions” on page 169
- “User-defined variables and variable tables” on page 454
The graphical user interface

Dynamic Workload Console is the interactive interface for Tivoli Workload Scheduler for z/OS. This console enables you to create, modify, and delete objects in Tivoli Workload Scheduler for z/OS database. It also enables you to monitor and control objects scheduled in the current plan.

Dynamic Workload Console runs on operating systems other than z/OS, such as on Microsoft Windows and UNIX. It communicates with, and present data from, active scheduler systems.

For more information, refer to the Dynamic Workload Console documentation.

Utilities

The long-term plan and the current plan are created and extended using Tivoli Workload Scheduler for z/OS batch programs. Other programs and utilities provide update and report on data in Tivoli Workload Scheduler for z/OS databases. You can submit these programs using Tivoli Workload Scheduler for z/OS panels or using normal job submission. For several of the batch processes, particularly extending the plans, it is convenient to define the job in an application of Tivoli Workload Scheduler for z/OS and use the full facilities of Tivoli Workload Scheduler for z/OS to schedule, submit, and track the processing.

When you submit the jobs from Tivoli Workload Scheduler for z/OS panel, you see the panel in Figure 30.

When you submit the jobs that update or report on Tivoli Workload Scheduler for z/OS data using the panel:

- The job is submitted using TSO submit; as a result, the authority of the submitting user is assigned to the job.
- The JCL for the job is not saved in Tivoli Workload Scheduler for z/OS JCL repository. If the job needs to be rerun, you must re-create the JCL.
• If you select E for the SUBMIT/EDIT option in the GENERATING JCL FOR A
  BATCH JOB panel, an ISPF edit panel containing the JCL is displayed. Enter
  SUBMIT to run the job. When you enter END or use PF3 from the ISPF edit
  panel, the job is not submitted. From the edit panel, you can save the job using
  the CREATE command.
Utilities
Chapter 4. Creating workstations

This chapter describes how to create and use workstations. It covers these topics:
• Workstation types
• Guidelines for creating workstations
• Creating a workstation
• Specifying workstation attributes
• Specifying workstation availability
• Specifying workstation fixed resources
• Controlling workstations

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

What types of workstation are there?

Each workstation supports one activity, which can be any DP activity, including:
• Manual job setup
• Job submission
• Started-task actions
• NetView communication
• Print operations
• Manual preprocessing or postprocessing activity
• Cross dependencies management
• Dynamic scheduling

These are the types of workstation:
• Computer workstations
• Printer workstations
• General workstations
• Remote engine workstations

Create one or more workstations for each activity that you want to schedule and control. For example, all operations representing jobs and started tasks are specified to run on a computer workstation, and print operations are specified to run on a printer workstation. Other operations are specified to run on general workstations.

In a multiple scheduling environments, remote engine workstations allow to define and monitor crossed environments job dependencies.

Computer workstations

The majority of operations are batch jobs and started tasks. These operations are specified to run on computer workstations where they are automatically started when the specified prerequisites are complete, or at a particular time of day, and when all required resources are available. Batch jobs and started tasks are automatically tracked to completion. To automate jobs and started tasks, create at least one job computer workstation and one started-task workstation, even though these might be the same physical processor.
Job computer workstations

**Job computer workstations**

A job computer workstation has the STC option in the panel shown in Figure 35 on page 63 set to N. Tivoli Workload Scheduler for z/OS cannot submit a job unless the computer workstation used by the operation representing the job is available.

To improve the automation of the workload submission to different destinations, you can define a virtual job computer workstation, by specifying a list of destinations for the workload submission (see “Virtual workstations” on page 55). A virtual workstation has the VIRTUAL option in the panel shown in Figure 35 on page 63 set to Y.

A computer workstation associated to a specific destination has the VIRTUAL option set to N. In this case, you can create your computer workstation open intervals (see “Specifying workstation open intervals” on page 75), and specify an alternate workstation where Tivoli Workload Scheduler for z/OS reroutes operations if the normal workstation becomes unavailable. You can also set the status of a workstation from the panel. See “Recovery of operations when a workstation becomes inactive” on page 350 for more details on rerouting computer workstation operations and workload restart.

A computer workstation can also submit operations to non-z/OS environments through a user-defined destination. The submission of these operations is handled by the operation-initiation exit, EQQUX009.

The way JCL is associated with a job or started-task operation is described in “Job statements and computer workstation operations” on page 165.

**Note:** When Tivoli Workload Scheduler for z/OS submits a job, it gives control to z/OS and JES, or some other operating environment. Tivoli Workload Scheduler for z/OS keeps track of what happens to the job or started task and acts accordingly, such as reporting successor operations as ready or starting recovery processing if there is an abnormal termination, but Tivoli Workload Scheduler for z/OS cannot affect how the job or started task executes.

**Started-task computer workstations**

A started-task computer workstation has the STC option in the panel shown in Figure 35 on page 63 set to Y. Operations you specify to run on this workstation are treated as started tasks, not as jobs.

To improve the automation of the workload submission to different destinations, you can define a virtual STC computer workstation, by specifying a list of destinations for the workload submission (see “Virtual workstations” on page 55). A virtual workstation has the VIRTUAL option in the panel shown in Figure 35 on page 63 set to Y.

The JCL for the started task is stored in the same way as the JCL for jobs (see “Job statements and computer workstation operations” on page 165). But, instead of submitting the job to the internal reader on the destination system, Tivoli Workload Scheduler for z/OS puts it into the procedure library allocated to the EQQSTC ddname, and issues the z/OS START command to invoke it.

If you specify the deadline-WTO option, described in “Started-task operations” on page 167, Tivoli Workload Scheduler for z/OS can automatically send a WTO
message to advise the system operator, or NetView, when an operation has passed its deadline. You could use this function to coordinate the shutdown of an online system that executes as a started task.

**Virtual workstations**

To spread the workload across your trackers, you can create a computer workstation with the automatic reporting attribute and the virtual option, defining a list of destinations for the workload submission. When the scheduler processes the operations submitted to a virtual workstation, it distributes the workload according to a sequenced turn criteria, based on a round-robin algorithm. To submit the job, at least one of the destinations in the list must be available.

You can associate open intervals, parallel servers, and fixed resources with each destination belonging to the defined pool. The association is disabled at virtual workstation level, because the jobs that you submit on a virtual workstation are actually run on a single destination. When you associate parallel servers with a virtual workstation destination, you can specify a value up to 65535.

The alternate workstation definition is not applicable either at workstation level or at single destination level.

For a procedure to define a virtual workstation including previously used destinations, see "Using virtual workstations" on page 60.

**Fault-tolerant workstations**

In IBM Tivoli Workload Scheduler for z/OS, a fault-tolerant workstation is a computer workstation configured to schedule jobs on a distributed agent. Thus, you use a fault-tolerant workstation so that you can schedule jobs on computers in the IBM Tivoli Workload Scheduler network.

When you create a fault-tolerant workstation, you must define it in the database and also in the CPUREC initialization statement in the parameter library. For more information about initialization statements and fault-tolerant workstations, see Scheduling End-to-end with Fault Tolerance Capabilities.

**Tivoli Workload Scheduler for z/OS agent workstations**

A Tivoli Workload Scheduler for z/OS agent is a computer automatic workstation and has the Z-CENTRIC AGENT option in the panel shown in Figure 35 on page 63 set to Y. Tivoli Workload Scheduler for z/OS agents are distributed workstations that run jobs scheduled from Tivoli Workload Scheduler for z/OS. Like fault-tolerant workstations, they are installed in a Tivoli Workload Scheduler distributed domain. Unlike fault-tolerant workstations, they do not:

- Have fault tolerance
- Require the end-to-end server
- Need topology definitions

Communication with the agents is handled directly by the controller.

A Tivoli Workload Scheduler for z/OS agent is a computer running the Tivoli Workload Scheduler agent. For more information about the end-to-end scheduling with fault tolerance capabilities, see Tivoli Workload Scheduler for z/OS: End-to-end with z-centric Capabilities.

**Dynamic workstations**

A dynamic workstation is a computer automatic workstation and has the DYNAMIC option in the panel shown in Figure 35 on page 63 set to Y. Dynamic
**Dynamic workstations**

Workstations are associated to the dynamic workload broker component that can schedule dynamic jobs in a z-centric end-to-end configuration. You associate the dynamic workstation to the dynamic workload broker component by specifying an HTTP or HTTPS destination in the ROUTOPTS statement. For details, see the description of the ROUTOPTS statement in *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

You define the dynamic workstation options in the WORKSTATION END-TO-END OPTIONS panel. Depending on your choice in this panel, you can have one of the following scenarios:

- If you select the BROKER option in the WORKSTATION END-TO-END OPTIONS panel, you can submit dynamic jobs by writing in the Tivoli Workload Scheduler for z/OS JCLs only the job name of the dynamic jobs. The dynamic jobs have been previously defined using the Dynamic Workload Console or the broker command line. This method is known as submission by reference because you only need to reference the job you want to submit, without having to write or import the whole job into the JCL.

- If you select the POOL option in the WORKSTATION END-TO-END OPTIONS panel, you can submit standard z-centric jobs to a pool of agents managed by the dynamic workload broker component. You can define the pool using the Dynamic Workload Console or the broker command line, and identify the pool in the Tivoli Workload Scheduler for z/OS workstation using ISPF panels or the Dynamic Workload Console.

- If you select the D-POOL option in the WORKSTATION END-TO-END OPTIONS panel, you can submit standard z-centric jobs with the added value of the resource requirements specified in the database. You can define the dynamic pool using the Dynamic Workload Console or the broker command line, and identify the dynamic pool in the Tivoli Workload Scheduler for z/OS workstation either using ISPF panels or the Dynamic Workload Console.

**Alternate workstations**

When you specify a job or started-task computer workstation as an alternate workstation for another workstation of the same type, try to ensure that their configurations are symmetrical. Tivoli Workload Scheduler for z/OS assumes this for special resources; you do not have to specify that resources are connected to alternate workstations.

An advantage of symmetry is that two workstations can be alternates for each other.

**Printer workstations**

With a printer workstation you can track (but not control) the production of print output. When a output group, tracked by Tivoli Workload Scheduler for z/OS, stops printing, Tivoli Workload Scheduler for z/OS is notified by an event record, and the corresponding operation is set to completed status. If the print operation completes successfully, any successor operations can be started.

**General workstations**

A general workstation lets you control operations that are normally not controlled automatically. To do this, the operation can create events to notify Tivoli Workload Scheduler for z/OS of its status, or an operator can report the status manually. For operations and workstations that are associated with z/OS jobs and started tasks, these events are created automatically by JES and SMF exits. On computer workstations with other non-z/OS operating systems, events are also created.
General workstations

automatically. Operations that are specified to run on general workstations, or computer workstations that have a user-defined destination, must use one of several methods to inform Tivoli Workload Scheduler for z/OS of the operation status:

- Tivoli Workload Scheduler for z/OS ISPF panels
- The TSO command OPSTAT, used either in TSO, or in a CLIST or REXX EXEC
- The EQQEVPGM batch program, using the OPSTAT command as input
- The program interface
- The EQGUIN subroutine, which lets you create events from your own programs

Tivoli Workload Scheduler for z/OS also uses three special general workstations to perform these tasks:

- Set up JCL for jobs and started tasks
- Issue WTO messages, possibly to trigger action by NetView
- Create dummy operations that run for specified periods of time (WAIT option).

Job setup general workstations

A job setup general workstation has the JOB SETUP option in the panel shown in Figure 35 on page 63 set to Y. The job setup workstation lets you prepare your job or started-task JCL manually before execution. Any variables in your JCL are resolved either automatically from information in the Tivoli Workload Scheduler for z/OS databases or manually using the panels. You do not need a job setup workstation if Tivoli Workload Scheduler for z/OS can resolve all the variables automatically.

To select an operation for setup, specify N (set next logical status) next to the job setup operation on the ready list.

In Tivoli Workload Scheduler for z/OS, the job setup operation must be an immediate predecessor of the computer workstation operation for the job. When you have prepared the job, the job operation can be started if it is not waiting for other predecessors.

If the setup operation has more than one succeeding processor operation with the same job name, Tivoli Workload Scheduler for z/OS displays a list of the operations that you can choose from.

WTO general workstations

A WTO general workstation has the WTO option in the panel shown in Figure 35 on page 63 set to Y. A WTO workstation lets you use the extensive scheduling and calendar facilities to issue a write-to-operator (WTO) message at the designated destination operator console. When an operation at a WTO workstation is started, Tivoli Workload Scheduler for z/OS builds a WTO message containing the text for the operation. NetView can then intercept the WTO message and take necessary action.

Tivoli Workload Scheduler for z/OS attempts to automatically dispatch a WTO operation when it becomes ready; however, like operations at any workstation, the scheduling criteria for the operation must be met before the operation can be started.

WAIT general workstations

A WAIT workstation is a general non reporting workstation with the WAIT option set to Y in the panel shown in Figure 35 on page 63. Using a WAIT workstation
you can create a dummy operation that runs for a specified period of time. This
time period is defined as the duration of the operation.

You can insert dummy operations on wait workstations between two dependent
operations to obtain a controlled delay within a sequence.

Operations started on wait workstations have the extended status set to Executing
on a WAIT workstation to remind users that a delay in the defined sequence of
operations is occurring.

Note: To change your system local time while operations are active on a WAIT
workstation, do not use the command /SET DATE=yyyy.ddd,CLOCK=hh.mm.ss
because setting the seconds could cause the operation to delay or anticipate
its completion by 1 minute. Change your system local time by using the
command:
/SET TIMEZONE=W/E.hh.mm

Remote engine workstations
Use remote engine workstations if you want to define in your scheduling
environment dependencies from batch activities managed by other Tivoli Workload
Scheduler environments. This kind of dependencies is called cross dependencies.

The remote engine workstation manages the exchange of information about cross
dependencies resolution between your environment and a remote Tivoli Workload
Scheduler for z/OS engine (controller) or a Tivoli Workload Scheduler engine
(master domain manager).

To define a cross dependency between a job defined in your environment and
another job defined on a different Tivoli Workload Scheduler engine, you must add
a dependency from a shadow job defined on the remote engine workstation. Using
an HTTP or HTTPS connection, the remote engine workstation requests the remote
gine to identify in the plan the specific job instance that must be bound with the
shadow job. The Tivoli Workload Scheduler for z/OS controller is then notified
about any changes of status related to the remote job instance. These status
changes are mapped into shadow job status changes to allow you to track locally
the remote dependency resolution. For more information about shadow jobs, see
“Specifying remote job information in shadow jobs” on page 164.

Because remote engine workstations use the HTTP or HTTPS protocol to let the
Tivoli Workload Scheduler environments communicate, before defining a new
remote engine workstation, you must define in ROUTOPTS a destination for the
remote scheduling environment. The destination must be specified in the remote
guide

workstations” on page 71.

If the remote Tivoli Workload Scheduler environment is z/OS based, a single
HTTP address is sufficient to manage in a transparent way failover scenarios
because the configuration with a hot standby controller is supported through VIPA
and controller and standby controller listen on the same hostname and port.

Guidelines for creating workstations
Here are some guidelines for the workstations you might want to create.
Remember that these are only guidelines and should be adapted to your
installation. Keep the model as simple as possible.
GUIDELINES FOR CREATING WORKSTATIONS

Note: Do not delete a workstation description or change its type if it has operations defined to it. First, request a report detailing the operations that use that particular workstation and would therefore be affected by the change. To request the report, use fastpath 1.4.4.3.

When updating a workstation description, changes to certain fields will never take effect in the current plan as long as that workstation exists in the plan: the old definition is always carried forward. These fields are workstation type, reporting attribute, control on servers, control on resources 1 and 2, and closed status. If you need the change to be reflected in the plan, make the same change in the plan using the MODIFYING CURRENT PLAN (MCP) panel.

Which workstation types are needed?

Although you must create at least one workstation to enable Tivoli Workload Scheduler for z/OS to run, you do not need all workstation types. Create the workstation types that fit the workload that runs at your installation. For example, if certain operations are dependent on the completion of prints produced by other operations, use a printer workstation. However, if the production of prints does not have any great impact on the running of the rest of your workload, you probably do not require a printer workstation.

Because the main function of Tivoli Workload Scheduler for z/OS is to control jobs and started tasks, at least two computer workstations will probably be required at any installation running Tivoli Workload Scheduler for z/OS: one for jobs and one for started tasks.

Use a remote engine workstation if you need to define dependencies from jobs running on a different Tivoli Workload Scheduler environment to accomplish your tasks.

How many workstations of each type?

Your installation configuration influences the number of workstations you need. Normally, you should have two computer workstations for each z/OS system in your complex (one for started tasks and one for jobs). Make the computer workstations available when the z/OS system they represent is available. If you are working in a shared spool environment with one job queue for several systems, you might consider having only two computer workstations for all systems in the JES complex.

Also consider the potential benefits that could be gained by creating further computer workstations to represent a single z/OS system. For example, you could create a computer workstation for IMS™-related batch processing and another for CICS processing. Specifying your operations in this way can give you significantly more control when handling both planned and unexpected outages that do not affect the z/OS system.

If you create individual computer workstations for the unique components of your z/OS batch workload, you have access to the Tivoli Workload Scheduler for z/OS controlled shutdown function to facilitate an orderly shutdown for planned outages. In the event of an online system failure and subsequent extended recovery facility (XRF) takeover, you can restart and reroute the workload to the specified alternate system even in situations where the z/OS system remains operational.
How many workstations of each type?

Because a remote engine workstation is assigned to an HTTP destination mapping a remote scheduling environment, use one remote engine workstation for each Tivoli Workload Scheduler engine you need to interoperate with.

Using virtual workstations

You can define virtual workstations running Version 8.3 with APAR PK46296 or higher. The supported combinations in terms of controller and tracker components are the following:

<table>
<thead>
<tr>
<th>Tracker without virtual workstations support</th>
<th>Controller without virtual workstations support</th>
<th>Controller with virtual workstations support</th>
</tr>
</thead>
<tbody>
<tr>
<td>No error message is issued.</td>
<td>• The tracker destination results active, waiting for connection.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• EQQE136E error message is issued in the controller message log.</td>
<td></td>
</tr>
</tbody>
</table>

| Tracker with virtual workstations support | No error message is issued. The controller ignores some information sent by the tracker. | Normal scenario |

Suppose you want to define a unique workstation matching the following JES2 MAS system configuration, including three computer automatic workstations: CPU1 with local destination, CPU2 with destination D2, and CPU3 with destination D3.

You can use the following procedure:
1. Define a computer automatic workstation in the database, with the VIRTUAL option set to Y. In this example, the name of the virtual workstation is V000. Add ****** to the destination list.
2. Save the current definition of the application descriptions. You can save it in a sequential data set, in batch loader format.
3. Using the mass update utility, change from CPU1 to V000 the workstation name for all the operations associated to CPU1.
4. Run a current plan extend or replan process.
5. Verify that the workload scheduling for V000 works as expected.
6. Consider removing CPU1 from the database once no more applications remain in the database referring to it and no associated operations remain in the current plan.
7. Add D2 to the destination list for V000 and iterate, for CPU2, the steps from 2. to 6.
8. Add D3 to the destination list for V000 and iterate, for CPU3, the steps from 2. to 6.

Notes:
1. If you activated the WLM interface and use a virtual workstation for an operation that has a scheduling environment defined, all the destinations in that virtual workstation must belong to the same Sysplex and JESplex.
2. Before assigning an operation to a virtual workstation, make sure the job can run correctly on all the destinations, checking for possible JCL errors or security violation on any virtual workstation destination.

Dummy workstations
You can sometimes simplify complex dependencies between operations in your applications by using a dummy workstation and creating dummy operations on it.

For example, assume that operations W, X, Y, and Z are all dependent on operations A, B, C, and D. That is, A, B, C, and D must complete before W, X, Y, or Z can start. Figure 32 shows this.

![Figure 32. Dependent operations with complex dependencies](image-url)

This arrangement can be simplified by using a dummy operation, O. You can make O dependent on A, B, C, and D and then make W, X, Y, and Z dependent on O. See Figure 33 on page 62.
Dummy workstations

To create such an operation, first set up a dummy workstation: a general workstation with the REPORTING ATTR field set to N in the panel shown in Figure 35 on page 63. Any operation that becomes ready on a nonreporting workstation is immediately set to status C, complete. If all other conditions are met, any successor operations to this dummy operation can then be started.

You can also create a dummy operation as a separate application, so that its dependencies are all external. This can simplify complicated dependencies between different applications.

In the same way, you can use dummy workstations to insert a delay between two operations in a sequence. Do this by creating operations on wait workstations.

For example, assume that operation B is a successor of operation A and your business policy requires that operation B wait at least 30 minutes before starting. Add a new operation C on a general non-reporting wait workstation. You define this operation as a successor of operation A and as a predecessor of operation B and you set its duration to 30 minutes. When operation A completes, dummy operation C starts and Tivoli Workload Scheduler for z/OS waits 30 minutes before completing it. Operation B is started only after operation C is complete.

Creating a workstation

Follow these steps to create a workstation:

1. Access the workstation panel by entering option 1.1 from the main menu. The Maintaining Work Station Descriptions menu is displayed.
2. Select option 2 (LIST). The LIST OF WORKSTATION DESCRIPTIONS panel, shown in Figure 34 on page 63, is displayed.
Creating a workstation

3. Enter the CREATE command. The CREATING GENERAL INFORMATION ABOUT A WORKSTATION panel, shown in Figure 35, is displayed.

4. Fill in the fields as described in "Specifying workstation attributes and options" on page 65. Note that:

   - A, O, and R commands are not available when the workstation is an FT workstation.
   - O and R commands are not available when the workstation is a remote engine workstation.

5. Enter the A command to specify:

   **Workstation availability and open time intervals**

   See "Specifying workstation availability" on page 74.
Creating a workstation

6. If you are using workstation fixed resources, described in “Specifying workstation fixed resources” on page 78, enter the R command to specify them.

7. If you are creating a virtual workstation, set the VIRTUAL option to Y, then enter the D command to specify a list of destinations associated to this workstation:

```
   VDEST2__ N N R1 N R2
   ******** N N R1 N R2
```

8. Enter the A row command to specify the destination availability and open time intervals:

```
   STANDARD__ DEFINED ________________________
   MONDAY________ STANDARD ________________________
   TUESDAY_______ STANDARD ________________________
   WEDNESDAY_____ STANDARD ________________________
   THURSDAY______ STANDARD ________________________
   FRIDAY________ STANDARD ________________________
   SATURDAY______ STANDARD ________________________
   SUNDAY________ STANDARD ________________________
```

Figure 36. EQQWMDES - Modifying virtual workstation destination

To indicate the system where the controller and a local tracker run, specify ******** in the destination name field.

8. Enter the A row command to specify the destination availability and open time intervals:

```
   STANDARD__ DEFINED ________________________
   MONDAY________ STANDARD ________________________
   TUESDAY_______ STANDARD ________________________
   WEDNESDAY_____ STANDARD ________________________
   THURSDAY______ STANDARD ________________________
   FRIDAY________ STANDARD ________________________
   SATURDAY______ STANDARD ________________________
   SUNDAY________ STANDARD ________________________
```

Figure 37. EQQWMAVV - Availability of a workstation
Creating a workstation

The same concepts apply as in "Specifying workstation availability" on page 74, but instead of a workstation, here, it is a destination that is defined as available or not available.

9. To change the STANDARD open interval, and to create other open intervals, and the associated number of resources, enter the ALL command. The ALL OPEN TIME INTERVALS panel is displayed:

```
Work station : WWS1
Destination   : YDEST2

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

```
Row Day of week or Open time interval Parallel Resources
1 STANDARD 00.00 24.00 65535 99 99

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

Figure 38. EQQWMTAL - All open time interval

The same concepts apply as in "Specifying workstation open intervals" on page 75, but instead of a workstation, here, it is a destination that is defined as available or not available.

Note: You can specify a value up to 65535 in the Parallel servers field.

10. If you are creating a dynamic workstation, set the DYNAMIC option to Y, then enter the O command to specify the end-to-end options associated to this workstation. For details, see "Specifying dynamic workstations" on page 69.

### Specifying workstation attributes and options

You also specify these workstation attributes and options:
- Reporting attributes
- Parallel servers
- Splittable attribute
- Destination
- Default transport time
- Default duration
- Use of a fault-tolerant workstation

These are described in the following sections. See also Table 1 on page 16 for the attributes used for the Paymore example.

#### Specifying workstation reporting attributes

Every operation in the current plan is assigned a status. The status of an operation describes its current condition. When all processing for an operation is finished, the operation is assigned status C (complete). Before it reaches completion, the operation will have many different statuses as it progresses through the system.

Tivoli Workload Scheduler for z/OS changes the status of the operation in response to:
- JES and SMF exits
- Tivoli Workload Scheduler for z/OS panel requests
Specifying workstation reporting attributes

- The EQQUSIN subroutine
- The program interface
- The OPSTAT and WSSTAT TSO commands

The sequence of statuses that an operation is assigned and the mechanism used for reporting status updates depends on the reporting attribute of the workstation the operation is defined on.

A workstation can have one of four reporting attributes:

A Automatic.

The status change of operations is normally reported automatically, in response to event records created by the JES and SMF exits in Tivoli Workload Scheduler for z/OS. You can also change the status of these operations using the panels, the OPSTAT command, the EQQUSIN subroutine, the EQQEVPGM batch program, or the program interface. This reporting attribute is normally used for computer and print workstations, or workstations that specify a user-defined destination.

When a general workstation with the WTO option is assigned automatic reporting, Tivoli Workload Scheduler for z/OS attempts to dispatch the operation as soon as all normal submission criteria have been satisfied. When the WTO has been submitted, the status is set to SQ. Use this attribute for synchronous WTO operations, where you want successor operations to wait until actions (that NetView, for example, takes when it intercepts the WTO) are reported complete. When you want the successor operations to run, report C (Complete) status using, for example, the OPSTAT command. If you need to measure the elapsed time of the operation that the WTO triggers, report S status when you want the timing to start (this changes the status to SS).

If an operation at an automatically reporting workstation is set to started status manually, this will not cause Tivoli Workload Scheduler for z/OS to submit the job, started task or WTO that the operation represents. Tivoli Workload Scheduler for z/OS selects work to be started from the queue of ready operations: those operations in A, R, or * status.

C Completion only.

The status change of operations is normally reported from the READY LIST panel. You can also change the status of operations with the OPSTAT command, the EQQUSIN subroutine, the EQQEVPGM batch program, or the program interface. This reporting attribute is normally used for general workstations that are not used for JCL preparation.

When a general workstation with the WTO option is assigned completion-only reporting, Tivoli Workload Scheduler for z/OS attempts to dispatch the operation as soon as all normal submission criteria have been satisfied. Use the completion-only attribute when you want to trigger actions asynchronously. When the WTO has been submitted, Tivoli Workload Scheduler for z/OS automatically sets the status to C (completed), so successor operations can run immediately.

N Non reporting.

Operations on a non-reporting workstation are set to complete as soon as they become eligible to be started. No job, started task, or WTO is submitted; instead the operation is set to completed status. You can use this reporting attribute for operations that do not require any processing.
Specifying workstation reporting attributes

For example, the operations can be used to hold the dependencies for successors, or the operation might represent a milestone in the processing. This is a good way to keep track of the important processing points in large installations.

When a general work station with non-reporting attribute has the WAIT option, Tivoli Workload Scheduler for z/OS does not complete the operation as soon as it becomes eligible. The operation is started and remains in started status for the time defined as operation duration.

This type of workstation is often used for dummy operations created to simplify the sequencing of other operations. (See “Dummy workstations” on page 61.)

Start and completion.

The status change of operations is normally reported from the READY LIST panel. You can also change the status of operations with the OPSTAT command, the EQQUSIN subroutine, the EQQEVPGM batch program, or the program interface. This reporting attribute is normally used for general workstations that are used for JCL preparation, or for other general workstations where the duration of the task needs to be tracked. You should consider using this reporting attribute for operations on a data entry workstation, where the operation represents a task that must be performed manually, and the duration of the task is important.

For general workstations with the WTO option, this has the same effect as automatic reporting.

Specifying fault-tolerant or Tivoli Workload Scheduler for z/OS agent workstations

You define fault-tolerant workstations or Tivoli Workload Scheduler for z/OS agent workstations when you want to schedule jobs on non-z/OS systems using IBM Tivoli Workload Scheduler distributed agents. In Tivoli Workload Scheduler for z/OS, fault-tolerant workstations and Tivoli Workload Scheduler for z/OS agent workstations are computer automatic workstations configured to schedule jobs on distributed agents that run on Windows, UNIX, or Linux.

Both fault-tolerant and Tivoli Workload Scheduler for z/OS agent workstations share some of the same properties with other workstations. Table 4 summarizes the settings for a fault-tolerant workstation:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation type</td>
<td>C</td>
<td>Computer</td>
</tr>
<tr>
<td>Reporting attribute</td>
<td>A</td>
<td>Automatic</td>
</tr>
<tr>
<td>Started task, STC</td>
<td>N</td>
<td>No</td>
</tr>
<tr>
<td>Server usage</td>
<td>N</td>
<td>Neither</td>
</tr>
<tr>
<td>Destination</td>
<td>Blank</td>
<td>–</td>
</tr>
</tbody>
</table>

Table 5 on page 68 summarizes the settings for a Tivoli Workload Scheduler for z/OS agent workstation:
### Specifying fault-tolerant or Tivoli Workload Scheduler for z/OS agent workstations

Table 5. Settings for Tivoli Workload Scheduler for z/OS agent workstations

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation type</td>
<td>C</td>
<td>Computer</td>
</tr>
<tr>
<td>Reporting attribute</td>
<td>A</td>
<td>Automatic</td>
</tr>
<tr>
<td>Started task, STC</td>
<td>N</td>
<td>No</td>
</tr>
<tr>
<td>Server usage</td>
<td>N</td>
<td>Neither</td>
</tr>
<tr>
<td>Destination</td>
<td>Alphanumerical string of up to 8 characters</td>
<td>The destination name that was defined with the HTTP parameter in the ROUTOPTS statement. You can add, modify, or delete the destinations. For details, see the description of the ROUTOPTS statement in Tivoli Workload Scheduler for z/OS: Customization and Tuning.</td>
</tr>
<tr>
<td>Job user (JOBUSR)</td>
<td>Alphanumerical string of up to 47 characters</td>
<td>For J2EE jobs, the WebSphere® Application Server administrator. For all the other jobs, the name of the user submitting the job. If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined. If you are defining a Windows domain user, use the following format: JOBUSR(domainName\user1) It can be overwritten by the user specified either through EQQUX001 or in a JOBREC statement in the job JCL.</td>
</tr>
<tr>
<td>Job password (JOBPWD)</td>
<td>Y</td>
<td>Specifies that the user name set in JOBUSR or by using the job-submit exit EQQUX001 is associated with a password. If you set JOBPWD to Y, Tivoli Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement. For details, see the description of the USRREC statement in Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities. Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to N if the user works with UNIX workstations. Valid values are Y or N. It can be overwritten by the user specified in a JOBREC statement in the job JCL.</td>
</tr>
</tbody>
</table>
Table 5. Settings for Tivoli Workload Scheduler for z/OS agent workstations (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Value Description</th>
</tr>
</thead>
</table>
| Job type (JOBTYPE)  |       | The type of job to be run. Supported job types are:  
|                     |       | • database  
|                     |       | • file transfer  
|                     |       | • web service  
|                     |       | • java  
|                     |       | • xajob  
|                     |       | • j2ee jms  
|                     |       | It can be overwritten by the user specified in a JOBREC statement in the job JCL. For information about supported job types, see the description of the JOBREC statement in Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities. |

Specifying dynamic workstations

You define dynamic workstations to schedule distributed jobs in a z-centric end-to-end configuration. Dynamic workstations are associated to the dynamic workload broker component that can schedule distributed jobs in a z-centric end-to-end configuration. You associate the dynamic workstation to the dynamic workload broker component by specifying an HTTP or HTTPS destination in the ROUTOPTS statement. For details, see the description of the ROUTOPTS statement in Tivoli Workload Scheduler for z/OS: Customization and Tuning.

The dynamic workstations share some of the same properties with other workstations.

Table 6 summarizes the settings for a dynamic workstation:

Table 6. Settings for dynamic workstations

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Workstation type</td>
<td>C</td>
<td>Computer</td>
</tr>
<tr>
<td>Reporting attribute</td>
<td>A</td>
<td>Automatic</td>
</tr>
<tr>
<td>Started task, STC</td>
<td>N</td>
<td>No</td>
</tr>
<tr>
<td>Server usage</td>
<td>N</td>
<td>Neither. Supported values are: Parallel server usage, B, N, P, or C</td>
</tr>
<tr>
<td>Destination</td>
<td></td>
<td>The destination name that was defined with the HTTP parameter in the ROUTOPTS statement. You can add, modify, or delete the destinations. For details, see the description of the ROUTOPTS statement in Tivoli Workload Scheduler for z/OS: Customization and Tuning.</td>
</tr>
</tbody>
</table>

Specifying fault-tolerant or Tivoli Workload Scheduler for z/OS agent workstations
### Specifying dynamic workstations

**Table 6. Settings for dynamic workstations (continued)**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Value Description</th>
</tr>
</thead>
</table>
| Job user (JOBUSR)      | Alphanumeric string of up to 47 characters | For J2EE jobs, the WebSphere Application Server administrator. For all other jobs, the name of the user submitting the job.  
If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined. If you are defining a Windows domain user, use the following format:  
JOBUSR(domainName\user1)  
It can be overwritten by the user specified either through EQQUX001 or in a JOBREC statement in the job JCL. |
| Job password (JOBPWD)  | Y                                  | Specifies that the user name set in JOBUSR or by using the job-submit exit EQQUX001 is associated with a password. If you set JOBPWD to Y, Tivoli Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement. For details, see the description of the USRREC statement in *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities*.  
Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to N if the user works with UNIX workstations.  
Valid values are Y or N.  
It can be overwritten by the user specified in a JOBREC statement in the job JCL. |
| Job type (JOBTYPE)     |                                    | The type of job to be run. Supported job types are:  
• database  
• file transfer  
• web service  
• java  
• xajob  
• j2ee jms  
It can be overwritten by the user specified in a JOBREC statement in the job JCL. For information about supported job types, see the description of the JOBREC statement in *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities*. |
## Specifying dynamic workstations

### Table 6. Settings for dynamic workstations (continued)

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
<th>Value Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BROKER</td>
<td></td>
<td>Specifies that the workstation is associated to the dynamic workload broker component and can schedule distributed jobs in a z-centric end-to-end configuration. Supports the submission by reference of jobs defined locally on the dynamic workload broker component. Valid values are Y, N or blank.</td>
</tr>
<tr>
<td>POOL</td>
<td></td>
<td>The name of the pool of dynamic workload broker component agents associated to this workstation.</td>
</tr>
<tr>
<td>D-POOL</td>
<td></td>
<td>The name of the dynamic pool meeting the resource requirements associated to this workstation.</td>
</tr>
</tbody>
</table>

### Specifying remote engine workstations

You define a remote engine workstation if you want to federate your Tivoli Workload Scheduler for z/OS environment with another Tivoli Workload Scheduler environment, either distributed or z/OS based, to add and monitor dependencies from jobs running in the other scheduling environment.

Table 7 summarizes the settings for a remote engine workstation:

### Table 7. Settings for remote engine workstations

<table>
<thead>
<tr>
<th>Field</th>
<th>Values</th>
<th>Value description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Remote engine type</td>
<td>Z or D</td>
<td>Z for z/OS remote engine, D for distributed remote engine</td>
</tr>
<tr>
<td>Destination</td>
<td>Alphanumeric string of up to 8 characters</td>
<td>The destination name defined with the HTTP or HTTPS parameter in the ROUTOPTS statement. You can add, modify, or delete the destinations. For details, see the description of the ROUTOPTS statement in Tivoli Workload Scheduler for z/OS: Customization and Tuning.</td>
</tr>
<tr>
<td>Reporting attribute</td>
<td>A</td>
<td>Automatic</td>
</tr>
</tbody>
</table>

For more information see section “Remote engine workstations” on page 58.

### Specifying workstation parallel servers

When you create an operation, you specify how many parallel servers it requires (see "Using parallel servers" on page 150). The workstation that the operation is using must have that number of parallel servers available before the operation can run. This option does not apply to fault-tolerant workstations.

The number of parallel servers a workstation owns limits the number of operations that can be in started status at any time. You set this value when you create the workstation (see Figure 42 on page 76), but you can change it using the MODIFY CURRENT PLAN (MCP) panel.
Specifying workstation parallel servers

You can specify, by setting server usage = P (planning only), that the number of parallel servers should not be considered when Tivoli Workload Scheduler for z/OS is deciding when to start an operation. If you do this, the number of parallel servers will be used only for planning purposes, and the plans that Tivoli Workload Scheduler for z/OS produces cannot accurately predict the behavior of real work in your system, because Tivoli Workload Scheduler for z/OS will submit as many jobs as are ready, regardless of its count of the number of servers in use. It is better to specify server usage = B (both planning and control), so that Tivoli Workload Scheduler for z/OS submits jobs only up to the limit of the number of servers specified.

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

If you specify a server usage of B (both) or C (control only), the number of parallel servers required by the operation must also be available on the workstation for the operation to be started. So the number of computer workstations and their availability are important factors when you specify your system to Tivoli Workload Scheduler for z/OS.

In Figure 39, jobs 1 through 6 were defined on a computer workstation, CPU1, which has four parallel servers. Server usage B was specified for this workstation. This means that CPU1 can start only four of the six waiting jobs, even though there are several JES initiators free to run the jobs that are waiting. It selects the jobs using the algorithm described in Chapter 13, “How work is selected for automatic submission,” on page 293.

Specifying splittable workstations

A workstation is described as splittable if operations on the workstation can be interrupted and then continued at a later time. A good candidate for this attribute is a job setup general workstation where you prepare JCL for submission. If the preparation of the JCL is interrupted by the person issuing the TSAVE command, the operation is given status I, interrupted. Preparation of the JCL can continue at a later time.

Printer workstations can also be splittable, but operations on computer workstations cannot be split.
Specifying workstation destinations

For workstations representing computer systems (computer workstations and WTO general workstations), the destination is the tracker. The tracker can communicate with the controller in several ways:

- Shared DASD containing a Tivoli Workload Scheduler for z/OS SUBMIT/RELEASE data set
- z/OS cross-system coupling facility (XCF) communication links
- A VTAM connection and the network communication function (NCF) of the tracker
- A TCP/IP communication link
- A user-defined method invoked from the operation-initiation exit, EQQUX009

Therefore, you can specify the destination as:

- The ddname of a SUBMIT/RELEASE data set
- The XCF member name of a tracker
- The NCF receivingLU name of a tracker
- The logical name associated with the IP address or host name of the tracker
- A user-defined name

For workstations representing a remote engine, a Tivoli Workload Scheduler for z/OS agent, or a dynamic domain manager, which are directly connected to the controller, the destination is defined by the fully-qualified host name or TCP/IP address and the port number of the agent. The remote engine, Tivoli Workload Scheduler for z/OS agent, and dynamic domain manager, communicate with the controller through an HTTP/HTTPS communication link.

When you create a workstation, specify a destination that corresponds to a destination specified in the ROUTOPTS initialization statement (for details about this statement, see Customization and Tuning). The default destination is the system where the controller is started.

Specifying the default workstation transport time

The transport time for an operation is the time that the system should allow between the end of a predecessor operation and the beginning of the present operation. This is the time needed for materials to be transported from one workstation to another.

For example, a tape might be written by operation A in New York and be required by operation B in Chicago. Both sites are controlled by Tivoli Workload Scheduler for z/OS. The transport time of operation B is the time that should be allowed for the tape to be transported from New York to Chicago.

The transport time of the workstation is the default transport time for all operations defined on that workstation. You can override this by specifying a transport time when you create an operation.

Note: The transport time is used only for planning purposes. Jobs, for example, are started regardless of the transport time specified. Transport time is not used even in planning if the predecessor and successor operations are on the same workstation.
Specifying the default duration

The duration specified for the workstation is the default estimated processing time for all operations on that workstation. You can override this by specifying a duration when you create an operation. The minimum value is one second, and the maximum value is 99 hours 59 minutes 00 seconds. If you specify 99 hours 59 minutes 01 seconds, you do not receive an alert message if the actual duration is greater than the planned duration.

Tivoli Workload Scheduler for z/OS uses the estimated processing time when creating the current plan, to work out a timetable for the operations. Each operation has a:
- Planned start time
- Latest start time
- Planned end time

It is not necessary to give an accurate figure, because Tivoli Workload Scheduler for z/OS can adjust this figure dynamically from its experience of the actual durations (see “Using duration feedback options” on page 156).

Specifying the Automation option

Set the AUTOMATION option to YES to enable the workstation to send commands to System Automation. Enter the command text and any additional automation information in the EQQAMAIP or EQQMAIP panel. For details about these panels, see “Creating a workstation” on page 62.

Note: The Automation option requires that the workstation type is general and the reporting attribute is automatic. The Automation option is incompatible with setting the workstation options FTA, WTO, STC, JCL, and Z-CENTRIC AGENT.

To select the target NetView by using a workstation, you can either:
- Use the correspondence between the workstation name and the target NetView domain ID that is set in the System Automation database.
- In the Destination field of the EQQWCGEP panel, specify the NetView target domain ID where to run the commands. In this case, you must also define this destination to Tivoli Workload Scheduler for z/OS by setting the destination parameter in the ROUTOPTS USER statement of the controller.

Specifying workstation availability

Some workstations are not available 24 hours every day; Tivoli Workload Scheduler for z/OS can run work on the workstation only when it is available. When it is unavailable, either as scheduled or unexpectedly, Tivoli Workload Scheduler for z/OS can reroute work to an alternate workstation. This feature does not apply for scheduling a job on non-z/OS systems using a fault-tolerant workstation. Fault-tolerant workstations are always available.

To be available for work, a workstation must be open, active, and connected. To specify when a workstation is available for processing, enter the A command on the CREATING GENERAL INFORMATION ABOUT A WORKSTATION panel (Figure 35 on page 63). This panel is displayed:
Specifying workstation availability

In Figure 40, one open interval, called STANDARD, is created, and all days use this interval. To close the workstation on Saturday, enter the C command beside this row, as shown.

If you enter the S row command on a row that refers to the STANDARD interval (such as TUESDAY in Figure 40), the OPEN TIME INTERVALS FOR ONE DAY panel, shown in Figure 41, is displayed.

If you press F3 (Exit) in this panel without specifying an open interval, the workstation will be closed for that day. Use F12 (Cancel) to leave the day connected to the STANDARD interval.

Specifying workstation open intervals

Before Tivoli Workload Scheduler for z/OS can start an operation, the workstation that the operation is defined on must be available. So, by controlling workstation availability, you control the running of operations defined on the workstation. Tivoli Workload Scheduler for z/OS establishes the availability of a workstation by using the open intervals in the workstation description database. These are the times
Specifying workstation open intervals

when workstation resources and parallel servers are available to process work. If no parallel servers or resources are available, no work is run at the workstation.

In Tivoli Workload Scheduler for z/OS, workstations are usually created to represent specific elements in your system configuration; the availability of these workstations should reflect the availability of those elements in the real world. For instance, a computer workstation might be created for each z/OS system in a Tivoli Workload Scheduler for z/OS complex. See “How many workstations of each type?” on page 59. So, the availability of the computer workstation should reflect the availability of the z/OS system it represents. This prevents Tivoli Workload Scheduler for z/OS from submitting work to a z/OS system that is not physically available. Also, the accuracy of any planning predictions that Tivoli Workload Scheduler for z/OS produces for you depends on how accurately you have described your installation to Tivoli Workload Scheduler for z/OS.

To change the STANDARD open interval, and to create other open intervals, and the associated number of resources, enter the ALL command on the AVAILABILITY OF A WORKSTATION panel (Figure 40 on page 75). The ALL OPEN TIME INTERVALS panel will then be displayed.

Figure 42 shows four intervals for STANDARD, and three intervals for SUNDAY. The CPU1 workstation has its availability and resources specified by STANDARD from Monday to Friday, it is closed on Saturday, and its availability and resources on Sunday are specified by the three SUNDAY intervals.

Closing workstations

Work and free days, as specified in the Tivoli Workload Scheduler for z/OS calendar, are used by Tivoli Workload Scheduler for z/OS to decide whether applications should be scheduled to start on a particular day. Tivoli Workload Scheduler for z/OS must also consider the availability of workstations. It takes this information from the workstation description database. Before it schedules an operation, Tivoli Workload Scheduler for z/OS checks that the workstation on which the operation will run is available throughout the estimated run time of the operation. Whether Tivoli Workload Scheduler for z/OS starts the operation depends on the result of this check and the shutdown policy that you have specified in the SHUTDOWNPOLICY parameter of the JTOPTS initialization statement (for details about JTOPTS, see Customization and Tuning).
Closing workstations

When you have a holiday, which is specified in the calendar as a free day, Tivoli Workload Scheduler for z/OS does not schedule operations on that day (unless the run cycle specifies free day rule = 3), but it schedules operations right up to the end of the previous day, regardless of their estimated duration. This might be what you want.

But when you have a z/OS maintenance period, for example, you want to avoid having any jobs running at the start of the period. In this case, make the workstations unavailable for this period. Tivoli Workload Scheduler for z/OS does not start an operation on a workstation if there is less time than its estimated duration until the start of a closed interval.

If no work at all is to be done on a particular day; that is, no work must be started on that day or carried forward from the previous day:
1. Specify it as a free day.
2. Check that no applications with free day rule 3 are scheduled on that day.
3. Do not have any open intervals for the workstation on the holiday, so that Tivoli Workload Scheduler for z/OS does not start any work on the previous day that will run over into the closed day. You can specify dates when all workstations should be closed on the MODIFYING ALL WORKSTATIONS CLOSED panel (Figure 43), which you reach by selecting option 4 from the MAINTAINING WORKSTATION DESCRIPTIONS panel.

You can specify when workstations are open with the WORKSTATION DESCRIPTION panel, in any of these ways. The list is in order of highest to lowest precedence.
1. For each workstation, you can create open time intervals for a specific date.
2. For all workstations, you can create intervals for specific dates when they are all closed (for example, 95/12/25 00:00–23:59).
3. For each workstation, you can create open time intervals for a particular weekday (for example, Friday).
4. For each workstation, you can create open time intervals for a standard day. You then specify which days of the week are to be considered standard days (for example, Monday, Tuesday, Wednesday, and Thursday).

Tivoli Workload Scheduler for z/OS never cancels a job: a job that is submitted is allowed to complete, even if it runs over to a day where no work is to be done.

When you have made a change with the WORKSTATION DESCRIPTION panel, you must run daily planning for the change to take effect. But a change to the
Closing workstations

closed status does not take effect, even after a daily planning EXTEND or REPLAN, if you have closed the workstation in the current plan (using the MCP panel, for example).

Specifying workstation fixed resources

Some workstations have limited resources:
- A computer workstation has a limited number of JES initiators.
- A printer workstation has limited number of printers.
- A job setup workstation has a limited number of people able to edit JCL.

Tivoli Workload Scheduler for z/OS can keep track of these limited resources and take account of them in its plan for jobs on z/OS systems. This feature does not apply for scheduling a job on non-z/OS systems using a fault-tolerant workstation.

When you set up a workstation, you can specify the quantity of two resources available to operations at the workstation, \( R_1 \) and \( R_2 \). You can also assign a 2-character name to these resources.

If you are new to Tivoli Workload Scheduler for z/OS, you are recommended not to use these fixed resources, but to use the special resources described in Chapter 5, “Creating special resources,” on page 81 instead, because these do not have the restrictions associated with workstation fixed resources, the most important being that:
- You might have no more than 99 of each resource.
- The name is limited to 2 characters.
- Workstations cannot share the resources.

To specify workstation fixed resources, enter the R command from the CREATING GENERAL INFORMATION ABOUT A WORKSTATION panel (Figure 35 on page 63). The RESOURCES FOR A WORKSTATION panel, shown in Figure 44, is displayed.

![Figure 44. EQQWMREP - Resources for a workstation](image)

Workstation resources can be used for scheduling purposes. When you create an operation, you can specify how many of the workstation resources (\( R_1 \), \( R_2 \), or both) the operation will use. If that quantity of the resources is not available, the operation will not be started by Tivoli Workload Scheduler for z/OS (see “Specifying workstation fixed resources” for exceptions to this rule).
Specifying workstation fixed resources

R1 and R2 can represent any physical resource in your system that is significant to you for scheduling purposes. For example, if you create computer workstation CPUA, which represents a z/OS system in your configuration, you can make R1 represent the tape drives on that system. If, when you create an operation, you also specify the number of tape drives (that is, how much of R1 the operation will use), Tivoli Workload Scheduler for z/OS will not schedule the operation unless its required number of tape drives is available.

Note: Tivoli Workload Scheduler for z/OS does not check the actual availability of resources. It reaches its scheduling decisions based on information contained in its database. From this information, it can decide whether Tivoli Workload Scheduler for z/OS-controlled operations are using a resource. If a resource is being used outside the control of Tivoli Workload Scheduler for z/OS or you have not specified to Tivoli Workload Scheduler for z/OS that an operation uses a resource, Tivoli Workload Scheduler for z/OS has no means of knowing that the physical resource is not available.

When Tivoli Workload Scheduler for z/OS builds the current plan, it uses information from the databases, and considers the workstation resources and their availability. If you do not want workstation resources to be considered when the plan is built, specify N for the PLANNING option.

The plan contains the best estimation of when operations will start. If something unexpected happens (for example, a job exceeds its expected run time or a tape drive needs repair) Tivoli Workload Scheduler for z/OS might need to reassess the start time of some of its operations. At this point, the Tivoli Workload Scheduler for z/OS CONTROL option becomes important. If you specified Y for this option, Tivoli Workload Scheduler for z/OS considers the workstation resources when rescheduling its operations. Otherwise, the workstation resources are ignored.

Controlling workstations

When you create workstations, this updates the workstation database, which is used when you update the long-term plan.

If you need to make immediate changes to the status of a workstation, or query its status, use the MODIFY CURRENT PLAN panel. These tasks dynamically control the availability of workstations:

- Modifying the workstation open intervals in the current plan from the MODIFY CURRENT PLAN panel (option 5.5 from the main menu)
- Setting the workstation to active
- Looking at the workstation status
- Closing a workstation
- Redirecting work from one workstation to an alternate workstation
Controlling workstations
Chapter 5. Creating special resources

This chapter explains special resources, and shows you how to create and use them. There are three types of resources:

Special resources
These resources are the most flexible. Create them using the panel that is described in this chapter.

Workstation fixed resources
These resources, called by default R1 and R2, are owned by one workstation only. A common use is for tape pools, but you cannot share these between workstations, so you might have problems if a computer workstation and a started task workstation share the same tape drives. Specify them using the workstation panel.

Parallel servers
These resources normally represent JES initiators for computer workstations running the z/OS operating system. Specify them using the workstation panel.

Understanding special resources

You can use special resources to represent any type of limited resource, such as tape drives, communication lines, or a database. You create resources using the SPECIAL RESOURCES panel, which is described in this chapter. The SPECIAL RESOURCES panel updates the resource database, which has these details of each resource:

Name
The resource name. It can be up to 44 characters.

Availability
Available (Y) or not available (N).

Connected workstations
A list of the workstations where operations can allocate the resource.

Quantity
It can be from 1 to 99999.

Used for
Specifies how Tivoli Workload Scheduler for z/OS is to use the special resource. Allowed values are:

P    Planning
C    Control
B    Both control and planning
N    Neither control nor planning

On-error action
Specifies the action to take if the operation that allocates this resource ends in error (and does not have an overriding keep-on-error specification in the operation definition). Possible values are:

F    Free all
FX   Free exclusively-held resources
FS   Free shared resources
Understanding special resources

K Keep all

Tivoli Workload Scheduler for z/OS uses the attribute specified at operation level first. If this is blank, it uses the attribute specified in the resource database. If this is also blank, it uses the ONERROR keyword of the RESOPTS statement.

On Complete Specifies the value to which the global availability is reset after the operation that uses the resource completes. It can be one of the following:

Y Sets the global availability to Yes.
N Sets the global availability to No.
R Sets the global availability to blank.

Blank Uses the system default, according to the following order:
1. The On Complete value set at operation definition level, if not blank.
2. The On Complete value set at special resource definition level, if not blank.
3. The ONCOMPLETE or DYNONCOMPLETE keyword value, respectively set for the not dynamically added resources or the dynamically added resources, in all the other cases.

Max Usage Limit

Specifies after how many allocations of the special resource, its availability is reset according to the value set for Max Usage Type. An internal usage counter is increased each time an operation allocates the resource. When the internal counter reaches the Max Usage Limit value, the global availability is reset to the value specified with Max Usage Type.

The default value is 0, meaning that no usage counter check is done.

Max Usage Type

Specifies the value to which the global availability is reset when the Max Usage Limit is reached. This value is valid only if the Max Usage Limit is nonzero. Possible values are:

Y Sets the global availability to Yes.
N Sets the global availability to No.
R Sets the global availability to blank.

The quantity, availability, and list of workstations, can vary with time. You can create time intervals to control each special resource.

You also specify, for each operation, the special resources that it uses: how (shared or exclusive), how many (quantity), and the on-error attribute.

The long-term plan is built without taking the special resources into account, but when you extend the current plan, it schedules operations taking account of all the special resources that are used for planning (though the daily planning program does not take manually changed availability, quantity, and deviation into account. This because they are assumed to be usually temporary changes and the values will reset to the normal values when, for example, an engineer has repaired a tape.
Understanding special resources

The special resource details of the operations in the current plan are copied from the database and held in the current plan extension data set. These details include the information from the resource database, but also have these fields:

**Quantity**
- It can be from 1 to 999999, or blank. If you specify a quantity, this overrides the scheduled quantity from the database.

**Availability**
- It can be Y, N, or blank. If you specify an availability, this overrides the scheduled availability from the database.

**Deviation**
- It can be from -999999 to 999999, or blank. You use the deviation to make a temporary alteration to the scheduled quantity.

To change the quantity and availability of a special resource, and the connected workstations, use the Special Resource Monitor, which is option 7 (SPECRES) in the MODIFY CURRENT PLAN panel. You might need to make a resource unavailable (to prevent the submission of all jobs needing a database, if some corruption is suspected), alter the quantity by specifying a deviation (if a tape drive is broken), or change the list of connected workstations (to include a workstation that will take over processing from the normal one). Refer to [“Using the special resource monitor” on page 610](#) for details.

Other ways of changing resource attributes are:

- **EQQUSIN subroutine** Refer to *Customization and Tuning*.
- **SRSTAT command** See [“SRSTAT” on page 664](#).

If the availability of a resource is known to the Resource Object Data Manager (RODM), Tivoli Workload Scheduler for z/OS can, by subscribing to RODM for that resource, be notified automatically of any changes. This is the best alternative where RODM is installed.

Changes to special resources using any of these methods override the scheduled quantity and availability, but you can at any time reset the values to those specified for the current interval.

For an operation running on a distributed agent, the usage of special resources causes the loss of fault tolerance. Only the controller determines the availability of a resource and consequently lets the distributed agent start the operation.

Thus if an operation running on a distributed agent uses a special resource, the following occurs:

- When the resource is available, the controller sets the state of the operation to started and the extended status to waiting for submission.
- The controller sends a release-dependency event to the distributed agent.
- The distributed agent starts the operation.

If the connection between the controller and the distributed agent breaks, the operation does not start on the distributed agent even if the resource becomes available.

**Note:** When you monitor a job on a fault-tolerant agent by means of the IBM Tivoli Workload Scheduler interfaces, you will not be able to see the special resources used by the job. Instead, you see the job, OPCMASTER#GLOBAL_SPECIAL_RESOURCES. The dependency on OPCMASTER#GLOBAL_SPECIAL_RESOURCES is set by the controller for every operation that is dependent on special resources. This dummy job is always added to the Symphony file with a priority of zero, even if no
operations depend on a special resource. As a result, the status of the schedule OPCMASTER1GLOBAL is always STUCK and the message AWS22010069E is shown in the STDLIST. Refer to IBM Tivoli Workload Scheduler Troubleshooting and Error Messages for an explanation of the message. When you monitor using the Tivoli Workload Scheduler for z/OS interfaces, you see the special resources as expected.

Example using data sets

The payroll application has many jobs that use the payroll database and must, therefore, not run together. You could let z/OS resolve the contention problem, using DISP=OLD in the JCL, but a job that waits in z/OS uses a JES initiator and other resources. This can reduce your batch throughput.

To prevent Tivoli Workload Scheduler for z/OS from scheduling or starting an operation that uses the payroll database when it is already being used, create a resource PAYROLL.DATABASE that represents the payroll database. Give each update job, such as PAYDAILY, exclusive control. Jobs that merely read the database, such as PAYQUERY, can have shared control.

In this case, the resource has a quantity of 1 (there is one database). Specify keep on error, because operators want to correct and resubmit a job without another job taking control of the database in the meantime.

When you extend the current plan, Tivoli Workload Scheduler for z/OS makes sure the jobs are not scheduled to run together or run at a time when the resources are unavailable. This is how the product makes use of the resource at the planning stage. When the product is ready to submit each job, it checks that the resource is available. This is how the product makes use of the resource at the control stage.

Specify the resource like this:

Name PAYROLL.DATABASE. Make sure that all operations specify this name exactly.

Quantity 1

Used for B (both planning and action).

On-error action Keep all.

Workstations Connect to all workstations that can use the data set; for Paymore, this is CPU1 and STC1.

Intervals The data set is always available.

Example using tape drives

Tape drives are usually owned by only one machine, but they can be used by a started task workstation and a computer workstation on the same machine, so you can, for example, allocate a tape pool to workstations CPU1 and STC1.

Do not keep this resource on error, because you normally want to release a tape drive for other work while you prepare a rerun of a failed job.

If you have 10 drives (quantity is 10), you do not need to allocate all 10 drives to both STC1 and CPU1. If CICS runs as a started task, you might need to reserve a tape drive for dumps of its journal, so give STC1 exclusive use of one tape drive in the intervals when CICS is available.
Example using tape drives

Specify the resource like this:

Name TAPES
Quantity 10 (for example). You need not make all the drives available for automatic allocation.
Used for B (both planning and action).
On-error action Free all.
Workstations Connect to all workstations that can use the tapes.
Intervals Reduce the number available when online systems might need a tape, and set the number to zero when the tape room is unstaffed.

Example using communication lines

Lines are often shared between processors, and can even be shared between operations. Lines are never allocated by z/OS, because they are owned by a communication controller, but you might want to ration the number of file transfer jobs, for example. If you have a number of lines from New York to London, with a total capacity of 256 kbaud, you can specify a quantity of 256. If you give a file transfer operation exclusive use of 20 units, for example, that gives the lines a limit of 12 file transfers.

You can protect online and voice systems that use the same lines by giving them a shared allocation of, for example, 50 units each. Because they share units, they do not compete with each other; you can have an unlimited number of operations each sharing 50 units, but this limits the quantity available for file transfers to 206.

Specify the resource like this:

Name LINES.TO.LONDON
Quantity 256
Used for B (both planning and action).
On-error action Free all.
Workstations Connect to all workstations that can use the lines.
Intervals Reduce the number available at peak hours, which will keep more transfer jobs out and improve the performance of online systems.

How are special resources used?

Tivoli Workload Scheduler for z/OS just keeps a record of the state of each resource and its allocation. The product does not know that PAYROLL.DATABASE is a database, and it is unaware that TAPES resources are tape drives. Only you know this, and you have the responsibility of making sure that the product knows the true availability of the objects that the resources represent.

VSAM will not tell Tivoli Workload Scheduler for z/OS when the payroll database is opened, and z/OS is not going to tell Tivoli Workload Scheduler for z/OS when you vary a tape drive offline. This is your responsibility.

The best way to tell Tivoli Workload Scheduler for z/OS about changes in special resources is through the RODM interface. System components such as Automated Operations Control (AOC) inform RODM about changes to their resources. You can
How are special resources used?

subscribe to RODM updates by setting the RODMTASK keyword on the OPCOPTS initialization statement, and using the RODMOPTS initialization statement. Refer to Customization and Tuning for details of initialization statements.

If you do not have RODM installed, and for resources that RODM does not know about, you can automatically notify Tivoli Workload Scheduler for z/OS about changes in resources by intercepting messages, for example, and issuing the Tivoli Workload Scheduler for z/OS SRSTAT command, or calling the EQQUSIN subroutine. If you cannot automatically notify Tivoli Workload Scheduler for z/OS of a change in resource status, use the Special Resource Monitor.

Updating the special resource database

Each resource is created with this structure:

Order (priority) in which the scheduler uses the data:

<table>
<thead>
<tr>
<th>Where?</th>
<th>CP</th>
<th>RD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overriding (global) Quantity Availability Deviation</td>
<td>x</td>
<td></td>
</tr>
<tr>
<td>Interval: Quantity Availability Connected workstations</td>
<td>x x</td>
<td></td>
</tr>
<tr>
<td>Interval: Quantity Availability Connected workstations</td>
<td>x x</td>
<td></td>
</tr>
<tr>
<td>Interval: Quantity Availability Connected workstations</td>
<td>x x</td>
<td></td>
</tr>
<tr>
<td>... (there can be many intervals)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Defaults: Quantity Availability Connected workstations</td>
<td>x x</td>
<td></td>
</tr>
<tr>
<td>Other header data</td>
<td>/SV040000</td>
<td></td>
</tr>
</tbody>
</table>

The interval and default (header) data is maintained in the resource database (RD) and copied to the current plan (CP), where you can change it with the MCP panel Resource Monitor. The overriding (global) fields are present only in the current plan.

The header has default values that are always used to fill in any missing values in the intervals. For example, suppose that you have created a special resource TAPES with these default values:

<table>
<thead>
<tr>
<th>Quantity</th>
<th>Availability</th>
<th>Connected workstations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defaults</td>
<td>8 Y CPU1 STC1</td>
<td></td>
</tr>
</tbody>
</table>

and you specify intervals with some values specified (shown in bold). The missing values are completed from the default (shaded) values:

<table>
<thead>
<tr>
<th>Day and time</th>
<th>Quantity</th>
<th>Availability</th>
<th>Connected workstations</th>
</tr>
</thead>
<tbody>
<tr>
<td>STANDARD 08.00 to 22.00</td>
<td>6</td>
<td>Y</td>
<td>CPU1 STC1</td>
</tr>
<tr>
<td>SATURDAY 00.00 to 24.00</td>
<td>8</td>
<td>Y</td>
<td>CPU1</td>
</tr>
<tr>
<td>SUNDAY 08.00 to 10.00</td>
<td>8</td>
<td>N</td>
<td>CPU1 STC1</td>
</tr>
</tbody>
</table>

STANDARD specifies the days of the week that are not otherwise specified (in this case, Monday to Friday). Days and times not covered by the intervals (such as
Updating the special resource database

Monday at 01.00 and Sunday at 01.00) take the default values of 8 drives available on the CPU1 and STC1 workstations. The STC1 workstation cannot use the TAPES resource on Saturdays. No workstation can use the resource (it is unavailable) between 08.00 and 10.00 on Sundays.

Do not confuse the default availability and quantity with the overriding (or global) values, which exist only in the current plan. When you alter the quantity with the SRSTAT command, for example, you always change the overriding quantity. If you use the MCP panel, you can change both default and overriding values.

Creating special resources

Operations hold and release special resources automatically, according to their descriptions in the current plan, and the resources are available and connected to workstations, also as scheduled in the current plan. You specify how operations use special resources when you create the operation (see “Standard applications and group definitions” on page 119). But first you must create the resource, its attributes, what workstations it is connected to, and the number available in each interval. That is described in this chapter.

Resources represent something, such as tape drives, and the current plan is built assuming that so many tape drives will be available. If one breaks down, Tivoli Workload Scheduler for z/OS continues to allocate the broken tape drive to a job that needs one. The job will wait, because z/OS knows that the tape drive is offline. Because availability can be affected in this way, there are ways for programs to automatically change resource status, and for operators to manually change resource status:

1. A program that detects a change in status can call the EQQUSIN subroutine to set an availability or a deviation (an alteration to the planned quantity). For example, if a program detects that online response times are poor, it can set a deviation of -40 for the resource representing lines for file transfer use.
2. An operator can use the SPECIAL RESOURCE MONITOR panel to set a resource unavailable. Refer to “Using the special resource monitor” on page 610 for details.
3. A program or an operator can issue the SRSTAT command, either directly from TSO or as input to the EQQEVPGM program.
4. If you have RODM, and Tivoli Workload Scheduler for z/OS subscribes to it, Tivoli Workload Scheduler for z/OS can keep in step with the system automatically for the resources defined to RODM.

All these methods can change the availability, the deviation, and the quantity of a resource from that specified in the planned availability intervals.

Table 8 shows how the planned availability of some resource such as tape drives is affected by unplanned events such as input from the Special Resource Monitor, the SRSTAT command, the EQQUSIN subroutine, and RODM. Manually altered overriding quantity, availability, and deviation values are honored across an interval boundary and batch daily planning EXTEND and REPLAN jobs. To make Tivoli Workload Scheduler for z/OS revert to a scheduled value after a manual alteration, you must reset the value, as at 11.20.
Creating special resources

Table 8. How resource quantity and availability can be changed

<table>
<thead>
<tr>
<th>Start of interval / time of event</th>
<th>Planned quantity</th>
<th>Planned availability</th>
<th>Actual quantity</th>
<th>Actual availability</th>
<th>Deviation</th>
<th>Number available</th>
</tr>
</thead>
<tbody>
<tr>
<td>08.00</td>
<td>8</td>
<td>N</td>
<td>8</td>
<td>N</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>08.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>8</td>
</tr>
<tr>
<td>09.00</td>
<td>8</td>
<td>N</td>
<td>8</td>
<td>Y</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>09.40</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>7</td>
</tr>
<tr>
<td>09.41</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>6</td>
</tr>
<tr>
<td>09.42</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>7</td>
</tr>
<tr>
<td>10.00</td>
<td>9</td>
<td>Y</td>
<td>9</td>
<td>Y</td>
<td>-1</td>
<td>8</td>
</tr>
<tr>
<td>10.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>5</td>
</tr>
<tr>
<td>11.00</td>
<td>8</td>
<td>Y</td>
<td>6</td>
<td>Y</td>
<td>-1</td>
<td>5</td>
</tr>
<tr>
<td>11.20</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-1</td>
<td>7</td>
</tr>
</tbody>
</table>

The number available (the last column) is the actual number available for allocation, taking into account the actual quantity, the deviation, and the actual availability.

The three events starting at 09.40 show the difference between altering the deviation with SRSTAT (or a subroutine) and with the Special Resource Monitor. SRSTAT adds the specified deviation to the current deviation, but the panel replaces the current deviation with the value you specify.

If you change values other than the overriding (global) quantity, availability, and deviation, or the values for an interval, you lose the changes at the next daily planning run, but the job issues a warning message about any manually changed values that will be lost. For example, if you change the default quantity (the quantity used where intervals are not specified) in the current plan, this is replaced at the next daily planning run with the value from the database.

Follow these steps to create the TAPES resource:

1. Select option 6 from the MAINTAINING TWSz DATA BASES menu [Figure 45 on page 89].
Creating special resources

2. Select option 3 (LIST) on the MAINTAINING SPECIAL RESOURCES menu (Figure 46). You can select option 2 (CREATE) instead, but option 3 gives you a chance to see the resources that already exist.

3. In the LIST OF SPECIAL RESOURCES panel (Figure 47), enter the CREATE command.

The CREATING A SPECIAL RESOURCE panel is displayed:

---

Select option 3 (LIST) on the MAINTAINING SPECIAL RESOURCES menu.

When you select 3 (LIST), you see the SPECIFYING SPECIAL RESOURCE LIST CRITERIA panel, where you can filter the resources shown in the list. To list all resources, enter * (asterisk) in the SPECIAL RESOURCE and SPECRES GROUP ID fields.

In the LIST OF SPECIAL RESOURCES panel, enter the CREATE command.

---
Creating special resources

4. Type values in the fields on the CREATING A SPECIAL RESOURCE panel:

**SPECIAL RESOURCE**
The name of the resource, up to 44 characters. The name of the special resource is translated to uppercase. You can include national characters in the name, but you are recommended not to include % and *, because Tivoli Workload Scheduler for z/OS uses these for filtering and searching in the panels. It is also good practice not to use the comparison operators: greater-than symbol (>), less-than symbol (<), caret (^), equals sign (=), or blank spaces. These might be used in search arguments passed by programming interface programs.

**TEXT**
A description of the resource, up to 46 characters.

**SPECRES GROUP ID**
The resource group, up to 8 characters. The group ID is for selecting subsets of resources in the panel (a list filter).

**HIPERBATCH**
Whether the resource represents a Data Lookaside Facility (DLF) object, Y or N. See “Hiperbatch and the Data Lookaside Facility” on page 100.

**USED FOR**
Whether the resource is used for:
- P Planning, when the current plan is extended
- C Control, when an operation starts
- B Both planning and control
- N Neither planning nor control

**ON ERROR**
What happens if an operation that allocates this resource ends in error (and does not have an overriding keep-on-error specification in the operation definition):
Creating special resources

F  Free the full allocation of this resource, both those allocated exclusive and those allocated shared
FS  Free the full shared allocation of this resource
FX  Free the full exclusive allocation of this resource
K  Keep the full allocation of this resource
Blank  Use the value specified in the ONERROR keyword of the RESOPTS statement. For details, refer to Customization and Tuning.

You might want critical jobs to keep their resources even when they fail, so that there is no delay waiting for resources when they are restarted.

The QUANTITY and AVAILABLE values, at the bottom of the panel, apply to intervals where a quantity or availability is not specified, and apply also to time ranges where there is no interval specified. You can save time by specifying the normal quantity and availability here, and specifying only the exceptions in intervals.

ON COMPLETE
The value to which the global availability is reset after the operation that uses the resource completes. It can be one of the following:
Y  Sets the global availability to Yes.
N  Sets the global availability to No.
R  Sets the global availability to blank.
Blank  Uses the system default, according to the following order:
a. The On Complete value set at operation definition level, if not blank.
b. The On Complete value set at special resource definition level, if not blank.
c. The ONCOMPLETE or DYNONCOMPLETE keyword value, respectively set for the not dynamically added resources or the dynamically added resources, in all the other cases.

MAX USAGE LIMIT
The number of allocations of this resource after which the resource global availability is changed to the value specified by Max Usage Type.

MAX USAGE TYPE
The value to which the global availability of the resource is reset, when its maximum usage limit is reached:
Y  Sets the global availability to Yes.
N  Sets the global availability to No.
R  Sets the global availability to blank.

QUANTITY
1 to 999 999.

AVAILABLE
Whether the resource is available, Y or N.

5. Enter option 2 on the command line to specify the default connected workstations. The MODIFYING CONNECTED WORK STATIONS FOR A SPECIAL RESOURCE panel, shown in Figure 49 on page 92, is displayed.
Creating special resources

When creating the resource, you see an asterisk (*) in the Ws column. This means that the resource is connected, by default, to all workstations. If you want to restrict the resource to named workstations, specify them as shown for TAPES in Figure 49.

Note: When an operation is switched to an alternate workstation, when the primary workstation is offline, the operation is still allowed to allocate the resource; the alternate workstation does not have to be in the list of connected workstations. Be sure to check that the resource is physically accessible from the alternate workstation.

6. Save the default connected workstations by pressing PF3 (End).
7. Enter option 1 on the command line of the CREATING A SPECIAL RESOURCE panel to create availability intervals. The MODIFYING INTERVALS FOR A SPECIAL RESOURCE panel, shown in Figure 50, is displayed.

8. Type values in the fields of the MODIFYING INTERVALS FOR A SPECIAL RESOURCE panel:

   Day of week or Date
   Specify a date in the format specified in the OPTIONS panel, or one of these:
   STANDARD (meaning default for the days not mentioned)
   MONDAY

---

**Figure 49. EQQDWMIL - Modifying connected work stations for a special resource**

When creating the resource, you see an asterisk (*) in the Ws column. This means that the resource is connected, by default, to all workstations. If you want to restrict the resource to named workstations, specify them as shown for TAPES in Figure 49.

**Figure 50. EQQDIMIL - Modifying intervals for a special resource**

When creating the resource, you see an asterisk (*) in the Ws column. This means that the resource is connected, by default, to all workstations. If you want to restrict the resource to named workstations, specify them as shown for TAPES in Figure 49.

---

When creating the resource, you see an asterisk (*) in the Ws column. This means that the resource is connected, by default, to all workstations. If you want to restrict the resource to named workstations, specify them as shown for TAPES in Figure 49.

**Figure 49. EQQDWMIL - Modifying connected work stations for a special resource**

When creating the resource, you see an asterisk (*) in the Ws column. This means that the resource is connected, by default, to all workstations. If you want to restrict the resource to named workstations, specify them as shown for TAPES in Figure 49.

**Figure 50. EQQDIMIL - Modifying intervals for a special resource**

When creating the resource, you see an asterisk (*) in the Ws column. This means that the resource is connected, by default, to all workstations. If you want to restrict the resource to named workstations, specify them as shown for TAPES in Figure 49.
Creating special resources

TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY

From time and To time
Specify a time range, with times in the format specified in the OPTIONS panel.

Qty The quantity of the resource in the time interval being specified. The default quantity and availability are those specified in step 4 on page 90.

A Available (Y) or unavailable (N).

Note: You cannot alter intervals that you have already modified in the current plan, the daily planning job never replaces a changed interval with values from the database.

9. Enter the S command beside the Saturday interval row to specify that STC1 operations cannot use the resource on that day. The MODIFYING CONNECTED WORK STATIONS FOR A SPECIAL RESOURCE panel, shown in Figure 51, is displayed.

10. Specify the CPU1 resource only. Press PF3 (End) to return to the MODIFYING INTERVALS FOR A SPECIAL RESOURCE panel.

11. When you have specified all the intervals, press PF3 (End) to return to the CREATING A SPECIAL RESOURCE panel.

12. Press PF3 (End) again to save the resource definition.

Table 9 shows the origin of each value in some intervals for the TAPES resource: whether from the default values, the STANDARD interval, or from a specific interval.

Table 9. Where values are taken from for each interval

<table>
<thead>
<tr>
<th>Time</th>
<th>Quantity</th>
<th>Availability</th>
<th>Workstations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monday 00.00 to 07.59</td>
<td>Default</td>
<td>Default</td>
<td>Default</td>
</tr>
<tr>
<td>Monday 08.00 to 22.00</td>
<td>Standard</td>
<td>Default</td>
<td>Default</td>
</tr>
<tr>
<td>Monday 22.01 to 24.00</td>
<td>Default</td>
<td>Default</td>
<td>Default</td>
</tr>
</tbody>
</table>
Creating special resources

Table 9. Where values are taken from for each interval (continued)

<table>
<thead>
<tr>
<th>Time</th>
<th>Quantity</th>
<th>Availability</th>
<th>Workstations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saturday</td>
<td>Default</td>
<td>Default</td>
<td>Interval</td>
</tr>
<tr>
<td>Sunday 00.00 to 07.59</td>
<td>Default</td>
<td>Default</td>
<td>Default</td>
</tr>
<tr>
<td>Sunday 08.00 to 10.00</td>
<td>Default</td>
<td>Interval</td>
<td>Default</td>
</tr>
<tr>
<td>Sunday 10.01 to 24.00</td>
<td>Default</td>
<td>Default</td>
<td>Default</td>
</tr>
</tbody>
</table>

Tivoli Workload Scheduler for z/OS uses, in order of priority:
1. A specific date and time, if specified
2. A specific day and time, if specified
3. The STANDARD entry
4. The default values

A more specific interval overrides the standard interval and the default values.

Setting the availability of a resource

If a resource is available at fixed times, use the RESOURCES panel to specify this. If the availability cannot be predicted, change the availability status with the SRSTAT command, entered either from TSO or from the batch program EQQEVPGM.

For example, if a resource represents a data set, you can make this resource available when the data set is created and loaded with valid data. If the data set is created and loaded by a TSO user, the user can set the availability status with the SRSTAT command immediately after the data set is loaded. If a batch job creates and loads the data set, add an extra step that executes EQQEVPGM to set the availability status to YES. Use condition codes to execute the EQQEVPGM step only if the create and load steps are successful.

When you set availability in this way, the daily planning program cannot know in advance when the resource is available, and therefore cannot predict accurate start times; it schedules operations according to the values in the database.

You can browse and modify the availability status of resources, and browse the allocation status, using the RESOURCES panel.

By default, Tivoli Workload Scheduler for z/OS starts an operation if a required resource is available, even if the operation will last one hour and the interval specifies that the resource will be unavailable after one minute. If you want the product to take the expected future availability and operation duration into account, use the LOOKAHEAD keyword of the RESOPTS initialization statement. The special resource must be used for control, for LOOKAHEAD to take effect. Refer to Customization and Tuning for details of the RESOPTS statement.

Using NetView to set the availability of a resource

Consider using resources to control the batch component of an online system. NetView can automatically generate an SRSTAT to make a resource unavailable if it is detected that the online system or the database has become unusable. This can prevent unnecessary abends in your batch processing if you specify all relevant operations as requiring shared access to the resource.
Using RODM

Using RODM to change the availability of a resource

If you have the Resource Object Data Manager (RODM) on your z/OS systems, you can subscribe to changes in resource availability using the RODMOPTS keyword on the OPCOPTS initialization statement. Refer to Customization and Tuning for details of the OPCOPTS statement. RODM can pass on to Tivoli Workload Scheduler for z/OS changes in resource status detected by such products as AOC.

Using On Complete to change the availability of a resource

Use the On Complete option to change the global availability of a special resource used by an operation that completes successfully. For example, to prevent other operations from using the special resource that is being used by the current operation, set the On Complete option to No. In this way, after the operation completes successfully, the special resource will no longer be available.

Suppose you plan to run an operation named JOBY every day. JOBY requires an input file FILEX that is updated daily by an external process. Represent FILEX as a special resource and define JOBY as an operation using FILEX in exclusive way, quantity 1, with default availability set to No. Each time the external process updates FILEX, an event is generated to set the resource global availability to Yes. If, after the current occurrence of JOBY is run, you do not want the next occurrence to run on the same day, set the On Complete option to No. In this way, after JOBY completes successfully, the global availability of the special resource FILEX is set to No and the next occurrence is run only on the following day, when another event is generated to set the resource global availability to Yes again.

The On Complete option can be specified at three different levels:

- Operation definition. Possible values are:
  - Y  Sets the global availability of the special resource to Yes.
  - N  Sets the global availability of the special resource to No.
  - R  Sets the global availability of the special resource to blank.
  - Blank  Uses the system default.

- Special resource definition. Possible values are:
  - Y  Sets the global availability to Yes.
  - N  Sets the global availability to No.
  - R  Sets the global availability to blank.
  - Blank  Uses the system default.

- RESOPTS initialization statement (either the ONCOMPLETE or DYNONCOMPLETE keyword). Possible values are:
  - YES  Sets the global availability of the special resource to Yes.
  - NO  Sets the global availability of the special resource to No.
  - RESET  Sets the global availability of the special resource to blank.
  - NOCHANGE  Does not change the global availability. This is the default.

When the operation completes, the On Complete value is applied according to the following order:
1. The On Complete value set at operation definition level, if not blank.
2. The On Complete value set at special resource definition level, if not blank.
Using On Complete

3. The ONCOMPLETE or DYNONCOMPLETE keyword set in the RESOPTS statement (respectively for resources added not dynamically or dynamically), in all the other cases.

The On Complete default value for special resources dynamically added is blank.

The On Complete option is valid only if the special resource is used for controlling purposes (the field Used For is set either to C or B).

Note: The On Complete action is performed any time the Complete status is set (either manually or as a consequence of a real run).

You can check if the global availability of a special resource was changed due to an On Complete action, in the Last Updated section of the BROWSING A SPECIAL RESOURCE and MODIFYING A SPECIAL RESOURCE panels.

Daily plan batch processes update the On Complete value:

In the current plan extension (CX) data set

With one of the following values, chosen according to this order:

1. The On Complete value specified in the RD database, if the special resource is defined in the RD database.
2. The On Complete value specified in the old CX data set, if the special resource exists in CX.
3. Blank, if the special resource is dynamically added.

In the current plan operation (CP) data set

With the Available On Complete value set in the application definition.

Using Max Usage Limit to change the availability of a resource

To change the global availability of a special resource after a specific number of operations have used it, independently of their successful completeness or actual run, use the Max Usage Limit option. For Max Usage Limit to be effective:

- Set the Max Usage Limit for the special resource to a value different from 0
- Set the Max Usage Type for the special resource to a meaningful value

When you set the Max Usage Limit, an internal usage counter is increased each time an operation allocates the resource. When the internal counter reaches the maximum usage limit, the global availability of the resource is reset to the value specified in the Max Usage Type field.

The usage counter is increased at each operation start and the resource availability is immediately reset if the maximum limit is reached. This means that the maximum usage limit is reached even if the operation fails or it is not submitted by the tracker, and the global availability of the resource is reset as soon as the operation is started.

For this reason, there are some cases where this option is not recommended. Consider, for example, that you are using WLM scheduling environment integration and your WLM job uses the special resource FILEX, whose maximum usage limit is 2 and maximum usage type is No. Suppose also that FILEX was already used once by another operation. When your WLM job is started and sent to the tracker, FILEX is allocated and the usage counter is increased to 2, reaching the maximum limit set. As a consequence, the FILEX global availability is immediately reset to No, preventing any operation from using it. If the scheduling
environment is not available, the WLM job is set back to Ready, waiting for scheduling environment availability and the special resource is deallocated. Nevertheless, its global availability remains unchanged (not available), therefore the WLM job will not be able to start when the scheduling environment becomes available.

You can monitor the usage counter in the BROWSING A SPECIAL RESOURCE panel, but you cannot change the value calculated.

The Max Usage Limit value is valid only if the special resource is used for controlling purposes (the field Used For is set either to C or B).

For dynamically added special resources, the default values are:

- **Max Usage Limit**: 0, meaning that the function is not active
- **Max Usage Type**: Reset

If you change the maximum usage limit so that the new value is incongruent with the current usage counter, the following actions are taken:

- If the new maximum usage limit is 0 and the usage counter is nonzero, the usage counter is reset to 0. A message is issued in EQQMLOG.
- If the new maximum usage limit is nonzero, but is lower than the usage counter, the usage counter is reset to 0 and the resource availability is reset according to the Max Usage Type. A message is issued in EQQMLOG.

The same actions are taken any time the daily plan batches make an update that causes incongruence with the current usage counter. Note that daily plan batch actions might be changed by the reapplying of the events generated during daily plan batch run. For details about current plan turnover, see the Diagnosis Guide and Reference.

You can check if the global availability of a special resource was changed due to maximum usage limit reached, in the Last Updated section of the BROWSING A SPECIAL RESOURCE and MODIFYING A SPECIAL RESOURCE panels.

Daily plan batch processes update the Max Usage Limit and Max Usage Type in the current plan extension (CX) data set with the first values available, taken in this order:

1. The Max Usage Limit and Max Usage Type specified in the RD database, if the special resource is defined in the RD database.
2. The Max Usage Limit and Max Usage Type specified in the old CX data set, if the special resource exists in CX.
3. Max Usage Limit=0 and Max Usage Type=Reset, if the special resource is dynamically added.

### Using SRSTAT LIFESPAN to change the availability of a resource

To change the global availability of a special resource, you can use the SRSTAT command with the LIFESPAN parameter. The LIFESPAN parameter sets:

- The interval of time, expressed in minutes, after which the global availability will be changed.
- The value to which the global availability will be changed.

For detailed information about the SRSTAT command, see “SRSTAT” on page 664.
Using SRSTAT LIFESPAN

When you issue an SRSTAT LIFESPAN command, the controller stores the information that there is a pending LIFESPAN action for the special resource that will be performed after the specified interval of time expires. You can check whether a pending LIFESPAN action exists by looking at the MODIFYING A SPECIAL RESOURCE and BROWSING A SPECIAL RESOURCE panels.

Only one LIFESPAN action can exist for a resource. If you issue an SRSTAT LIFESPAN command for a resource that already has a pending LIFESPAN action, the old LIFESPAN action is replaced with the current one. To delete a pending LIFESPAN action, you can issue an SRSTAT with LIFESPAN interval of time set to 0.

Using event-triggered resource handling to change the availability of a resource

To set the global availability of a special resource, you can use an event-driven process that triggers special resource changes based on a configuration file that correlates system activities and required changes.

For detailed information, refer to “Data set triggering” on page 426.

Creating a resource dynamically

If an operation in one of your applications needs a resource but the resource it requires is not created, Tivoli Workload Scheduler for z/OS can create the resource, either when it creates the current plan or later.

Creating missing resources during batch planning

Use the DYNAMICADD keyword of the BATCHOPT initialization statement to specify whether Tivoli Workload Scheduler for z/OS should create resources that are not in the database when it creates or extends the current plan. Refer to Customization and Tuning for details of BATCHOPT.

Use the DYNAMICDEL keyword of the BATCHOPT initialization statement to specify whether Tivoli Workload Scheduler for z/OS should consider eligible for deletion those resources, which have been dynamically added, when it creates or extends the current plan. Refer to Customization and Tuning for details of BATCHOPT.

Creating missing resources during the life of the plan

Tivoli Workload Scheduler for z/OS distinguishes between these two cases:
1. An operation or event refers to a resource that is in the database but is not in the current plan. See “Dynamically copying resources from the database.”
2. An operation or event refers to a resource that is neither in the database nor in the current plan. See “Dynamically adding resources that are not in the database” on page 99.

Dynamically copying resources from the database

These are some cases where a resource is dynamically copied to the current plan:
- An SRSTAT command refers to a resource that is not in the plan. The details (including interval data) are copied to the current plan. Tivoli Workload Scheduler for z/OS ignores the CREATE keyword on the SRSTAT command, and the DYNAMICADD keyword on the RESOPTS initialization statement.
Dynamically copying resources from the database

- You add an occurrence to the current plan using the MCP panel. One of the operations in the occurrence references the resource. Tivoli Workload Scheduler for z/OS ignores the DYNAMICADD keyword on the RESOPTS initialization statement. The details (including interval data) are copied to the current plan. This happens as soon as the occurrence is added; Tivoli Workload Scheduler for z/OS does not wait until it starts the operation.

Tivoli Workload Scheduler for z/OS always creates a resource in the plan if it is in the database but was not added during daily planning. Resources are only added during daily planning if there is an operation in the plan that references the resource. When you refer to the resource during the life of the plan, Tivoli Workload Scheduler for z/OS adds the resource from the database.

Dynamically adding resources that are not in the database
These are some cases where a resource might be dynamically added to the current plan:
- An SRSTAT command refers to a resource that is not in the plan. If the resource is not in the database, Tivoli Workload Scheduler for z/OS looks at the CREATE keyword of SRSTAT (it must be YES) and the DYNAMICADD keyword of RESOPTS (it must be EVENT or YES) to decide whether to build an entry for the resource in the current plan.
- You add an occurrence to the current plan using the MCP panel. One of the operations in the occurrence references the resource. Tivoli Workload Scheduler for z/OS looks at the DYNAMICADD keyword of RESOPTS (it must be OPER or YES) to decide whether to build an entry for the resource in the current plan.

Refer to Customization and Tuning for details of RESOPTS.

Setting the global values
Values that you specify on an event (for example, on the SRSTAT command) update the overriding (or global) quantity, availability, and deviation, which are stored in the current plan separately from the default values from the database.

The batch daily planning process (REPLAN or EXTEND) never deletes resources from the current plan if you have set the overriding quantity, availability, or deviation, even though there might be no operations in the plan that reference the resource.

An SR is considered eligible for deletion if all of the following conditions are satisfied:
- The global quantity is not set.
- The global availability is not set.
- The deviation is not set.
- No intervals are modified.
- RODM Availability, Quantity, and Deviation are set to N.

To make a resource eligible for deletion at the next REPLAN or EXTEND, reset the manually set value. For example, consider this sequence:
1. The resource TAPES is not in the current plan.
2. You issue the command:
   \[ \text{SRSTAT 'TAPES' SUBSYS(OPC1) AVAIL(YES)} \]
3. Tivoli Workload Scheduler for z/OS adds TAPES to the current plan, taking the values from the resource database.
Setting the global values

4. Tivoli Workload Scheduler for z/OS sets the overriding (global) availability to YES. It leaves the overriding quantity and deviation fields blank, so that Tivoli Workload Scheduler for z/OS uses the quantity from the intervals (or default quantity).

5. You run a daily planning REPLAN or EXTEND. There are no operations in the plan that reference TAPES.

6. The TAPES resource remains in the plan, because you have set its availability.

7. You issue the command:
   SRTSTAT 'TAPES' SUBSYS(OPC1) AVAIL(RESET)

8. Tivoli Workload Scheduler for z/OS resets the availability of TAPES to its default value.

9. You run a daily planning REPLAN or EXTEND. There are no operations in the plan that reference TAPES.

10. The TAPES resource is deleted from the plan.

Note: The previously-described process applies to dynamically-added SRs only when a daily planning REPLAN or EXTEND is run with DYNAMICDEL(YES). In this instance, dynamically-added SRs are deleted, provided they satisfy the same conditions to be considered eligible for deletion.

Hiperbatch and the Data Lookaside Facility

Hiperbatch is a z/OS performance enhancement that works with the Data Lookaside Facility (DLF) to allow batch jobs and started tasks to share access to a data set, or data object. Tivoli Workload Scheduler for z/OS provides control information to DLF concerning which operations are allowed to connect to which DLF object, and which data sets are eligible for Hiperbatch.

Within Tivoli Workload Scheduler for z/OS, a data set that is eligible for Hiperbatch is treated as a resource. Using the RESOURCES panel, you can define data sets with the DLF attribute. The DLF exit sample, EQQDLFX, can then make the following decisions about DLF processing:

- Is this data set eligible for Hiperbatch?
- Should this operation be connected to this data object?

Tivoli Workload Scheduler for z/OS issues enqueues on the job and data set name to notify the DLF exit that the job to be scheduled will use Hiperbatch. When the job ends, Tivoli Workload Scheduler for z/OS checks if the same data set will be used by the immediate successor operation or by any other ready operation. If so Tivoli Workload Scheduler for z/OS does not purge the data object. Otherwise, Tivoli Workload Scheduler for z/OS initiates purge processing of the data object (that is, Tivoli Workload Scheduler for z/OS removes it from Hiperspace™). For details about installing Tivoli Workload Scheduler for z/OS Hiperbatch support, refer to Customization and Tuning.

Note: The controller can create DLF objects on any system in the controller’s global resource serialization (GRS) ring, but operations that need to connect to the object must run on the same system as the controller.
Reporting on special resources

When you run the daily planning job (EXTEND or REPLAN), Tivoli Workload Scheduler for z/OS reports on special resources that are included in the new plan, but limits the report to resources specified on RESOURCE statements. Refer to Customization and Tuning for information on the RESOURCE statement, which you include in the same member as your BATCHOPT statement.

You can cross-reference special resources (for example, listing the operations that reference each special resource) by selecting option 6 (XRF OF ITEMS) from the AD panel print menu, or option 1.4.4.6 from the main menu.

Using availability changes for workload automation

Based on system activities that occur at tracker side, you can:
- Start an operation automatically when some resource becomes available.
- Define special resource dependencies to make a regularly scheduled job (one which is already in the current plan) dependent on the activity that affects a data set. For example, you can make a job wait for a data set to be created before it starts.

For details, refer to Chapter 21, “Running event-driven workload automation,” on page 425.
Using availability changes for workload automation
Chapter 6. Creating calendars and periods

This chapter shows how you create your installation calendar, create period definitions, and how these work together with run cycles to specify when work will run. Run cycles are described in detail in "Creating run cycles" on page 124, where you specify these as part of the application definition.

When you run work manually without Tivoli Workload Scheduler for z/OS, you submit jobs according to a calendar. Some jobs run every day, some run every working day, some run on a fixed date each month, and so on. There are jobs that run at the end of each tax year, which varies from country to country. If Tivoli Workload Scheduler for z/OS is to be able to automate this, you must specify when jobs are to run. To achieve this, Tivoli Workload Scheduler for z/OS has three important objects:

Calendars
The calendar specifies normal working days and public holidays. Tivoli Workload Scheduler for z/OS uses the calendar to determine when applications are scheduled, and to calculate dates for JCL tailoring.

You can specify the calendar when you create an application. If no calendar is specified for the application, Tivoli Workload Scheduler for z/OS uses the calendar in the CALENDAR keyword of the BATCHOPT initialization statement, for batch services such as extending the long-term plan, or the calendar specified under the OPTIONS panel (0.2 from the Tivoli Workload Scheduler for z/OS main menu), for online services such as testing a rule with GENDAYS. If no calendar is specified, a calendar with the name DEFAULT will be used. If the DEFAULT calendar does not exist, all days are considered work days. You might have several calendars, but always call your default calendar DEFAULT, and specify the same calendar name on BATCHOPT and in the panel.

A calendar is created only if it contains at least one work day.

Periods
Periods are either cyclic, such as a week or 28-day period, or noncyclic, such as an academic semester. Cyclic periods are defined by their origin date and their length, and noncyclic periods by the origin date of each interval. Noncyclic periods can optionally have an end date for each interval.

Run cycles
When you create an application, you specify when it should run using a run cycle, which has one of two forms:

1. A rule with a format such as The SECOND TUESDAY of every MONTH, where the words in capitals are selected from lists of ordinal numbers, types of day, and common calendar intervals or period names, respectively.

2. A combination of period and offset. For example, an offset of 1 in a weekly period specifies Monday. An offset of 10 in a monthly period specifies the tenth day of each month.

Run cycles generate either positive or negative occurrences in the long-term plan. A negative occurrence always cancels any matching positive occurrences. You can specify an negative occurrence only if the positive equivalent already exists. You use negative occurrences to identify the days when an application
Creating calendars and periods

would normally be scheduled but is not required. For example, when you normally schedule the application every Friday but not if Friday is also the last day of the month.

The run cycle type identifies whether positive or negative occurrences will be generated. You can specify many run cycles, positive and negative, when you create an application. You can mix run cycles that use offsets and run cycles that use rules in the same application.

The long-term planning process uses the calendar information, the period definitions, and the run cycle, to determine the days on which an application should be scheduled. The daily planning process does not use the calendar and period definitions—Tivoli Workload Scheduler for z/OS uses the input arrival date and time in the long-term plan when it extends the current plan to include a new application occurrence. This does not determine when the operations run, because this depends on other factors such as predecessor completion, resource availability, deadline time, and priority.

Creating the default calendar

A calendar in Tivoli Workload Scheduler for z/OS defines the status of each day and the workday end time. Tivoli Workload Scheduler for z/OS uses the calendar when it creates the long-term plan. When it creates the current plan, and when it decides whether to submit work, it does not refer to the calendar, so you cannot use the calendar to close down a system at short notice. Instead, use the MODIFY CURRENT PLAN panel to change the workstation open intervals as described in “Changing workstation availability” on page 584.

To create the default calendar:
1. Select option 1.2.2 from the Tivoli Workload Scheduler for z/OS main menu.
2. You see a list of calendars, if any exist. You can modify an existing calendar or to create a new calendar, enter CREATE at the command prompt. The CREATING A CALENDAR panel (Figure 52) is displayed.

To create the default calendar:

3. Type DEFAULT for the calendar ID.

Figure 52. EQQTCAL - Creating a calendar

104 IBM Tivoli Workload Scheduler for z/OS: Managing the Workload
4. Enter W for the status of the days of the week that are work days, and F for the non-work days and holidays. A day is specified as either:

<table>
<thead>
<tr>
<th>Status</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Work day</td>
<td>When Tivoli Workload Scheduler for z/OS schedules work as normal</td>
</tr>
<tr>
<td>Free day</td>
<td>When Tivoli Workload Scheduler for z/OS plans work according to the free-day rule on each application run cycle. See “Selecting a free-day rule” on page 130 for detailed information about the free-day rule.</td>
</tr>
</tbody>
</table>

The status specified for a particular date overrides the specification for the corresponding day of the week.

5. Specify the workday end time. If your installation shuts down, it probably does not do so at midnight. Normally, a system shuts down at the end of a shift, for example 6 a.m. This could be considered as the workday end time for your installation: the time that a normal work day ends. The workday end time is the time of day that a work day ends and a free day begins, and is the time of day that a free day ends and a work day begins, but be careful if you specify a time before midnight: the work day after a free day starts at the first workday end time on the work day. If the workday end time is after midnight, the workday end time determines when work is planned to run. If the workday end time is before midnight on the free day, Tivoli Workload Scheduler for z/OS waits for the next workday end time: the one on the work day after the free day, which is probably not what you expect, so it is recommended that you specify a workday end time at 00.00 or a later time.

Note: Specify all times of day in 24-hour notation: never use a.m. and p.m.

**Example 1: the workday end time is 02.00, and Saturday and Sunday are free days**

- The long-term plan (LTP) has work until 02.00 Saturday morning.
- There is no work in the long-term plan starting between 02.00 Saturday morning and 01.59 Monday morning, unless the free-day rule in the run cycle specifies that the application can run on a free day.
- Work for Monday begins at 02.00 Monday morning.
- Run cycles that specify an input arrival time between 00.00 and 01.59 generate an occurrence on the following day. For example, the rule Every Monday in the Year generates occurrences early on every Tuesday morning (taking the free-day rule into account if Monday is a free day).

**Example 2: the workday end time is near midnight**

In this example, Sunday is a free day and Monday is a work day. Assume that the free-day rule in the run cycle specifies that no applications are run on free days.

<table>
<thead>
<tr>
<th>End time</th>
<th>Effect on scheduling</th>
</tr>
</thead>
<tbody>
<tr>
<td>00.00</td>
<td>No applications will be scheduled to run on a Sunday. Applications scheduled to run on Monday will start just after midnight between Sunday and Monday.</td>
</tr>
<tr>
<td>00.01</td>
<td>The Sunday free day runs from 00.01 on Sunday to 00.00 (midnight between Sunday and Monday) inclusive. If you specify a day with an input arrival time of 00.00 on the run cycle, Tivoli Workload Scheduler for z/OS assumes that you mean midnight between that day and the next day—times before the workday end time come at the end of the work day.</td>
</tr>
<tr>
<td>23.59</td>
<td>This work day end time is not recommended. The free day,</td>
</tr>
</tbody>
</table>
Creating the default calendar

Sunday, starts at 23.59 on Sunday and ends at 23.59 on Monday. The following work day, Monday, starts at 23.59 on Monday and lasts for 1 minute. In other words, almost no applications would be scheduled for Monday.

24.00

This work day end time is not recommended. The work day, Monday, would be lost because the start of the day would be midnight between Monday and Tuesday.

The workday end time has no effect on the daily planning process; when Tivoli Workload Scheduler for z/OS creates the current plan, it is the workstation open hours that determine if work will start within a particular period. If you do not want Tivoli Workload Scheduler for z/OS to plan or start work on a free day, close all workstations. If workstations are open on a free day, operations that have not finished processing by the workday end time will continue to process on the free day.

When you use an extra calendar, specify the calendar name for the application, job, or group definitions that do not use the default calendar.

When you modify the current plan using the ETT, ARC, or PIF, the subsystem has no access to the OPTIONS panel calendar name, or to the calendar specified in the BATCHOPT initialization statement. Unless you explicitly specify a calendar in the application description, the subsystem tries to use the calendar called DEFAULT. It is strongly recommended that you use the name DEFAULT for your normal calendar.

Creating periods

Periods can be cyclic or noncyclic. A cyclic period starts on a specific date and has a specified number of days. There are two kinds of cyclic periods; work-days-only cyclic periods where only work days are counted, and all-days cyclic periods where all the days are counted.

A noncyclic period such as an academic semester, has a varying interval, so you must specify the origin date of each interval.

If you run work at fixed days of the week, month, or year, and take one of the standard Tivoli Workload Scheduler for z/OS actions when this day falls on a free day (holiday), you do not need to create your own periods. You can describe most cases with rules such as:

- First Sunday in June
- First work day in the week
- Last Friday in the year
- Last free day in the month

If you do need to create your own periods, either for use in rules or in the older (offset-based) type of run cycle definition, follow this procedure:

1. Create periods using the PERIOD panel. From the main menu, select option 1.3, to display the MAINTAINING THE PERIODS menu.
2. Select option 2 to display the LIST OF CALENDAR PERIODS panel:
Creating periods

3. To create a new period, enter CREATE at the command prompt. The following panel is displayed:

```
Figure 53. EQQTPERL - List of calendar periods

3. To create a new period, enter CREATE at the command prompt. The following panel is displayed:

4. Fill in the fields in the CREATING A CALENDAR PERIOD panel:

   PERIOD NAME
   The period name can be up to 8 characters.

   PERIOD TYPE
   There are three types:
   A  All-days cyclic. Tivoli Workload Scheduler for z/OS counts both work days and free days when calculating period dates.
   W  Work-days-only cyclic. Tivoli Workload Scheduler for z/OS counts only work days when calculating period dates. However, it is still possible for an offset-based run cycle to select a free day. See “Using work-days-only cyclic periods” on page 111.
   N  Noncyclic.
```
Creating periods

For example, if the calendar you are using specifies that 25 December 2003 is a free day, as are every Saturday and Sunday, and you create a type \( \mathbb{W} \) period with an origin of December 1 and a length of 10 days, cyclic intervals will begin on these dates:

- 1 December, 2003
- 15 December, 2003
- 30 December, 2003
- 13 January, 2004, and every 10 work days after that

If you create a run cycle for an application that references this period with an offset of 1, the application will be scheduled on these dates. Because 25 December is a free day, it is not counted when Tivoli Workload Scheduler for z/OS calculates the run dates of the period.

**DESCRIPTION**
Type a description of the period.

**INTERVAL**
For cyclic periods, type the length of the period. For noncyclic periods, leave blank.

**VARIABLE TABLE**
If you use an offset-based run cycle, and do not specify an overriding table name on the run cycle, Tivoli Workload Scheduler for z/OS uses the JCL variable table that you specify here. If you use this period with a rule-based run cycle, Tivoli Workload Scheduler for z/OS ignores any table name specified here. See [Chapter 22, “Job tailoring,” on page 441](#) for a full description of job tailoring and variable substitution.

You can override the variable table, specified here or on the run cycle, using the MODIFY CURRENT PLAN (MCP) panel, the LONG TERM PLAN panel, or by using Tivoli Workload Scheduler for z/OS statements in the job.

**Interval origin**
For cyclic periods, specify the date of the start of the first interval in the period. For noncyclic periods, specify the start of every interval for, say, the next year. Remember to update the period definition and add more interval dates each year: Tivoli Workload Scheduler for z/OS issues a warning message when you extend or modify the long-term plan if you have not created intervals beyond the end of the long-term plan.

**Interval end**
For cyclic periods, leave this blank. For noncyclic periods, specify the end of every interval; the intervals must not overlap. If you leave this blank for noncyclic periods, Tivoli Workload Scheduler for z/OS assumes that the interval ends the day before the next interval.

When a run cycle specifies a negative offset in the period, or a rule includes the LAST specification, the interval end date is used to base the calculation. If interval end is not specified, Tivoli Workload Scheduler for z/OS assumes that the interval ends the day before the next interval begins. However, if you have only one interval origin specified and no interval end, a negative offset will be calculated from the day before the interval origin date.

When interval end is specified and a run cycle rule, or period and offset definition, results in a date outside the interval, an occurrence will not be generated. For example, a noncyclic period MYJAN specifies the start and end dates of January in every year. If you create a rule
that specifies the 5th Friday in MYJAN and there are not 5 Fridays, the occurrence will not be generated in February. Similarly, if you specify offset 25 excluding free days and there are not 25 work days in a January, the occurrence will not be generated sometime in early February.

Notes:
1. If the free day rule causes an occurrence to be moved beyond interval end, or prior to interval start, the long-term planning process issues a warning message. The occurrence will be scheduled by Tivoli Workload Scheduler for z/OS outside the specified interval.
2. From earlier versions of Tivoli Workload Scheduler for z/OS, work-days-only cyclic periods are handled differently. If you have defined the interval origin on a free day for a work-days-only cyclic period, a run cycle using that period will not generate the same dates as it did in previous releases of Tivoli Workload Scheduler for z/OS. To obtain the run dates you are used to in the long-term plan, move the interval origin to the nearest work day before the old interval origin. See "Using work-days-only cyclic periods" on page 111 for more information.
3. If you use a rule-based run cycle with a user-defined noncyclic period and do not specify an explicit end date for the last interval defined, GENDAYS and LTP batch treat the last specified origin date as an end date. Therefore, no occurrences are generated on or after the last specified origin date.
4. If a noncyclic period is defined with more than one interval, that is, with more than one starting date, then to specify offsets you can count days from the starting of each interval and generate dates for the occurrences in the long-term plan.

Examples of periods
If you use rules with their built-in calendar cycles (days of the week, months of the year, and so on), you probably need to create only special noncyclic periods, such as university semesters and tax years. This section illustrates the concept of periods with some examples.

Cyclic periods
Examples of cyclic periods are a day, a week, and a fortnight, with fixed intervals of 1 day, 7 days, and 14 days, respectively. An academic semester cannot be described as a cyclic period, because spring, summer, and fall semesters have different lengths.

This example shows a lunar calendar month, assumed to be 28 days:
Cyclic periods

**Cyclic period: work days and free days**

<table>
<thead>
<tr>
<th>Period Name</th>
<th>Period Type</th>
<th>Interval Origin</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>MOON</td>
<td>A</td>
<td>7 February 2003</td>
<td>28 days</td>
</tr>
</tbody>
</table>

This example shows a period that is useful if you need to take a backup every 4 work days:

**Cyclic period: work days only**

<table>
<thead>
<tr>
<th>Period Name</th>
<th>Period Type</th>
<th>Interval Origin</th>
<th>Interval</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKUP</td>
<td>W</td>
<td>2 January 2003</td>
<td>4 work days</td>
</tr>
</tbody>
</table>

Noncyclic periods

Examples of noncyclic periods are a quarter, and a payroll period. You specify the start of each interval of a noncyclic period with an origin date.

This example shows a period for university semesters, with the interval origin and end specified for each semester:

**Noncyclic period**

<table>
<thead>
<tr>
<th>Period Name</th>
<th>Period Type</th>
<th>Interval Origin</th>
<th>Interval End</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester</td>
<td>N</td>
<td>26 August 2002</td>
<td>13 December 2002</td>
</tr>
<tr>
<td></td>
<td></td>
<td>13 January 2003</td>
<td>16 May 2003</td>
</tr>
<tr>
<td></td>
<td></td>
<td>9 June 2003</td>
<td>28 June 2003</td>
</tr>
</tbody>
</table>

Noncyclic periods have a once-a-year maintenance overhead when you must create the intervals for the coming months. For this reason, carefully consider how flexible your period definitions are, and remove potentially duplicate definitions.

**How run cycles use periods**

To have Tivoli Workload Scheduler for z/OS automatically schedule an operation, create a run cycle in the application, job, or group definition.

When you create run cycles using the SEMESTER period, for example, you can use either rules or offsets:

**Run cycles that use rules**

First Day in Semester
Last Sunday in Semester
Seventh Sunday in Semester (This may result in an error)
Last Day in Semester

These are the equivalent selections using offsets:
How run cycles use periods

<table>
<thead>
<tr>
<th>Run cycles that use offsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester, offset 1</td>
</tr>
<tr>
<td>Semester, offset 105</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Semester, offset 38</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Semester, offset -1</td>
</tr>
</tbody>
</table>

Notice that using offsets is more difficult and makes run cycles harder to maintain.

Notes:
1. The start (origin) of each period is offset 1: there is no offset 0.
2. You cannot specify rules or offsets that select a day outside the interval. If the free-day rule causes a selected day to be shifted outside the interval, however, Tivoli Workload Scheduler for z/OS schedules the occurrence and issues a warning message.
3. If you omit the end date from the interval, the interval extends to the beginning of the next interval, so an offset of -1 (Last Day in Semester) selects the day before the next semester.
4. If you add occurrences to the current plan using ETT (event-triggered tracking), and you want external dependencies to be resolved as though the added occurrence had a fixed input arrival time, create a noncyclic period ETTRCY1 with an interval origin of 71/12/31, 31 December 2071. Run cycles that specify this special period are never scheduled in the long-term plan, but dependencies are resolved using the input arrival time on the run cycle, instead of the actual input arrival time, which is the time when the occurrence is added. See “Adding occurrences by event-triggered tracking” on page 432 for more information.
5. If you add occurrences to the current plan using ETT (event-triggered tracking), and you want to set its deadline by adding an offset to the actual input arrival time, then create a noncyclic period ETTRCY2 with an interval origin of 71/12/31, 31 December 2071. Then, for your ETT application, create a run cycle that specifies this offset period: an input arrival time of 00.00 and, as the deadline, the day and time you want the offset to be added to the actual input arrival time. These run cycles are never scheduled in the long-term plan, but when the occurrence is added to the plan, the occurrence deadline is calculated as the actual input arrival time plus the run cycle deadline. See “Adding occurrences by event-triggered tracking” on page 432 for more information.

Using work-days-only cyclic periods

A work-days-only cyclic period (type W) has only work days. Figure 55 on page 112 shows a 5-day period MYWEEK that starts on a Monday.
As you can see, the period does not always start on the same day of the week, because of public holidays. If you want a fixed day of the week, use a rule such as “Every Monday in the year,” using the free-day rule to avoid free days.

**Difference between offset-based and rule-based cycles**

If you have a work-days-only cyclic period MYWEEK, for example, with a length of 5 days, and specify an offset greater than 1, Tivoli Workload Scheduler for z/OS can select different days depending on the type of run cycle. Figure 56 on page 112 shows how offset-based (top of diagram) and rule-based (bottom of diagram) run cycles treat an offset of 3. Day 1 in the period is a Thursday, and Saturdays and Sundays are free days.

When you specify offset 3 in an offset-based run cycle, the long-term planning program selects the third calendar day from the period origin and takes action depending on the free-day rule. See “Selecting a free-day rule” on page 130 for information about free-day rules.

This behavior differs from using the same period in the rule-based run cycle “Only 3rd day in MYWEEK,” which selects Monday. When you specify a rule-based run cycle using a work-days-only cyclic period, Tivoli Workload Scheduler for z/OS jumps over the free days, counting only the work days in the period.

**Greater consistency**

When you create a work-days-only cyclic period where the interval origin is a free day, the long-term planning program ignores this day and starts the first period on the first work day after this.
Figure 57 on page 113 shows an example of this, where the origin of the 5-day period is a Saturday, and Saturdays and Sundays are free days.

Greater consistency

Figure 57. Results when the work-days-only cyclic period origin is a free day
Chapter 7. Creating applications and groups

This chapter explains applications and groups, and shows you how to create and use them. It covers these topics:

- "Things you should consider before creating applications"
- "Standard applications and group definitions" on page 119
- "Creating job descriptions" on page 169
- "Listing applications" on page 174

Things you should consider before creating applications

This section provides general information to consider before creating applications:

- What are applications
- Application types
- Naming standards in applications
- Rules for creating applications
- Subdivision of applications
- Methods for creating applications

Detailed procedures for creating applications are described, with the help of an example payroll application, in "Standard applications and group definitions" on page 119 and "Creating job descriptions" on page 169 (online panels), and in Chapter 8, "Defining applications in batch," on page 185.

What are applications?

The tasks that you want to control, such as running a job, issuing a WTO, or preparing JCL for a job, are called operations. An application is a group of related operations. The operations in each application are always run together: when an operation in the application is run, all other operations must also be run.

An application contains some or all of these parts:

<table>
<thead>
<tr>
<th>General data</th>
<th>Application name, description, and owner representative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run schedules</td>
<td>Period references, run time, and deadlines</td>
</tr>
<tr>
<td>Operation data</td>
<td>Job names, workstations, and resources</td>
</tr>
<tr>
<td>Dependencies</td>
<td>Between operations within the application and other applications</td>
</tr>
</tbody>
</table>

All applications contain general data. The requirement for other parts depends on your processing schedules and also on the type of application created. You can create one of three application types:

- Standard application descriptions
- Job descriptions
- Group definitions

Standard application descriptions

Use the APPLICATION DESCRIPTION panel to create and modify standard applications. Standard applications can contain more than one job.
Job descriptions

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

You can browse or modify job descriptions using both the JOB DESCRIPTION and the APPLICATION DESCRIPTION panels, but if you change a job description using the APPLICATION DESCRIPTION panel so that it no longer obeys the rules for job descriptions, it becomes a standard application description.

Standard applications and group definitions can be viewed or modified only with the APPLICATION DESCRIPTION panel.

To be a job description, an application description must satisfy the following conditions:

- It consists of no more than three operations.
  - One of the operations is a computer workstation operation, or a general workstation WTO operation. This operation is called the main operation of the description.
  - The other two operations, if present, are immediate predecessors of the main operation, and run on general workstations, either as manual operations or as JCL preparation operations.
- The job name of the main operation is the same as the job description ID and the same as its two general workstation predecessors, if present.
- The operation is ready to be included in the schedule: you cannot specify pending status for job descriptions.
- The application ID and owner ID should not be specified in double-byte character set (DBCS) bracketed format.

If you create an application that meets these rules, Tivoli Workload Scheduler for z/OS classes it as a job description, whether you create it with the APPLICATION DESCRIPTION panel, the JOB DESCRIPTION dialog box, the batch loader, or a program interface (PIF) application.

Like standard applications, you can specify a run cycle for the job description, or make it part of a group, in which case Tivoli Workload Scheduler for z/OS uses the calendar and run cycles associated with the group definition.

Group definitions
You can group applications and job descriptions that are always scheduled together by creating them as members of an application group. With an application group, the description of the application is recorded in a group definition, and not in individual applications. By doing this, you avoid having to specify the same calendar and run policy information for each application.

The use of application groups can save you time, in the initial specification of your work to Tivoli Workload Scheduler for z/OS and in ongoing maintenance to the applications. You can also use groups in the MODIFY CURRENT PLAN panel, to add, delete, and complete all or part of an application group in the current plan.
Having applications in a group also gives some protection against unintentional deletion or modification of individual group members in the plans. For example, to delete an application occurrence that is part of a group in the current plan, you must first remove it from the group.

**Naming standards in applications**

Naming standards are important when you are creating application descriptions. A sensible standard at the beginning of the application-definition process can save you time later.

The major factor that will influence your naming conventions is the ability to use generic names in Tivoli Workload Scheduler for z/OS panels and security products such as RACF®. Consider naming standards for these items:

- Application group IDs
- Application IDs
- Job names
- Operation numbers
- Owner IDs
- Authority group IDs

**Note:** Job descriptions are identified by their associated job or started-task procedure name.

**Application IDs**

Related applications should have the same application ID prefix. For example, all payroll applications could begin with P. If you use group definitions, you might consider using a unique prefix, like GRP, to identify all group definitions. The application ID field is 16 characters.

Avoid numbering application IDs. For example, HOUSEKEEP£1, HOUSEKEEP£2, and so on, because you might need a HOUSEKEEP£1.5.

**Job names**

Try to make each job name unique when you specify a job or started task. Ideally, the names of Tivoli Workload Scheduler for z/OS jobs should also be different from those of jobs not in Tivoli Workload Scheduler for z/OS.

Tivoli Workload Scheduler for z/OS imposes some restrictions on your choice of job names:

- Print operations must have the same job name as the job operations to which they refer.
- An operation on a job setup workstation must have the same job name as the succeeding computer workstation operation.

**Operation numbers**

Give each operation in an application a number in the range 1 to 255. Consider reserving number ranges for specific types of operations. For example, job setup operations could be from 1 to 9, computer processing operations from 10 to 240, and print operations from 241 to 255. When specifying operation numbers within these ranges, leave gaps for growth. For example, the first operation could be 05, the second operation 15, and so on.

**Owner IDs**

An owner ID must be specified for each application. The owner ID is a convenient label under which you can identify applications. This ID can then be used as a
search argument. For example, all Paymore applications have an owner ID of SAMPLE. The owner ID of SAMPLE can be used as selection criteria in the panels and reports.

**Rules for creating applications**

This section outlines the basic rules that govern standard applications and group definitions. The panels prevent you from breaking these rules.

A *group definition* is the central member in an *application group*. It groups related applications as a unit, holding common run cycles and calendar information for all applications in the group. The benefits are that you:

- Can perform modify actions on the group as a single unit within the plan instead of having to perform actions on many individual applications. This makes it simpler to add, delete, and complete many applications at once.
- Maintain only a single source for run cycle and calendar information.

Maintaining your application description database is simpler and less prone to error.

<table>
<thead>
<tr>
<th>Specifications</th>
<th>Standard application</th>
<th>Group definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>How many operations can it have?</td>
<td>From 1 to 255, each with an operation number in the range 001 to 255, all directly or indirectly linked to each other, and in such a way that they do not form a loop.</td>
<td>None directly, but it has applications, each of which can have 255 operations</td>
</tr>
<tr>
<td>How do you get Tivoli Workload Scheduler for z/OS to schedule it?</td>
<td>Specify the calendar name (if you are not using the default calendar) and the run cycles.</td>
<td>Specify the calendar name (if you are not using the default calendar) and the run cycles. The applications that belong to the group must not specify these.</td>
</tr>
</tbody>
</table>
| What fields must you specify? | • Application ID  
• Type, which must be A  
• Owner ID  
• Valid-from date  
• Status  
• Priority  
• At least one operation | • Application ID (the group ID)  
• Type, which must be G  
• Owner ID  
• Valid-from date  
• Status |

**Subdivision of applications**

If possible, divide large applications that might run over several days—or even months—into smaller applications that run daily. You can link these smaller applications with a network of *external dependencies* to provide a more meaningful picture of the overall run plan for an extended period. See “Specifying dependencies” on page 139.

Do not subdivide applications too finely. If you do, you will have to access many applications descriptions, instead of just a few, to make changes.

You can split large jobs consisting of many steps into smaller, interrelated jobs. This lets you use your installation resources more efficiently. For example, assume
that job steps A, B, C, and D are all in the same job to ensure that B, C, and D run after A. Tivoli Workload Scheduler for z/OS can schedule the steps in the correct order and let B, C, and D run at the same time. This lets more work run in parallel and can give you more spare time in your batch window.

Also consider using dummy workstations to serialize operations. See “Dummy workstations” on page 61 for more details.

**Methods of creating applications**

You can use any of these methods to create your application:

- **APPLICATION DESCRIPTION panel**
  
  You enter this panel by selecting the AD option from the MAINTAINING TWSz DATA BASES panel. See “Standard applications and group definitions.”

- **JOB DESCRIPTION panel**
  
  The JOB DESCRIPTION panel compresses most of the function of the APPLICATION DESCRIPTION panel into one panel by making some assumptions about the application that you are creating. See “Creating job descriptions” on page 169.

- **Batch loader**
  
  This program lets you enter your application descriptions in batch. See Chapter 8, “Defining applications in batch,” on page 185 for details.

- **Program interface**
  
  This standard interface lets you write your own programs to update information held by Tivoli Workload Scheduler for z/OS. For details, see Developer’s Guide: Driving Tivoli Workload Scheduler for z/OS.

- **LIST OF APPLICATIONS panel**
  
  If you are using the advanced panels (see “Panels Style” on page 41) in the LIST OF APPLICATIONS panel choose **Action > Create** (see Figure 92 on page 175).

**Standard applications and group definitions**

This section describes how you use the APPLICATION DESCRIPTIONS panel to create standard applications and group definitions.

It covers these topics:
- Creating an application and its operations
- Specifying when your application should be scheduled
- Creating operations
- Specifying operation details
- Associating job statements with operations
- Using print operations
- Using WTO operations
- Creating started-task operations
- Creating time-dependent operations

**Creating an application and its operations**

1. From the main menu, select option 1.4. You see the MAINTAINING APPLICATION DESCRIPTIONS panel, shown in Figure 58 on page 120.
Creating an application and its operations

2. Select option 2 to show the CREATING AN APPLICATION panel, shown in Figure 59.

   Note: If you are using the advanced panels (see “Panels Style” on page 41), select Create in the Action menu of the LIST OF APPLICATIONS panel (EQQNALSL).

3. Specify the Application ID (for details, see “Specifying the application ID” on page 121) and, optionally, specify a description of the application (up to 24 characters) in the TEXT field.

4. Specify A or G in the TYPE field and a group definition ID in the GROUP DEFINITION field as required, according to the table in Table 11.
Creating an application and its operations

Table 11. How to use the RUN and OPER commands and the GROUP DEFINITION field

<table>
<thead>
<tr>
<th>If you are creating</th>
<th>RUN command</th>
<th>OPER command</th>
<th>Group Definition field</th>
</tr>
</thead>
<tbody>
<tr>
<td>A group definition TYPE = G</td>
<td>Use the RUN command if you want to schedule the group definition.</td>
<td>Do not use the OPER command to specify operations: specify these when you create the applications in the group.</td>
<td>Leave blank. Specify the name of the group in the ID field (Application ID).</td>
</tr>
<tr>
<td>An application that is part of a group TYPE = A</td>
<td>Do not use the RUN command, because Tivoli Workload Scheduler for z/OS uses the run cycles that are part of the group definition.</td>
<td>Use the OPER command to specify the operations in the group.</td>
<td>Set to the name of the group that this application belongs to.</td>
</tr>
<tr>
<td>An application that stands alone TYPE = A</td>
<td>Use the RUN command if you want to schedule the application.</td>
<td>Use the OPER command to specify the operations in the application.</td>
<td>Leave blank.</td>
</tr>
</tbody>
</table>

For detailed information about run cycles and the RUN command, see “Specifying when your application should be scheduled” on page 123.

For detailed information about operations and the OPER command, see “Creating operations” on page 137.

5. Specify the name of the application owner (1-16 characters) in the Owner ID field and, optionally, a description of the application owner (up to 24 characters) in the Owner TEXT field.

6. For non-group definitions, specify the priority in the PRIORITY field, where:
   1    Low
   2-7  Medium
   8    High
   9    Urgent

7. Specify a VALID FROM date (for details, see “Specifying the valid-from date” on page 122).

8. Specify A or P in the STATUS field as required (for details, see “Specifying the status” on page 122).

9. Optionally, specify the name of the authority group for the application (1-8 characters, where the first character is alphabetic or national) in the AUTHORITY GROUP ID field.

10. Optionally, specify the name of a calendar (1-16 characters, where the first character is alphabetic or national) in the CALENDAR ID field. If a calendar name is not specified, the name DEFAULT is used. A calendar ID cannot be specified for an application in an application group.

11. Optionally, specify the SMOOTHING FACTOR and FEEDBACK LIMIT options for the deadline. For details, see “Using deadline feedback options” on page 122.

**Specifying the application ID**

The name of the application, the date to which it is valid, and the status uniquely identify each application. The application ID can be from 1 to 16 characters, the first of which must be either an alphabetic or national character. All other characters must be alphanumeric. For example, these are valid application IDs:

  MYAPPLICATION1, $MYAPPL, A

while these are invalid application IDs;
Specifying the application ID

MYAPPLICATION=1, IMYAPPL,

Specifying the valid-from date
You can create several applications with the same ID but with different valid-from dates. Tivoli Workload Scheduler for z/OS picks the correct version for the day it is planning.

For example, suppose that your LTP covers 98/01/01 to 98/01/10, that you have an application APPL1 with a valid-from date of 98/01/01, and that you have an amended version of APPL1 with a valid-from date of 98/01/05. When creating the LTP, Tivoli Workload Scheduler for z/OS uses the first version for the period 98/01/01 to 98/01/04 and the second version from 98/01/05.

Notes:
1. You can tailor the date format in the panel to suit your installation. The default format is used here.
2. When listing applications with a filter by date, do not confuse the lower boundary in the specified range with the valid-from date. For example, listing with 01/01/2008 and 01/01/2009 as filter boundaries, the output correctly includes an application with 01/01/2001 as valid-from date.

You specify a valid-from date when you create or copy an application. The application is automatically assigned a valid-to date of 71/12/31, 31 December 2071. When you copy an application, the valid-to date of the old version is automatically set to the day before the valid-from date of the new version. If the new version is valid only for dates earlier than other versions, its valid-to date is set to one day earlier than the valid-from date of the earliest old version.

Specifying the status
You can create an application or group as active or pending. An active application or group can be included in the plans, whereas a pending application or group cannot.

When you are creating a complicated application, it is useful to give it a status of pending. You can then change the status to active when you have completed the application description. If you follow this procedure, there is no risk that your uncompleted application will be included in the plans.

If you change a status from active to pending, consider deleting any occurrences in the LTP before extending the current plan, because the daily plan program tries to find a valid version of an application when processing an occurrence in the LTP. If the latest version is pending, and therefore not valid, Tivoli Workload Scheduler for z/OS uses the previous version, if there is one. If an application description that is active at the input arrival date is not found during daily planning, message EQQ0317W is issued and the occurrence is not included in the plan; instead it is marked as deleted in the LTP.

Using deadline feedback options
Tivoli Workload Scheduler for z/OS automatically monitors the actual deadline of operations. It can use these deadlines to modify the estimates in the application description database.

Two parameters, the smoothing factor and the limit for feedback, control how measured deadlines are used. Any value you specify here overrides the installation default specified on the JTOPTS keyword.
Using deadline feedback options

**Deadline smoothing factor:** The deadline smoothing factor is a value, 0 to 999, that determines how much a measured deadline will change existing values in the application description database, both at run cycle and at operation level.

**Note:** If the measured deadline is outside the limits established by the limit for feedback, the smoothing factor will not be applied and the AD data set is not updated.

The new estimated deadline is calculated as follows:

\[
NDL = ODL + \left(\frac{ADL - ODL}{DSF/100}\right)
\]

Where:

- **NDL** The new estimated deadline for the occurrence or operation, considered as offset from the IA to be stored in the AD database.
- **ODL** The old estimated deadline for the occurrence or operation, considered as offset from the IA stored there.
- **ADL** The actual deadline, considered as the elapsed minutes between the IA and the occurrence or operation completion time.
- **DSF** The deadline smoothing factor.

**Limit for deadline feedback:** The limit for deadline feedback is a value, 100 through 999, that establishes the limits within which measured values are regarded as normal and acceptable. A measured value outside the limits is ignored; that is, no smoothing factor is applied and the application description database is not updated.

The limits are calculated as follows:

- Lower limit = \( ODL \times \frac{100}{DLF} \)
- Upper limit = \( ODL \times \frac{DLF}{100} \)

Where:

- **ODL** The old estimated deadline for the run cycle or operation, considered as offset from the IA currently stored in the application description database.
- **ADL** The actual deadline, considered as the elapsed minutes between the IA and the occurrence or operation completion time.
- **DLF** The limit for deadline feedback.

**Specifying when your application should be scheduled**

When you create an application that is not part of a group, and when you create a group, specify when Tivoli Workload Scheduler for z/OS is to schedule the application or group by creating one or more run cycles. If you do not create a run cycle, the application can still be run on demand, and added to the plans by:

- Panel users
- The program interface (PIF)
- Event-triggered tracking (ETT), or
- Automatic recovery, which adds only to the current plan

For example, an application to recover a database is added to the current plan only when required.

To add and run a group on demand through the panel:

1. Add a member to it.
2. Specify option G on panel EQQMAADL.

For details, see "Adding an application group to the current plan" on page 559.
Creating run cycles

Creating run cycles

Run cycles use the calendar and periods that are described in Chapter 6, “Creating calendars and periods,” on page 103. Run cycles have one of two forms:

1. A rule with a format such as The SECOND TUESDAY of every MONTH, where the words in capitals are selected from lists of ordinal numbers, types of day, and cycle or period names, respectively.

2. A combination of period and offset. For example, an offset of 1 in a weekly period specifies Monday. An offset of 10 in a monthly period specifies the tenth day of each month.

Run cycles can also be negative, when they specify the days that the application must not run. You can specify many run cycles, both positive and negative, when you create an application. Add positive run cycles to generate more days, or to have multiple occurrences on the same day, and add negative run cycles to exclude days already generated. You can mix run cycles that use offsets and run cycles that use rules.

A negative run cycle prevents any positive run cycle from generating an occurrence with an input arrival date and time that matches any date/time combination that the negative run cycle generates.

If you run work at fixed days of the week, month, or year, and take one of the standard Tivoli Workload Scheduler for z/OS actions when this day falls on a free day (holiday), you do not need to create any periods. These are examples of rules:

### Table 12. Examples of rules

<table>
<thead>
<tr>
<th>Rule</th>
<th>Interpretation and notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Only first Sunday in June</td>
<td>The first Sunday in June every year.</td>
</tr>
<tr>
<td>Only work day in the week</td>
<td>The (first) work day (specified by the calendar) in every week. First is default.</td>
</tr>
<tr>
<td>Only last Friday in the year</td>
<td>The last Friday in every year.</td>
</tr>
<tr>
<td>Only second Friday in the year and in April</td>
<td>The second Friday in the year and the second Friday in April. This is a complex rule specifying two cycles (year and April) that could equally well be split into two simple rules: “Only second Friday in year,” and “Only second Friday in April.”</td>
</tr>
<tr>
<td>Only second free day in semester</td>
<td>The second free day (specified by the calendar) in every interval of the period called semester.</td>
</tr>
<tr>
<td>Every second Monday in the month</td>
<td>The first, third, and fifth (if present) Monday in every month. Be careful not to specify this when you mean “Only the second Monday in the month.” When you specify “Every second x,” Tivoli Workload Scheduler for z/OS starts at the origin of x and builds a series 1, 3, 5, and so on.</td>
</tr>
<tr>
<td>Every first work day in the year</td>
<td>Every work day in every year. The “first” means that the series has an interval of one. Do not confuse this with “Only (first) work day in the year.”</td>
</tr>
<tr>
<td>Only 2nd last work day in the month</td>
<td>The last work day but one (second to last) in every month.</td>
</tr>
<tr>
<td>Every third fourth day in the year</td>
<td>Every third day gives January 1, 4, 7, 10, 13 etc. Every fourth day gives January 1, 5, 9, 13, 17 etc. The rule combines these and removes duplicates, giving January 1, 4, 5, 7, 9, 10, 13, 17 etc.</td>
</tr>
<tr>
<td>Every last Monday in the month</td>
<td>This generates every Monday, because Tivoli Workload Scheduler for z/OS generates a series of Mondays starting from the last Monday. If you want only the last Monday in the month, specify Only last Monday.</td>
</tr>
</tbody>
</table>
Table 12. Examples of rules (continued)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Interpretation and notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Every 3rd last work day in the month shift origin 1</td>
<td>This generates a series from the last day but 1 of each month. For July, for example, July 31 is not selected, but the first work day before July 31 is selected. The next two work days are skipped, then the next work day is selected, and so on (the free day rule is not relevant here).</td>
</tr>
<tr>
<td>Every third last free day in the month shift origin 1</td>
<td>This also generates a series from the last day but 1 of each month. For July, for example, July 31 is not selected, but the first free day before July 31 is selected. The next two free days are skipped, then the next free day is selected, and so on. The actual days generated, however, depend on the free-day rule. The result can be confusing unless you use rule 3 when selecting free days.</td>
</tr>
</tbody>
</table>

Note: The only period that you need for the above examples is semester. Words such as week, month, year, April, June, and July are part of the cycle definition, and you do not have to create periods with these names.

For an application to be automatically selected by the planning process, it must contain a run cycle, specified either in the application itself or in its group definition, if it belongs to a group. The run cycle must also be in effect for the duration of the plan (that is, the in-effect dates of the run cycle must fall within the range covered by the plan).

Creating run cycles with rules: Follow these steps to create run cycles using rules:

1. In the CREATING AN APPLICATION panel (Figure 59 on page 120), enter the RUN command. You see the RUN CYCLES panel, shown in Figure 60.

![Figure 60. EQQAMRPL - Run cycles](image)

2. Specify the name of the rule, for example TAXYEAR, and the input arrival time on the day or days specified by the rule. The name should be unique within the application or group, and it is less confusing if you do not use the name of a period; choose a naming convention, perhaps with the rule type as the first character.

The input arrival time specified on the run cycle applies to the occurrence, and is the default time for each operation. For additional information, see “Specifying the input arrival time” on page 133.
Creating run cycles

Note: For details of the effect of specifying an input arrival time earlier than the workday end time, see “How days are generated using rules” on page 128.

3. Specify the deadline day and time, which is the latest time that all operations in any occurrence of the application should be completed by. The deadline day is relative to the input arrival date for the occurrence. Specify a number from 0 to 99, where 0 means that the deadline date is also the input arrival date. When the deadline day is greater than 0, Tivoli Workload Scheduler for z/OS considers only work days when calculating the deadline date. For example, if day is 1, the deadline date is the first work day (as specified by the calendar) after the input arrival date.

The deadline day and time that you specify is also the default for all operations in the application.

4. Specify the type of run cycle, which can be:

R Regular rule.

E Exclusion rule. Specify one or more of these ONLY after you specify an R or N type run cycle.

N Normal run cycle using periods and offsets. See “Creating run cycles with offsets” on page 129 for more information about offset-based run cycles.

X Exclusion run cycle using periods and offsets. Specify one or more of these ONLY after you specify an R or N type run cycle.

5. Specify the free-day rule. See “Selecting a free-day rule” on page 130 for a full description.

6. Specify the in-effect and out-of-effect dates. If you leave these blank, Tivoli Workload Scheduler for z/OS fills the valid- and 71/12/31, 31 December 2071. See “Using in-effect/out-of-effect dates” on page 132 for hints on using these dates.

7. Specify the variable table that will be used on the days selected by this run cycle. For information about job tailoring, see Chapter 22, “Job tailoring,” on page 441.

Note: When you use rule-based run cycles with periods that you have created, Tivoli Workload Scheduler for z/OS ignores any variable table associated with the period, so you must specify any table name here, unless:

• The occurrence operations do not use variable substitution.
• You specify the table name on the LONG TERM PLAN panel.
• You specify the table name on the MODIFY CURRENT PLAN (MCP) panel when you add the occurrence manually to the plan.
• You specify the table name in the job itself.
• The operations use the global variable table.

8. Type a description of the rule in the line below the other fields.

9. Enter the S row command to specify the days that this rule will select (type R) or deselect (type E). You see the MODIFYING A RULE panel, shown in Figure 61 on page 127.
10. In the Frequency column, select Only or Every.

11. Also in the Frequency column, select an ordinal number from First to Fifth and 6 to 999 or its Last equivalent. Type numbers higher than 5th in the blanks below Fifth or 5th Last, and use numerics here not ordinals (for example, 6 instead of 6th). You can select more than one number.

12. In the Day column, select the type of day. You can select more than one type.

13. In the Cycle Specification column, select the type of cycle or period. You can make multiple selections. You can shift the origin of any cycles or periods that you specify by 1 to 999 days, if you are using EVERY. The default origin of each week is Monday. Week 1 is specified as the first week with at least 4 days of the new year.

You can also specify a shifted origin together with EVERY LAST: the origin is shifted from the end, so that EVERY 2nd LAST DAY in JANUARY with shifted origin 1 means January 30, 28, 26, and so on. You cannot specify a shifted origin with ONLY, and do not need to, because ONLY FIRST with shifted origin 1 is equivalent to ONLY SECOND.

EVERY SECOND DAY in YEAR, with no origin shift specified, means January 1, 3, 5, and so on: the series always starts on the first day of the cycle or period unless you shift the origin.

Note: If you select July, for example, the rule does not look for a period called JULY; the cycle is predefined and not alterable. If you really need to use your own JULY period, type the name JULY in the Period name field.

14. Enter the GENDEAYS command to check that the rule causes the days’ selection that you expect. This is especially important for selections with Every, Last, and multiple selections in the Day and Cycle columns. You see the LIST OF GENERATED DATES panel.

Figure 61. EQQRULEP - Modifying a rule
Creating run cycles

**EQQRULSL ------------------ LIST OF GENERATED DATES ------------------**

Command ==> Scroll ==> PAGE
Goto Year ===>

**Rule :** THIRDTHU Third Thursday of the Month
**Calendar :** DEFAULT Work day end time: 00.00
**Interval :** 97/01/30 - 99/12/29 Free day rule: 1

<table>
<thead>
<tr>
<th>January 1997</th>
<th>February 1997</th>
<th>March 1997</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mo Tu We Th Fr Sa Su</td>
<td>Mo Tu We Th Fr Sa Su</td>
<td>Mo Tu We Th Fr Sa Su</td>
</tr>
<tr>
<td>01 02 03 04 05</td>
<td>01 02 03</td>
<td>01 02</td>
</tr>
<tr>
<td>06 07 08 09 10 11 12</td>
<td>03 04 05 06 07 08 09</td>
<td>03 04 05 06 07 08 09</td>
</tr>
<tr>
<td>13 14 15 16 17 18 19</td>
<td>10 11 12 13 14 15 16</td>
<td>10 11 12 13 14 15 16</td>
</tr>
<tr>
<td>20 21 22 23 24 25 26</td>
<td>17 18 19 20 21 22 23</td>
<td>17 18 19 20 21 22 23</td>
</tr>
<tr>
<td>27 28 29 30 31</td>
<td>24 25 26 27 28</td>
<td>24 25 26 27 28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>29 30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>April 1997</th>
<th>May 1997</th>
<th>June 1997</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mo Tu We Th Fr Sa Su</td>
<td>Mo Tu We Th Fr Sa Su</td>
<td>Mo Tu We Th Fr Sa Su</td>
</tr>
<tr>
<td>01 02 03 04 05 06 07</td>
<td>01 02 03 04 05 06 07</td>
<td>01 02 03 04 05</td>
</tr>
<tr>
<td>08 09 10 11 12 13</td>
<td>05 06 07 08 09 10 11</td>
<td>02 03 04 05 06 07 08</td>
</tr>
<tr>
<td>14 15 16 17 18 19 20</td>
<td>12 13 14 15 16 17 18</td>
<td>09 10 11 12 13 14 15</td>
</tr>
<tr>
<td>21 22 23 24 25 26 27</td>
<td>19 20 21 22 23 24 25</td>
<td>16 17 18 19 20 21 22</td>
</tr>
<tr>
<td>28 29 30</td>
<td>26 27 28 29 30 31</td>
<td>23 24 25 26 27 28 29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>30</td>
</tr>
</tbody>
</table>

Figure 62. EQQRULSL - List of generated dates

Notes:

a. If you have not specified a calendar for the application, GENDAYS uses
   the calendar specified under the OPTIONS panel (0.2 from the main
   menu), or the calendar named DEFAULT if no calendar is specified. If the
   DEFAULT calendar does not exist, all days are considered work days. For
   long-term plan batch jobs, however, Tivoli Workload Scheduler for z/OS
   uses the calendar specified on the CALENDAR keyword of the
   BATCHOPT initialization statement, or the DEFAULT calendar. Make sure
   that you specify the same calendar in the panel and BATCHOPT (if you do
   not specify the calendar for the application on the CREATING AN
   APPLICATION panel in Figure 59 on page 120) or you might find that
   GENDAYS shows you different days to those generated when you extend
   the long-term plan.

b. The interval displayed by the GENDAYS command is defined as the
   shortest time period that these dates overlap:
   - Application validity range
   - Run cycle validity range
   - Four years forward from 1 January current year

15. Enter PF3 (End) to return to the RUN CYCLES panel.
16. Repeat steps 2 on page 125 to 15 for regular (R) and exception (E) rules, until
    you have completely specified when the application should be scheduled.

How days are generated using rules: Tivoli Workload Scheduler for z/OS uses
both the free-day rule and the workday end time (specified in the calendar) when
generating days from a rule.

When you fill in the rule on the MODIFYING A RULE panel, Tivoli Workload
Scheduler for z/OS:
1. First generates the days regardless of the free-day rule and workday end time.
2. Adds one day to the generated dates, if the application input arrival time is
   before the workday end time. (If an application input arrival time is 02.00 and
   the workday end time is 03.00; when you specify that the application runs on
Monday, Tivoli Workload Scheduler for z/OS assumes you mean Tuesday at 02.00, because 02.00 on Monday belongs to Sunday.)

3. Takes the workday end time and free-day rule into account. Days can be moved outside the interval because of the free-day rule, although the long-term planning process generates a warning message when this happens.

4. Shows the days resulting from this, if you use the GENDAYS command.

Examples: These examples assume that only Saturday and Sunday are free days, and the workday end time in the calendar is 06.00. You have specified EVERY DAY in WEEK 21 on the MODIFYING A RULE panel. Table 13 shows the generated days for different combinations of free-day rule and input arrival time.

<table>
<thead>
<tr>
<th>Free-day rule</th>
<th>Input arrival time</th>
<th>Days generated</th>
</tr>
</thead>
<tbody>
<tr>
<td>3 (schedule on the free day)</td>
<td>08.00</td>
<td>Monday Tuesday Wednesday Thursday Friday Saturday Sunday</td>
</tr>
<tr>
<td>1 (closest work day before)</td>
<td>08.00</td>
<td>Monday Tuesday Wednesday Thursday Friday</td>
</tr>
<tr>
<td>3 (schedule on the free day)</td>
<td>04.00 (before the workday end time)</td>
<td>Tuesday Wednesday Thursday Friday Saturday Sunday Monday</td>
</tr>
<tr>
<td>1 (closest work day before)</td>
<td>04.00</td>
<td>Tuesday Wednesday Thursday Friday Saturday (considered part of Friday, which is not a free day)</td>
</tr>
<tr>
<td>2 (closest work day after)</td>
<td>04.00</td>
<td>Tuesday Wednesday Thursday Friday Saturday Tuesday (Saturday and Sunday are transferred to Tuesday at 04.00, considered part of Monday)</td>
</tr>
</tbody>
</table>

For details about the workday end time, see “Creating the default calendar” on page 104.

Selecting the last work day before a free day: Use the rule EVERY FREE DAY of the MONTH, which selects all free days. But use free-day rule 1, so that all the selected free day occurrences are moved to the closest work day before. For example, the Saturday and Sunday occurrences of a normal week are both moved back to Friday. Tivoli Workload Scheduler for z/OS does not allow run cycles to schedule multiple occurrences with the same date and time, so the effect is that one occurrence is generated each Friday (or Thursday, if Friday is also free).

Creating run cycles with offsets: You can specify run cycles only by specifying periods, and offsets from the start of the period. This is convenient for some run cycles, where the period is cyclic (a daily or weekly job), but is less convenient for cycles based on calendar months and particular dates in the year, because the period is noncyclic and its origin must therefore be manually maintained from time to time. So you are recommended to use rule-based run cycles.

Follow this process to create run cycles that use offsets:

1. On the RUN CYCLES panel (Figure 60 on page 125), fill in the fields as described in “Creating run cycles with rules” on page 125. In the Name of period/rule field, however, specify the name of a period that you have created, and in the Type field, specify N or X.

You can specify that an application be associated with more than one period by creating more than one run cycle for the application. For example, if your
Creating run cycles

inventory application runs monthly and quarterly, you can associate it with both the monthly and quarterly periods by creating two run cycles for the application, each specifying one of the periods.

If you want a job to run more than once a day, specify two run cycles using the same period and offset, but with a different input arrival time.

2. Enter the S row command to specify the offsets to the start of the period. You see the RUN DAYS panel, shown in Figure 63.

```
Application : CLASSLST  Class lists
Period name  : SEMESTER
Type         : Normal
Free day rule: Schedule on free days

Offsets
: 1
=-1
```

3. Specify a positive or negative offset from the period origin dates.

For example, using the SEMESTER period on page 110, you can specify that application CLASSLST will be scheduled on the first and last day of each semester by specifying offsets of 1 and -1.

**Using work-days-only cyclic periods with offsets:** If you have a work-days-only cyclic period, be careful when you specify offsets greater than 1. For details, see “Using work-days-only cyclic periods” on page 111.

**Selecting a free-day rule**

Specify whether the rule or offset-based run cycle selects work days only or calendar days (that is, work days and free days). Tivoli Workload Scheduler for z/OS must know what action to take if a selected day results in a date that is a free day. You do this by specifying a free-day rule on the RUN CYCLES panel (Figure 60 on page 125).

The possible values of the free-day rule are:

<table>
<thead>
<tr>
<th>Rule</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>Count only work days when using the rule or offset. That is, free days are excluded. This option ensures that the scheduled day will always be a work day. This is the default for offset-based run cycles.</td>
</tr>
<tr>
<td>1</td>
<td>Count work days and free days when using the rule or offset. If this gives a free day, schedule the application on the closest work day before the free day.</td>
</tr>
<tr>
<td>2</td>
<td>Count work days and free days when using the rule or offset. If this gives a free day, schedule the application on the closest work day after the free day.</td>
</tr>
</tbody>
</table>
Selecting a free-day rule

Count work days and free days when using the rule or offset. If this gives a free day, schedule the application on the free day. This is the default for rule-based run cycles.

Count work days and free days when using the rule or offset. If this gives a free day, do not schedule the application at all.

The free-day rule provides the flexibility to schedule your applications precisely when they are required. Sometimes you will find that you must work out on paper which free-day rule you should select. When you do this, consider what would happen if a normal work day is declared a holiday and, therefore, is specified in the calendar as a free day.

When an application is normally due to run but the calendar definition identifies the day as free, the free-day rule in the run cycle for that application determines the effect.

Figure 64 shows what happens for each free-day rule if you specify July with offsets 1, 6, 11, 16, and so on, or the equivalent rule, every fifth day in July.

<table>
<thead>
<tr>
<th>Every fifth day in July</th>
<th>Jul 1997</th>
</tr>
</thead>
<tbody>
<tr>
<td>Free days in calendar</td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule</th>
<th>Free days excluded</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Closest work day before</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Closest work day after</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Schedule on the free day</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Do not schedule at all</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 64. The effect of the free-day rule

Specifying when your application should not run

You might want to prevent Tivoli Workload Scheduler for z/OS from scheduling occurrences of an application on a particular day or set of dates.

There are two ways of using run cycles to prevent Tivoli Workload Scheduler for z/OS from scheduling an application on particular days:

- Negative run cycles
- In-effect/out-of-effect dates

Using negative run cycles: A normal run cycle specifies the day on which an application occurrence should be generated, relative to the dates of a period. Suppose you have an application INVENTORY, which runs on the first day of each month. You might not want INVENTORY to run every calendar month, for example, because an inventory is not taken in December. You can specify this by creating a negative run cycle for the application INVENTORY. A negative run cycle specifies the days that you do not want an application to run, even though the application normal run cycle has specified these days as run days.

Negative run cycles are types E and X, as opposed to normal run cycles, which are types R and N. When the long-term planning process finds a run cycle of type E or X, it generates a negative occurrence. To stop an occurrence generated by a run
cycle of type R or N, the input arrival times on the normal and negative run cycles must be identical.

Figure 65 shows a common example of negative run cycles. In this example, an application that normally runs weekly, on Friday, is not required if Friday is also the last work day of the month. The regular rule specifies:

Only first FRIDAY in the WEEK

and the exception or negative rule specifies:

Only LAST WORK DAY in the MONTH.

The selected keywords are shown in capitals; the other words in the rules are implied or defaulted.

Stopping a normal job if the following day is free: Suppose you run a daily job from Monday to Thursday, and a weekly job on Friday. If Friday is free, you need to run the weekly job on Thursday instead of the daily job. For the daily job, you need run cycles that suppress the job when the following day is free, so use these run cycles:

<table>
<thead>
<tr>
<th>Type</th>
<th>Rule</th>
<th>Free day rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>Daily job run cycles:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>EVERY (or ONLY) MONDAY TUESDAY WEDNESDAY</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>THURSDAY of WEEK</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>EVERY (or ONLY) FRIDAY of WEEK</td>
<td>1</td>
</tr>
</tbody>
</table>

Weekly job run cycles:

| R     | EVERY (or ONLY) FRIDAY of WEEK            | 1             |

The run cycles with free-day rule 1 ensure that the weekly job is brought forward, and the daily job is cancelled, when the Friday is a holiday.

Using in-effect/out-of-effect dates: You can use run cycles that have in-effect/out-of-effect dates to cause an automatic change after a certain date, such as changing the frequency of an application from daily to weekly, as shown in
Specifying when your application should not run

Figure 66

Specifying the input arrival time

When you specify a run cycle for an application, you specify an input arrival time for each occurrence of the application; that is, a time of day to be associated with each run of the application. The input arrival time forms part of the key that uniquely identifies each occurrence of the application in the LTP and current plan; it is not the time that Tivoli Workload Scheduler for z/OS attempts to start the occurrence, unless you specify the first operation as time-dependent (for details, see "Creating time-dependent operations" on page 168). Tivoli Workload Scheduler for z/OS also uses the input arrival time to resolve external dependencies. See "External dependencies between multiple occurrences" on page 146.

The input arrival time of the occurrence is the default input arrival time for operations making up that occurrence. The input arrival time is also used by the daily planning process to calculate the planned start time of an operation. A planned start time cannot be earlier than the input arrival time.

When the daily planning process is selecting occurrences from the long-term plan, it selects only those occurrences whose input arrival times fall within the planning period.

Specifying the EVERY options for a run cycle

To specify how frequently an application must be scheduled, use the EVERY options. To set the EVERY options:

1. In the RUN CYCLES panel (Figure 60 on page 125), enter the S row command next to the run cycle you want.

2. In the MODIFYING A RULE (Figure 61 on page 127) or RUN DAYS (Figure 63 on page 130) panel, enter the E command to set the EVERY options. You see the EVERY OPTIONS panel, shown in Figure 67 on page 134.
Specifying the EVERY options for a run cycle

3. In this panel, the FROM field is already set with the input arrival time of the run cycle. Specify the following settings:

**REPEAT EVERY**
The frequency at which the application is to be scheduled, in the format HH.MM. For example, 00.30 means that the application is run every 30 minutes.

**UNTIL**
The end time until which the application is to be scheduled, in the format HH.MM. It must be a value between the IA time of the run cycle and the calendar work day end time of the application.

For example, consider you have an application whose calendar work day end time is 00.00, run cycle IA time 10:00, and deadline time 22.00. If you set REPEAT EVERY to 01.00 and REPEAT END TIME to 13.00, the long-term plan batch job adds to the long-term plan four occurrences for each day generated:

- IA time 10.00 - deadline time 22.00
- IA time 11.00 - deadline time 23.00
- IA time 12.00 - deadline time 24.00
- IA time 13.00 - deadline time 01.00 - deadline day offset 01

As shown in the example, the deadline time and day offset proceed at the same rate as the input arrival time.

**Note**: If you specified an operation deadline and input arrival time, these values are used unchanged.

As another example, consider you have an application with a calendar work day end time 05.00 and a run cycle whose input arrival time is 11.00. In this case, the EVERY end time must be a value between the IA time 11.00 and the work day end time 05.00, meaning that it must be either a value between 11.01 and 23.59 or between 00.00 and 4.59.

The EVERY options take effect at long-term plan level. Therefore, depending on the current plan end time, the EVERY options might generate occurrences in the current calendar day. For example, suppose that:

- An application scheduled to run every day has the REPEAT EVERY set to 01.00 and REPEAT END TIME to 15.00
- The application IA time is 10:00
- The current plan end date is 31 October, at 12:30

the current plan contains three occurrences of this application, with IA date 31 October and IA time 10:00, 11:00, and 12:00. If you extend the current plan by 24 hours, the new current plan will contain:

---
Specifying the EVERY options for a run cycle

- The application occurrences with IA date 31 October and IA time 13:00, 14:00, 15:00. These are the remaining occurrences set by the long-term plan for the date 31 October.
- The application occurrences with IA date 1 November and IA time 10:00, 11:00, and 12:00.

Note: The EVERY options are not considered when an application is dynamically added to the current plan.

If the application description does not specify a calendar, the default calendar is used to check whether the REPEAT END TIME is valid, according to the tool or function used:

Panels
The default calendar can be set in the SETTING DATE AND TIME FORMAT panel [Figure 20 on page 37]. If the calendar ID is set to DEFAULT and a calendar named DEFAULT does not exist, the default work day end time 00.00 is used.

Mass Update
The default calendar can be set in the CALENDAR parameter of the BATCHOPT statement. If this parameter is not set, a calendar named DEFAULT is used. If a calendar named DEFAULT does not exist, the default work day end time 00.00 is used.

If the calendar that is set does not exist, Mass Update does not perform any check on the REPEAT END TIME.

PIF, BCIT, batch loader, Dynamic Workload Console
The default calendar can be set in the CALENDAR parameter of the INIT statement. If this parameter is not set or the default calendar that is specified does not exist, no check on the REPEAT END TIME is performed. If the calendar is set to DEFAULT, the default work day end time 00.00 is used.

The long-term plan batch job checks whether the REPEAT END TIME is consistent with the work day end time; if not, the LTP batch job resets the REPEAT END TIME to the latest valid time possible (which is one minute before the calendar work day end time) and issues a warning message. The job completes with a return code of 4, at least. For example, if the application has a calendar work day end time 00.00, IA time 10.00, and REPEAT END TIME 09.00, the long-term plan batch job resets the REPEAT END TIME to 23.59 and calculates the occurrences according to this time.

When you create a run cycle with rules, you can check the generated days by issuing the GENDAYS command. These days are generated according to the input arrival of the run cycle. If you set the EVERY options, the LIST OF GENERATED DATES panel shows the number of occurrences that are run on each generated day. Note that if the calendar work day end time is not 00.00, the REPEAT END TIME could be later than midnight; in this case, the LIST OF GENERATED DATES panel also shows the occurrences that is run on the following days.

Specifying run cycles with offsets
Four processing cycles are required to schedule the Paymore system:
- Daily processing-Monday to Friday
- Weekly processing-Thursday
- Monthly processing-third Thursday of the month
- Yearly processing-third Thursday in July
Specifying run cycles with offsets

It is easiest to implement these using rules, but this section shows how you can use offsets instead.

Assume that the calendar specifies Monday to Friday as work days, and public holidays are specified as free days. The PAYROLL system requires three period definitions; one cyclic and two noncyclic. Although monthly processing is required on the third Thursday of the month, you should specify the first Thursday as the period origin, because this ensures that you get maximum benefit from the period definition. When the period is created with an origin of the first Thursday, you can easily specify offsets to select the second, third, fourth, or last Thursday of the month.

Periods should not be created for individual business systems; instead, a period should have a processing cycle that can be used by any run cycle in your applications database. Noncyclic periods incur a yearly overhead when you must enter the origin dates for the following year. For this reason you should make the definition as flexible as possible.

Table 14 shows the period definitions that you need, if you specify the Paymore run cycles using periods with offsets.

Table 14. Period definitions required for offset examples

<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Origin dates</th>
</tr>
</thead>
<tbody>
<tr>
<td>WEEK</td>
<td>All-days cyclic</td>
<td>970102 with an interval of 007 days</td>
</tr>
<tr>
<td>FIRSTTHU</td>
<td>Noncyclic</td>
<td>970102 970206 970306 970403 970501 and so on</td>
</tr>
<tr>
<td>FSTTHU7</td>
<td>Noncyclic</td>
<td>970703</td>
</tr>
</tbody>
</table>

Daily run cycle example: The payroll jobs required daily (PAYDAILY and PAYBACKP) can use this run cycle:

Type = NORMAL  Period = WEEK  Offset = 1,2,3,4,5
Free-day rule = 4  Input arrival = 17.00  Deadline = 01 06.00
Description = Run Monday to Friday except public holidays

Weekly run cycle example: The weekly group GPAYW runs on Thursday. Include the negative run cycle if you want to omit the GPAYW applications on the day that the monthly GPAYM group runs.

Type = NORMAL  Period = WEEK  Offset = 4
Free-day rule = 1  Input arrival = 17.00  Deadline = 01 06.00
Description = Run Thursday or day before if Thursday is free

Type = NEGATIVE  Period = FIRSTTHU  Offset = 14
Free-day rule = 1  Input arrival = 17.00  Deadline = 01 06.00
Description = Exclude third Thursday of the month

The monthly group GPAYM runs on the third Thursday of the month. If you already have a monthly period with an origin date of the first of each month, you can specify a series of offsets and a negative run cycle to avoid the need for a separate period. For example, you could specify a normal run cycle MONTH with offsets 15, 16, 17, 18, 19, 20, 21 with a negative run cycle of WEEK with offsets 1, 2, 3, 5, 6, 7. The normal run cycle is selecting all days in the third week of the month, and the negative run cycle excludes all of those occurrences except the one generated for the third Thursday. This is a correct, if tedious, way of creating run cycles; however, these run cycles cannot reschedule the application to any other day because the negative run cycle always cancels the generated occurrence.
Specifying run cycles with offsets

Monthly run cycle examples: GPAYM is required to run on the Wednesday if the third Thursday is a public holiday; so a period is created and offset calculated that will enable the application to be rescheduled correctly when there are public holidays.

Type = NORMAL  Period = FIRSTTHU  Offset = 14
Free-day rule = 1  Input arrival = 17.00  Deadline = 01 06.00
Description = Run third Thursday of the month or day before if free

The yearly processing PAYTAXYR also requires a unique period definition to be sure the application is always scheduled correctly.

Type = NORMAL  Period = FSTTHU7  Offset = 14
Free-day rule = 1  Input arrival = 17.00  Deadline = 01 06.00
Description = Run third Thursday in July or day before if free

In the sample run cycles for the Paymore system, identical input arrival and deadline times are specified. This can be the most effective method to handle several applications that share a service level but cannot be specified as a group definition. In any case, you need not consider individual operation execution times. The daily planning process handles this for you and assigns latest-out times for each operation based upon the deadline time of the last operation in the dependency chain.

Creating operations

The type of operation determines the type of workstation you use:

- Job operations run on computer workstations.
- Started-task operations run on computer workstations that have the STC attribute.
- Operations on distributed agents run on Tivoli Workload Scheduler for z/OS agents, dynamic domain managers, or fault-tolerant workstations.
- Job setup operations for jobs and started tasks run on general workstations that have the SETUP attribute.
- Print operations run on printer workstations.
- WTO operations run on a general workstation that has the WTO attribute.
- Shadow jobs run on remote engine workstations.
- Operations that have events reported by EQQUSIN or OPSTAT can run on any workstation type except one that specifies no reporting.
- Dummy operations, which are used to simplify dependencies, run on nonreporting general workstations.
- Other tasks that you want to be represented by operations usually run on general workstations.

Follow these steps to specify the operations in an application:

1. In the CREATING AN APPLICATION panel (Figure 59 on page 120), enter the OPER command. You see the OPERATIONS panel, shown in Figure 68 on page 138.
Creating operations

2. For each operation, type the workstation and operation number in the **Oper ws** and **no.** fields.

3. Estimate the duration of each operation, or use the default that you specified for the workstation. When an operation runs, the real duration is fed back to the application description database. The minimum value of planned duration is 1 second, and the maximum value is 99 hours 59 minute 00 seconds. If you specify 99 hours 59 minutes 01 seconds, you do not receive an alert message if the actual duration is greater than the planned duration. For details about the duration feedback, see "Using duration feedback options" on page 156.

4. If the workstation is a job setup, computer, or printer workstation, specify the name of the job or started task that the operation represents. Tivoli Workload Scheduler for z/OS uses this to find the JCL for job or started-task operations. See "Associating job statements with operations" on page 165. The job name for setup and print operations must be the same as the associated job or started-task operation.

5. Type, in the Operation text field, a line of text to be associated with the operation. This text forms part of any WTO message that is issued for an operation. For more details, see "Using WTO operations" on page 167 and the description of the DEADLINE WTO option on page 155. The operation text can be used to further document the processing; it can also direct operators to perform certain functions. For example, you could have an operation on a general workstation to remind the day-shift operators to collect the office supplies.

6. Enter the PRED command to specify any internal and non conditional predecessors to the operation you are creating. For more details of operation predecessors and successors, see "Specifying dependencies" on page 139.

7. Specify the details of each operation by entering S next to the operation you want to specify. For details, see "Specifying operation details" on page 143.

8. Check the JCL of the associated job name by entering J next to the operation you want to specify. This is only possible if you have a tool for editing JCL, and you have specified it in the 0.6 OPTION panel. For details, see "Editing JCL operations" on page 160. This option applies only to jobs running in z/OS environments.
Examples of operations

There is no need to restrict operations to those relating to batch processing. If you rely on people remembering to perform a certain task, consider specifying the task as an operation.

The operations in the example for Figure 68 on page 138 are in the PAYDAILY application for the Paymore system. At the specified time, operation 005 will automatically be set to ready status. When that occurs, the product builds a WTO message and sends the message to the destination specified on the WTO1 workstation definition. The resulting status of the operation depends on the reporting attribute of the workstation. If the workstation reporting attribute is completion-only, the application sets the WTO operation to complete as soon as it is sent, but if the successor operation depends on some action, as in this case, it is better to give the WTO workstation the automatic reporting attribute, so that the product waits for some event (such as an OPSTAT command) to complete the operation.

NetView can be used to intercept the WTO, issue the appropriate CICS commands and check for successful deallocation of the online application. When NetView has determined the online system has successfully shut down, it can execute the EQQEVPGM program specifying this input:

```
OPSTAT JOBNAME(PAYDAILY) STATUS(C) WSNAMES(WTO1)
```

The EQQEVPGM program creates a job-tracking event that is communicated to Tivoli Workload Scheduler for z/OS by the same means that job start and job end events are. When the controller receives the event, operation 005 is set to completed status, allowing the PAYDAILY job to be submitted.

This process can be very quick, and it ensures that the online application is shut down at the correct time and without delay. It further ensures that the batch processing does not start until the online application has completely shut down.

Specifying dependencies

You can specify that operations are dependent on other operations. If operation A1 must complete before operation A2 can start, then operation A1 is a predecessor of operation A2. A2 is a successor of A1. An operation can have many predecessors and successors. These relationships between operations, based on the completion of a successor job, are called normal dependencies.

You can define also dependencies based on the status or return code of other jobs or on the return code of other jobs' steps, these dependencies are named conditional dependencies. See Chapter 19, “Conditioning the processing of operations,” on page 373 for more information on this type of dependencies.

Dependencies are specified between operations. You cannot specify a dependency between applications or application occurrences using the APPLICATION DESCRIPTION panel. You can create dependencies between occurrences with the LONG TERM PLAN panel, but these occurrences are converted to operation dependencies when the occurrences become part of the current plan.

A job setup operation must be an immediate predecessor of the related computer workstation operation. This means that there is a direct dependency between the two operations; it does not mean that the operation number of the job operation is next in sequence to the operation of the setup operation. This dependency is
Specifying dependencies

automatically generated when you create job descriptions using the JOB
DESCRIPTION panel, but you must specify it yourself when creating a standard
application description.

Operations 040 and 050 are in the same occurrence of the PAYM1 application, the
dependency is called internal. Operation 040 depends on an operation in the
PAYDAILY application, the dependency is called external.

You can have an external dependency even between two operations in the same
application description. For instance, suppose you have an application SYSLOG,
which runs every day and consists of two operations, X and Y. You can make X
dependent on Y of the SYSLOG occurrence of the previous day. That is, a SYSLOG
occurrence will not start until the previous SYSLOG occurrence has completed.

Enter the PRED command to enter up to eight internal predecessors for an
operation on the OPERATIONS panel, shown in Figure 69.

You can specify external predecessors, additional internal predecessors, or
conditional predecessors by selecting an operation using the S row command. For
details, see “Specifying predecessors to condition the processing of operations” on
page 144.

If predecessors are added (or other changes are made) to an application in the AD
database, this does not take effect in the current plan until both the long-term plan
and the current plan batch programs have run, or until they are manually added
using the panels.

When an operation is on a fault-tolerant workstation and it does not use a
centralized script, the operation cannot have a predecessor of job setup and a print
successor. In this situation, the fault-tolerant workstation jobs are not in the z/OS
environment. For the same reason, you cannot edit the JCL.

Specifying cross dependencies and shadow jobs

A cross dependency is a dependency of a local job, for example Job_A from another
job, Job_B running in a different scheduling environment. The cross dependency
indicates that Job_A cannot start until the remote job Job_B is completed.
To define a cross dependency, perform the following steps:

1. Make sure that a remote engine workstation referring to the remote scheduling environment is defined, as described in “Specifying remote engine workstations” on page 71.

2. Define the **shadow job**, which runs on the remote engine workstation, that is used to identify the remote job Job_B. For detailed information, see “Specifying remote job information in shadow jobs” on page 164.

3. Add a dependency for Job_A, either internal or external, on the shadow job.

Shadow jobs can be added to the plan by the plan creation process or dynamically at run time. The shadow job input arrival time is used to identify the job instance in the remote engine plan. The matching criteria, regardless whether the remote engine is z/OS or distributed, is **closest preceding**.

The resolution of the cross dependency depends on the status of the shadow job, which reflects at any time the status of the remote job instance. For more information about the possible values of the shadow job status, see “Monitoring a cross dependency resolution in the current plan” on page 410.

**Setting an operation as a critical path target**
If an operation is critical for your business and must complete by the deadline set, you can specify that the operation be considered as the target of a **critical path**. In this way, during daily planning processing, a critical path including the internal and external predecessors of the target operation is calculated, and a specific dependency network is created. While the plan is running, the scheduler monitors the critical paths that are consuming their slack time, becoming more critical than the paths calculated at plan generation.

You set the operation as critical in the JOB, WTO, AND PRINT OPTIONS panel (EQQAMJBP), shown in [Figure 76 on page 151]. You can also specify the WLM policy and service class that will be applied to promote the target operation, together with all the operations belonging to the critical path (unless otherwise stated at operation level).

**Calculating and monitoring critical paths:** For each operation defined as critical, a specific critical path is calculated in the dependency network during daily planning processing. Starting from the target operation, for each level of predecessors, the process chooses the most critical one and includes it in the critical path. Among all of the internal and external predecessors, the most critical one is planned to be the predecessor with the latest end time. Calculating the critical path starts from the target operation and moves backward among its predecessors; it ends when an operation without predecessor is reached.

The planned end time is calculated as the output of the Tivoli Workload Scheduler for z/OS planning processing, which is based on the information you set for the operation or occurrence (for example the IA time, duration, and deadline). The workstation parallel servers, open intervals, special resources and their availability are also taken into account. For this reason, the more accurate the application definition, the more accurate the critical path calculation.

**Note:** Keep in mind that for shadow jobs involved in a critical path, the estimated duration is always calculated as one minute.

To have Tivoli Workload Scheduler for z/OS update your estimates in the application description database with actual run values, see “Using duration feedback options” on page 156.
Calculating and monitoring critical paths

While the plan is running, the scheduler checks if any predecessor of a critical job is starting to delay and monitors the critical paths that are consuming their slack time. In particular:

- The scheduler maintains a hot list for critical jobs, monitoring any critical job predecessor that starts to delay in one of the following conditions:
  - It is late, meaning that it did not start by its latest start time
  - It is long running, meaning that it is running longer than its estimated duration
  - It ends in error
  - It is suppressed by condition

To do this, the scheduler applies the same internal logic that is used to monitor alert conditions.

Before recalculating a critical path, the scheduler estimates new start and end times for all the successors of the job that started to delay.

- When any job in a critical path completes or a dynamic update changes the path, before recalculating the critical path the scheduler estimates new start and end times for the successors of the changed job.
- When the estimated start and end times determine a critical path change, the scheduler updates the current plan.

The deadlines of the operations in a critical job network affect the accuracy of the dynamic critical path handling. To optimize this, set these deadlines to the same or later than the critical job deadlines.

You can monitor critical paths by selecting option 7 (Critical Jobs) from the CURRENT PLAN AND STATUS INQUIRY panel. You get a list of all the critical target operations included in the current plan. Select the operation you want, to display the list of operations belonging to the corresponding critical path in BROWSING THE CRITICAL PATH. For details about option 7 (Critical Jobs), see "CRITICAL JOBS" on page 541.

You can monitor critical paths also through the TEP monitor. For details, see "Enabling monitoring by IBM Tivoli Monitoring" on page 624.

**Managing critical paths at plan execution:** While the plan is running, to increase the possibility of having the target critical job completed by the specified deadline, the scheduler takes the following actions on operations in the critical path:

**z/OS and end-to-end**
- If a ready operation is about to miss its latest start time, the scheduler raises its internal scheduling priority to the highest possible value. In this way, the Workstation Analyzer subtask submits the operation as soon as possible.

**z/OS only**
- According to the specified policy, a started operation which is becoming late is promoted to the WLM service class even if the CRITICAL field is not set to W, if the following conditions apply:
  - The WLM keyword of the OPCOPTS statement is set
  - The operation is in execution state (started status)
  - The requirements of the WLM policy to be applied are met
Managing critical paths at plan execution

At run time, the current plan can be modified in a way that impacts the critical paths. If, for example, you remove a dependency between two operations belonging to a critical path, the path is no longer valid and the scheduler automatically recalculates it.

When you run a daily planning job, the scheduler completely replans the entire network, considering also database updates and removing all completed operations. This means that the critical paths that are recalculated at planning time might differ from the last dynamic updates.

Specifying operation details

In the OPERATIONS panel, enter the row command S to select an operation. You see the OPERATION DETAILS panel.

This section describes the following operation details, in the order they are shown in the OPERATION DETAILS panel.

1. External predecessors, additional internal predecessors, or conditional predecessors.
2. Workstation resources and servers (operations on z/OS only)
3. Special resources
4. Automation options
5. Feedback options
6. Time specifications
7. Operator instructions
8. Edit JCL
9. Restart and cleanup options (operations on z/OS only)
10. Extended information about the operation
11. System automation information
12. User fields
13. Remote job information
Specifying predecessors to condition the processing of operations

You specify external predecessors, additional internal predecessors, and conditional predecessors by selecting option 1 in the OPERATION DETAILS panel. The PREDECESSORS panel (Figure 71) is displayed:

```
Application   PAYM2  MONTHLY PAYROLL TRANSFER
Operation     CPU1 040
Jobname       
Duration      00.01.00  Number of int preds: 0
               Number of ext preds: 0
               Number of conditions: 2
```

Figure 70. EQQAMSDP - Operation details

Specifying predecessors to condition the processing of operations

You specify external predecessors, additional internal predecessors, and conditional predecessors by selecting option 1 in the OPERATION DETAILS panel. The PREDECESSORS panel (Figure 71) is displayed:

```
Application   PAYM2  MONTHLY PAYROLL TRANSFER
Operation     CPU1 040 PAYTRANS
Number of Conds: 0
Row     Oper     Transport time     Application id     Jobname
       cmd         ws no. HH.MM (for ext pred only)       
```

```
  ' SETP 010       PAYTRANS
  ' CPU1 040       PAYMONTH
  ' CPU1 020       PAYWEEK
```

Figure 71. EQQAMPDL - Predecessors

Specifying conditional dependencies: From the PREDECESSORS panel (Figure 71), enter the COND command to define a set of conditions. The CONDITIONS LIST panel (Figure 72 on page 145) is displayed:
Specifying conditional dependencies

Each row corresponds to a list of condition dependencies and is identified by a condition number. The scheduler evaluates the set of conditions according to the Boolean logic of the AND operator.

To define each list of condition dependencies, enter the S row command. The CONDITION DEPENDENCIES DEFINITIONS panel is displayed (Figure 73):

As condition rule, you can specify one of the following:
- All the condition dependencies in the list must be true. It corresponds to the AND operator in the Boolean logic.
- At least \( n \) out of all the condition dependencies must be true. In this case set the rule input field to \( n \). It corresponds to the OR operator in the Boolean logic.

To specify a step dependency, use:
- ProcStep field if the step is not in a procedure.
- Both StepName and ProcStep fields if the step is in a procedure.

ProcStep must correspond to a step specifying the EXEC PGM= statement.

Depending on the type of check that you require on the predecessor, enter one of the following values in the CoTy column:
- RC To check the predecessor return code.
Specifying conditional dependencies

ST To check the predecessor status. It does not apply to step dependencies.

In the CoOP column, specify one of the following logical operators for the required check:

GE Greater than or equal to. Valid only for RC condition type.
GT Greater than. Valid only for RC condition type.
LE Less than or equal to. Valid only for RC condition type.
LT Less than. Valid only for RC condition type.
EQ Equal to.
NE Not equal to. Use it to specify conditions on final statuses only.
RG Range.

To set the value fields, specify:

- An error code, if you set CoTy to RC. You can specify a 4-digit number or one of the supported error codes, listed in "Error codes" on page 745.
- An operation status, if you set CoTy to ST. Allowed values are:
  C Complete.
  E Ended-in-error.
  S Started, according to the job-start event reported by the tracker component. If CONDSUB (YES) is specified in JTOPTS, the condition dependency is evaluated when the operation status becomes S (started) without waiting for the job-start event.
  X Suppressed by condition, meaning that the operation did not run because any condition is false.

For more information on how the scheduler evaluates condition dependencies at run time, see "Evaluating conditions and conditional successor status" on page 374.

Notes:
1. If you use the NE operator, specify a final status (C, E, or X).
2. If you use the NOERROR or highest return code capabilities, the scheduler saves the original return code when setting an operation to complete, because of the NOERROR or highest return code processing. It then uses the original return code to evaluate a condition dependency.

External dependencies between multiple occurrences: If you specified an external dependency between two operations, Tivoli Workload Scheduler for z/OS must decide which occurrences of the applications should be linked by the dependency relationship. This is not always obvious, because there might be several occurrences of each application in the LTP and current plan. The relationship is set up (in other words, the dependency is resolved) by Tivoli Workload Scheduler for z/OS during the LTP planning process.

To resolve an external dependency, Tivoli Workload Scheduler for z/OS uses the input arrival times of the occurrences or, if they have been specified, the input arrival times of the individual operations. For an explanation of input arrival times, see "Specifying the input arrival time" on page 133 and "Specifying input
External dependencies between multiple occurrences

Arrival times and deadlines for operations” on page 158. In all cases, a dependency will be resolved if the successor's input arrival is later than, or equal to, the predecessor's input arrival.

Consider this example:
- Application LTPAPPL1 is scheduled at 09:00 and 12:00 every day.
- Application LTPAPPL2 is scheduled at 11:00 every day.

Application LTPAPPL2 specifies LTPAPPL1 as a predecessor. By default, the daily occurrence of LTPAPPL2 will be a successor of the 09:00 occurrence of LTPAPPL1. If the operation input arrival time of the successor, LTPAPPL2, is modified to be later than or equal to 12:00, the dependency would be resolved to the 12:00 occurrence of LTPAPPL1.

Note: When the LTP selects the predecessor occurrence to resolve a dependency, the input arrival time of the occurrence is always used. Input arrival time specified at operation level is not considered when identifying the predecessor.

Tivoli Workload Scheduler for z/OS resolves external dependencies during the LTP planning process. If you manually change the input arrival time of an occurrence, or an individual operation, in the LTP or current plan (for example, using the panels), this change does not affect dependencies that have already been resolved. To change the dependency relationships for occurrences that are already scheduled, you must explicitly modify them.

Specifying operation resource usage

An operation can use these types of resources:
- Special resources
- Workstation fixed resources
- Parallel servers

Using these resource types, you can avoid allocation failures and other problems caused by contention for resources.

Using special resources: In the application description, you can specify the resources that are used by each operation in your application. You also specify whether the operation requires the resource exclusively or whether it can share the resources with other operations. For information about resources, see "Creating special resources,” on page 81.

The resource name you choose can be any character string up to 44 characters. For documentation purposes, the string should be a meaningful name. If the resource represents a data set, it is good practice to make the resource name the data set name.

When you create an operation that uses a special resource, select option 3 (SPECIAL RESOURCES) in the OPERATION DETAILS menu. You see the SPECIAL RESOURCES panel, shown in Figure 74 on page 148.
### Special resources

<table>
<thead>
<tr>
<th>Operation</th>
<th>CPU1 030</th>
</tr>
</thead>
</table>

#### Type values in the fields on the SPECIAL RESOURCES panel:

**Special Resource**

- **Name**: You can use the global search characters `%` and `*` if you are unsure of the name. Tivoli Workload Scheduler for z/OS will display a list of matching names. For example, if you specify `PAY*`, Tivoli Workload Scheduler for z/OS displays all special resources beginning with `PAY`, and you can select `PAYROLL.DATABASE`.

**Qty**: The number of resources that the operation allocates. If you leave this field blank, the operation is allocated the whole quantity currently available (the adjust quantity), unless this value is not greater than 0, in which case the operation cannot start until the adjust quantity becomes greater than 0. This adjust value is the difference between the current and global values. Once started, the operation allocates the whole quantity available, even if this value later increases.

**Shr Ex**: Whether the operation needs shared or exclusive access to the resource:
- S (shared)
- X (exclusive)

A resource unit allocated to an operation as shared can be used by other sharing operations at the same time, but is unavailable to operations that need exclusive allocation. Resource units allocated as exclusive can be used only by that operation. Tivoli Workload Scheduler for z/OS does not start another operation that requires the allocated units until the first operation ends.

This is like data set disposition. However, if a resource unit is allocated as shared by an operation and an operation requiring exclusive use becomes ready, further shared requests are not delayed behind the waiting exclusive operation. Waiting operations, in other words, have no allocated resources (unless they are failed operations that have the keep-on-error attribute for some resources).

**Keep on error**

- Y: The resource is kept
- N: The resource is freed
- Blank: The default value set for the resource is taken
**Avail on complete**

Whether the global availability of the resource is to be changed when the operation completes.

Y The global availability is changed to Yes.

N The global availability is changed to No.

R The global availability is changed to blank.

**Blank** Uses the system default, according to the following order:

1. The On Complete value set at operation definition level, if not blank.
2. The On Complete value set at special resource definition level, if not blank.
3. The ONCOMPLETE or DYNONCOMPLETE keyword value, respectively set for the not dynamically added resources or the dynamically added resources, in all the other cases.

Use the SPECIAL RESOURCE panel to create and modify special resources (option 1.6 from the main menu).

**Using workstation fixed resources:** When you specify an operation on a workstation, you can specify how many of the workstation fixed resources the operation will use. Tivoli Workload Scheduler for z/OS knows how many of each resource is available on that workstation; for more information, see “Specifying workstation fixed resources” on page 78. When deciding whether to start the operation, it considers the available workstation resources and the resources required for this operation. Normally, Tivoli Workload Scheduler for z/OS starts an operation only if the needed quantity of a resource is available at the workstation. In case of virtual workstations, the scheduler considers the fixed resources as a criterion to select the submission destination: the next destination to be selected, as part of the round-robin algorithm, is checked for fixed resources. If the check fails, it moves on the next destination in the list.

However, when creating the workstation, you can specify that resource usage will not be considered when Tivoli Workload Scheduler for z/OS is either producing its schedule or starting operations. Fixed resources do not apply for jobs on fault-tolerant workstations and remote engine workstations.

**When you create workstations** (for details, see “What types of workstation are there?” on page 53), you specify the quantity of workstation resources available. Tivoli Workload Scheduler for z/OS recognizes two workstation resource types, by default called R1 and R2, although you can choose other names. You decide what these resources represent. They are commonly used to represent tape or cartridge drives.

For example, assume that workstation CPU1 has an R1 value of 10. This means that, for all operations started at any one time on this workstation, the total use of R1 by these operations cannot exceed 10.

Note that although workstation resources represent some resource pool, Tivoli Workload Scheduler for z/OS has no knowledge of the actual state of the resources. That is, Tivoli Workload Scheduler for z/OS can only keep track of resource users that are operations in the current plan. In addition, Tivoli Workload Scheduler for z/OS is only aware of the total value specified on the workstation definition, or the modified amount in the current plan. Tivoli Workload Scheduler
for z/OS also has no way of checking that the actual resource usage of an operation matches the planned resource usage.

Suppose, for example, that R1 on CPU1 represents tape drives. Your system has 6 tape drives, so in the workstation description you have specified a value of 6 for R1. You have specified in the application description that operations A, B, and C (all jobs) use 2 tape drives each. Tivoli Workload Scheduler for z/OS submits the operations to JES. There is no contention for resources, so they can all run at the same time. However, the JCL for operation B has been changed since the application description for B was set up, so that it now uses 3 tape drives. If the jobs are submitted by Tivoli Workload Scheduler for z/OS in the order A, B, C, this means that C is waiting for a tape drive.

Tivoli Workload Scheduler for z/OS bases its decisions on the operation resource usage, as recorded in the application database, so the change to the JCL did not affect its planning or starting of the operations. Because Tivoli Workload Scheduler for z/OS cannot tell whether the tape drives are being used by jobs outside its control, it might submit jobs on the assumption that the resources represented by the workstation resources are available, even though the real resources are all in use.

If some resources become unavailable, for example, because of a hardware problem, Tivoli Workload Scheduler for z/OS continues to schedule according to the original amount until the current plan is modified with a revised amount.

You can specify workstation resource usage for an operation on the WORK STATION RESOURCES AND SERVERS panel (Figure 75), displayed by selecting option 2 on the OPERATION DETAILS panel (Figure 70 on page 144).

| EQQAMWRP -------- WORK STATION RESOURCES AND SERVERS -------- |
| Command ====> |
| Enter/Change data below: |
| Operation : CPU1 020 Runs pay04 and pay06 |
| Work station resource: |
| Resource 1 ===> _0 Amount of ws resource |
| Resource 2 ===> _0 Amount of ws resource |
| SERVERS ===> _1 Number of parallel servers |

Figure 75. EQQAMWRP - Work station resources and servers

Using parallel servers: It is common practice to regard parallel servers for computer workstations as JES initiators. Parallel servers work like workstation fixed resources: the number of servers required by an operation must be available before Tivoli Workload Scheduler for z/OS will start the operation. Operations on non-virtual workstations use one server. In case of virtual workstations, the scheduler considers the parallel servers as a criterion to select the submission destination: the next destination to be selected, as part of the round-robin algorithm, is checked for the number of parallel servers. If the check fails, it moves on the next destination in the list. Parallel servers do not apply for jobs on fault-tolerant workstations.

You can choose whether Tivoli Workload Scheduler for z/OS takes parallel servers into consideration when it plans the schedule, when it starts an operation, at both
Parallel servers

times or at neither. This is the server usage, specified as P, C, B, or N on the workstation definition. See “Specifying workstation fixed resources” on page 78.

Tivoli Workload Scheduler for z/OS assumes that operations on computer workstations always use a single server.

Specify the number of servers for an operation on the WORK STATION RESOURCES AND SERVERS panel. You can display this panel by selecting option 2 on the OPERATION DETAILS panel.

Specifying options for automation

Selecting AUTOMATIC OPTIONS, option 4, in the OPERATION DETAILS panel takes you to the JOB, WTO, AND PRINT OPTIONS panel.

The following sections describe options that apply to:
- Jobs and started tasks
- Jobs only
- Print operations only
- All operations

Options that apply to jobs and started tasks: These options apply to jobs and started tasks:

ERROR TRACKING (default: Y)

If you specify Y, an error in the job or started task (for example, an abend or JCL error) causes the operation representing the job or started task to be marked E (ended-in-error).

If you specify N, the operation is marked C (complete) when the operation ends, regardless of the outcome except for restarted jobs failing in the EQQCLE character step, that is a step inserted into the job by the Restart and Cleanup function. In this case, the operation is marked E (Ended in Error).

Figure 76. EQQAMJBP - Job, WTO, and print options

The following sections describe options that apply to:

- Jobs and started tasks
- Jobs only
- Print only
- All operations

Options that apply to jobs and started tasks: These options apply to jobs and started tasks:

ERROR TRACKING (default: Y)

If you specify Y, an error in the job or started task (for example, an abend or JCL error) causes the operation representing the job or started task to be marked E (ended-in-error).

If you specify N, the operation is marked C (complete) when the operation ends, regardless of the outcome except for restarted jobs failing in the EQQCLE character step, that is a step inserted into the job by the Restart and Cleanup function. In this case, the operation is marked E (Ended in Error).
Options that apply to jobs and started tasks

For fault-tolerant workstations and automation workstations, the only valid value is Y.

The error tracking option is applied only to error conditions reported by job tracking events. An operation which specifies N for error tracking can be set to error status manually by a panel user or the OPSTAT command. Errors reported during job submission; for example missing JCL, job card inconsistencies, JCL variable errors and the installation policy for suppress-if-late option can all result in an E status for an operation which specifies error tracking N.

HIGHEST RETURNCODE (default: value of HIGHRC on JTOPTS)

This field specifies the highest acceptable return code from any step in the job or started task. It is not applicable to the EQQCLEAN step in a restarted JCL.

For fault-tolerant workstations and automation workstations, the valid value is either zero or blank. If a return code for a step in the job or started task exceeds this value, the operation is set to E (ended-in-error) status, unless there is a match with a statement in the NOERROR initialization statement. If you must specify an acceptable nonzero return code for a particular step, or steps, use the NOERROR statement.

For more details, see “Using error codes to set operations to ended-in-error” on page 311.

CRITICAL (default: N)

This field specifies if the operation must be considered a critical path target, meaning that at DP processing a critical path is calculated for the job to complete by its deadline. You can specify:

- P The operation is to be considered a critical path target.
- W The operation is eligible for WLM assistance.
- N The operation is not eligible for WLM assistance.

For details about the critical path, see “Setting an operation as a critical path target” on page 141.

POLICY (default: ‘’) and CLASS

The Workload Manager (WLM) policy and service class. If an operation is defined as a critical path target or eligible for WLM assistance, the scheduler automatically sends a request to promote the job or started task to the high-performance service class, when the conditions of the specified assistance policy are met. You can specify the following policies:

- L Long duration. The job is assisted if it runs beyond its estimated duration time.
- D Deadline. The job is assisted if it has not finished when its deadline time is reached.
- S Latest start time. The job is assisted if it is submitted after the latest start time.
- C Conditional. An algorithm calculates whether to apply the Deadline or the Latest start time policy.
- ‘ ‘ Default. WLM uses the policy specified in the OPCOPTS statement.

This option does not apply to fault-tolerant workstations. For a description of WLM, see Chapter 23, “Job scheduling and WLM,” on page 493.
Options that apply to jobs and started tasks

SUBMIT (default: Y)
If you specify Y, Tivoli Workload Scheduler for z/OS automatically starts the job or started task or issues the WTO message when all predecessors have been satisfied and all required resources are available. Usually, this is the option you choose. However, if the JCL for the job is not under Tivoli Workload Scheduler for z/OS control, for example, when the job arrives via a RJE link, specify N.

Note: For jobs and started tasks to be automatically submitted, the JOBSUBMIT parameter on the JTOPTS initialization statement must be set to YES, and job submission must not be deactivated using the SERVICE FUNCTIONS panel (see Chapter 12, “Using service and optional functions,” on page 289).

The N option does not apply to fault-tolerant workstations.

RESTARTABLE (default: take the installation default)
This option determines what the status of the operation will be if its workstation becomes inactive (failed or offline). This option applies to the operation only while it has status S (started). This option does not apply to fault-tolerant workstations.

Y The operation is reset to status R (ready) if its workstation becomes inactive. That is, the operation is restarted from the beginning on the alternate workstation, or on this workstation when it becomes active again. The operation is reset to ready status only when:
1. You specify this in the MCP panel when you manually set the workstation failed or offline.

or,
2. You specify this in the WSSTAT or EQQUSIN subroutine when you set the workstation failed or offline.

and, if not otherwise specified in the panel, command, or subroutine,
3. The first parameter of the installation default on the WSFailure or WSoffline keyword on the JTOPTS initialization statement allows operations to be restarted.

N This operation will not be restarted even if the installation default, as specified in the WSFailure or WSoffline parameter on the JTOPTS initialization statement, is to restart started operations on workstations that become inactive.

If the installation default is to put started operations into error status on workstations that become inactive, the operation is given status code E (ended-in-error).

blank The operation takes the installation default action on the OPRESTARTDEFAULT keyword of the JTOPTS statement if the workstation that it is started on becomes inactive.

REROUTABLE (default: take the installation default)
This option specifies what action Tivoli Workload Scheduler for z/OS should take for this particular operation if the computer workstation that it is scheduled to run on is inactive and an alternate workstation has been specified. This option applies to the operation only when it is in status R (ready) or W (waiting). Once the operation is in status S (started), the RESTARTABLE option determines the action. This option does not apply to fault-tolerant workstations.
Options that apply to jobs and started tasks

Y  The operation is eligible to be rerouted if the workstation becomes inactive, and:
   1. You specify this in the MCP panel when you manually set the workstation failed or offline.

   or,
   2. You specify this in the WSSTAT or EQQUSIN subroutine when you set the workstation failed or offline.

   and, if not otherwise specified in the panel, command, or subroutine,
   3. The second parameter of the installation default on the WSFAILURE or WSOFFLINE keyword on the JTOPTS initialization statement allows operations to be rerouted.

N  The operation is not rerouted, even when the workstation has an alternate destination. For more information about directing work to alternate workstations, see “Redirecting work to alternate workstations” on page 586.

blank  The operation takes the installation default action on the OPREROUTEDEFAULT keyword of the JTOPTS statement if the workstation becomes inactive.

Options that apply only to jobs:  These options apply only to operations that represent jobs:

JOB CLASS (no default)
   Specify the JES input class of the job that the operation represents. The job class that you specify here is only for documentation purposes. It need not correspond to the actual class of your job, although it is good practice if it does. This option does not apply to fault-tolerant workstations.

HOLD/RELEASE (default: Y)
   Use this option to control held jobs that are not submitted by Tivoli Workload Scheduler for z/OS.

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

   If HOLD/RELEASE is set to N, Tivoli Workload Scheduler for z/OS releases a held job immediately without reference to its schedule. You cannot set this option to N for fault-tolerant workstations.

   Note: Specifying a HOLD/RELEASE value Y for a job operation does not cause Tivoli Workload Scheduler for z/OS to put the job into HOLD status. If the job is already in HOLD status, however, Tivoli Workload Scheduler for z/OS releases it at the scheduled time.

Options that apply to print operations: The FORM NUMBER and SYSOUT CLASS fields in the operation options, combined with the job or started-task name, identify the JES output group that you want to track. The operation is marked as complete when an output group with a matching job name, form number, and SYSOUT class has either completed printing or been purged from the spool.
Options that apply to print operations

Ensure that the combination of job or started-task name, form number, and SYSOUT class is unique to the print data set that you want to monitor. If this is not the case, the print operation might be marked as complete when another print data set, which matches the selection criteria, is printed or purged. These options do not apply to fault-tolerant workstations.

If some of your jobs or started tasks conditionally create output, or if FREE=CLOSE is specified on the SYSOUT DD statement, consider specifying PRTCOMPLETE(YES) in the JTOPTS initialization statement. For more information about the PRTCOMPLETE keyword, see Customization and Tuning.

Options that apply to all operations: These options apply to all operations:

TIME-DEPENDENT (default: N)
If you specify Y, the operation becomes time-dependent. That is, Tivoli Workload Scheduler for z/OS will not start the operation until the operation input arrival time is reached. If Tivoli Workload Scheduler for z/OS cannot start the operation at the input arrival time, the operation is considered late. This can happen if your system has been unavailable for some period or when the operation also has predecessors that do not finish in time. See “Creating time-dependent operations” on page 168.

If you specify N, the operation will not be time-dependent. Tivoli Workload Scheduler for z/OS will start the operation as soon as its predecessors are completed and resources are available. If there are no predecessors, and resources are available, Tivoli Workload Scheduler for z/OS starts the operation immediately as it is added to the current plan.

SUPPRESS IF LATE (default: N)
Specify Y to stop this time-dependent operation being started if it is late. See “Creating time-dependent operations” on page 168 for details of what happens to late time-dependent operations.

If you specify N, Tivoli Workload Scheduler for z/OS ignores the fact that the operation is late and tries to start it as soon as possible.

DEADLINE WTO (default: N)
If you specify Y for this option, Tivoli Workload Scheduler for z/OS issues the operator message EQQW776I when a z/OS operation passes its deadline and the operation is in started status (that is, the operation has been started but has not been marked as completed within the deadline time). The message is routed to the operator console of the workstation that the operation runs on. The message is also written to the message log (EQQMLOG). The WTO is issued only for z/OS operations that have status S (started).

Besides the standard message, the user-defined text that you specified on the OPERATIONS panel, using the TEXT command, is issued as part of the WTO (Figure 68 on page 138).

You can use this to stop a job or started task automatically at a certain time. See “Scheduling the closedown of online systems and started tasks” on page 168 for details.

Specifying Y for an operation on a workstation that is not a z/OS destination will have no effect. That is, a deadline WTO will not be routed to the workstation destination.

EXTERNAL MONITOR (default: N)
Specifies if the operation is monitored by an external product (for example, Tivoli Business Systems Manager or Tivoli Enterprise Portal). If you specify Y for this option, then monitoring is enabled. For more information about...
Options that apply to all operations

---

**Centralized Script (default: N)**

By default, the script is not centralized and therefore is stored locally on the agent and a job definition is added to the SCRPTLIB. In contrast, a centralized script is stored in the JOBLIB. Using a centralized script implies loss of fault tolerance, because an operation might depend on a script that can be released only by the IBM Tivoli Workload Scheduler for z/OS controller. Extra processing is needed because the script must be taken from the JOBLIB and sent to the agent.

Specify Y to use a centralized script. This option applies to fault-tolerant workstations only. If you specify Y for workstations that are not fault-tolerant, the value is forced to Y when you generate the daily plan.

**COND RECOVERY JOB (default: N)**

Specify Y if you use this operation as recovery job, meaning that it might recover a conditional predecessor. For details, see “Handling recovery using conditional dependencies” on page 392.

**Using duration feedback options**

Tivoli Workload Scheduler for z/OS automatically monitors the actual duration of operations. It can use these durations to modify the estimates in the application description database.

For example, if a job processes a data set that is getting bigger, the job is likely to take longer each time it runs. Using the feedback option helps to ensure that the job is started soon enough to process any new records and to still meet its deadline.

Two parameters, the smoothing factor and the limit for feedback, control how measured durations are used. Any value you specify here overrides the installation default specified in the JTOPTS statement.

**Notes:**

1. For shadow jobs, the smoothing factor and limit for feedback are ignored.
2. The value used to select the operations for which a long duration alert must be issued, is set with the ALEACTION keyword of JTOPTS. If ALEACTION is not set, the LIMFDBK value is used instead. In this case, the value for the feedback limit that you can optionally enter in the application description is ignored.
3. The feedback limit value applies also to the WLM policy DURATION.

Select option 5 in the OPERATION DETAILS panel. Specify the feedback options in the panel shown in Figure 77 on page 157 to automatically adjust the estimated duration in the database after the job completes.
Duration smoothing factor

The duration smoothing factor is a number, 0 to 999, that determines how much a measured duration will change existing values in the application description database. Note that if the measured duration is outside the limits established by the limit for feedback, the smoothing factor will not be applied and the AD data set will not be updated.

The new estimated duration is calculated as follows:

\[ ND = OD + ((AD - OD) \times \frac{SF}{100}) \]

Where:
- **ND**: The new estimated duration to be stored in the AD database.
- **OD**: The old estimated duration stored there.
- **AD**: The measured duration.
- **SF**: The smoothing factor.

Table 15 shows some examples of how the smoothing factor algorithm works.

<table>
<thead>
<tr>
<th>Factor</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>There will be no feedback.</td>
</tr>
<tr>
<td>10</td>
<td>The new estimated duration will be the old estimated duration, plus one-tenth the difference between the measured and old estimated duration.</td>
</tr>
<tr>
<td>50</td>
<td>The new estimated duration will be the old estimated duration, plus one-half the difference between the measured and old estimated duration.</td>
</tr>
<tr>
<td>100</td>
<td>The measured duration replaces the old estimated duration.</td>
</tr>
<tr>
<td>999</td>
<td>The new estimated duration will be the old estimated duration, plus 10 times the difference between the measured and old estimated duration.</td>
</tr>
</tbody>
</table>

Limit for duration feedback: The limit for duration feedback is a number, 100 through 999, that establishes the limits within which measured values are regarded as normal and acceptable. A measured value outside the limits is ignored; that is, no smoothing factor is applied and the application description database is not updated.

The limits are calculated as follows:

Lower limit = \( OD \times \frac{100}{LF} \)

Upper limit = \( OD \times \frac{LF}{100} \)

Where:
- **OD**: The old estimated duration stored in the application description database.
- **LF**: The limit for duration feedback.
Limit for duration feedback

Table 16 shows some examples of how the limit for feedback algorithm works.

**Table 16. Examples of limits for feedback**

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

Specifying input arrival times and deadlines for operations

You can specify an input arrival time for each operation making up an application. If you do not specify an input arrival time for an operation, the operation input arrival time defaults to the input arrival time of the application.

You can specify an input arrival time and deadline for an operation on the TIME SPECIFICATIONS panel (Figure 78) which you can reach by selecting option 6 in the OPERATION DETAILS panel (Figure 70 on page 144).

**Figure 78. EQQATMP - Time specifications**

The input arrival and deadline day are relative to the input arrival date for the occurrence. Specify a number from 0 to 99, where 0 means that the day is also the input arrival day. When the specified day is greater than 0, Tivoli Workload Scheduler for z/OS by default considers only work days when calculating the date. For example, if day is 1, this specifies the first work day (as specified by the calendar) after the input arrival date. You can change Tivoli Workload Scheduler
Specifying input arrival times and deadlines for operations

for z/OS to include free days when calculating deadline dates. For a description of the OPERIALL and OPERDALL keywords of the BATCHOPT statement, see *Customization and Tuning*.

**Specifying operator instructions**

You can specify an operator instruction to be associated with an operation in an application. This could be, for example, special running instructions for a job operation.

The instruction can be permanent or temporary. A temporary instruction has VALID FROM and VALID TO dates associated with it, which specify when the instruction is valid.

Operator instructions can be handled from either the APPLICATION DESCRIPTION panel or the OPERATOR INSTRUCTION panel.

**Note:** When you delete an application, a confirmation panel is displayed, on which you can choose whether to retain or delete the associated operator instructions (see Figure 81 on page 160).

If you select option 7 in the OPERATION DETAILS panel (Figure 70 on page 144), then the LIST OF OPERATOR INSTRUCTIONS panel (Figure 79) is displayed.

---

**Figure 79. EQQALSML - List of operator instructions**

From here it is possible to create, update, delete, browse operator instructions in the same way as in the OPERATOR INSTRUCTION panel (option 1.5 from the main menu), except that the operator instruction key (application name and operation number) cannot be changed.

You can specify up to 443 lines of operator instructions for an operation in the CREATING AN OPERATOR INSTRUCTION panel (Figure 80 on page 160).
Specifying operator instructions

While handling operator instructions in the APPLICATION DESCRIPTION panel menu, and returning from the LIST OF OPERATOR INSTRUCTIONS panel, the operator instruction might have been modified (with a create operator instruction, for example) while the Application Description has yet to be confirmed. This might lead to inconsistencies, such as having an operator instruction referring to a nonexistent application. For this reason, optional consistency checks have been added each time an application is deleted, created, or modified. These checks look for operator instructions having no match in the application description data base (at least one application with the same name and operation number). If any are found, they are deleted.

You can specify (option 0.5) whether to perform these checks and whether to display a confirmation panel [Figure 81].

Editing JCL operations

You can edit the JCL of an operation if a job name exists, and if you have a tool for editing JCL and have specified its name in the 0.6 OPTION panel. This function applies to fault-tolerant workstations only if they use the centralized script.
Restart and cleanup of operations

You can specify the cleanup action to be taken on computer workstations for operations running on z/OS. Also, you can specify if Tivoli Workload Scheduler for z/OS will use the JCL extracted from the JESJCL sysout. If necessary, you can specify if user sysout support is needed. [Figure 82]

The following are the cleanup types:

A Automatic. When the operation is ready to be submitted and the controller selects it for submissions, the controller automatically finds the cleanup actions to be taken and also inserts them as the first step in the JCL of the restarted job. Whenever the operation is started from the panels, the cleanup actions are shown to the user for confirmation, only if the CLEANUP CHECK option was set to YES through the DEFINING PARAMETERS AND OPTIONS panel.

I Immediate. Data set cleanup is immediately performed if the operation ends in error. The operation is treated as if it had the automatic option when it is rerun.

M Manual. Data set cleanup actions are deferred for the operation. They are performed when initiated manually from the panel.

N None. No data set cleanup actions are performed.

These options apply to expanded JCL:

Y Use the fully expanded JCL.

N Use the JCL contained in the libraries of Tivoli Workload Scheduler for z/OS.

These options apply to user sysout support:

Y Data store logs user sysout.

N Data store does not log user sysout.

For more information, see Chapter 15, “Planning for recovery and restart,” on page 303 and Chapter 17, “Restart and cleanup,” on page 313 for additional information.

Specifying extended information

You can specify additional information in the operation description. You can also use the strings to filter queries of operations.
**Specifying extended information**

```
EQQAMXDP---------------------- OPERATION EXTENDED INFO ----------------------
Command ===> 
Application : PAYDAILY daily payroll jobs
Operation : CPU1 020 runs pay04 and pay06
Job name : PAYDAILY

Enter change data below:
Use extended info: Y Y or N
Operation Extended Name:
daily payroll data for accounting_________________________________
Use SE name: Y Y or N
Scheduling Environment Name:
DB2ACTIVE_________________________________
```

*Figure 83. EQQAMXDP - Operation extended info*

Specify Y to have the Extended Name field appear for the operation in the current plan after you generate the daily plan or perform a dynamic addition.

Specify Y to have the Scheduling Environment Name field appear for the operation in the current plan after you generate the daily plan or perform a dynamic addition.

After it is added to the plan, the Scheduling Environment name is used at job submission time to check if the job can be submitted and if so, to appropriately tailor the JCL (by adding the JOB keyword SCHENV=SE name).

For more information, see Chapter 24 “Job Scheduling and WLM”.

**Specifying system automation information**

For operations running on Automation workstations, you must specify at least the command text to be run by System Automation for z/OS. All the other system automation information is optional.
Specifying system automation information

Command Text
Text of the command to be routed to System Automation. It is free format and you can specify Tivoli Workload Scheduler for z/OS variables, which are replaced before the command is passed to System Automation for z/OS. If an error occurs during this phase, the operation is set to E with code OJCV. No syntax checking on text content is performed on the Tivoli Workload Scheduler for z/OS side. You can specify a System Automation command, NetView command, or z/OS system command (it must be issued within a NetView PIPE command), or any user-defined command (such as, for example, CLIST or REXX exec).

Completion Info
Completion information. You can optionally specify the following information, in the following order, separated by a comma:

• Maximum wait time, in NetView notation (for example, hh:mm:ss). If specified, System Automation for z/OS waits for the completion of the command for the specified time interval. If the command does not complete, System Automation for z/OS posts the operation in error. Note that for INGREQ and INGMOVE commands the following applies:
  – The command is considered complete when the specified resource has reached or is already in the requested state.
  – If more than one resource is specified in the INGREQ or INGMOVE command, all resources must be in the requested state before the command is considered complete.

• Maximum return code accepted as successful execution.

• Name of an optional user-supplied completion checking routine. The completion checking routine ensures that the command achieved the expected results, before posting the operation as complete.

Automated Function
Automated function (for operation). This parameter is optional. If specified, the
command is run on the NetView task associated with this automated function in System Automation for z/OS. You can use this parameter to serialize commands. If this parameter is not specified, the command is run by any of the local NetView tasks available.

Security Element
Security element. It is an optional parameter used for security tracking of the operation. You can use it in alternative to or in conjunction with the job name, for security validation of the operation on the System Automation side.

For additional details about how to set and use these parameters, see Tivoli Workload Scheduler Automation Reference and Operator’s Guide.

Creating user fields to specify additional information
You can create your own fields to specify additional information that you find useful to store about the operation. This information is stored in the Application Definition and Current Plan. It is not processed by Tivoli Workload Scheduler for z/OS and does not affect the operation execution, however it is read by the following user exits:

- EQQUX001
- EQQUX002
- EQQUX007
- EQQJVXIT

The user fields that you create are not checked by any process, and are kept exactly as you type them. You cannot set the user field name to blank.

Specifying remote job information in shadow jobs
The operations that you run on a remote engine workstation are named shadow jobs. A shadow job is used to identify the job, in the remote engine plan, to which the processing must point. To identify the remote engine job, you must specify the appropriate information in the panel that opens according to the type of remote engine.
Specifying remote job information in shadow jobs

z/OS remote engine
The z/OS REMOTE JOB INFO panel opens. To identify the remote job, specify the application ID and operation number.

Distributed remote engine
The DISTRIBUTED REMOTE JOB INFO panel opens. To identify the remote job, specify the job stream workstation, job stream name, and job name.

In the COMPLETE IF BIND FAILS field, specify how to set the shadow job status if the matching with the remote job instance in the remote engine plan fails. The allowed values are:

Y  The operation status is set to Complete and the successor starts.
N  The successor is not started until you delete the dependency manually or complete the operation manually. This is the default.

Associating job statements with operations
The main purpose of Tivoli Workload Scheduler for z/OS is to start work according to a predefined schedule, the current plan. The current plan is produced from information in the Tivoli Workload Scheduler for z/OS database and LTP. Before Tivoli Workload Scheduler for z/OS can submit a job or start a started task, it must have job statements (JCL in the case of the z/OS operating system). Tivoli Workload Scheduler for z/OS offers some powerful facilities for tailoring JCL (or the equivalent for other operating systems) to fit each run of your operations.

Job statements and computer workstation operations
For jobs on z/OS systems, store job statements for submitted jobs in the partitioned data sets that are allocated to the ddbname EQQJBLIB. Tivoli Workload Scheduler for z/OS associates an operation with a member of one of these libraries, using the JOB NAME field of the OPERATIONS panel, shown in Figure 69 on page 140. That is, the job statements for a job or started task are stored in the member identified by the JOB NAME field. For jobs on distributed agents, EQQSCLIB is used. The members of EQQSCLIB contain a JOBREC statement that describes the job to be executed. The rest of this section addresses the details of jobs on z/OS systems.

Tivoli Workload Scheduler for z/OS submits the job statements if the operation is a job, or starts it as a procedure if the operation is a started task. So the job name of each job or started task operation must be unique to ensure that the correct job is picked up. Each time an operation is run, its job statements are picked up from EQQJBLIB, and are then stored in the job repository (a cycle of VSAM data sets with the ddbname EQQJSnDS). The job statements remain there until the successful completion of the next occurrence of the application, and are then deleted. So the job repository should contain the job statements for the most recent completed run of any application in the current plan. Tivoli Workload Scheduler for z/OS always uses the job from the repository for reruns; this job is modified by the variable substitution and recovery functions. The original job, in EQQJBLIB, is always used for the first run of operations in an occurrence.

Variable substitution can be automatically invoked at job or started-task submission time, if you require. Alternatively, operators can be prompted for the values of variables before submission. See Chapter 22, “Job tailoring,” on page 441 for more details.
Job statements and computer workstation operations

Notes:
1. Tivoli Workload Scheduler for z/OS checks the JCL for a valid job card if you specify JOBCHECK(YES) on the JTOPTS initialization statement. This applies only to jobs destined for z/OS. If you specify JOBCHECK(SAME), it also checks that the job name is the same as the operation name.
2. It is not recommended that you have several jobs in the same member. If you do this, put the job that matches the operation (member) name last, or Tivoli Workload Scheduler for z/OS cannot track the job. You must not specify JOBCHECK(SAME). This applies only to jobs destined for z/OS.
3. Do not use JCL that has been packed by ISPF in EQQJBLIB, because Tivoli Workload Scheduler for z/OS does not use ISPF routines to read it.
4. Do not include JCL with TYPRUN=SCAN in EQQJBLIB. Tivoli Workload Scheduler for z/OS does not track these jobs. To test JCL, do this outside Tivoli Workload Scheduler for z/OS, or add TYPRUN=SCAN using the MCP panel and type the SUBMIT command; then remove TYPRUN=SCAN or cancel the edit. This applies only to jobs destined for z/OS.

If you use Tivoli Workload Scheduler for z/OS to submit jobs to non-z/OS operating systems, you need not store the JCL equivalent information in the EQQJBLIB. The operation-initiation exit, EQQUX009, which handles submission for operations at workstations that specify a user-defined destination ID, can be used to locate job statements from another file, or you can request the receiving operating environment to locate them. If the job statements are in EQQJBLIB, you can use the job tailoring and automatic recovery functions for these operations.

Note: Started task operations on workstations that specify a user-defined destination ID will be treated in the same manner as normal computer operations on user-defined destinations. That is, all job-statement information is passed to the operation-initiation exit, EQQUX009. You decide how the exit chooses to handle the information.

JCL and job setup operations
When manual tailoring of job statements is necessary, specify a setup operation to precede a job or started task operation. By using a setup operation you can ensure that jobs and started tasks are not submitted until the job statements have been suitably tailored.

A setup operation is an operation that is specified on a general workstation that has the JCL setup attribute. The setup operation is associated with the subsequent submission operation by two factors:
1. The job name of the setup operation is the same as the job name of the submission operation.
2. The setup operation is an immediate predecessor of the submission operation.
   This does not mean that the setup operation must immediately precede the submission operation in operation number order.

The setup operation should not itself have predecessors, because it is started manually and is therefore under control of the operator.
Using print operations

If the time that a SYSOUT data set is printed or purged is important at your installation, consider using print operations. A print operation must complete before the occurrence that it is part of is considered complete.

A print operation does not require any action by Tivoli Workload Scheduler for z/OS. Tivoli Workload Scheduler for z/OS monitors the print and reports on the status, but the handling of print operations is still under the control of the JES subsystem.

A print operation must be a successor of a job or started-task operation and must have the same job name as its predecessor operation. If the job or started task produces more than one SYSOUT data set, you can create a print operation for each unique combination of job and started-task name, form number, and SYSOUT class that is produced to track the print output accurately. See “Options that apply to print operations” on page 154.

Using WTO operations

A WTO operation is created on a general workstation that has the WTO option. This causes message EQQW775I to be sent to the workstation destination and from there the message is issued as a WTO to the system console. Besides the standard message, the user-defined text that you specified on the OPERATIONS panel, using the TEXT command (Figure 68 on page 138), is also issued as part of the WTO.

You can schedule a WTO operation like any other operation: it can have predecessors, specify resources, and be time-dependent. The WTO is issued only if the operation is automatically started by Tivoli Workload Scheduler for z/OS, so the submit option of a WTO operation must be set to Yes, and job submission must be active. Then, for example, NetView can intercept this operation and take action.

If the workstation is nonreporting, the operation is marked complete but no WTO is issued. The reporting attribute of the workstation affects the way in which the operation is tracked. See “General workstations” on page 56 for details.

Consider using WTO operations to trigger your console automation software. Although most console automation systems include some scheduling capabilities, these are normally limited to AT and EVERY type constructs. Many of the tasks performed by the console automation software must be coordinated with batch processing or online systems, and Tivoli Workload Scheduler for z/OS can manage this effectively.

Started-task operations

Started-task operations are created and scheduled like job operations, except that they run on workstations that have the STC option.

The procedure that will be used to invoke the started task is specified in the JOB NAME field for the operation. The JCL for the started task is stored like the JCL for jobs (see “Job statements and computer workstation operations” on page 165). Instead of the JCL being submitted to the internal reader on the destination system, however, it is temporarily placed in the JES procedure library that is allocated to the EQQSTC ddname in the Tivoli Workload Scheduler for z/OS procedure, and a START command is issued to invoke it. For details about the required procedure libraries, see Planning and Installation.
Scheduling the closedown of online systems and started tasks

Scheduling the closedown of online systems and started tasks

Tivoli Workload Scheduler for z/OS can initiate and track a job or started task but cannot directly stop it. The recommended method for scheduling the stop of a z/OS job or started task is to:

1. Specify the time that you want the task to be stopped as the DEADLINE TIME for the operation. You specify this under Time Specifications, which you can select from Operation Details (see “Specifying input arrival times and deadlines for operations” on page 158).

2. Enter text identifying the started-task in the OPERATION TEXT field for the operation. The text could be the command required to stop the task.

3. Specify a DEADLINE WTO for the operation. You specify this on the JOB, WTO, AND PRINT OPTIONS panel (see “Options that apply to all operations” on page 155).

   After the operation is started, it will reach its deadline for completion, and Tivoli Workload Scheduler for z/OS issues message EQQW776I as a WTO (if the workstation is a z/OS system). In this way, the system operator is automatically informed that the started task should be ended. The message can be intercepted by NetView, which can then automatically stop the task.

4. Add code in NetView to intercept the EQQW766I message and stop the job or started task. Refer to the member EQQNGETW1 sample library for an example in the REXX language.

When the started task has terminated normally, Tivoli Workload Scheduler for z/OS sets the operation to complete.

Creating time-dependent operations

The input arrival time of an operation is usually not the time that Tivoli Workload Scheduler for z/OS will start an operation. Tivoli Workload Scheduler for z/OS seeks to maximize the throughput of work in your system by starting as many operations as quickly as possible on any one day. Some operations cannot be started because the resources they require are not available, they have a predecessor that has not completed, or the workstation they run on is closed. If an operation can run (that is, if all its predecessors are complete and resources are available), Tivoli Workload Scheduler for z/OS will usually start the operation regardless of its input arrival time or the time of day.

For example, when you run jobs on distributed agents, the SUPPRESSPOLICY keyword also affects the actual start time. If you specify the suppress if late option, the value of the SUPPRESSPOLICY keyword also impacts the start of late operations. Refer Customization and Tuning for more information about the SUPPRESSPOLICY keyword on the JTOPTS statement.

For most operations, this is exactly what you want. However, you often need to run jobs or started tasks at a particular time of day or at regular intervals throughout the day. To do this, make the job operation time-dependent on the MVS JOB OPTIONS panel (Figure 76 on page 151). The following describes what happens when Tivoli Workload Scheduler for z/OS starts a time-dependent operation, and what happens if it is late:

1. Is the operation time-dependent?
   b. No: Tivoli Workload Scheduler for z/OS starts the operation as soon as it is added to the current plan or as soon as its dependencies are met.
2. Is it the input arrival time?
   a. Yes: continue with step 3.
3. Is the operation ready?
   b. No: wait until the operation is ready. Continue at step 3.
4. Does the operation have the suppress-if-late attribute (see “Options that apply to all operations” on page 155)?
   a. Yes: continue with step 5.
   b. No: Tivoli Workload Scheduler for z/OS starts the operation.
5. Does the SUPPRESSPOLICY allow more time for this operation?
   a. Yes: Tivoli Workload Scheduler for z/OS starts the operation.
   b. No: give the operation a status according to the SUPPRESSACTION option (C, E, or RL).

An operation with the status RL can be started only after manual intervention. For details of the SUPPRESSPOLICY and SUPPRESSACTION keywords of the JTOPTS statement, see Customization and Tuning.

Creating job descriptions

This section describes the JOB DESCRIPTION panel, which lets you create job, started-task, and WTO operations using a faster path than the one available with the APPLICATION DESCRIPTION panel.

A job description is an application consisting of a job, started task, or WTO main operation that can have an internal predecessor job preparation or a manual preparation operation, or both, but no other operations. Applications with more operations are standard applications, which are described in “Standard applications and group definitions” on page 119. The Job Description dialog automatically specifies the internal dependencies between its operations; there is no need to specify them as you would if you were creating a standard application using the APPLICATION DESCRIPTION panel.

You can change a job description with the APPLICATION DESCRIPTION panel, but if you add operations so that it is no longer eligible for the Job Description dialog, it becomes a standard application. On the other hand, if you remove operations from a standard application until it meets the criteria for a job description (and it has operations with the standard numbers 005, 010, and 015), you can change it using the Job Description dialog.

Using the Job Description panel

The information that you must provide to create a job description is described more fully in “Standard applications and group definitions” on page 119, and you should read that chapter if you have never created applications. The JOB DESCRIPTION dialog compresses the fields onto one panel, and makes some assumptions about the application that you creating.

Follow these steps to create a job description:
1. Enter the JOB DESCRIPTION dialog by selecting option 8 (JD) from the MAINTAINING TWSz DATA BASES panel, or by entering 1.8 from the main
Using the Job Description panel

menu. You see the MAINTAINING JOB DESCRIPTIONS panel:

![EQQJSUBP - Maintaining job descriptions](image)

2. Select option 2 (CREATE) from the MAINTAINING JOB DESCRIPTIONS panel. You see the CREATING A JOB panel with the settings from the previous job description that you worked with:

![EQQJCGPP - Creating a job](image)

3. On the CREATING A JOB panel, specify the characteristics of your main operation and up to two predecessor operations:

    **JOBNAME - TEXT**
    
    The name of the main operation. This is also the name of the job description, and this operation is assigned number 015. You can add operator text.

    You cannot use the JOB DESCRIPTION panel if the APPLID keyword of the DBCSOPTS initialization statement specifies DBCS input, because the job name cannot be entered in DBCS bracketed format: use the APPLICATION DESCRIPTION panel instead if you need a double-byte character set application ID.
### Using the Job Description panel

**OWNER: ID – TEXT**

The owner ID and a description.

**CALENDAR ID**

If you leave this blank, Tivoli Workload Scheduler for z/OS uses the calendar in the CALENDAR keyword of the BATCHOPT initialization statement, for batch services such as extending the long-term plan, or the calendar specified under the OPTIONS panel (0.2 from the main menu), for online services such as testing a rule with GENDAYS. If no calendar is specified, or the specified calendar does not exist, a calendar with the name DEFAULT will be used. If the DEFAULT calendar does not exist, all days are considered work days. You might have several calendars, but always call your default calendar DEFAULT, and specify the same calendar name on BATCHOPT and in the panel.

**AUTHORITY GROUP ID**

This field can be used for security grouping and for reporting.

**VALID FROM – TO**

The date range that this job description is valid.

**DURATION**

The estimated duration of the main operation.

**RUN TIME FROM – TO**

The input arrival time of the main operation (the FROM time) and the deadline time of the main operation (the TO time). If the TO time is less than the FROM time, the TO time is considered to be for the day following the run day of the operation.

**Note:** Tivoli Workload Scheduler for z/OS attempts to start an operation at the FROM time only if you specify that the operation should be time-dependent.

**TIME DEPENDENT**

See “Creating time-dependent operations” on page 168 for details of time-dependent operations.

**WORK STATION**

The workstation of the main operation.

**PRIORITY**

The priority of the main operation, from 1 (lowest) to 9 (urgent).

**JCL PREPARATION and JCL WS**

Specifies whether a JCL preparation operation will precede the main operation and specifies its workstation. If you fill in this field, a JCL preparation operation is created and is assigned operation number 005. This option applies for main operations on fault-tolerant workstations only if they use the centralized script.

**HIGHEST RETURN CODE**

The highest acceptable return code from any step in the main operation. If a return code for a step in the job or started task exceeds this value, the operation is set to E (ended-in-error) status, unless there is a match against a statement in the NOERROR initialization statement. If you must specify an acceptable nonzero return code for a particular step, or steps, use the NOERROR statement.

See “Using error codes to set operations to ended-in-error” on page 311 for more details.
Using the Job Description panel

MANUAL INTERACTION and MANUAL WS
The text for a manual operation that precedes this computer operation, and
its workstation. If you fill in these fields, a manual operation is created and
is assigned operation number 010.

RUN CYCLES
Specifies up to six offset-based run cycles. See “Creating run cycles for job
descriptions” on page 173 for more information.

PREDECESSORS
Names of other job descriptions that are made external predecessors of the
main operation. You can specify a generic name in this field: if there is more
than one job description with this job name, you get a list from which to
choose the predecessor you need. If the plus sign (+) is displayed to the
right of the list, one of these is true:
• Some of the predecessors are standard application descriptions.
• There are more than five job-description predecessors.
• The external dependency is not to the main operation of a job description.
The external dependency that is set up between the job descriptions is
between their main operations. See “Specifying dependencies” on page 139
for details of the predecessor relationship between operations.
Enter the DETAILS command if you need to specify other predecessors.

SPECIAL RESOURCES
Specifies up to three resource names, with a quantity from 1 to 999 999 or
blank (blank means allocate the whole quantity available), allocation type S
(shared) or X (exclusive), keep-on-error N (free), Y (keep), or blank (default),
and on complete N (no), Y (yes), or blank (use system default).
If you specify more than three special resources for a job description, the
plus sign (+) is shown to the right of the last resource:

<table>
<thead>
<tr>
<th>PREDECESSORS</th>
<th>SPECIAL</th>
<th>RESOURCES</th>
<th>GROUP DEFINITION</th>
</tr>
</thead>
<tbody>
<tr>
<td>===&gt;</td>
<td>===&gt;</td>
<td>LINES.TO.LONDON</td>
<td>===&gt;</td>
</tr>
<tr>
<td>PAYROLL.DATABASE</td>
<td>______</td>
<td>000002 X Y N</td>
<td></td>
</tr>
<tr>
<td>TAPES</td>
<td>000000</td>
<td>000001 X Y N</td>
<td></td>
</tr>
</tbody>
</table>

Figure 88. Specifying special resources for a job description

GROUP DEFINITION
The group name, if this job description is to be part of a group.

SMOOTHING FACTOR
A value, from 0 to 999, that determines how much a measured deadline will
change existing values in the application description database, both at run
cycle and at operation level. For more details about deadline feedback
options, see “Using deadline feedback options” on page 122.

LIMIT
The limit for deadline feedback. You can specify a value from 100 to 999
that establishes the limits within which measured values are regarded as
normal and acceptable. A measured value outside the limits is ignored. For
more details about deadline feedback options, see “Using deadline feedback
options” on page 122.

See “Standard applications and group definitions” on page 119 for further
information on these fields.
Creating run cycles for job descriptions

You can specify up to six run cycles on the CREATING A JOB panel:

<table>
<thead>
<tr>
<th>RUN CYCLES</th>
<th>MON - 001</th>
<th>TUE - 001</th>
<th>WED - 001</th>
<th>THU - 001</th>
<th>FRI - 001</th>
<th>SAT - 001</th>
</tr>
</thead>
</table>

Figure 89. Specifying run cycles for a job description

If you are not familiar with run cycles, see “Specifying when your application should be scheduled” on page 123. You can specify up to six periods and offsets, as shown in Figure 89, if these run cycles have:

- The same in-effect dates as the job description
- The same input arrival times and deadline times as the job description (as specified in the RUN TIME FROM – TO field)
- No more than one offset specified
- Free-day rule E (exclude free days)

If you want to create more than six run cycles, rule-based run cycles, or run cycles without these restrictions, enter the RUN command. The RUN CYCLES panel (Figure 60 on page 125) is displayed, where you can specify run cycles in the standard way.

If you specify run cycles with the RUN command, a plus sign (+) is displayed when you return to the CREATING A JOB panel, as shown in Figure 89.

Specifying additional operation details for job descriptions

If you enter the DETAILS command in the JOB DESCRIPTION panel, the OPERATION DETAILS panel is displayed. From this menu, you can specify the operation details as you would for a standard application. The difference is that you can specify operation details only for the processor operation. See “Specifying operation details” on page 143 for more information.
How job descriptions affect the current and long-term plans

Each job description produces several operation occurrences in the LTP and current plan. If you have several job descriptions, especially if they are linked using external dependencies, your LTP and current plan can become too big and can take a lot of computer time to produce. If this occurs, consider combining related job descriptions into one standard application using internal dependencies. This can also simplify your schedules.

Listing applications

This section describes how to create a list of all the applications in the Application Description database both by using the basic panels (default) and by using the advanced panels (see "Panels Style" on page 41).

Creating a list of applications

If you are using either the basic panels (default) or the advanced panels, to create a list of applications, enter fast path 1.4 from the main menu. The MAINTAINING APPLICATION DESCRIPTIONS panel opens, as shown in Figure 58 on page 120. Select option 3 to display the SPECIFYING APPLICATION LIST CRITERIA panel. Leave all the criteria fields blank to see the entire list of applications in the database, or enter criteria to create a more specific list. The LIST OF APPLICATIONS panel opens, as shown in Figure 91 on page 175.
Creating a list of applications

The advanced LIST OF APPLICATIONS panel (EQQNALSL) provides more extensive information than the basic panel style. To distinguish between applications and groups of applications, you can set a different color for the letter in the column T (type). You can customize the type color in the ISPF options (see “Setting options” on page 36). In the first panel, you see the basic information (see Figure 92). By scrolling to the right, using F11, you can see additional data such as priority, run cycle, and number of operations, as shown in Figure 93 on page 176, Figure 94 on page 176, and Figure 95 on page 176. Use F10 to scroll back to the left.

Figure 91. EQQALSTL - List of Applications (default panel style)

The advanced LIST OF APPLICATIONS panel (EQQNALSL) provides more extensive information than the basic panel style. To distinguish between applications and groups of applications, you can set a different color for the letter in the column T (type). You can customize the type color in the ISPF options (see “Setting options” on page 36). In the first panel, you see the basic information (see Figure 92). By scrolling to the right, using F11, you can see additional data such as priority, run cycle, and number of operations, as shown in Figure 93 on page 176, Figure 94 on page 176, and Figure 95 on page 176. Use F10 to scroll back to the left.
Performing tasks from the List of Applications panel

When using the advanced panels, the LIST OF APPLICATIONS panel has a menu bar that allows you to perform tasks without browsing away from the panel. The menu bar has the following three menus:

**Action**

You can perform the following actions by either entering the number next to the action or by entering, in the primary command line, the abbreviated name shown in parentheses:

**Create (CREATE)**

Displays the CREATING AN APPLICATION panel for creating a new application (see Figure 59 on page 120).
Performing tasks from the List of Applications panel

Print Applications (PRINTA)
Displays the PRINTING APPLICATIONS panel.

Mass Update (MASSUP)
Displays the MASS UPDATING OF APPLICATION DESCRIPTIONS panel.

View
You can choose between the Full or Graph views of the application data. The name of the template the panel is using is displayed next to the name of the view. There is a different template for each type of view of each type of advanced panel. See “Specifying panel views” on page 44 for more information.

Help
You can learn more about the LIST OF APPLICATIONS panel, including:
• General help topics
• Primary commands
• Row commands

Enter forward slash (/) in the Row cmd column next to an application (see Figure 94 on page 176) to open the TABLE ROW COMMANDS panel. The Table row commands panel lists all the available commands for the selected application, as shown in Figure 96.

<table>
<thead>
<tr>
<th>EQSRSCLP</th>
<th>Table row commands</th>
</tr>
</thead>
<tbody>
<tr>
<td>Choose one of the following commands by entering the number.</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>1. Browse (B/S)</td>
<td></td>
</tr>
<tr>
<td>2. Modify (M)</td>
<td></td>
</tr>
<tr>
<td>3. Copy (C)</td>
<td></td>
</tr>
<tr>
<td>4. Delete (D)</td>
<td></td>
</tr>
<tr>
<td>5. Print (P)</td>
<td></td>
</tr>
<tr>
<td>6. Run Days (A)</td>
<td></td>
</tr>
<tr>
<td>7. Modify LTP (L)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 96. Table row commands

Enter the number of a table row command to open the corresponding panel. For example, enter 4 to open the CONFIRMING DELETION OF AN APPLICATION panel to delete the selected application.

Alternatively, without browsing away from the LIST OF APPLICATIONS panel, you can enter any of the command letters listed in parentheses (see Figure 96). For example, instead of entering forward slash (/) in the Row cmd column, as shown in Figure 97 on page 178, enter C to go directly to the COPYING AN APPLICATION panel.
Browsing an application

From the LIST OF APPLICATIONS panel (EQQNALSL), enter B or S in the Row command column next to an application to browse its description and corresponding data. Alternatively, you can enter forward slash (/) in the Row command column of the application, and then enter 1 to select Browse in the TABLE ROW COMMANDS panel. The APPLICATION IN THE DATABASE panel (EQQNABGP) opens. This panel is scrollable down and to the right, with each part of the panel providing various types of information (see Figure 98).

The first part of the panel shows the basic information such as the application status, valid dates, type, owner, and so on.

By scrolling down, you can see information about the latest updates to the data and the run cycles data.

Figure 97. EQQNALSL - Enter command letters in Row cmd column

Figure 98. EQQNABGP - Browsing the application description in the database (part 1)
By scrolling down even further, you see the list of operations in the application. The Run Cycles section, and all of the sections that follow, are prefixed by a number so you can easily jump to the required section using the `find` command, without having to scroll through each section. For example, to quickly display the Operations section, type `find 2-` at the command prompt.

The APPLICATION IN THE DATABASE panel (EQQNABGP) has a menu bar that allows you to perform tasks without browsing away from the panel. The menu bar has the following three menus:

**Action**
You can perform the following actions by either entering the number next to the action or by entering, in the primary command line, the abbreviated name shown in parentheses:

**Link**
**View**
**Help**

---

**Figure 99. EQQNABGP - Browsing the application description in the database (part 2)**

By scrolling down even further, you see the list of operations in the application. The Run Cycles section, and all of the sections that follow, are prefixed by a number so you can easily jump to the required section using the `find` command, without having to scroll through each section. For example, to quickly display the Operations section, type `find 2-` at the command prompt.

---

**Figure 100. EQQNABGP - Application description in the database (part 3)**

The APPLICATION IN THE DATABASE panel (EQQNABGP) has a menu bar that allows you to perform tasks without browsing away from the panel. The menu bar has the following three menus:

**Action**
You can perform the following actions by either entering the number next to the action or by entering, in the primary command line, the abbreviated name shown in parentheses:
**Browsing an application**

**Print Application (P)**
Displays the GENERATING JCL FOR A BATCH JOB panel, where the data is ready to be generated for printing an application.

**Print Run Cycles (A)**
Displays the GENERATING JCL FOR A BATCH JOB panel, where the data is ready to be generated for calculating and printing run days.

**Modify LTP (L)**
Displays the GENERATING JCL FOR A BATCH JOB panel, where the data is ready to be generated for modifying the LTP of an application.

**Link**
The Link menu options enable you to quickly open the Period, Calendar, Workstation, and Special Resources databases.

- **Link Period (LINKPER)**
  Displays the Maintaining the TWSz Periods panel, where you can manage periods.

- **Link Calendar (LINKCAL)**
  Displays the Maintaining the TWSz Calendars panel, where you can manage calendars.

- **Link Workstation (LINKWS)**
  Displays the Maintaining Work Stations Descriptions panel, where you can manage workstations.

- **Link Special Resources (LINKSR)**
  Displays the Maintaining Special Resources panel, where you can manage the special resources.

**View**
You can choose between the Full or Graph views of the application data. The name of the template the panel is using is displayed next to the name of the view. There is a different template for each type of view of each type of advanced panel. See "Specifying panel views" on page 44 for more information.

**Help**
You can learn more about sections of the APPLICATION IN THE DATABASE panel, including:
- General help topics
- Primary commands
- Run cycle
- Operations

For example, selecting Run cycle shows information specific to the section of the panel that contains the run cycle data. Selecting Operations shows specific information about the section of the panel that contains operation data, such as workstation number, duration, number or dependencies, and so on.

**Browsing run cycle information**

The APPLICATION IN THE DATABASE panel has a section for run cycles (see Figure 99 on page 179). Enter forward slash (/) in the Row cmd column next to a run cycle to open the TABLE ROW COMMANDS panel. The commands available for that run cycle are listed, as shown in the example in Figure 101 on page 181.
Browsing run cycle information

**Browsing application operations**

The APPLICATION IN THE DATABASE panel has a section for the list of operations in that application (see Figure 100 on page 179). Enter forward slash (/) in the Row cmd column next to an operation to open the TABLE ROW COMMANDS panel. The commands available for the selected operation are listed, as shown in the example in Figure 102.

---

1. **Show Run Cycle details (B/S)**
   - Displays the BROWSING A RULE basic panel.

2. **Display the dates generated by this rule (GEN)**
   - Displays the LIST OF GENERATED DATES basic panel.

3. **Display the Every Options for this rule (E)**
   - Displays the EVERY OPTIONS basic panel.

4. **Show Period (BPE)**
   - Displays the BROWSE PERIOD panel for the period of the selected row.

5. **Modify Period (MPE)**
   - Displays the MODIFY PERIOD panel for the period of the selected row.

---

You can enter the abbreviations shown in parentheses in the primary command-line in the APPLICATION IN THE DATABASE panel, without having to list the TABLE ROW COMMANDS.

---

**Figure 101. Table row commands for a run cycle**

---

**Figure 102. Table row commands for an operation**
Browsing application operations

2. **Browse JCL (BJ)**
   Displays the EDITING/BROWSING JCL FOR A COMPUTER OPERATION basic panel.

3. **Browse operation instructions (O)**
   Displays the BROWSING AN OPERATION INSTRUCTOR panel, if operator instructions exist. If there are no operation instructions, option 3 takes you back to APPLICATION IN THE DATABASE panel (EQQNABGP).

4. **Browse workstation (BWS)**
   Displays the BROWSING A WORK STATION DESCRIPTION panel for the workstation of the selected row.

5. **Modify workstation (MWS)**
   Displays the MODIFYING GENERAL INFORMATION ABOUT A WORK STATION panel for the workstation of the selected row.

You can enter the abbreviations shown in parentheses in the primary command line in the APPLICATION IN THE DATABASE panel, without having to list the TABLE ROW COMMANDS.

The APPLICATION IN THE DATABASE advanced panel allows you to see, in a single panel, the comprehensive information for the selected operation in an application. You can scroll both right and down to view more information. While the operation details section of the panel remains static, you can scroll down to see the following sections:

- Automatic options
- Time options
- Predecessors
- Conditions
- Resources and servers
- Special resources
- User fields
- Feedback options
- Clean up options
- Extended info
- Automation info
- Remote job info

The sections are prefixed by a code so that you can easily jump to the required section using the `find` command, without having to scroll through each section. For example, to quickly display the Predecessors section, type `find PRED-` at the command prompt. Figure 103 on page 183 shows an example of the first part of an OPERATION IN THE DATABASE advanced panel.
Browsing application operations

Figure 103. EQQNABSP - Showing operation details and some automatic options

Figure 104 shows an example of the predecessors and conditions information for the selected operation in the database.

Figure 104. EQQNABSP - Showing time options and predecessors

Figure 105 on page 184 shows an example of the resources and servers, special resources, and user field information for the selected operation in the database.
Browsing application operations

**Figure 105. EQQNABSP - Showing resources and user field information**
Chapter 8. Defining applications in batch

This chapter describes the following:
- How to choose between the mass update and batch-loader utilities
- How to run the mass update utility
- What the batch loader is
- How to code batch-loader control statements
- How to run the batch-loader program
- Batch-loader sample

See "Batch-loader control statements" on page 200 for the syntax, explanation, and examples of the batch-loader control statements.

Mass update or batch loader?

There are two utilities that process applications in batch; the batch loader and the mass update utility. Use Table 17 to decide which is best for your purpose.

<table>
<thead>
<tr>
<th>If you want to ...</th>
<th>BL</th>
<th>MU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maintain a sequential data set with your application descriptions, group definitions, and operator instructions.</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Change application descriptions in batch.</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Change group definitions or operator instructions in batch.</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>Make changes to application descriptions conditional on the original values of options and attributes.</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Produce a report showing all applications that have specified values in certain fields.</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>

Note: If you want to create only operator instructions in batch, you can use the batch program EQQOIBLK. See "Layout of the operator instruction data set" on page 699 for details.

Maintaining a sequential data set

You might want to maintain a sequential data set for loading your AD and OI databases, because:
1. It is a backup that you can use if you lose the databases.
2. You can use it when you use Tivoli Workload Scheduler for z/OS for the first time, if you already have application descriptions in some format that you can convert.
3. You might find it convenient to use the ISPF editor to change the control statements.
4. You can easily delete and recreate your AD and OI databases, for example if you are new to Tivoli Workload Scheduler for z/OS and are developing naming standards.

Changing application descriptions in batch

You can use either the batch loader or mass update to do this. The batch loader:
- Replaces the application description without paying attention to the original values.
Changing application descriptions in batch

- Can perform syntax and validity checking on your new application description.

The mass update function:

- Changes individual fields in your application description, and you can specify what the old value must be for the change to occur.
- Can do the update in trial mode, so that you see what applications will be changed.

Changing group definitions and operator instructions

You must use the batch loader to change group definitions in batch, a sample program is provided in the EQQYCBAG member of the SEQQSAMP library. To change operator instructions in batch, use the batch loader or the program EQQOIBLK. See “Layout of the operator instruction data set” on page 699 for details of EQQOIBLK.

Making changes conditional on the original values

You might want to change all priority 5 applications to priority 4, for example.

Searching applications for values in specified fields

You might want to list all time-dependent operations on workstation CPU1, for example. Use the mass update function as though you are changing the time-dependency from Y to N, but in trial mode. You get a list of all the hits, which is the report that you need. Use the mass update function to do this.

How to run the mass update utility

If you need to make many updates to application or job descriptions, consider using the mass update function to perform the updates to the database in batch. Perform mass updates in trial mode first. Trial mode produces a report from which you can verify the proposed changes before submitting the request in update mode, which implements the changes in the application description database.

This example shows how you might use the panel to change the priority of all Paymore applications (which have an OWNER ID of SAMPLE) from priority 5 to priority 4, or simply make a list of applications that have priority 5:

1. Select option 5 from the MAINTAINING APPLICATION DESCRIPTIONS or MAINTAINING JOB DESCRIPTIONS menu to display the panel in Figure 106 on page 187.
How to run the mass update utility

2. Enter the CLEAR command on the command line to delete pending updates that you no longer need.
3. Scroll down through the list of items that can be modified using the mass update function. You can update these data items:
   - GEN - General information
   - RUN - Run cycle definitions
   - OPR - Operation data
   - INT - Internal predecessors
   - EXT - External predecessors
4. Type s beside GEN – PRIORITY and press Enter. The UPDATING DATA ITEM panel, shown in Figure 107, is displayed:

5. Enter the S row command to select a subset of applications. The SPECIFYING FILTER CRITERIA panel, shown in Figure 108 on page 188, is displayed:
How to run the mass update utility

6. Type sample in the OWNER ID field. You can also use global search characters to limit the scope of a change, typing sam*, for example, for all IDs beginning with SAM. Press PF3 (End) to return to the UPDATING DATA ITEM panel. You now see the word Filter in the Target criteria column.

7. Enter 5 in the Old data field and 4 in the New data field:

```
Row Target
  cmd criteria
    ' Filter Old data: 5
        New data: 4
```

If you specify no old data value, all applications matching the filtering criteria that have the field defined are changed to the new data value. For example, if you specify a new value of NEW RESOURCE for a resource name, and leave the old value blank, any operation with a resource defined will be updated with the new resource name: operations that did not originally have a resource defined remain unchanged.

**Note:** If the old data string is found anywhere in the target field data, it is replaced by the new data string. If the new data string is longer than the old data string, the target field can be truncated, as shown in this example:

Before the mass update:

8. The target field is 8 bytes long.

9. Three instances of the target field contain PAYUPD, PAYUPDAT, and UPDATE.

10. Old data is UPD.

11. New data is BKUP.

After the mass update:

12. The updated fields contain PAYBKUP, PAYBKUPA, and BKUPATE respectively.

13. Press PF3 (End). You return to the MASS UPDATING OF APPLICATION DESCRIPTION panel.

14. When you request a trial run, a batch job is submitted that produces a report of all specified pending updates without updating the database. An update run generates a batch job that updates the database and produces a report of
How to run the mass update utility

all changed fields. Type t in the TYPE OF BATCH JOB field to specify a trial run, and press PF3 (End). The GENERATING JCL FOR A BATCH JOB panel, shown in Figure 109, is displayed:

<table>
<thead>
<tr>
<th>Command</th>
<th>Enter/change data below and press ENTER to submit/edit the JCL.</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCL to be generated for: MASS UPDATE OF APPLICATIONS</td>
<td></td>
</tr>
<tr>
<td>SYSOUT CLASS</td>
<td>====&gt; _ (Used only if output to system printer)</td>
</tr>
<tr>
<td>LOCAL PRINTER NAME</td>
<td>====&gt; _______ (Used only if output on local printer)</td>
</tr>
<tr>
<td>DATASET NAME</td>
<td>====&gt; XRAYNER.EIDA.ADMUP.LIST (Used only if CLASS and LOCAL PRINTER are both blank). If blank default name used is XRAYNER.EIDA.ADMUP.LIST</td>
</tr>
<tr>
<td>SUBMIT/EDIT JOB</td>
<td>====&gt; S S to submit JOB, E to edit</td>
</tr>
<tr>
<td>Job statement :</td>
<td></td>
</tr>
<tr>
<td>====&gt; //XRAYNERE JOB (890122,NOBO),SIMON RAYNER,</td>
<td></td>
</tr>
<tr>
<td>====&gt; // MSGCLASS=H,NOTIFY=XRAYNER,CLASS=A</td>
<td></td>
</tr>
<tr>
<td>====&gt; //OUTPUT1 OUTPUT DEST=LAB21,DEFAULT=YES</td>
<td></td>
</tr>
<tr>
<td>====&gt; //*</td>
<td></td>
</tr>
</tbody>
</table>

Figure 109. EQQXSUBP - Generating JCL for a batch job

15. Check the report to see the applications that have been found with priority 5.

16. To update the AD database, rerun the job, this time with u in the TYPE OF BATCH JOB field.

Note: If you want to use mass update to change operation input arrival day or time, only those operations are changed where all operations in the application already have a value in both fields. You cannot leave these fields blank and then change them with mass update. Also, note that with mass update, no check is made to ensure that the deadline day and time is later than the operation input arrival day and time.

What is the batch loader?

Use the batch-loader program in Tivoli Workload Scheduler for z/OS to create and update application descriptions, group definitions, and operator instructions, in the AD database and the OI database. With the batch loader, you can perform in the batch environment some of the functions you would otherwise perform online with the Tivoli Workload Scheduler for z/OS panels.

This chapter does not describe fully all the application options. Read "Things you should consider before creating applications" on page 115 and "Standard applications and group definitions" on page 119 before using the batch loader.

Input

Input to the batch loader is a set of control statements in an input data set. These control statements, which you provide, describe your applications or operator instructions, or both.
Output

Output from the batch loader can be for either an AD database or an OI database, or both. The output can be directed to:
- Databases allocated to an active Tivoli Workload Scheduler for z/OS subsystem
- A database that you set up yourself, that is, a VSAM key-sequenced data set (KSDS)

If the batch-loader output is directed to an active Tivoli Workload Scheduler for z/OS subsystem, you specify which Tivoli Workload Scheduler for z/OS subsystem the information should be directed to. You can also specify if your AD or OI information is new and should be added to the database, or if it should replace existing information.

If the batch-loader output is directed to a VSAM data set, the batch-loader program is run independently of any Tivoli Workload Scheduler for z/OS subsystem; it is not necessary to start the Tivoli Workload Scheduler for z/OS subsystem. This independently created database can then be allocated to an active Tivoli Workload Scheduler for z/OS subsystem at a later time.

Validity checking
The batch loader checks the validity of the information you provide in the input data set. However, validity checking is optional if the output is directed to a VSAM data set. You can also instruct the batch loader to perform only validity checking, without producing output. In this case, only basic syntax checking of the control statements is performed.

Description of the databases
A brief description of the information in, and structure of, the AD and OI databases follows. This background information should help you understand and code the batch-loader control statements.

AD database
The basic unit of information in the AD database is an application description (AD), containing all the information about one application. An AD can contain:

General information
This section contains information, such as the application name, status, and owner, which uniquely identifies the application and must appear only once in each AD. You build this section with an ADSTART control statement.

Run cycle definitions
These definitions describe when the application should run, such as daily or weekly. The Tivoli Workload Scheduler for z/OS planning functions convert this information into calendar run dates. There can be more than one run cycle in each AD; for example, if a particular application must be run daily with an additional run at the end of each week. You specify a run cycle with an ADRUN control statement.

Operations
An application usually consists of a series of related operations, such as:
- JCL setup
- Execution of a job or started task
- Printing and dispatch of reports
AD database

Each operation is described by its own section, which contains a unique operation number and the name of the workstation. There are normally multiple operation sections in each AD. You build an operation section with an ADOP control statement.

An operation section can have one or more of these subsections:

Dependencies
Each operation in an application can have one or more dependencies, or predecessor operations, which must be completed before it can start. A predecessor operation can be part of the same occurrence of the application (internal dependency), part of another application (external dependency), or part of another occurrence of the same application (external dependency). Within each operation section in the AD, there can be one or more dependency subsections, one for each dependency. You create a dependency subsection with an ADDEP control statement.

Special resources
Each operation can use one or more special resources. A special resource subsection contains details of how the operation uses the resource. There can be one or more resource subsections within each operation section. You create a resource subsection with an ADSR control statement. See Chapter 5, “Creating special resources,” on page 81 for information about special resources.

Operation Extended Info
Within an application every operation can have its own extended information area, so that there can be one operation extended name subsection per operation. You create or delete an operation extended information subsection with an ADOPEXTN control statement. The extended information area contains two types of information:

- Extended job name
- Scheduling Environment name

System Automation Info
Within an application every operation can have its own System Automation information area, so that there can be one operation system automation subsection per operation. The subsection is present only for operations running on automation workstations. The System Automation information area contains the following data:

- Command text
- Automated function
- Security element
- Completion information

User Fields
Each operation can use one or more user fields. A user field subsection contains details about user field name and value. There can be one or more user fields subsections within each operation section. You create a user field subsection with an ADUSF control statement.

Remote Job Info
Within an application, every operation can have their own Remote Job information area. The subsection is present only for operations running on remote workstations. The Remote Job information area contains the following data:

- ADID or job stream name
**AD database**

- Operation number (Tivoli Workload Scheduler for z/OS only)
- Job stream workstation (Tivoli Workload Scheduler only)
- Job name (Tivoli Workload Scheduler only)
- Complete on failed bind (Y | N)

**OI Database**

The basic unit of information in the OI database is the *operator instruction* (OI), which contains instruction text for a particular operation in an application. Each OI contains these sections:

**Identification**

The information in this section identifies the application operation that the operator instruction belongs to. The application is identified by its name. The operation can be identified by the operation number, the job name, or the name of the workstation where the operation takes place. This section can appear only once in each OI. You build the identification section with an OISTART control statement.

**Text**

Each OI can contain one or more text sections. Each text section contains one line of text and is built with an OIT control statement.

---

**How to code batch-loader control statements**

The batch-loader control statements closely reflect the structure of the AD and OI databases. Each AD or OI consists of one or more sections. One batch-loader control statement is required for each section that describes your environment.

**Structure of AD control statements**

The ADSTART control statement tells the batch loader that you want to start building a new AD and builds the general information section. Also, when an ADSTART is found in the input data set, it signals the batch loader to complete the previous AD or OI being built and write it to the database.

Put statements representing the remaining sections of the AD record after ADSTART. The sequence of control statements describing one AD is not important, except:

- ADDEP, ADOPEXTEN, ADSR, ADOPSAI, ADUSF, and ADRE must follow the ADOP to which they belong, because they build subsections of an operation section.
- An ADRULE statement must immediately follow its associated ADRUN statement.

The “[Example of control statement sequence” on page 193](#) illustrates this. Place control statements in a logical sequence to avoid mistakes and to make the input easier to read.

**Structure of OI control statements**

The OI control statements are used to build an operator instruction in the OI database. You start with an OISTART control statement, followed by up to 443 OIT control statements that provide the text for the operator instruction (or the text can be in a separate file).
Example of control statement sequence

This example shows the structure and sequence of batch-loader control statements. Do not worry if you do not understand all of it, because only the basic structure is important. The parameters are described in detail later.

```
OPTIONS SUBSYS(EIDA) CHECK(Y)
ADSTART ACTION(SETDEFAULT) ADTYPE(A) DESC('PAYROLL SAMPLE')
ODYSCR('SAMPLE APPLICATION') PRIORITY(5)
OWNER(SAMPLE)
ADRULE ACTION(SETDEFAULT) TYPE(R)
ADSTART ADID(PAYDAILY) DESC('DAILY PAYROLL JOBS')
ADRULE EVERY DAY(WORKDAY) YEAR
ADOP WSID(SETP) OPNO(10) JOBN(PAYDAILY)
DESC('SETUP FOR PAYDAILY') DURATION(5)
ADOP WSID(CPU1) OPNO(20) JOBN(PAYDAILY)
DESC('PAYDAILY JOB') DURATION(5) PREOP(10)
ADOP WSID(SETP) OPNO(20) JOBN(PAYW)
DESC('SETUP FOR PAYWEEK') DURATION(5)
ADOP WSID(CPU1) OPNO(20) JOBN(PAYW)
DESC('PAYWEEK JOB') DURATION(5) PREOP(10)
ADDEP PREWSID(OPNO(10)) PREADID(PAYDAILY)
DESC('WAIT FOR PAYDAILY')
ADOPEXTN EXTNAME(' ') OISTART ADID(PAYW) OPNO(020)
OIT 'Please note...'
OIT 'If this job (PAYWEEK) fails, automatic recovery will be attempted for some situations.'
```

Table 18 illustrates the differences between updating the active subsystem or using a VSAM data set.

<table>
<thead>
<tr>
<th>If you update the active subsystem:</th>
<th>If you use independent VSAM data sets:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Tivoli Workload Scheduler for z/OS subsystem must be active.</td>
<td>The Tivoli Workload Scheduler for z/OS subsystem need not be active.</td>
</tr>
<tr>
<td>Code the SUBSYS keyword on the OPTIONS statement.</td>
<td>Provide the EQQADDS, EQQAD2DS, EQQOIDS, EQQWSDS ddnames in the batch-loader JCL, and the associated data sets. See &quot;How to run the batch-loader program&quot; on page 194.</td>
</tr>
<tr>
<td>You can use the added application descriptions and operator instructions as soon as the job ends.</td>
<td>You must shut down the Tivoli Workload Scheduler for z/OS subsystem, allocate the new data sets, and restart the subsystem, before the new application descriptions and operator instructions can be used.</td>
</tr>
<tr>
<td>You might have problems if the batch loader goes wrong, and leaves incomplete application descriptions and operator instructions in the active database.</td>
<td>You can delete and redefine the VSAM data sets and rerun the batch loader if things go wrong.</td>
</tr>
</tbody>
</table>
Example of control statement sequence

<table>
<thead>
<tr>
<th>If you update the active subsystem:</th>
<th>If you use independent VSAM data sets:</th>
</tr>
</thead>
<tbody>
<tr>
<td>There can be performance problems if your Tivoli Workload Scheduler for z/OS subsystem is busy writing to the active database. You might have to turn off AUDIT for the AD and OI databases to stop the audit file filling up.</td>
<td>There is little impact on the active Tivoli Workload Scheduler for z/OS subsystem.</td>
</tr>
</tbody>
</table>

**Batch-loader security considerations**

If you are updating the databases of a Tivoli Workload Scheduler for z/OS subsystem, you need the same authority that would be required if you were to make the updates using the Tivoli Workload Scheduler for z/OS panels. You will need access to these resource codes:

<table>
<thead>
<tr>
<th>AD</th>
<th>For the application descriptions</th>
</tr>
</thead>
<tbody>
<tr>
<td>OI</td>
<td>For the operator instructions</td>
</tr>
</tbody>
</table>

**How to run the batch-loader program**

This section describes the requirements for executing the batch loader. It also describes the batch-loader report and provides sample JCL.

**Coding the JCL**

The Tivoli Workload Scheduler for z/OS sample library (SEQQSAMP) contains samples to execute the batch loader. For the data set name of the library, see your system programmer.

These samples are provided:

- EQQBSCAN uses the scan action to syntax check an AD.
- EQQBSUBS creates three new application descriptions and operator instructions through the controller.
- EQQBVVSAM updates VSAM data sets directly with new application descriptions and operator instructions.

If you use the batch loader a lot, you might find it easier to allocate a PDS and create a member for each batch-loader request type. This ensures that syntax errors are not repeated, and could save you time. You can use the samples provided in SEQQSAMP as the basis for your batch-loader PDS.

The following is a sample JCL for executing the batch loader:

```jcl
//XRAYNERE JOB (890122,N0BO),'SIMON RAYNER',
// MSGCLASS=H,NOTIFY=XRAYNER,CLASS=A
//STEP1 EXEC PGM=EQQYLTOP
//STEPLIB DD DSN=OPCESA.VvRrMm.LOAD,DISP=SHR
//EQQMLIB DD DSN=OPCESA.VvRrMm.MSGLIB,DISP=SHR
//EQQMLOG DD SYSOUT=*
//EQQDUMP DD SYSOUT=*
//EQQUDUMP DD SYSOUT=*
//** -- Required if your operator instructions are in
//** -- separate members
//** //EQQOPDS DD DSN=XRAYNER.OPCESA.MESSAGES,DISP=SHR
//SYSIN DD DSN=XRAYNER.OPCESA.BATCH(PAYROLL),DISP=SHR
//** -- Required if you are writing to VSAM data sets
```
The program name EQQYLTOP is required on the EXEC statement in your JCL. EQQYLTOP should reside in the Tivoli Workload Scheduler for z/OS load module library, or at least in an APF-authorized library. If this library is not defined in the active z/OS LNKLIST member of SYS1.PARMLIB, you must provide a STEPLIB DD statement in your JCL so that EQQYLTOP can be found.

The data set requirements depend on whether you are directing the batch-loader output to a Tivoli Workload Scheduler for z/OS subsystem or to VSAM data sets. If the output is directed to VSAM data sets, you must allocate them with the correct VSAM data set characteristics for an AD or OI database. The Planning and Installation and the sample library have JCL and IDCAMS control statements for creating an AD or OI database.

The ddnames for the required data sets are listed here with a description of their function:

- **SYSIN**
  The batch-loader input data set that contains your control statements. You can use a disk or tape data set, with fixed blocked 80-byte records, or use SYSIN DD * and place the control statements in your JCL.

  - **SYSIN** data cannot exceed 72 characters. For details, see "Syntax and construction" on page 200.

- **EQQMLIB**
  The Tivoli Workload Scheduler for z/OS message library.

- **EQQMLOG**
  Message log. This can be allocated to a data set or to SYSOUT.

- **EQQDUMP**
  Diagnostics data set. If batch-loader output is directed to a VSAM data set, diagnostic information is written to this data set during validity checking of the records in the input data set. If output is directed to a Tivoli Workload Scheduler for z/OS subsystem, diagnostic information is written to the diagnostics data set of that subsystem.

- **SYSUDUMP**
  For z/OS dumps.

- **EQQOIPDS**
  Required if the MEMBER keyword is specified on an OISTART control statement. EQQOIPDS specifies a partitioned data set, fixed blocked with a record length of 80 bytes, which contains operator instruction text.

- **EQQADDS**
  Required if output is being directed to a VSAM data set. Specifies the VSAM data set that is to receive the AD information. This DD statement is also used to check:
  - That applications and operations identified on ADDEP control statements exist.
  - That applications and operations identified on OISTART control statements exist.
  - That JCL variable tables identified on ADRUN control statements exist.

- **EQQAD2DS**
  This optional DD statement is used only if batch loader output is directed to a VSAM data set. If present, it is used with EQQADDS to check that applications and operations on ADDEP and OISTART statements exist. You might need to specify this ddname if you are...
merging two sets of application descriptions (the batch-loader statements and an old database, represented by this ddname) into a new database, represented by the EQQADDS ddname.

If no match is found on the EQQADDS data set, Tivoli Workload Scheduler for z/OS searches:

- EQQAD2DS for an application description that can be the predecessor.
- The operator instruction database for a matching application description.

No information is ever written to EQQAD2DS.

**EQQOIDS**
Required only if output is directed to a VSAM data set. EQQOIDS specifies the VSAM data set that is to receive OI information.

**EQQWSDS**
Required only if output is directed to a VSAM data set. EQQWSDS specifies the data set name of your existing workstation description database. This is required so that the batch loader can check that the specified workstations, calendars, and periods exist.

**Batch-loader reports**

The only report produced by the batch loader is a list of messages, written to the data set specified by the EQQMLOG DD statement. Refer to this report to find syntax or validation errors in your control statements. The batch-loader message log must not be the same message log that your Tivoli Workload Scheduler for z/OS subsystem uses, but if you have directed the batch-loader output to a Tivoli Workload Scheduler for z/OS subsystem, some messages might be written to the EQQMLOG data set that is allocated to that subsystem. Errors related to the request itself, such as a syntax error in a parameter argument, result in a message being written only to the EQQMLOG that is allocated in the batch-loader JCL.

If the batch loader terminates with message EQQX268W, the database is being used by another function. When the batch loader has a completed database record to store, it enqueues on the database. If the database is busy, the batch loader waits for a maximum of 90 seconds for the database to become available. Normally, only the planning programs or mass update hold the database for more than 90 seconds.

To avoid failures like this, allocate a data set such as the dump data set with disposition OLD or MOD in all Tivoli Workload Scheduler for z/OS batch programs. This ensures that should there be a severe error, multiple dumps are not overwritten, and a beneficial side effect is serialization of batch programs requiring access to the databases.

To verify that the AD or OI information that has been created accurately reflects your environment, use the Tivoli Workload Scheduler for z/OS ISPF panel to print application descriptions or operator instructions. Several reports are available.

**Batch-loader sample**

These batch-loader sample statements create applications for the Paymore example.

```
OPTIONS ACTION(SCAN) /* SYNTAX CHECK ONLY */
SUBSYS(EIDA) /* SUBSYSTEM IS EIDA */
ADSTART /* SET DEFAULTS FOR SUBSEQUENT */
ACTION(SETDEFAULT) /* ADSTART STATEMENTS. */
ADTYPE(A) /* APPLICATION TYPE */
DESC('PAYROLL SAMPLE') /* DESCRIPTION TEXT */
```
Batch-loader sample

ODESC('SAMPLE APPLICATION') /* OWNER DESCRIPTION */
PRIORITY(5) /* PRIORITY */
OWNER(SAMPLE) /* OWNER ID */
ADRUN /* SET DEFAULTS FOR SUBSEQUENT */
ACTION(SETDEFAULT) /* ADRUN STATEMENTS. */
TYPE(R) /* USE RULE-BASED RUN CYCLES */
ADSTART ADID(PAYDAILY) /* DEFINE PAYDAILY */
DESC('DAILY PAYROLL JOBS') /* */
ADRUN NAME(DAILY) /* SPECIFY RULE-BASED RUN CYCLE */
DESCR('RUN EVERY WORK DAY') /* */
IATIME(1200) /* INPUT ARRIVAL TIME */
DLTIME(1600) /* DEADLINE TIME */
ADRULE EVERY DAY(WORKDAY) /* SPECIFY EVERY WORK DAY IN THE YEAR */
YEAR /* SELECT ALL DAYS IN THE WEEK */
ADOP WSID(WTO1) /* SEND A WTO TO CLOSE THE PAYROLL */
OPNO(05) /* DATASET. */
JOBN(PAYDAILY) /* THE DESC KEYWORD SPECIFIES THE */
DESC('PAYX CLOSE DATASET') /* OPERATION TEXT, WHICH CAN */
DURATION(1) /* BE USED BY THE RECEIVER OF THE WTO */
TIME(Y) /* START THIS JOB AT IATIME */
ADOP WSID(SETP) /* DEFINE THE SETUP OPERATION (010) */
OPNO(10) /* FOR THE PAYDAILY JOB */
JOBN(PAYDAILY) /* */
DESC('SETUP FOR PAYDAILY') /* */
DURATION(5) /* */
PREOP(05) /* WTO IS PREDECESSOR */
ADOP WSID(CPU1) /* DEFINE THE PAYDAILY JOB (020) */
OPNO(20) /* */
JOBN(PAYDAILY) /* */
DESC('PAYDAILY JOB') /* */
DURATION(5) /* ESTIMATED JOB DURATION */
PREOP(10) /* JCL SETUP IS PREDECESSOR */
ADSR RES(PAYROLL.DATABASE) /* ASK FOR EXCLUSIVE USE OF THE */
USAGE(X) /* PAYDAILY DOES NOT RUN ALONGSIDE OTHER */
ADOPEXTN EXTNAME('CALCULATE SIMPLE PAYWEEK JOB') /* */
ADSTART ADID(GPAYW) /* DEFINE THE WEEKLY PAYROLL GROUP */
DESCR('WEEKLY PAYROLL GROUP') /* */
ADTYPE(G) /* APPLICATION TYPE IS G (GROUP) */
ADRUN NAME(THURS) /* SPECIFY THE RUN CYCLE FOR THE */
RULE(1) /* WEEKLY PAYROLL GROUP DEFINITION */
DESCR('RUN ON THURSDAY') /* RUN ON THURSDAY OR ON THE PREVIOUS */
IATIME(1200) /* DAY (FREE-DAY RULE 1) IF THURSDAY IS */
DLTIME(1600) /* A FREE DAY. */
ADRULE EVERY DAY(THURSDAY) /* */
YEAR /* THIS COULD ALSO BE WEEK OR MONTH */
ADSTART ADID(PAYW) /* THIS WEEKLY JOB IS PART OF GPAYW, SO */
DESCR('WEEKLY PAYROLL JOBS') /* IT DOES NOT HAVE ITS OWN ADRUN. */
ADGROUP(GPAYW) /* */
ADOP WSID(CPU1) /* */
OPNO(20) /* */
JOBN(PAYWEEK) /* */
DESC('PAYWEEK JOB') /* */
DURATION(5) /* */
ADOP PREWSID(CPU1) /* */
PREOP(020) /* */
PREADID(PAYDAILY) /* */
DESC('WAIT FOR PAYDAILY') /* */
ADSR RES(PAYROLL.DATABASE) /* ASK FOR EXCLUSIVE USE OF THE */
USAGE(X) /* PAYWEEK DOES NOT RUN ALONGSIDE OTHER */
ADOP WSID(CPU1) /* THE PAYSILPS JOB. */
OPNO(30) /* */
JOBN(PAYWSLIP) /* */
DESC('PAYWSLIP JOB') /* */
DURATION(5) /* */
PREOP(20) /* */
Batch-loader sample

ADOP WSID(PRT1) /* TRACK THE PRINTING OF THE JES */
  OPNO(90) /* OUTPUT GROUP. */
  JOBN(PAYWSLIP) /* SAME NAME AS THE PREDECESSOR JOB. */
  DESC('PAYWSLIP PRINT') /* */
  DURATION(20) /* */
  PREOP(30) /* */
  PRTCLASS(D) FORM(SLIPS) /* TRACKS THIS SYSOUT CLASS AND FORM NO */
ADOP WSID(PAY1) /* TRACK THE COLLATION OF PAY SLIPS */
  OPNO(95) /* AND PREPARATION OF PAY PACKETS */
  JOBN(PAYWSLIP) /* IN THE PAY OFFICE. */
  DESC('PAY PACKETS') /* */
  DURATION(60) /* */
  PREOP(90) /* */
ADOPEXTN EXTNAME(' ') /* DELETE THE EXTENDED NAME INFO */
ADSTART ADID(GPAYM) /* DEFINE THE MONTHLY PAYROLL GROUP */
  DESC('MONTHLY PAYROLL GROUP') /* */
  ADTYPE(G) /* APPLICATION TYPE IS G (GROUP) */
ADRUN NAME(FIRSTTHU) /* SPECIFY THE RUN CYCLE FOR THE */
  RULE(1) /* MONTHLY PAYROLL GROUP DEFINITION */
  DESC('RUN ON THURSDAY') /* RUN ON THE THIRD THURSDAY IN MONTH */
  IATIME(1200) /* OR DAY BEFORE(RULE 1) IF THURSDAY IS */
  DLTIME(1600) /* A FREE DAY. */
ADRUL ONLY(3) /* */
DAY(THURSDAY) MONTH /* */
ADSTART ADID(PAYM1) /* THIS MONTHLY JOB IS PART OF GPAYM, SO*/
  DESC('MONTHLY PAYROLL JOBS') /* IT DOES NOT HAVE ITS OWN ADRUN. */
  ADGROUP(GPAYM) /* */
ADOP WSID(CPU1) /* */
  OPNO(40) /* */
  JOBN(PAYMONTH) /* */
  DESC('PAYMONTH JOB') /* */
  DURATION(5) /* */
ADDEP PREWSID(CPU1) /* */
  PREOP(020) /* */
  PREADID(PAYDAILY) /* */
  DESC('WAIT FOR PAYDAILY') /* */
ADSR RES(PAYROLL.DATABASE) /* */
  USAGE(X) /* */
ADOP WSID(CPU1) /* THE PAYSILPS JOB. */
  OPNO(50) /* */
  JOBN(PAYMSLIP) /* */
  DESC('PAYMSLIP JOB') /* */
  DURATION(5) /* */
ADDEP PREWSID(CPU1) /* */
  PREOP(020) /* */
  PREADID(PAYDAILY) /* */
  DESC('PAYMSLIP PRINT') /* */
ADOP WSID(PRT1) /* TRACK THE PRINTING OF THE JES */
  OPNO(99) /* OUTPUT GROUP. */
  JOBN(PAYMSLIP) /* SAME NAME AS THE PREDECESSOR JOB. */
  DESC('PAYMSLIP PRINT') /* */
  DURATION(20) /* */
  PREOP(50) /* */
  PRTCLASS(D) FORM(SLIPS) /* TRACKS THIS SYSOUT CLASS AND FORM NO */
ADSTART ADID(PAYM2) /* THIS MONTHLY JOB IS PART OF GPAYM, SO*/
  DESC('MONTHLY GIRO') /* IT DOES NOT HAVE ITS OWN ADRUN. */
  ADGROUP(GPAYM) /* */
ADOP WSID(CPU1) /* PAYTRANS IS THE JOB THAT SENDS */
  OPNO(40) /* INFORMATION ABOUT MONTHLY PAYMENTS */
  JOBN(PAYTRANS) /* TO THE BANKS. */
  DESC('GIRO TRANSFER') /* */
  DURATION(5) /* */
ADDEP PREWSID(CPU1) /* */
  PREOP(040) /* */
  PREADID(PAYMONTH) /* */
  DESC('WAIT FOR PAYMONTH') /* */
ADDEP PREWSID(CPU1) /* */
  PREOP(020) /* */
  PREADID(PAYWEEK) /* */
Batch-loader sample

DESC('WAIT FOR PAYWEEK') /* */
ADSR RES(PAYROLL.DATABASE) /* */ USAGE(X) /* */
ADSR RES(TAPES) /* */ USAGE(X) QUANTITY(1) /* EXCLUSIVE USE OF ONE TAPE DRIVE */
ADSTART ADID(PAYTAXYR) /* DEFINE THE YEARLY TAX RUN */
DESC('YEARLY TAX RUN') /* */
ADRUN NAME(FSTTHU7) /* SPECIFY RUN CYCLES */
DESCR('RUN 3RD THU IN JULY') /* */
RULE(1) /* RUN DAY BEFORE IF A FREE DAY */
IATIME(1200) /* INPUT ARRIVAL TIME */
DLTIME(2000) /* DEADLINE TIME (SAME DAY ASSUMED) */
ADRULE ONLY(3) /* GENERATE THE THIRD THURSDAY IN JULY */
DAY(THURSDAY) MONTH(JULY) /* */
ADOP WSID(CPU1) /* DEFINE THE PAYTAXYR JOB */
OPNO(15) /* */
JOBN(PAYTAXYR) /* MUST BE THE SAME NAME AS JCL MEMBER */
DESC('PAYTAXYR JOB') /* */
DURATION(5) /* ESTIMATED JOB DURATION */
ADDEP PREWSID(CPU1) /* */
PREOP(040) /* */
PREADID(PAYM2) /* DEPEND ON THE MONTHLY GIRO JOB */
DESC('WAIT FOR PAYTRANS') /* */
ADSR RES(PAYROLL.DATABASE) /* ASK FOR EXCLUSIVE USE OF THE */
USAGE(X) /* RESOURCE PAYROLL.DATABASE */
ADSTART ADID(PAYRECOV) /* RECOVERY JOB. */
DESCR('RECOVER PAYROLL') /* RUN AS REQUIRED, SO NO ADRUN. */
ADOP WSID(CPU1) /* */
OPNO(20) /* */
JOBN(PAYRECOV) /* */
DESC('PAYRECOV JOB') /* */
DURATION(5) /* */
ADSR RES(PAYROLL.DATABASE) /* */
USAGE(X) /* */
ADSR RES(TAPES) /* */
USAGE(X) QUANTITY(1) /* EXCLUSIVE USE OF ONE TAPE DRIVE */
ADSTART ADID(PAYQUERY) /* QUERY JOB. */
DESCR('AD-HOC QUERY') /* RUN AS REQUIRED, SO NO ADRUN. */
ADOP WSID(CPU1) /* */
OPNO(50) /* */
JOBN(PAYQUERY) /* */
DESC('PAYQUERY JOB') /* */
DURATION(5) /* */
ADSR RES(PAYROLL.DATABASE) /* */
USAGE(X) /* */
ADSTART ADID(PAYBACKP) /* BACKUP JOB. */
DESCR('BACKUP PAYROLL') /* RUN AFTER ALL OTHER PAYROLL JOBS. */
ADRUN NAME(DAILY) /* SPECIFY RULE-BASED RUN CYCLE */
DESCR('RUN EVERY WORK DAY') /* */
IATIME(1200) /* INPUT ARRIVAL TIME */
DLTIME(1600) /* DEADLINE TIME */
ADRULE EVERY DAY(WORKDAY) /* SPECIFY EVERY WORK DAY IN THE YEAR */
YEAR /* SELECT ALL DAYS IN THE WEEK */
ADOP WSID(CPU1) /* */
OPNO(15) /* */
JOBN(PAYBACKP) /* */
DESC('PAYBACKP JOB') /* */
DURATION(5) /* */
ADSR RES(PAYROLL.DATABASE) /* */
USAGE(X) /* */
ADDEP PREWSID(CPU1) /* ALL THESE */
PREADD(PAYDAILY) /* DEPENDENCIES */
PREOP(020) /* ARE EFFECTIVE */
DESC('WAIT FOR PAYDAILY') /* ONLY */
ADDEP PREWSID(CPU1) /* IF THE */
Batch-loader sample

```
PREAD(PAYSLIP) /* SPECIFIED */
PREOP(030) /* OPERATION */
DESC('WAIT FOR PAYSLIP') /* IS SCHEDULED */
ADDEP PREWSID(CPU1) /* FOR THE SAME DAY */
PREAD(PAYSPLIP) /* AS PAYBACKP. */
PREOP(050) /* SO, IF IT IS A NORMAL */
DESC('WAIT FOR PAYMSLIP') /* MONDAY, FOR EXAMPLE, */
ADDEP PREWSID(CPU1) /* PAYBACKP IS DEPENDENT */
PREAD(PAYTRANS) /* ONLY ON PAYDAILY. */
PREOP(040) /* */
DESC('WAIT FOR PAYTRANS') /* */
ADDEP PREWSID(CPU1) /* */
PREAD(PAYTAXYR) /* */
PREOP(015) /* */
DESC('WAIT FOR PAYTAXYR') /* */
ADOP WSID(WTO1) /* SEND A WTO WHICH TRIGGERS A CICS */
OPNO(30) /* TRANSACTION TO REOPEN THE DATASET */
JOBN(PAYBACKP) /* */
DESC('PAYX OPEN DATASET') /* THIS TEXT CAN BE USED AS THE COMMAND */
DURATION(1) /* */
PREOP(15) /* INTERNAL DEPENDENCY ON BACKUP JOB */
/*
```

Batch-loader control statements

The batch-loader control statements are:

- **ADCNC**  Specifies a condition for the operation being built
- **ADCNS**  Specifies a condition dependency for the operation being built
- **ADDEP**  Specifies an operation that is a predecessor to the operation being built
- **ADOP**   Defines an operation in the application
- **ADOPEXTN**  Specifies extended information for the operation (extended name, Scheduling Environment name, or both)
- **ADOPSAI**  Defines the system automation information for the operation
- **ADRE**   Defines the remote job information for the operation
- **ADRULE**  Provides additional information to the ADRUN statement for rule-based run cycles
- **ADRUN**  Describes when the application is to run
- **ADSR**   Specifies special resources required by the operation
- **ADSTART**  Starts the creation of a new application description
- **ADUSF**  Creates a user field for the operation
- **OISTART**  Starts the creation of a new operator instruction
- **OIT**    Specifies a line of text in the operator instruction that is under construction
- **OPTIONS**  Defines execution options for the batch loader

Syntax and construction

Batch loader control statements follow the same syntax rules as TSO commands.
Syntax and construction

Begin each statement on a new line. A control statement is the statement name followed by keyword parameters with the value of the parameter in parentheses, like this:

```
controlstatementname  keyword(value)  keyword(value) ...
```

Some keywords can have more than one value, for example IADAYS(1,2,3,4).
Blanks separate the control statement name and the keywords. For example:
ADSTART  ADID(APPLNAME)  DESCR('Application Description')

Control statements must not exceed column 72 and can continue over two or more lines. Continuation characters are not required. Any information in columns 73 through 80 is ignored. Any parameter that begins before column 72, and ends after this column, is ignored with no error message or non-zero return code being issued.

You can abbreviate keywords down to their shortest unambiguous form in the current statement. Control statement names cannot be abbreviated.

The statements can include comments in the form

```
/* comment */
```

Keyword values containing delimiters, like blanks, should be written with single quotation marks. See *TSO Extensions Command Language Reference* for details of the syntax rules.

Defaults

When coding batch-loader control statements, you do not need to provide every keyword and a corresponding value. Keywords that are required are clearly noted in the text. For other keywords, a default value is used for that keyword if you do not supply it.

Tivoli Workload Scheduler for z/OS selects default values in this sequence:

1. Some control statements have the ACTION keyword. By specifying 
   ACTION(SETDEFAULT), you can provide your own default values for that 
   control statement. These default values will remain in effect for all later 
   occurrences of the control statement in the input data set, or until you specify 
   another ACTION(SETDEFAULT). For example:

   ADOP  ACTION(SETDEFAULT)  OPNO(10)  DURATION(5)

   In this example, the default values set for the keywords OPNO and 
   DURATION are used for all following ADOPs in the input data set. When you 
   specify a default value for the OPNO keyword (operation number), it is used as 
   an increment rather than an absolute value. This is because each operation 
   within an application must have a unique operation number. A statement with 
   ACTION(SETDEFAULT) as the only keyword removes all defaults that have 
   been previously set in this way and returns the defaults to their normal values. 
   A control statement with ACTION(SETDEFAULT) specified does not cause any 
   creation of AD or OI information.

   Keywords that cannot be set by the SETDEFAULT command are listed in the 
   text.

2. A keyword parameter can have a specific default as shown in the description of 
   that control statement.

3. If no default value is obtained from the preceding two steps, the following 
   rules apply when you omit the keyword:
Defaults

a. A character value defaults to blank.
b. An integer value defaults to 0.

The following sections describe purpose and format of the supported statements.
ADCNC

Purpose
Use the ADCNC statement when defining conditional dependencies for an operation, to specify a condition.

Format

```
ADCNC
  ACTION(SETDEFAULT)
  CONDID(condition ID)
  CONDDEPNO(number of condition dependency)
  CONDCOUNT(condition counter)
  CONDDESC('descriptive text')
```

Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for these keywords:
- CONDID
- CONDDEPNO

Parameters

**ACTION (SETDEFAULT | ADD)**
If you specify SETDEFAULT, all the other values that you set in the ADCNC statement become the default values for all the ADCNC statements that follow. The application description database is not updated. The parameters that you do not set, are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an update of the database.

**CONDID(condition ID)**
Condition identifier. Valid values are from 1 to 999.

**CONDDEPNO(number of condition dependencies)**
Number of condition dependencies to be included in this condition, each one corresponding to an ADCNS statement.

**CONDCOUNT(condition counter | 0)**
Condition counter. Use it to define the rule type:
- 0 All the condition dependencies in this condition must be true.
- n At least n out of the condition dependencies in this condition must be true.

**CONDDESC('descriptive text')**
A free-format description of the condition. It can be up to 16 characters and must be contained within single quotation marks. Do not include delimiters, such as parentheses and single quotation marks, in the description.

Examples
This example defines a condition including two condition dependencies:
ADCNC

ADCNC CONDID( 3) CONDDEPNO(2) CONDCOUNT(1)
CONDDESCR('WITH 2 COND.DEPS')
ADCNS ...
ADCNS ...
ADCNS

Purpose
Use the ADCNS statement when defining condition dependencies for an operation.

Format

```
ADCNS
  ACTION(SETDEFAULT | ADD)
  CONDEPCONDID(owning condition ID)
  CONDEPPREADID(predecessor application ID)
  CONDEPPREWSID(predecessor workstation)
  CONDEPPREOPNO(predecessor operation number)
  CONDEPPROCSNAME(step name)
  CONDEPSTEPNAME(procedure invocation step name)
  CONDEPDEPTYPE(e | ST)
  CONDEPLOG(EQ | GE | GT | LE | LT | NE | RG)
  CONDEPVALST(status)
  CONDEPVALRC1(return code)
  CONDEPVALRC2(second boundary return code)
```

Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for these keywords:
  CONDEPCONDID
  CONDEPPREADID
  CONDEPPREWSID
  CONDEPPREOPNO
  CONDEPDEPTYPE
  CONDEPLOG

Parameters
ACTION (SETDEFAULT | ADD)

If you specify SETDEFAULT, all the other values that you set in the
ADCNS statement become the default values for all the ADCNS statements
that follow. The application description database is not updated. The
parameters that you do not set, are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an
update of the database.
CONDDEPCONDID(*owning condition ID*)
Owning condition identifier. Valid values are from 1 to 999.

CONDDEPPREADID(*predecessor application ID*)
If the predecessor operation is in a different application from the one being built, or is in a different occurrence of the same application, you must identify the application ID with this keyword. If you use DBCS characters, you must enter them as a quoted string started by a shift-out and ended by a shift-in.

CONDDEPPREWSID(*predecessor workstation*)
The four-character workstation name of a predecessor operation to this operation.

CONDDEPPREOPNO(*predecessor operation number*)
The operation number of a predecessor operation to this operation.

CONDDEPPROCSTEP(*step name*)
Use it to define a step level dependency. If the step is not in a procedure, this parameter identifies the job step name, otherwise it identifies the step name in the JCL procedure. It must correspond to the name of an EXEC PGM= statement.

CONDDEPSTEPNAME(*procedure invocation step name*)
Use it in conjunction with CONDDEPPROCSTEP when defining a step level dependency, only if the step is in a procedure, to identify the name of a step that invokes an in-stream or cataloged procedure. It must correspond to the name of an EXEC PROC= statement.

CONDDEPDEPTYPE(E | I)
Depending on the dependency type (internal or external), specify one of the following values:

E External predecessor.
I Internal predecessor.

CONDDEPDEPTYPE(RC | ST)
Depending on the type of check that you require on the predecessor, specify one of the following values:

ST To check the predecessor status. It does not apply to step dependencies.
RC To check the predecessor return code.

CONDDEPLOG(GE | GT | LE | LT | NE | RG | EQ)
Depending on the type of check that you require on the predecessor, specify one of the following logical operators:

EQ Equal to.
GE Greater than or equal to. Valid only for RC as CONDDEPDEPTYPE value.
GT Greater than. Valid only for RC as CONDDEPDEPTYPE value.
LE Less than or equal to. Valid only for RC as CONDDEPDEPTYPE value.
LT Less than. Valid only for RC as CONDDEPDEPTYPE value.
NE Not equal to. Use it to specify conditions on final statuses only.
RG Range.
**CONDDEPVALST** *(status)*  
Status. Valid only for ST as CONDDEPTYPE value.

**CONDDEPVALRC1** *(return code)*  
Return code value. Valid only for RC as CONDDEPTYPE value.

**CONDDEPVALRC2** *(second boundary return code)*  
Return code value, as second boundary in a range expressed by the RG logical operator.

**Examples**  
This example defines a condition including two condition dependencies:

```
ADCNC CONDID( 3) CONDDEPNO(2) CONDCOUNT(1)  
CONDESCTR('WITH 2 COND.DEPS')
ADCNS CONDDEPCONID( 3)  
CONDDEPPREADID(APPL1)  
CONDDEPPREWSID(CPU1)  
CONDDEPPREOPNO(2)  
CONDDEPDEPTYPE(E)  
CONDDEPTYPE(ST)  
CONDDEPLOG(EQ)  
CONDDEPVALST(X)
ADCNS CONDDEPCONID( 3)  
CONDDEPPREADID(APPL1)  
CONDDEPPREWSID(CPU1)  
CONDDEPPREOPNO(2)  
CONDDEPPROCSSTEP(STEP100)  
CONDDEPDEPTYPE(E)  
CONDDEPTYPE(RC)  
CONDDEPLOG(RG)  
CONDDEPVALRC1(4)  
CONDDEPVALRC2(8)
```
**ADDEP**

**Purpose**
Use the ADDEP control statement to specify a predecessor operation to the operation that is being built by the previous ADOP statement. The ADOP statement lets you specify one internal predecessor to the operation being defined. So you need to use ADDEP control statements if the operation has more than one predecessor or if it has any external predecessors.

If the predecessor operation is in a different AD from the one being built, you must specify the PREADID keyword to identify the AD containing the predecessor. You should remember, though, that there can be up to four versions of an AD with different validity dates. If there is more than one version, the batch loader searches the ADs defined by your PREADID keyword to find an AD with a current validity date. ADs with active status are searched first, followed by those with pending status.

If you do not supply the PREADID keyword, Tivoli Workload Scheduler for z/OS assumes that the predecessor operation is in the AD that is currently being built (that is, the one defined by the last ADSTART statement).

You identify the predecessor operation by one or more of the following:
- Operation number (PREOPNO)
- Workstation name (PREWSID)
- Job name (PREJOBN)

You must specify enough of these keywords to uniquely identify the predecessor operation. If no operation, or more than one operation, matches, an error message is produced and no dependency is created.

If the output is directed to a VSAM data set, the predecessor operation can be defined by an ADOP statement occurring later in the input data set. This is because most validity checking occurs after all statements in the input data set are read.

If the output is directed to an active Tivoli Workload Scheduler for z/OS subsystem and the predecessor operation does not already exist in the AD database, specify both the operation number and the workstation name in the ADDEP statement.

**Format**

```
ADDEP
ACTION(SETDEFAULT)

DESCR('predecessor description')
```
Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for the following keywords:
PREWSID
PREOPNO
PREJOBN
PREADID

Parameters
ACTION (SETDEFAULT | ADD)
If you specify SETDEFAULT, the remaining keyword values that you specify on the ADDEP statement become default values for all ADDEP statements that follow. No application description is updated. Keywords that you do not specify are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an update of the database.

DESCR (‘predecessor description’)
A free-format description of the dependency. It can be up to 50 characters and must be contained within single quotation marks. Do not include delimiters, such as parentheses and single quotation marks, in the description.

Tivoli Workload Scheduler for z/OS holds descriptions only for external dependencies. This field cannot be used to hold a description for an internal dependency.

PREADID (predecessor application ID)
If the predecessor operation is in a different application from the one being built, or is in a different occurrence of the same application, you must identify the application ID with the PREADID keyword. If you use DBCS characters, you must enter them as a quoted string started by a shift-out and ended by a shift-in.

PREWSID (predecessor workstation)
The four-character workstation name of a predecessor operation to this operation.

PREOPNO (predecessor operation number)
The operation number of a predecessor operation to this operation.
ADDEP

PREJOBN (predecessor job name)
The job name of a predecessor operation to this operation.

TRANSPT (transport time)
When Tivoli Workload Scheduler for z/OS creates the plan, it allows this
many minutes between the completion of the predecessor and the start of
the successor operation that is being defined. You must specify an integer.
The default is the time specified for the workstation.

Examples
In this example, the operation on workstation CPU1 with job name SMFCHK is
made an internal predecessor to the operation being defined. An external
predecessor is also defined: operation 020 in application PAYDAILY:

ADDEP ...
ADDEP PREADID(CPU1) PREJOBN(SMFCHK)
ADDEP PREADID(PAYDAILY) PREOP(020) DESC('WAIT FOR DAILY PAYROLL')
ADOP

Purpose
Use the ADOP control statement to add operations to an application description. Provide an ADOP statement for each operation that is to be defined in the current application.

On this statement, you can specify one internal predecessor for the operation being defined. If you need to specify external predecessors, or more internal predecessors, use one or more ADDEP statements immediately after the ADOP statement.

Note: You cannot add operations to a group definition.

Format
Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for these keywords:
- WSID
- OPNO
- JOBN
- PREWSID
- PREOPNO
- PREJOBN

Parameters

ACTION (SETDEFAULT | ADD)
If you specify SETDEFAULT, the remaining keyword values that you specify on the ADOP statement become default values for all ADOP statements that follow. No application description is updated. Keywords that you do not specify are assigned their standard defaults.
If you specify ADD or use it by default, the statement can result in an update of the database.

ADOPCATM (A | I | M | N)
The cleanup type for this operation:
- A Automatic. When the operation is ready to be submitted and the controller selects it for submissions, the controller automatically finds the cleanup actions to be taken and also inserts them as the first step in the JCL of the restarted job. Whenever the operation is started from the panels, the cleanup actions are shown to the user for confirmation, only if the CLEANUP CHECK option was set to YES through the DEFINING PARAMETERS AND OPTIONS panel.
- I Immediate. Data set cleanup is immediately performed if the operation ends in error. The operation is treated as if it had the automatic option when it is rerun.
- M Manual. Data set cleanup actions are deferred for the operation. They are performed when initiated manually from the panel.
- N None. No data set cleanup actions are performed.

ADOPEXPJCL (Y | N)
Specifies if the scheduler uses the JCL extracted from the JES JCL sysout.
ADOP

Y  Uses the fully expanded JCL.
N  Uses the JCL contained in the libraries of the scheduler.

ADOPJOBCRT (W | P | N)
Specifies if the operation is to be considered critical.
W  The operation is considered eligible for Workload Manager (WLM) assistance, if late.
P  The operation is considered the target job of a critical path and eligible, with all the operations belonging to that critical path, for WLM assistance.
N  The operation is not to be considered critical. This is the default.
For W and P, the scheduler automatically sends a request to WLM to promote the job or started task to the specified WLM service class, whenever the conditions of the specified assistance policy are met.

ADOPJOBPOL (L | D | S | C)
Specifies which policy is to be applied for WLM assistance, if the job is defined as critical.
L  Long duration. The job is assisted if it runs beyond its estimated duration time.
D  Deadline. The job is assisted if it has not finished when its deadline time is reached.
S  Latest start time. The job is assisted if it is submitted after the latest start time.
C  Conditional. An algorithm calculates whether to apply the Deadline or the Latest start time policy.
   ' ' Default. WLM uses the policy specified in OPCODES.
If no WLM policy is specified and the operation belongs to a critical path, the policy of the critical path target job is applied.

ADOPPWTO (Y | N)
If you specify Y, a WTO message is issued if the operation passes its deadline and is in started status.

ADOPUSRSYS (Y | N)
Specifies if user sysout support is needed. If you specify Y, the data store logs user sysout.

ADOPWLMCLASS (WLM service class)
The name of the WLM service class to which late critical jobs are promoted. It can be an existing service class or a new service class created for this purpose. If you do not set this keyword and the operation belongs to a critical path, the WLM service class of the critical path target job is used.

AEC (N | Y)
For operations on automatic reporting workstations, Tivoli Workload Scheduler for z/OS does some processing when a job completes to determine whether the operation should be given error status or completed status. If you specify AEC(N), Tivoli Workload Scheduler for z/OS will not check for errors and will give the operation completed status, regardless of any error reported by job tracking.
AJR (N | Y)
Jobs can be placed in HOLD status on the job queue by the event writer (an event writer option). Such jobs can either be released when all Tivoli Workload Scheduler for z/OS scheduling conditions are met, or be released immediately. AJR(Y) means that Tivoli Workload Scheduler for z/OS should control the scheduling. AJR(N) means that Tivoli Workload Scheduler for z/OS will release the job immediately.

The automatic job release option is applicable only when the HOLDJOB keyword of the EWTROPTS is set to USER or YES.

AJSUB (N | Y)
Automatic job submission.

CLATE (Y | N)
Specify Y to cancel this operation if it is time-dependent and late.

Note: Tivoli Workload Scheduler for z/OS never cancels a job that has already started running.

CONDRJOB (Y | N)
Specifies if the operation might recover a conditional predecessor.

CSCRIPT(Y | N)
Use this keyword to set the centralized script flag.

DESCR (‘operation description’)
A free-format description of the operation. It can be up to 24 characters and must be contained within single quotation marks.

DLDAY (deadline day | 0)
Specifies the number of days, relative to the start of the application, when this operation must be completed. This must be an integer. 0 means that the deadline is on the same day as the occurrence input arrival.

DLTIME (hhmm)
Required if you have specified DLDAY. DLTIme specifies the time, on the day specified by the DLDAY keyword, by which this operation should be completed. This must be in the format hhmm.

DURATION (operation duration)
The estimated duration of this operation in minutes or seconds according to the DURUNIT value in OPTIONS. It must be an integer greater than 0. The maximum value is 99 hours 59 minutes 00 seconds. If you specify 99 hours 59 minutes 01 seconds, you do not receive an alert message if the actual duration is greater than the planned duration.

FORM (form number)
If this operation is a printing operation, the printer form number that will appear on the daily plan and on ready lists. For printer workstations with automatic reporting, the printer class and form number let Tivoli Workload Scheduler for z/OS identify the different print operations belonging to a specific job. This can be up to 8 characters.

HIGHRC (return code)
If this operation is a z/OS job, the highest return code that should not be considered an error. If the job ends with this return code or less, the operation will be treated by Tivoli Workload Scheduler for z/OS as completed. This must be an integer less than 4096. If you leave out the parameter, Tivoli Workload Scheduler for z/OS takes the value you specified in the JTOPTS statement.
ADOP

**JOBCLASS** (*jobclass*)
A single character that appears on workstation ready lists for information only. This should be the z/OS job class from the JCL.

**JOBN** (*job name*)
The job name for this operation, if applicable.

**LIMFDBK** (*feedback limit*)
The default is the value you set in the job-tracking initialization statement JTOPTS. The feedback limit must be an integer in the range 100-999.

**MONITOR** (*Y | N*)
If you specify *Y*, the operation is monitored by an external monitor, for example by Tivoli Business Systems Manager.

**OPNO** (*operation number*)
The operation number for this operation, in the range 1 to 255. Each operation within an application must have a unique number. If you do not specify an operation number, the default value will be the previous operation in this application, plus 1. If this is the first operation in an application, the default OPNO is 1. You can also specify your own default operation number increment with ACTION(SETDEFAULT).

If ACTION(SETDEFAULT) is specified, the value entered is an increment to be used when allocating a default operation number.

**PREJOBN** (*predecessor job name*)
The job name of an internal predecessor operation to this operation.

**PREOPNO** (*predecessor operation number*)
The operation number of an internal predecessor operation to this operation.

**PREWSID** (*predecessor workstation*)
The workstation name of an internal predecessor operation to this operation.

**PRTCLASS** (*print class*)
If this operation is a printing operation, the printer SYSOUT class that appears on the daily plan and on ready lists. For printer workstations with automatic reporting, the printer class and form number let Tivoli Workload Scheduler for z/OS identify the different print operations belonging to a specific job. This is a single character, and must be specified for print operations.

**PSNUM** (*number of servers | 0*)
The number of workstation parallel servers required by this operation. This must be an integer.

**R1NUM** (*amount of R1 | 0*)
The amount of workstation type 1 resources required by this operation. This must be an integer.

**R2NUM** (*amount of R2 | 0*)
The amount of workstation type 2 resources required by this operation. This must be an integer.

**REROUTABLE** (*Y | N*)
This keyword specifies the reroute option for the operation. The default is that the operation is reroutable if the WSFAILURE initialization statement RESTART keyword is set to REROUTE.

*Y* The operation is always reroutable.

*N* The operation is not reroutable.
The operation is never reroutable.

RESTARTABLE (Y | N)
This keyword specifies the restart option for the operation. The default is that the operation is restartable if the WSFAILURES initialization statement RESTART keyword is set to RESTART.
Y The operation is always restartable.
N The operation is never restartable.

SMOOTHING (smoothing factor)
The default is the value you set in the job-tracking initialization statement JTOPTS. The smoothing factor must be an integer in the range 0–999.

STARTDAY (start day | 0)
Specifies the input arrival day of this operation, as a number of days offset from the occurrence input arrival day (0 means the same day). This must be an integer. Specifying a separate input arrival day and time for an operation can be useful if the operation is time-dependent and you want to ensure that it will not start before the specified time.

STARTTIME (hhmm)
Required if you have specified STARTDAY. STARTTIME specifies the input arrival time of this operation, on the day specified with the STARTDAY keyword. This must be in the format hhmm. If you specify STARTTIME but not STARTDAY, STARTDAY defaults to 0 (zero), which is the occurrence input arrival day.

TIME (Y | N)
If you specify Y, the job is made time-dependent. See “Creating time-dependent operations” on page 168 for more details.

USESASI (N | Y)
Determines whether the system automation information for the operation is used in the current plan. This keyword must be set to Y (yes) if the workstation has the system AUTOMATION option set to Y (yes).

USEXTNAME (N | Y)
Determines whether or not the operation extended name is used in the current plan.

USEXTSE (N | Y)
Determines if the Scheduling Environment name of the operation is used in the current plan.
Y Scheduling Environment name specified and stored in CP by the DP or dynamic addition process.
N Scheduling Environment name not specified or specified in AD and not stored in CP by the DP or dynamic addition process.

WSID Specifies the name of the workstation where you are scheduling the operation.

Examples
In this example, a job SMFCHK is given an operation number of 060. It has an internal predecessor—operation 030:
ADOP WSID(CPU1) JOBN(SMFCHK) OPNO(060) PREOP(030)

The following example creates three operations:
ADOP WSID(SETP) JOBN(BACKUP) OPNO(010)
ADOP WSID(CPU1) JOBN(BACKUP) TIME(Y) STARTDAY(0) STARTTIME(2000)
PREOPNO(010) OPNO(015) DLTIME(2100) ADOPPWTY(O) ADOPCATM(Y)
ADOP WSID(PRT1) JOBN(BACKUP) PRTCLASS(H) PREOPNO(015)

Chapter 8. Defining applications in batch  217
The first, number 010, is on the job setup workstation SETP. The second, number 015, depends on the setup operation, and cannot start before 20.00 because it is time-dependent. If it is still running at 21.00, Tivoli Workload Scheduler for z/OS issues a DEADLINE WTO. Tivoli Workload Scheduler for z/OS performs immediate cleanup if the job fails.

The last operation has no number specified, but Tivoli Workload Scheduler for z/OS gives it operation number 016 (the default operation number increment is 1). It is a print operation, and depends on the predecessor job 015. Tivoli Workload Scheduler for z/OS will track the output with class H, and report the operation completed when this output has finished printing.

The setup and print operations have the same job name as the main operation; this is required.
ADOPEXTN

Purpose
Use the ADOPEXTN statement to give an operation a descriptive name.

Format
```
ADOPEXTN
  ACTION(SETDEFAULT | ADD)
  EXTNAME('extended name')
  EXTSE('Scheduling Environment')
```

Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for the keyword EXTNAME.

Parameters
ACTION (SETDEFAULT | ADD)
If you specify SETDEFAULT, the remaining keyword values that you specify on the ADOPEXTN statement become default values for all ADOPEXTN statements that follow. No application description is updated. Keywords that you do not specify are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an update of the database.

EXTNAME('extended name')
A free-format name for the operation. It can include blanks and special characters for a maximum of 54 characters. It must be contained in single quotation marks. Do not include delimiters, such as parentheses and single quotation marks for the extended name.

EXTSE ('Scheduling Environment name')
The name of the scheduling environment for this operation. Special characters are allowed. To delete the name, enter EXTSE followed by blank.

Examples
This example sets the extended name daily payroll job for the operation:
```
ADOP ...
ADOP ...
ADOPEXTN ACTION(ADD) EXTNAME('DAILY PAYROLL JOB')
```

This example removes the extended name for the operation:
```
ADOP ...
ADOP ...
ADOPEXTN ACTION(ADD) EXTNAME('')
```
ADOPSAI

Purpose
Use the ADOPSAI statement to define the system automation information for an operation.

If you specify this statement, set to Y the USESAI parameter of the corresponding ADOP statement.

Format

```
>>ADOPSAI
  ACTION(ADD)
  AUTFUNC('automated function')
  COMMTEXT('command text')
  COMPINFO('completion information')
  SECELEM('security element')
```

Parameters

**ACTION (SETDEFAULT | ADD)**
- If you specify SETDEFAULT, all the other values that you set in the ADOPSAI statement become the default values for all the ADOPSAI statements that follow. The application description database is not updated. The parameters that you do not set, are assigned their standard defaults.
- If you specify ADD or use it by default, the statement can result in an update of the database.

**AUTFUNC('automated function')**
- Alphanumeric name for the automated function field of the operation. It can be up to 8 characters. It must be specified between single quotation marks (see the following examples).

**COMMTEXT ('command text')**
- Free-format name for the command text of the operation. It can include blanks and special characters for a maximum of 255 characters. It must be specified between single quotation marks. This parameter is required.

**COMPINFO ('completion information')**
- The completion information for the operation. It can be up to 64 characters. This parameter is positional. You can specify the following information, in the following order, separated by commas:
  - Maximum wait time, in the format hh:mm:ss.
  - Maximum return code accepted as successful execution.
  - Name of an optional user-supplied completion checking routine. It must be contained in single quotation marks.

To delete this information, set COMPINFO to blank.
SECELEM ('security element')
Free-format name for the security element of the operation. It can be up to
8 characters and include blanks and special characters. It must be specified
between single quotation marks.

Examples
This example sets the command text and security element for the operation:

ADOP ...
ADOPSAI ACTION(ADD) COMMTEXT('DAILY PAYROLL JOB')
SECELEM('PPPP')

This example removes the completion information set for the operation:

ADOP ...
ADOPSAI ACTION(ADD)
COMMTEXT('DAILY PAYROLL JOB') SECELEM('PPPP') COMINFO (' ')

ADOPSAI
ADRE

Purpose
Use the ADRE control statement to define the remote job information for an operation.

Format

```
ADRE
    ACTION( ADD | SETDEFAULT )
    RECOMPL( Y | N )
    REJOBNAME( 'remote jobname' )
    REJSNAME( 'remote adid or jobstream name' )
    REJSWS( 'remote jobstream workstation' )
    REOPNO( 'remote operation number' )
```

Parameters

**ACTION (SETDEFAULT | ADD)**
If you specify SETDEFAULT, the remaining keyword values that you specify on the ADRE statement become default values for all ADRE statements that follow. No application description database is updated. Keywords that you do not specify are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an update of the database.

**RECOMPL (Y | N)**
Specifies if the shadow job status must be automatically set to complete, if the remote job does not exist:
Y Sets the operation status to complete.
N Sets the operation status to error.

**REJOBNAME (remote jobname)**
Specifies the remote job name. It can be up to 40 characters and must be specified between single quotation marks. This parameter is required if the remote job runs on a Tivoli Workload Scheduler remote engine.

**REJSNAME (remote adid or jobstream name)**
Specifies the name of the remote application (for Tivoli Workload Scheduler for z/OS) or of the remote job stream (for Tivoli Workload Scheduler). It can be up to 16 characters and must be specified between single quotation marks.

**REJSWS (remote jobstream workstation)**
Specifies the name of the remote job stream workstation. It can be up to 16 characters and must be specified between single quotation marks. This parameter is required if the remote job runs on a Tivoli Workload Scheduler remote engine.

**REOPNO (remote operation number)**
Specifies the remote operation number. It must be a number in the range...
1-255. This parameter is required if the remote job runs on a Tivoli Workload Scheduler for z/OS remote engine.

**Examples**

In the following example, you set the application name and operation number of a remote job that runs on Tivoli Workload Scheduler for z/OS:

```
ADOP ...
ADRE ACTION(ADD)
REJSNAME('APPLREMOTE ')
REOPNO( 1)
RECOMPL(N)
```
**Purpose**
Use the ADRULE control statement to specify a rule, which generates a set of
dates. The ADRULE control statement must immediately follow the ADRUN
statement. If you have many rules, use a pair of ADRUN/ADRULE statements for
each one.

**Note:** The only abbreviations allowed for the keywords on this statement are the
first letter and OS for ORIGINSHIFT.

**Format**

```
ADRULE
 EVERY
  ( day number )
  ( day number )

LAST
 DAY-(
  ( day number )

WEEK
 MONTH
  ( week number )
  JANUARY
  FEBRUARY
  MARCH
  APRIL
  MAY
  JUNE
  JULY
  AUGUST
  SEPTEMBER
  OCTOBER
  NOVEMBER
  DECEMBER
```
Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for this control statement.

Parameters
DAY(DAY | WORKDAY | FREEDAY | MONDAY | TUESDAY | WEDNESDAY | THURSDAY | FRIDAY | SATURDAY | SUNDAY)
Specifies the day or days. You can abbreviate the names of the days to MON TUE WED THU FRI SAT SUN WORK and FREE.

EVERY | ONLY
Use EVERY to specify a series of days. For example, EVERY(2) DAY(DAY) FEBRUARY specifies days 1, 3, 5, 7 etc. in February. The origin of the series is 1 unless you also specify ORIGINSHIFT.

Use ONLY to specify the days precisely. For example, ONLY(2) DAY(DAY) FEBRUARY specifies only February 2.

(day number)
 Specifies the number (for ONLY) of the day or days to be selected. For EVERY, this specifies the interval of the series. The number is in the range 1 to 999.

LAST (day number)
Specifies the number (for ONLY) of the day or days to be selected. For EVERY, this specifies the interval of the series, starting from the end, so EVERY LAST(3) DAY(DAY) JANUARY specifies January 31, 28, 25 etc.

The origin of the series is the last day unless you also specify ORIGINSHIFT. The number is in the range 1 to 999.

MONTH(JANUARY | FEBRUARY | MARCH | APRIL | MAY | JUNE | JULY | AUGUST | SEPTEMBER | OCTOBER | NOVEMBER | DECEMBER)
Specifies the month or months. You can abbreviate the names of the months to the first three characters. If you omit the name of the month, the rule selects every month.

ORIGINSHIFT (number)
Specifies the origin shift in days. The number is in the range 1 to 999. Use this only with the EVERY keyword, when the origin is not the first (or, with LAST, the last) day of the cycle or period. If you specify EVERY(4) DAY(DAY) MONTH ORIGINSHIFT(1), for example, the rule selects a series starting at the second day of each month, with an interval of 4 days: January 2, 6, 10 etc., then February 2, 6, 10 etc., for each month in the year.

PERIOD (period name)
The name of a user-defined period, which must be in the period database. If you specify a period name such as JULY, which is the same name as a predefined cycle, Tivoli Workload Scheduler for z/OS looks for a user-defined period JULY, and gives an error if one does not exist.
WEEK

Specifies the week number or numbers. The number might range from 1 to 53. Week 1 is defined as the first week with at least 4 days of the new year. If you omit the number, the rule selects every week.

YEAR

Used to specify that the cycle is a year, as in EVERY(2) DAY(DAY) YEAR, which gives January 1, January 3, January 5 etc. for each year. ONLY LAST DAY(FRIDAY) YEAR gives the last Friday in each year.

Examples

In the following example, EX1A (type R) selects the fourth Thursday in each month, or the closest work day before, if the Thursday is a free day (free-day rule 1). EX1B (type E) excludes the last Thursday of the year. Note that the exclusion rule has exactly the same input-arrival time and free-day rule as EX1A:

```
ADRUN NAME(EX1A) TYPE(R) RULE(1) IATIME(1800) DLTIME(2300)
ADRULE ONLY(4) DAY(THU) MONTH
ADRUN NAME(EX1B) TYPE(E) RULE(1) IATIME(1800) DLTIME(2300)
ADRULE ONLY LAST DAY(THU) YEAR
```

In the next example, EX2A (type R) selects February 3 and February 5, whether or not these are free days (rule 3). EX2B (type R) selects March 8, whether or not this is a free day (rule 3). The total effect of the two rules is to select these three days each year:

```
ADRUN NAME(EX2A) TYPE(R) RULE(3) IATIME(1800) DLTIME(2300)
ADRULE ONLY(3,5) DAY(DAY) MONTH(FEB)
ADRUN NAME(EX2B) TYPE(R) RULE(3) IATIME(1800) DLTIME(2300)
ADRULE ONLY(8) DAY(DAY) MONTH(MARCH)
```

EX3A (type R) is a complex rule which is a result of two series. EVERY(2) DAY(DAY) YEAR specifies January 1, 3, 5, 7, 9 etc., and EVERY(3) DAY(DAY) YEAR specifies January 1, 4, 7, 10, 13 etc. Tivoli Workload Scheduler for z/OS adds these two series and removes the duplicates, creating this series: January 1, 3, 4, 5, 7, 9, 10, 13 etc. It then ignores any days that are free days (free-day rule 4):

```
ADRUN NAME(EX3A) TYPE(R) RULE(4) IATIME(1800) DLTIME(2300)
ADRULE EVERY(2,3) DAY(DAY) YEAR
```

EX4A (type R) specifies every Monday, even if Monday is a free day. Be careful with the work day end time defined in the calendar, because the input arrival time is early (01.00). If the work day end time is 01.00 or later, this rule will cause the application to be scheduled early on Tuesday morning, because times up to the work day end time are considered part of the previous day:

```
ADRUN NAME(EX4A) TYPE(R) RULE(3) IATIME(0100) DLTIME(2300)
ADRULE EVERY DAY(MONDAY) YEAR
```

EX5A (type R) specifies every week day (Monday to Friday), even if a free day. EX5B (type E) excludes all work days. The result is to select all week days that are free days:

```
ADRUN NAME(EX5A) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)
ADRULE EVERY DAY(MON, TUE, WED, THU, FRI) YEAR
ADRUN NAME(EX5B) TYPE(E) RULE(3) IATIME(0800) DLTIME(2300)
ADRULE EVERY DAY(WORK) YEAR
```

In the following example, EX6A (type R) specifies the first free day in each week and the first free day in each year, because ONLY without a number is equivalent to ONLY(1). This is really a combination of two simple rules: ONLY(1) DAY(FREEDAY) WEEK and ONLY(1) DAY(FREEDAY) YEAR. The days generated
are also affected by the free day rule, so you normally use rule 3 (select the free
day) with FREEDAY. If EX6A had specified free-day rule 1 (closest work day
before), for example, Tivoli Workload Scheduler for z/OS would have generated
the closest work day before the first free day in each week and the first free day in
the year.

ADRUN NAME(EX6A) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)
ADRULE ONLY DAY(FREE) WEEK YEAR

Note: You are not recommended to build complex rules that are hard to
understand. It is better to create two simple rules:
ADRUN NAME(EX6A) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)
ADRULE ONLY(1) DAY(FREE) WEEK
ADRUN NAME(EX6B) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)
ADRULE ONLY(1) DAY(FREE) YEAR

EX7A (type R) selects February 29. In years that are not leap years, no day is
selected:
ADRUN NAME(EX7A) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)
ADRULE ONLY(29) DAY(29) MONTH(FEB)

EX8A (type R) selects the first day in the year and the first day in week 4. This is
another combination rule, like example 6. The free-day rule is 1, so Tivoli
Workload Scheduler for z/OS generates the closest work day before if either
January 1 or Monday in week 4 is a free day, generating a day in December or in
week 3.
ADRUN NAME(EX8A) TYPE(R) RULE(1) IATIME(0800) DLTIME(2300)
ADRULE 0 D(DAY) W(4) Y /* USING KEYWORD ABBREVIATIONS */
**Purpose**
Use the ADRUN control statement to add a run cycle specification to an application description. To specify a rule-based run cycle, follow the ADRUN statement with an ADRULE statement. See "Creating run cycles" on page 124 for a description of run cycles. If you are specifying an offset-based run cycle, you supply all the information on the ADRUN statement.

**Note:** The ADRUN statement cannot be used to add run cycles to an application that is a member of a group; specify ADRUN when you create the group.

**Format**
- `ADRUN`  
  - `ADD`  
    - `ACTION(SETDEFAULT)`
  - `ADRJTAB( variable table name )`
  - `DESCRT('run cycle description')`
    - `DLDAY( deadline day )`
  - `DLTIME( hhmm )`
  - `EIADAYS( start day from end of period )`
  - `IADAYS( start day from start of period )`
  - `IATIME( hhmm )`
  - `NAME( rule name )`
  - `PERIOD( period name )`
  - `RPTENDT( hhmm )`
  - `RPTEVRY( hhmm )`
  - `RULE( 1 )`
  - `TYPE( X )`
  - `VALFROM( yymmdd )`
  - `VALTO( yymmdd )`
Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for the NAME and PERIOD keywords.

Parameters
ACTION (SETDEFAULT | ADD)
If you specify SETDEFAULT, the remaining keyword values that you specify on the ADRUN statement become default values for all ADRUN statements that follow. No application description is updated. Keywords that you do not specify are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an update of the database.

ADRJTAB (variable table name)
A 16-character field that identifies the JCL variable table to be used for the occurrences generated. For offset-based run cycles, this JCL variable table overrides the one specified for the period. For rule-based run cycles, specify the JCL variable table here, because Tivoli Workload Scheduler for z/OS ignores any JCL variable table associated with the period. The first character must be alphabetic.

DESCR (‘run cycle description’)
A free-format description of the run cycle, up to 50 characters and contained in single quotation marks.

DLDAY (deadline day 1 0)
The number of days from the input arrival day that the application should be completed in: 0 means that the deadline is on the same day as the input arrival day. This must be an integer.

DLTIME (hhmm)
The time on the deadline day that the application should be completed by, in the format hhmm.

EIADAYS (start day from end of period 1 0)
Depending on the TYPE keyword, this keyword defines one or more days relative to the start of the period when the application should be scheduled (TYPE N) or when it should not be scheduled (TYPE X). The numbers count from the end of the period; 1 is the last day.

IADAYS (start day from start of period 1 1)
Depending on the TYPE keyword, this keyword defines one or more days relative to the start of the period when the application should be scheduled (TYPE N) or when it should not be scheduled (TYPE X). The numbers count from the start of the period; 1 is the first day.

IATIME (hhmm)
The time, in the format hhmm, that the application is to arrive at the first workstation.

NAME (rule name)
For rule-based run cycles, this is the name of the rule—up to 8 characters and unique for this application. Specify NAME, and type R or E, if you are creating a rule-based run cycle.

PERIOD (period name)
For offset-based run cycles, this is the name of a cyclic or noncyclic period defined in the calendar database. Specify PERIOD, and type N or X, if you are creating an offset-based run cycle.
ADRUN

RPTENDT (hhmm)
The repeat end time for the EVERY options, in the format hhmm. It must be a time between the IA time of the run cycle and the calendar work day end time of the application.

RPTEVRY (hhmm)
The repeating frequency for the EVERY options, in the format hhmm. It specifies that the application has an occurrence in the long-term plan every hhmm, starting from the IA time to the repeat end time (RPTENDT keyword). If this keyword is not set, only the occurrence related to the IA time is added to the long-term plan.

RULE (1 | 2 | 3 | 4 | E)
Defines which free-day rule is in effect. See "Selecting a free-day rule" on page 130.

TYPE (X | E | R | N)
Specify R or E: without IADAYS or EIADAYS when you create a rule-based run cycle. You must specify the NAME keyword, and an ADRULE statement must follow this ADRUN statement. R (regular) means that the ADRULE statement specifies days when the application should be scheduled. E (exclusion) means that the ADRULE statement specifies days when the application should not be scheduled.

Specify N or X together with IADAYS or EIADAYS when you create an offset-based run cycle. You must specify the PERIOD keyword. N (normal) means that the IADAYS and EIADAYS parameters define days when the application should be scheduled. X (negative) means that the IADAYS and EIADAYS parameters define days when the application should not be scheduled.

VALFROM (yymmdd | current date)
The start date of validity of this run cycle, in the format yymmdd. See the note under VALTO.

VALTO (yymmdd | 711231)
The end date of validity of this run cycle, in the format yymmdd.

Note: Tivoli Workload Scheduler for z/OS interprets the yy part in the VALTO and VALFROM keywords as follows:

<table>
<thead>
<tr>
<th>YY</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>72 - 99</td>
<td>1972 - 1999</td>
</tr>
<tr>
<td>00 - 71</td>
<td>2000 - 2071</td>
</tr>
</tbody>
</table>

Examples
In the following sample, the first example is type R, which is a regular rule-based run cycle. The ADRULE statement immediately following this statement specifies the last work day of each month. The deadline for the application is 21.00 on the same day. The input arrival time is 9.00, but this is not necessarily the time that the application will start.

The second example is type N, which is a normal offset-based run cycle, so you must have defined a period called WEEK. The AD that is being built will be scheduled for the first day in each week. If the first day is a free day, the application is not scheduled for that week.

ADRUN NAME(LASTWD) IATIME(0900) DLTIME(2100) TYPE(R) RULE(1)
ADRULE ONLY LAST(1) DAY(WORKDAY) MONTH
;
ADRUN PERIOD(WEEK) IATIME(0900) DLTIME(2100) IADAYS(1) RULE(4)
Purpose
Use the ADSR control statement to specify the use of special resources by an operation.

Format
```
/SM590000
ADSR
AD
SETDEFAULT
KEEPONERR(N|Y)
QUANTITY(quantity required)
RESOURCE(resource name)
USAGE(X|S)
/SM590000
```

Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for this keyword: RESOURCE.

Parameters

**ACTION (SETDEFAULT | ADD)**
If you specify SETDEFAULT, the remaining keyword values that you specify on the statement become default values for all ADSR statements that follow. No application description is updated. Keywords that you do not specify are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an update of the database.

**KEEPONERR (N | Y)**
Defines whether the resource should be kept if the operation ends in error. If you do not specify this keyword, the default action is taken from the resource definition or RESOPTS statement.

**ONCOMPLETE (N | Y | R)**
Defines the value to which the global availability of the resource is reset at operation completion. If you do not specify this keyword, the default action is taken from the resource definition or RESOPTS statement.

**QUANTITY (quantity required)**
The number of this resource that the operation needs, in the range 1 to 999999. If you do not specify this keyword, the operation takes all the resource exclusively, if USAGE is X, or prevents the exclusive use of any of this resource by any other operation, if USAGE is S.

**RESOURCE (resource name)**
The name of the resource required by this operation. You can specify up to 44 characters. If the name contains special characters, enclose the string in quotes.

**USAGE (X | S)**
Defines whether the resource should be allocated as shared (S) or exclusive (X).
**Examples**

In this example, the operation needs two of the resource 'LINE.LONDON' exclusively. The **KEEPONERR** action is taken from the definition in the special resource database:

```adop
ADSR RESOURCE(LINE.LONDON) USAGE(X) Q(2)
```

In this example, the operation requires shared access to the whole quantity of the special resource PAYROLL.DATABASE, and keeps its allocation if the operation fails:

```adop
ADSR RESOURCE(PAYROLL.DATABASE) KEEP(Y)
```
ADSTART

Purpose
Use the ADSTART control statement to signal the start of an application
description. When this statement is found in the input data set, it signals the batch
loader to complete the preceding AD or OI being built and write it to the database.

If you have specified OPTIONS ACTION(ADD) and an application description
with the same ADID, STATUS, and ADVALFROM already exists, these actions are
taken:
• The processing of the ADSTART is terminated.
• An error message is issued.
• The statements that follow are ignored until the next ADSTART or OISTART
statement is found in the input data set.

Format

```plaintext
ADSTART
  ACTION(SETDEFAULT)
ADID(application ID)
ADGROUPID(group definition ID)
ADTYPE(GUI)
ADVALFROM(yymmd)
CALENDAR(calendar name)
DESCR('descriptive text')
OLIMFDGBK(limit for deadline feedback)
OSMOOTHING(deadline smoothing factor)
GROUP(authority group)
ODESCR('owner description')
OWNER(owner ID)
PRIORITY(priority)
```

Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for this keyword:
ADID.

Parameters

ACTION (SETDEFAULT | ADD)
If you specify SETDEFAULT, the remaining keyword values that you
specify on the ADSTART statement become default values for all
ADSTART statements that follow. No application description is updated.
Keywords that you do not specify are assigned their standard defaults.
If you specify ADD or use it by default, the statement can result in an update of the database.

**ADID (application ID)**

The job name or identifier of the new application, in either EBCDIC or DBCS format, as specified on the OPTIONS statement. If you use DBCS characters, you must enter them as a quoted string started by shift-out (X'0E') and ended by shift-in (X'0F').

An ADID must be specified.

**ADGROUPID (group definition ID)**

The name of the group definition used by this application to generate run cycle information. The name can be in either EBCDIC or DBCS format, as specified on the OPTIONS statement. If you use DBCS characters, you must enter them as a quoted string started by shift-out (X'0E') and ended by shift-in (X'0F').

This keyword is valid only for an ADTYPE of A and should not be specified with CALENDAR.

**ADSTAT (A | P)**

Application status indicator. A is for active, P is for pending application description.

**ADTYPE (A | G)**

Application type indicator. A is for applications, G is for group definitions.

**ADVALFROM (yymmdd | current date)**

Defines the start date of the validity period of the AD. Only the start date can be specified. The end of the validity period is set so that the time up to 31 December 2071 is covered, taking existing application descriptions into account. Tivoli Workload Scheduler for z/OS interprets the yy part as follows:

- **YY Year**
  - 72 - 99 1972 - 1999
  - 00 - 71 2000 - 2071

**CALENDAR (calendar | default calendar)**

The name of the calendar to be used when run days are calculated for this application or group definition. Do not specify this keyword for applications that are members of a group.

**DESCR (descriptive text)**

A free-format description of the application up to 24 characters and contained in single quotation marks.

**DLIMFDBK (limit for deadline feedback)**

The deadline limit for feedback. This keyword determines if the estimated deadline in the application description run cycle or operation is updated when an occurrence of the application reaches the complete status. The DLIMFDBK keyword value you set in this keyword is used only if no value is set in the application description.

Feedback values are in the range 100 through 999. The value 0, meaning that the deadline must always be updated, regardless of the estimated and actual values, cannot be specified at application level in batch loader, PIF, and ISPF panels. It can be specified only in the DLIMFDBK keyword of the JTOPTS statement.

The feedback limits for **ADL** are calculated as follows:
• Lower limit = \( ODL \times 100 / DLF \)
• Upper limit = \( ODL \times DLF / 100 \)

Where:

\( ADL \)  
The actual deadline, considered as the elapsed minutes between 
the IA and the completion time of the occurrence or operation.

\( ODL \)  
The old deadline estimated for the run cycle or operation 
(considered as offset in minutes from the IA) currently stored in 
the application description database.

\( DLF \)  
The deadline limit for feedback.

When the deadline feedback limit is set to 100, no new estimated deadline 
is stored in the application description database. If the actual deadline lies 
within the feedback limits, a smoothing factor is applied before the 
application description is updated.

If the completion time occurs before the IA time, the deadline is not 
updated and a missed feedback record is generated.

When the occurrence is generated, an identifier of the run cycle that 
generates the occurrence is stored in the occurrence record. This identifier 
is used to determine which run cycle must be updated. If the application 
description or the occurrence input arrival was modified, the run cycle 
might no longer be matchable. In this case, the deadline is not updated 
and a missed feedback record is generated.

\textbf{DSMOOTHING (deadline smoothing factor)}

The smoothing factor. It determines how much the actual deadline 
influences the new deadline estimated for a run cycle or operation in the 
application description database. The smoothing factor is applied only if 
the actual deadline lies within the deadline feedback limits. The 
\texttt{DSMOOTHING} keyword value is used only if you did not set a smoothing 
factor in the application description.

The smoothing factor is in the range 0 through 999. A value of 0 means 
that the deadline is not updated, a value of 100 means that the actual 
deadline replaces the existing estimated deadline.

The new deadline is calculated as follows:
\[
NDL = ODL + (ADL - ODL) \times DSF / 100
\]

Where:

\( NDL \)  
The new deadline estimated for the run cycle or operation 
(considered as offset in minutes from the IA) to be stored in the 
application description database.

\( ODL \)  
The old deadline estimated for the run cycle or operation 
(considered as offset in minutes from the IA) currently stored in 
the application description database.

\( ADL \)  
The actual deadline, considered as the elapsed minutes between 
the IA and the completion time of the occurrence or operation.

\( DSF \)  
The smoothing factor.

\textbf{GROUP (authority group)}
The name of the application authority group to be used for additional 
authority checking. You can specify up to 8 characters.
ADSTART

ODESCR ('owner description')
A free-format description of the application owner up to 24 characters and contained in single quotation marks.

OWNER (owner ID)
The name of the application owner. You can specify up to 16 characters for EBCDIC, or 7 characters for DBCS. The lowercase alphabetic characters, a–z, are translated to uppercase A–Z. If you use DBCS characters, you must enter the name as a quoted string started by a shift-out and ended by a shift-in. This keyword is required.

PRIORITY (priority | 5)
The scheduling priority of the application. Must be a single digit in the range 1–9. This keyword is valid only for an ADTYPE of A.

Examples
This example sets defaults for all following ADSTART statements:
ADSTART ACTION(SETDEFAULT) OWNER(PAYGRP) PRIORITY(6)
   ODESCR('PAYROLL GROUP')

In this example, the batch loader will create application REORG7, which is an urgent application. It will be valid for inclusion in Tivoli Workload Scheduler for z/OS plans on the first day of 1999:
ADSTART ADID(REORG7) PRIORITY(9) DESCR('Reorganize IMS databases')
   ADVALFROM(990101) OWNER(XDARV00)
ADUSF

Purpose
Use the ADUSF statement to define user fields for the operation. For detailed information about user fields, refer to "Creating user fields to specify additional information" on page 164.

Format

```
ADUSF
UFNAME(‘user field name’)  
ACTION(SETDEFAULT)
UFVALUE(‘user field value’)
```

Restrictions
You cannot use ACTION(SETDEFAULT) to set a default value for the UFNAME keyword.

Parameters

**Parameters (SETDEFAULT | ADD)**

**ACTION (SETDEFAULT | ADD)**
ACTION(SETDEFAULT) is applicable only to the UFVALUE keyword. If you specify SETDEFAULT, the value that you set for UFVALUE becomes the default for all the UFVALUE keywords that you set in the ADUSF statement. The application description database is not updated.

If you specify ADD or use it by default, the statement can result in an update of the database.

**UFNAME(‘user field name’)**
The name for the user field, up to a maximum of 16 characters and enclosed in single quotation marks.

**UFVALUE(‘user field value’)**
The value for the user field, up to a maximum of 54 characters and enclosed in single quotation marks. You can set the user field value to blank.

Examples
This example defines the user field Path, which is set to /TWS/local/zcentric, and the user field Workstation name, which is set to Lab1328:

```
ADOP...
ADUSF
  UFNAME (‘Path’)  
  UFVALUE (‘/TWS/local/zcentric’)
ADUSF
  UFNAME (‘Workstation name’)  
  UFVALUE (‘Lab1328’)
```
Purpose
Use the OIT control statement to add one line of operator instruction text to the current operator instruction.

Format

```
OIT 'text'
```

Parameters

'text'  A line of text to be added at the end of the current operator instruction. You can have up to 443 lines in each instruction. Each line can be up to 66 characters and must be contained within single quotation marks.

Examples

OISTART ...
OIT 'Enter password for update'
OIT 'of IMS database'
Purpose
Use the OISTART control statement to start the creation of a new operator
instruction. The text contained in the OIT statements that follow it will form the
new operator instruction.

To identify which AD the operator instruction relates to, supply the ADID
keyword. To identify the operation within the AD, also specify at least one of the
following:
- Operation number (OPNO)
- Job name (JOBN)

You must specify enough of these keywords to uniquely identify the operation. If
more than one operation or no operations match your specification, an error
message is issued and no OI is created. This also happens if an OI already exists
for the same application ID and operation (with a validity time that overlaps the
time specified by this request) unless REPLACE is specified on the OPTIONS
statement.

If the output is directed to a VSAM data set, the operation can be defined by an
ADOP statement occurring later in the input data set. This is because most validity
checking occurs after all statements in the input data set are read.

If the output is directed to an active Tivoli Workload Scheduler for z/OS
subsystem, the operation specified must already exist on the AD database. It
cannot be defined later in the input data set.

Format

```
OISTART
  ADID(application ID)
  ACTION(SETDEFAULT)
  OPNO(operation number)
  JOBN(job name)
  MEMBER(member name)
  VALFROMD(yymmdd)
  VALFROMT(hhmm)
  VALTOD(yymmdd)
  VALTOT(hhmm)
```

Restrictions
You cannot use ACTION(SETDEFAULT) to set default values for these keywords:
- ADID
- OPNO
OISTART

Parameters

ACTION (SETDEFAULT | ADD)

If you specify SETDEFAULT, the remaining keyword values that you specify on the OISTART statement become default values for all OISTART statements that follow. No OI is updated. Keywords that you do not specify are assigned their standard defaults.

If you specify ADD or use it by default, the statement can result in an update of the database.

ADID (application ID)

The identifier of the application. If you use DBCS characters, they must be entered as a quoted string started by a shift-out and ended by a shift-in.

You must specify ADID.

JOBN (job name)

The job name of the operation that this OI is for.

MEMBER (member name)

If you specify the MEMBER keyword, the OI text must reside in the partitioned data set (PDS) defined by the EQQOIPDS DD statement. It must be free format in columns 1 to 72. The MEMBER keyword specifies which member in the PDS contains the OI text.

OPNO (operation number)

The operation number of the operation that this OI is for.

VALFROMD (yymmdd | current date)

The start date of validity of this OI. You must specify this in the format yymmdd. See the notes after VALTOT.

VALFROMT (hhmm | current time)

The start time of validity of this OI. You must specify this in the format hhmm. See the notes after VALTOT.

VALTOD (yymmdd | 711231)

The end date of validity of this OI. You must specify this in the format yymmdd. See the notes after VALTOT.

VALTOT (hhmm | 2359)

The end time of validity of this OI. You must specify this in the format hhmm. See the following notes.

Notes:

1. If you do not supply any of these VAL keywords, Tivoli Workload Scheduler for z/OS assumes that the operator instruction is permanent.

2. Tivoli Workload Scheduler for z/OS interprets the yy part as follows:

   YY Year
   72 - 99 1972 - 1999
   00 - 71 2000 - 2071

Examples

In this example, the OISTART specifies that operator instruction text for operation 020 in the application PAYDAILY will follow this statement:

OISTART ADID(PAYDAILY) OPNO(020)
OIT ...
In this example, the OISTART specifies that operator instruction text for operation 020 in the application PAYDAILY is in the PAYDAILY member of the PDS defined by the EQQOIPDS ddname:

OISTART ADID(PAYDAILY) OPNO(020) MEMBER(PAYDAILY)
**OPTIONS**

**Purpose**
Use the OPTIONS control statement to define execution options for the batch loader. None of the keywords are required. It is no longer true that there can be only one options statement in the input data set and that it must be the first not commented card. It is now allowed to have more options statements in the sysin file, with the rule that the subsequent not commented card must be another options card or an ADSTART or an OISTART. Note that this does not mean that you can have different values of the options operands based on the position of the different options statements. If an operand of the options statement is present more than once in the sysin file, the entire processing will take place with the last value specified, not just from the point where the operand has been changed.

**Format**

```
OPTIONS
  ACTION(ADD|REPLACE|SCAN)
  ADID(DBCS|EBCDIC)
  ADVALFROMCHG(N|Y)
  CHECK(N|Y)
  DURUNIT(MINUTES|SECONDS)
  MSGLEVEL(1|2|3)
  OWNER(DBCS|EBCDIC)
  SUBSYS(subsystem name)
  STATUSCHANGE(N|Y)
```

**Parameters**

**ACTION (SCAN| REPLACE| ADD)**
If you specify ACTION(SCAN), only basic syntax validity checking is performed on the remaining control statements in the input data set. The batch loader does not produce any output other than error messages.

**ACTION(ADD)** specifies that you are adding new ADs and OIs. If you attempt to add an AD or OI that already exists in the database, an error message is produced, and the AD or OI is not added.

**ACTION(REPLACE)** specifies that ADs and OIs defined in the input data set can replace existing ones in the database if they have the same name.

**ADID (DBCS | EBCDIC)**
Specifies the format of the application ID data that you will use in ADID and PREADID keywords in your control statements. You specify whether you will use EBCDIC or DBCS characters. DBCS means that the application ID should consist of double-byte character set characters only. If you have specified the SUBSYS keyword, you must use the format specified for that subsystem. If you have not specified the SUBSYS keyword, EBCDIC will be the default.

**ADVALFROMCHG (N | Y)**
The ADVALFROMCHG keyword instructs the batch loader about whether
OPTIONS

to replace the existing version of an application description whose validity interval is defined by the ADVALFROM of the application being added through the ADSTART statement.

Specify ADVALFROMCHG(Y) if the existing version of the AD is to be replaced by the version created by the batch loader and ADVALFROM is to be changed accordingly. You cannot specify ADVALFROMCHG(Y) if ACTION(ADD) or STATUSCHANGE(Y).

The default is ADVALFROMCHG(N). The existing version of the application description is replaced only if the application ID, type, valid from date, and status are consistent. Otherwise, a new version is added.

CHECK (Y | N)
The CHECK keyword instructs the batch loader to perform validity checking for the application descriptions, for example by checking whether a workstation exists in the workstation database. If CHECK(Y) is specified and an error is found, the application description record is not updated. When CHECK(N) is specified, the application is updated regardless of errors.

CHECK(N) should be used with care, because it can result in storing invalid applications. This can cause problems in the long-term and current plans. If CHECK(N) is unavoidable, the application descriptions should be given a future valid-from date, using the ADVALFROM keyword of the ADSTART statement, that makes the applications unavailable to the long-term or current plans until the applications can be verified.

Validity checking (other than basic syntax checking) is not done when OPTIONS ACTION(SCAN) is specified, because no output data set is produced, so do not specify the CHECK keyword.

DURUNIT (MINUTES | SECONDS)
Sets the unit in which duration is entered. You specify whether you use MINUTES or SECONDS. DURUNIT(SECONDS) causes the value entered by means of the DURATION keyword of the ADOP statement to be treated as a number of seconds. If DURUNIT(MINUTES) is specified or DURUNIT is left out of the DURATION value of the ADOP statement, it is treated as a number of minutes.

MSGLEVEL (1 | 3 | 2)
Controls the messages generated by the batch loader:

Level 1 This is the lowest level. Error messages and exceptional information messages are written.

Level 2 Includes level 1. A message is also written each time the statement making up one record is received and the syntax has been checked and passed.

Level 3 Includes level 2. In addition, each statement is written to the message log when the statement is processed.

By setting MSGLEVEL to 3, the input SYSIN for the batch loader JCL is scanned twice; in this way, some inconsistencies found during the first scan might be removed. For example, if during the first scan the system finds a successor operation defined before its predecessor, the successor is not inserted into the AD and message EQQY221E is displayed. Continuing the first scan, the system finds the predecessor and adds it to the AD. By
specifying MSGLEVEL(3), this inconsistency is resolved
during the second scan of the input SYSIN, because when
the system scans the successor operation, the predecessor is
already in the AD.

OWNER (DBCS | EBCDIC)
Specifies the format of the owner data that you will use in the OWNER
keyword of ADSTART control statements. You specify whether you will
use EBCDIC or DBCS characters. DBCS means that the owner ID should
consist of double-byte character set characters only. If you have specified
the SUBSYS keyword, you must use the format specified for that
subsystem. If you have not specified the SUBSYS keyword, EBCDIC will
be the default.

STATUSCHANGE (N | Y)
Determines whether the application description record is modified
according to a new version, or a new record is created. Generally, when the
batch loader creates a new version of an application description whose ID,
type, valid-to date, and status are already recorded, the application
description record is updated. If you specify STATUSCHANGE(Y), the
application description record is updated even if the application status is
different, and the status itself is modified to the new value. This is true,
only if another application with the same credentials and status does not
already exist.

If you specify STATUSCHANGE(N), the application description already
recorded is updated if all the credentials of the new application match,
otherwise a new record is created. This is the default.

SUBSYS (subsystem name)
The SUBSYS keyword indicates which Tivoli Workload Scheduler for z/OS
subsystem the output will be directed to. If this keyword is not specified,
output will be directed to the VSAM data sets defined by the EQQADDS
and EQQOIDS DD statements. If the JCL includes either or both of these
DD statements when output is directed to a Tivoli Workload Scheduler for
z/OS subsystem, they are ignored.

Note: Do not use the data sets defined by EQQADDS and EQQOIDS if
they are allocated to an active subsystem. Otherwise, an error
message is issued and batch-loader processing ends.

Examples
In this example, the batch-loader output is directed to a subsystem called OPC1.
Default values selected are ACTION(ADD), CHECK(Y), ADID(EBCDIC),
OWNER(EBCDIC), MSGLEVEL(2), and DURUNIT(MINUTES).

OPTIONS SUBSYS(OPC1)
Chapter 9. Overview of the long-term and current plans

Once you have described your installation to Tivoli Workload Scheduler for z/OS and defined your work in the application description (AD) database, Tivoli Workload Scheduler for z/OS can build the plans required to control your production workload. The long-term plan (LTP) contains a high-level description of the work scheduled for the coming weeks or months. The current plan (CP) is the Tivoli Workload Scheduler for z/OS schedule, providing the detail required to control the processing. The current plan is derived from a section of the long-term plan and contains the work that Tivoli Workload Scheduler for z/OS will run. As your batch processing is submitted and run, the current plan is updated to reflect the actual status of the work. This chapter describes the two plans and provides some guidelines for creating and maintaining them.

When you create an application, it is stored in the AD database. The database contains information on the operations that make up an application and describes how often the operations should run. This data is used by the batch planning functions to schedule the application or job description on the required days, at the specified time.
Figure 110 shows the relationship between the database information and the planning processes.

The task of creating the long-term plan and current plan is normally performed once. After creation, the plans are continually extended using batch functions.

When you make changes to the Tivoli Workload Scheduler for z/OS database, such as creating a new application, this is not reflected in the plans until you have incorporated the changes into the long-term plan and current plan. However, last-minute or unplanned changes can be made directly to the plans using the panels.

The plans can be used to produce reports which can help you:
- Reach service level agreements with your customers
- Measure the slack time in your batch window
- Assist workload management and tuning exercises
- Forecast and plan for the effect of heavy processing periods
- Demonstrate the effect on the batch window if more or fewer resources are available

When creating the long-term plan and current plan, Tivoli Workload Scheduler for z/OS calculates planned start and end times based on completion of predecessor
processing and resource availability. In addition, the latest out time is calculated. This is the latest possible time the operation can start, to meet the deadline time. Operations that run close to their deadlines are given priority when competing for resources. If you have defined accurate resource utilization and deadline times in the application description, Tivoli Workload Scheduler for z/OS is in the best position to schedule your work and optimize the use of resources.

The Tivoli Workload Scheduler for z/OS planning functions are powerful; problems that affect the batch window can be highlighted before service levels are missed. Careful examination of plan output can give you the opportunity to prevent such problems.

## Long-term plan

The long-term plan is a high-level plan that can cover a length of time from 1 day up to 4 years. You produce the long-term plan using Tivoli Workload Scheduler for z/OS batch jobs, which are usually submitted from the panels. The batch jobs use data from:

- The application description database
- Calendar and period definitions
- The current long-term plan, if one exists

When an application or job description is scheduled in the long-term plan or current plan, it becomes an occurrence. Every occurrence is uniquely identified by its name, input arrival date, and time.

Tivoli Workload Scheduler for z/OS examines every application and job description to determine if an occurrence should be generated in the long-term plan. An occurrence is generated if an active application has a run cycle with a period and calendar combination that falls within the long-term plan range. If an application or job description has been specified as belonging to a group definition, the run cycles and calendar definition are extracted from the group definition, and an occurrence group is created in the long-term plan. An occurrence group consists of application occurrences that reference a group definition and have the same input arrival date and time.

Occurrences from the old long-term plan that you have added, deleted, or changed manually are carried over to the new long-term plan, except when Tivoli Workload Scheduler for z/OS is creating a new long-term plan.

Figure 111 on page 248 describes the data required for the long-term planning process.
Long-term plan

The long-term plan contains an occurrence entry for every planned run of an application. Most applications generate several occurrences in the long-term plan; for example, applications that are required daily or weekly. Every occurrence in the long-term plan is uniquely identified by the occurrence name, input arrival date, and time. The input arrival time is calculated from the run cycle defined to the application or job description or from the group definition if the application is a member of a group.

The long-term plan also contains external dependency information produced from data in the AD database. It does not, however, know the relationships between operations making up an application.

As time goes on, you must extend the long-term plan to cover future periods. For example, if you wanted to plan for 4 weeks, you could initially set up the long-term plan to cover a period of 35 days into the future. You could then extend the long-term plan by an additional 7 days every week. Your long-term plan would always stretch at least 28 days into the future.

When you create your long-term plan, you specify the start date. Although this information is recorded and shown as the long-term plan start date in the Tivoli Workload Scheduler for z/OS panel, occurrences are not maintained as far back as this date. If this were the case, the long-term plan data set would expand indefinitely. When you create a new current plan (NCP) part of the process is to update the long-term plan with any occurrences that are now eligible for removal from the current plan. For details, refer to “Removing data from NCP by running daily planning” on page 250.

The length of the long-term plan is calculated from the date of the earliest occurrence it contains, that is, from the date of the earliest uncompleted occurrence. The long-term plan length is not calculated from the long-term plan start date.
Tivoli Workload Scheduler for z/OS does not delete occurrences from the long-term plan that are uncompleted or any occurrences that follow an uncompleted occurrence. Tivoli Workload Scheduler for z/OS deletes occurrences on a day-by-day basis. That is, all occurrences—or none—are removed for a particular day. This means that if you have occurrences in your long-term plan that are not completed, all occurrences scheduled for that day and all following days are retained, whether the occurrences are completed or not. As part of Tivoli Workload Scheduler for z/OS maintenance, you should assess such uncompleted occurrences and take suitable action. For example, you can manually mark the occurrences as complete, or delete the occurrences if they are not required. This saves space in the VSAM data set that the long-term plan is stored in and reduces the time and resources required to perform long-term plan batch and panel functions.

The long-term plan contains run time and external dependency information for application occurrences. The current plan, which is the Tivoli Workload Scheduler for z/OS detailed schedule, is based on the occurrence information stored in the long-term plan.

Current plan

The long-term plan does not contain all the information that Tivoli Workload Scheduler for z/OS needs to submit work to your system. Before Tivoli Workload Scheduler for z/OS can schedule work, you must produce a detailed plan, called the current plan. The process of producing the current plan is called daily planning (see Chapter 11, “Producing the current plan,” on page 267).

The current plan covers a section of the long-term plan. Typically, the current plan covers a period of 24 hours, although it can range from 1 minute to 21 days. Tivoli Workload Scheduler for z/OS expands the information held in the long-term plan, using the workstation description data and the operation data specified in the application description database. If a current plan already exists, the planning process carries forward any uncompleted occurrences into the new plan.

Tivoli Workload Scheduler for z/OS creates networks of operations using the dependency information for each operation. Planned start and end times are calculated for each operation. This calculation is based on completion of predecessors, resource availability, and occurrence input arrival time. In addition, when you are scheduling jobs in the Tivoli Workload Scheduler environment, the current plan also includes information for the Symphony file. For more information, see “Renewing the symphony file” on page 273.

When the current plan is created, it is updated in real time by events produced by the SMF and JES exits. User programs can automatically report status of operations by calling a Tivoli Workload Scheduler for z/OS routine, or TSO users can issue the OPSTAT command. The current plan can also be updated from the Tivoli Workload Scheduler for z/OS panel. Users with sufficient authority can add, change, complete, or delete operations and occurrences. Work running outside Tivoli Workload Scheduler for z/OS control can trigger an application occurrence to be included in the current plan. Figure 112 on page 250 shows the data used in daily planning.
Removing data from NCP by running daily planning

When creating an updated current plan, daily planning removes selected objects that do not require further processing. These objects are:

- Any completed occurrence and any occurrence in error status including only operations in one of the following statuses:
  - Complete.
  - Suppressed by condition.
  - Ended-in-error, with the RECOVERED BY CONDITION field set to Y.

- Following kinds of dependencies:
  - Standard dependencies on completed predecessors.
  - Evaluated condition dependencies belonging to evaluated conditions.
    Daily planning does not remove any evaluated condition dependency that belongs to an undefined condition. It applies also if daily planning removed the predecessor that the condition dependency refers to. In this case, the predecessor appears as removed when monitoring condition dependencies in the current plan.
  - Evaluated conditions.
  - Conditions defined for operations in suppressed by condition status.

Resolving pending occurrences

To handle occurrences that extend beyond the current plan and dependencies whose predecessors are not yet in the current plan, Tivoli Workload Scheduler for z/OS uses tail plans and pending occurrences.

Tail plans

If an occurrence in the long-term plan has an input arrival time within the date range of the current plan, it is included in the current plan when the plan is created or extended. If the occurrence cannot be completed before the end of the current plan, the current plan is extended internally, up to 28 days beyond the start of the specified planning period to include the entire occurrence. This period beyond the end of the normal plan is called the tail plan.

The tail plan includes work that is planned to start during or before the current planning period, but is not planned to complete before the end of the planning period. New applications with input arrival during the time covered by the tail plan are not included; instead, these will be added by the subsequent daily plan.
These applications are included in the current plan as usual when you extend the current plan beyond their input arrival times.

Operations that belong to included occurrences, but are not scheduled to start within the period of the tail plan, are given the date and time of the end of the tail plan as a start date and time.

**Pending occurrences**

If an operation in an occurrence has an external predecessor, the dependency is resolved when the long-term plan is created or extended. That is, the link is established between the two dependent occurrences.

Although the dependency has been resolved in the long-term plan, it can sometimes happen that the successor of a dependency relationship is included in the current plan but the predecessor is not. This can occur, for example, when a dependency is added manually to the long-term plan. To ensure that the dependency is honored, Tivoli Workload Scheduler for z/OS creates a dummy occurrence in the current plan, called a pending occurrence. The successor operation is temporarily made dependent on this pending occurrence. When the true predecessor of the operation is included in the current plan, the pending occurrence is replaced.

In Figure 113 the application A is included in the new current plan because its input arrival time X falls within the period of the current plan. So the operation Y is also in the current plan. Operation Y has a predecessor Z in application B (dependency D), but application B is not in the current plan, because its input arrival time is later. So Y has a predecessor Z that is not in the current plan, and a pending occurrence is created in the current plan until application B is included.

![Figure 113. Example of a missing predecessor](image)

**Creating the plans for the first time**

See "Creating plans" on page 30 for an example of creating plans for the first time. This is a summary of the steps:

1. Decide how far the long-term plan will extend.

   The long-term plan can span 1 day to 4 years from the date of the last uncompleted occurrence. If you begin with a plan covering too long a period,
Creating the plans for the first time

the task of creating the plan can become unnecessarily cumbersome. However, if you do not look far enough ahead, you will not benefit from the planning functions of the long-term plan.

A good compromise is 5 weeks or 75,000 occurrences, whichever is less. It provides planning capabilities for the near future without being too great. This can be particularly useful, for example, at end-of-month or holiday processing. With this period, the long-term plan can be extended every 7 days for 7 days and will always cover a calendar month. However, if 5 weeks proves to be too short or too long, you can always adjust the look-ahead period to a more suitable length.

2. Decide when the current plan will start.

The current plan is a second-by-second schedule of all operations. It drives all automatic Tivoli Workload Scheduler for z/OS activities, such as:
- Submitting and tracking of jobs and started tasks
- Completing operations on nonreporting workstations
- Providing information to workstation operators
- Recovering failed jobs

When you decide the start time for the current plan, you should note that:
- All scheduled jobs with no predecessors are normally submitted immediately by Tivoli Workload Scheduler for z/OS when the plan is created, unless they are time-dependent operations.
- Long-term plan occurrences with input arrival time before the current plan start time but deadline time after the current plan start time, are given the status UNDECIDED. Tivoli Workload Scheduler for z/OS does not automatically submit jobs with this status. This situation applies only when a current plan does not yet exist.
- A long-term plan occurrence with input arrival time before the current plan start time and deadline time also before the current plan start time is assumed by Tivoli Workload Scheduler for z/OS to have completed. It will be marked as completed in the long-term plan and excluded from the current plan. This situation applies only when a current plan does not yet exist.

Avoid scheduling occurrences in the long-term plan before the start of your first current plan. The current plan can then start at the date and time that you want Tivoli Workload Scheduler for z/OS to begin its scheduling activities.

3. Decide how far the current plan will extend.

The current plan can span from 1 minute to 21 days from the time of its creation or extension. Tivoli Workload Scheduler for z/OS brings in from the long-term plan all occurrences with an input arrival time that is within the period you specify. Tivoli Workload Scheduler for z/OS then creates a detailed schedule for the operations that are contained in these occurrences.

If you are extending the current plan, Tivoli Workload Scheduler for z/OS also carries forward, into the new plan, any uncompleted occurrences in the existing plan. Therefore, the current plan could contain information on occurrences back to the creation of the plan if there are uncompleted occurrences.

To decide what time span your current plan should have, consider:
- The longer the plan, the more computing resources required to produce it.
- Changes in the long-term plan are reflected in the current plan only after the current plan is extended.
- You cannot amend occurrences in the long-term plan that have an input arrival time before the end of the current plan.
- Plans longer than 24 hours will contain two occurrences of daily applications and can cause confusion for your operations staff.
Creating the plans for the first time

- Short plans must be extended more frequently.
- The current plan can contain a maximum of 32,760 application occurrences.

Normally, a plan of 24 hours is a good compromise. However, in very large installations, 24 hours might be too long. You might then consider a current plan covering one shift.

4. Create the plans as described in “Creating plans” on page 30.

5. Consider getting Tivoli Workload Scheduler for z/OS to schedule the batch jobs that extend the plans. For example, you could run the job that extends the current plan every day at 06:00, and run the job to extend the long-term plan weekly.

6. Extend the current plan a few hours before it ends, so you have time to look at the reports.

The following describes what to do when you make changes:

1. Is the change temporary?
   a. Yes: continue with step 3.
   b. No: change the application database. Do you want the change to affect the current plan?
      1) Yes: continue with 1c.
      2) No: modify or extend the long-term plan.
   c. Are you adding or deleting occurrences that are in the current plan?
      1) Yes:
         - Modify or extend the long-term plan.
         - Use the MCP panel to add or delete occurrences.
         - Replan or extend the current plan.
      2) No:
         - Modify or extend the long-term plan.
         - Replan or extend the current plan.

2. Does the change affect the current plan?
   a. Yes: use the MCP panel.
   b. No: modify the long-term plan online.
Creating the plans for the first time
Chapter 10. Producing and modifying the long-term plan

You can use Tivoli Workload Scheduler for z/OS to perform these tasks:

- Create a long-term plan, if one does not exist.
- Extend the long-term plan, if the time period covered by the plan is not far enough into the future.
- Create a trial long-term plan.
- Produce reports on all or part of the long-term plan.
- Check the contents of the long-term plan online.
- Change the long-term plan to fix errors or include last-minute changes.
- Prepare job control language (JCL) for computer operations belonging to occurrences in the long-term plan.
- Modify the long-term plan to reflect changes in the Tivoli Workload Scheduler for z/OS databases.

You access the long-term plan by selecting option 2 from the main menu. The MAINTAINING THE LONG TERM PLAN menu is displayed (Figure 114).

This menu presents you with a list of functions, divided between those that run batch jobs and those that perform online updates.

You use the APPLICATION DESCRIPTION panel to change the AD database. This might cause changes in the run dates, input arrival times, or dependencies of occurrences that are included in the long-term plan. These changes are not reflected in the schedule until the long-term plan and current plans have been updated. You do not need to discard your long-term plan completely. Instead, you can change the long-term plan to reflect these changes. See “Modifying the long-term plan in batch” on page 259 for details. Then use the DAILY PLANNING panel to amend the current plan to include these changes. See Chapter 11, “Producing the current plan,” on page 267 for details.
Running long-term plan batch jobs

Running long-term plan batch jobs

For some functions, such as creating or printing the long-term plan, the panel submits a batch job. You can save the job and submit it outside the Tivoli Workload Scheduler for z/OS panels. You can use Tivoli Workload Scheduler for z/OS to schedule and submit the long-term plan extend job. This ensures that the long-term plan is always updated at the correct time and on the correct date.

The batch jobs normally affect all occurrences in the long-term plan. If you specify the MODIFY ONE or PRINT ONE options, however, only the selected occurrences are modified or printed. Run long-term plan batch jobs by selecting option 2 (BATCH) from the MAINTAINING THE LONG TERM PLAN menu. You see the menu shown in Figure 115.

The long-term plan data sets

The long-term-planning function creates or modifies the long-term plan data set. The long-term plan is stored in a VSAM data set that is defined when Tivoli Workload Scheduler for z/OS is installed. The long-term plan is referenced in the long-term planning jobs and by the Tivoli Workload Scheduler for z/OS address space by ddname EQQLTDS.

The long-term plan VSAM data set does not require regular reorganization because the entire data set is rewritten by long-term plan batch create, modify, and extend functions. A VSAM work data set (ddname EQQLDDS) and a backup data set (ddname EQQLTBP) are used by the long-term plan batch jobs during the planning process.

Long-term plan messages

Many messages issued by the long-term planning process can be dismissed immediately; others will require action. For example, a message indicating that a daily application could not find a once-a-year dependency is not considered an issue for 364 days of the year. Warning messages are generated for all occurrences that have been manually added or changed. Messages are also generated if the long-term plan planning process detects a period definition that has no future origin dates.

Warning messages are generated for all run cycles with an EVERY end time inconsistent with the application calendar work day end time. In this case, the EVERY end time is automatically reset by the long-term plan batch job to the latest valid value possible. For details, see “Specifying the EVERY options for a run cycle” on page 133.
Long-term plan messages

Messages are written to the message-log data set defined by the EQQMLOG DD statement.

How a long-term plan is created

Tivoli Workload Scheduler for z/OS examines every application and job description. If an application is a member of a group, Tivoli Workload Scheduler for z/OS extracts the run cycle and calendar information from the group definition.

When the planning process finds a valid run cycle, it checks for generated dates within the long-term plan range. The planning process must check the defined calendar to determine if the free-day rule should be applied. See “Selecting a free-day rule” on page 130 for details. Tivoli Workload Scheduler for z/OS also checks the workday end time on the calendar; the next day does not start until that time is reached. For example, if you specified a work-day end time of 06:00 hours, a free day following a work day would not start for Tivoli Workload Scheduler for z/OS purposes until 06:00. This means that work might be scheduled on a free day. See “Creating the default calendar” on page 104 for more information.

Tivoli Workload Scheduler for z/OS ignores multiple occurrences that fall on the same day with the same input arrival time because of the free-day rule. It schedules only one occurrence and writes a warning message to EQQMLOG.

Tivoli Workload Scheduler for z/OS creates an occurrence in the long-term plan for each required instance of the application or job description. Application occurrences that are part of an occurrence group are uniquely identified by a reference to the group definition and by a common input arrival date and time.

Tivoli Workload Scheduler for z/OS detects and reports a duplicate occurrence, but does not store it in the long-term plan. This can happen when the run cycles for an application specify that an occurrence should be generated every Friday and also on the last day of the month. In this example, when the last day of the month is also a Friday, only one occurrence of the application is stored in the long-term plan; the other occurrence is canceled because it is a duplicate.

When the planning process creates an occurrence, the operations are examined. If an operation has one or more external predecessors, the planning process tries to make the dependency in the long-term plan. The dependency links the closest occurrence, that is, the occurrence with an equal or the nearest earlier input arrival time. If a predecessor application occurrence does not exist in the long-term plan, no dependency is created. In this case, the planning process issues a message to indicate a potential problem.
Creating the long-term plan

Create a new long-term plan by selecting option 7 from the SELECTING LONG TERM PLAN BATCH JOB menu (Figure 115 on page 256).

```
EQQLCREP ---------------- CREATING THE LONG TERM PLAN ------------------
Command ===> 
Enter/Change data below:
Long term plan:
START ===> 03/01/27 Date in format YY/MM/DD
END ===> 03/04/30 Date in format YY/MM/DD
```

Figure 116. EQQLCREP - Creating the long-term plan

See "Creating the plans for the first time" on page 251 for general guidance on creating the first plan. When creating the long-term plan, you specify the start and end date of the period that the plan is to cover on the CREATING THE LONG TERM PLAN panel (see Figure 116).

If a long-term plan already exists, you will receive a warning message when creating a new long-term plan. If occurrences in the existing long-term plan still exist in the current plan and are not yet complete, you cannot use the create function. To create a long-term plan in this situation, you must refresh the long-term plan. You perform an LTP REFRESH from the SERVICE FUNCTIONS panel (option 9 from the main menu), not from the LTP panel.

**Attention:** A REFRESH of the long-term plan deletes your current plan and you can lose the statuses of occurrences in the long-term plan. When you create the current plan, the status of uncompleted operations might be undecided.

A batch job that you can edit, or submit directly, is generated when you specify the required dates on the CREATING THE LONG TERM PLAN panel. If a long-term plan already exists, you can specify an end date earlier than that of the old long-term plan.

The create job does not use the existing long-term plan as input. Therefore, any occurrences or occurrence groups that are manually added are not included in the new long-term plan. Nor will manual updates that have deleted or changed occurrences be reflected in the new long-term plan. The long-term plan-create batch job performs the planning process as described in “How a long-term plan is created” on page 257.

To decide what time span your long-term plan should have, consider:
- The longer the plan, the more computing resources required to produce it.
- A long long-term plan probably has many occurrences and operations, so there can be a long response time when you edit the long-term plan with the panel.
Extending the long-term plan

Extend the long-term plan by selecting option 3 from the SELECTING LONG TERM PLAN BATCH JOB menu, shown in Figure 115 on page 256. The EXTENDING THE LONG TERM PLAN panel, shown in Figure 117, is displayed.

```
      EQQLEXTP ---------------- EXTENDING THE LONG TERM PLAN ----------------
      Command =>>
      Enter/Change data below:
      Current end : 03/04/30
      NEW END DATE ==> ______ New long term plan end date in format YY/MM/DD
      EXTENSION LENGTH ==> 28__ DDDD Extend plan by
```

*Figure 117. EQQLEXTP - Extending the long-term plan*

To extend the long-term plan, specify a new end date, or extension length in days, for the long-term plan. Tivoli Workload Scheduler for z/OS generates a batch job that you can edit or submit directly. Long-term planning adds only occurrences that have input arrival times outside the period covered by the current plan. When resolving dependencies, Tivoli Workload Scheduler for z/OS uses the long-term plan that already exists. Occurrences that fall within the current plan are also considered in the dependency resolution process.

This procedure is part of your normal maintenance of Tivoli Workload Scheduler for z/OS; extend the long-term plan at regular intervals. Because Tivoli Workload Scheduler for z/OS can schedule the job of extending the long-term plan in the same way as any other job in your system, consider using Tivoli Workload Scheduler for z/OS to regularly schedule a job that extends the long-term plan by a fixed duration each time it runs.

Modifying the long-term plan in batch

You can modify the long-term plan to reflect the latest data available in the Tivoli Workload Scheduler for z/OS database by selecting option 1 from the SELECTING LONG TERM PLAN BATCH JOB panel (see Figure 115 on page 256). This generates modify-long-term plan JCL to execute the planning process. Only occurrences that have input arrival times after the end of the current plan are modified.

Occurrences that have been changed in any way through the LTP panel function are left unchanged by the modify process. This includes manually deleted occurrences. If you delete an occurrence, this is noted in the long-term plan. The same occurrence (that is, an occurrence with the same application ID, input arrival date, and input arrival time) will not then be re-created by the modify-long-term plan batch job.

Any occurrences that are involved in a manually added dependency chain are also left unmodified. A note is also made of manually deleted dependencies. Modifications that would reinstate such deleted dependencies are not permitted.
Modifying the long-term plan in batch

MODIFY can be performed for a single application by selecting option 2 from the SELECTING LONG TERM PLAN BATCH JOB menu (Figure 115 on page 256). If you perform MODIFY for a single application, dependencies are not resolved.

Creating reports about the long-term plan

The long-term planning batch programs generate these reports:
- A list of occurrences by run date
- Workload by workstation
- Run date and error reports

You can also produce reports from the long-term plan by selecting option 5 (print) or option 6 (print one) from the SELECTING LONG TERM PLAN BATCH JOB menu (Figure 115 on page 256). Both functions extract data from the long-term plan and produce a report showing occurrences of the selected applications. Option 5 generates a batch job that produces a report for all occurrences in the long-term plan, whereas option 6 produces a report for one application.

The report provides detailed information that you can inspect before using the long-term plan as input to daily planning.

The full report lists all occurrences in the long-term plan. It shows application IDs, owner IDs, input arrival times and deadlines for the application, priorities, application text, and external dependencies. This report also lists the total duration of the work that is planned for each workstation.

You can request two types of report. Specify F for a full report, or D (dependencies) for a report on predecessors and successors. For each dependency, the dependency report contains the optional text string called reason for dependency. In the application description, you can specify whether this string should always be printed or should be printed only when the dependency is resolved between occurrences on different dates.

Figure 118. EQQLEXTP - Printing the long-term plan, all applications

The report provides detailed information that you can inspect before using the long-term plan as input to daily planning.
Creating reports about the long-term plan

You can specify these sort orders:
I Run date, input arrival time, application ID
O Owner ID, run date, input arrival time, application ID
A Owner ID, application ID

When you request a printout of the long-term plan, you also get histograms showing the planned workstation utilization for the period of the printout for each workstation.

See “Long-term plan reports” on page 716 for examples of the various reports that are generated.

Generated comments in long-term plan reports

The long-term plan reports include information that Tivoli Workload Scheduler for z/OS adds about the way each occurrence has been planned. Tivoli Workload Scheduler for z/OS can add these comments:

DEADLINE – START > 24 HRS
There are more than 24 hours between occurrence input arrival and deadline. The application description perhaps should be split into multiple applications, which could be planned more efficiently.

MOVED (FREE DAY RULE)
An occurrence that would otherwise have been scheduled on a free day has been moved by the free-day rule specified in the run cycle for the application description.

AD NOT FOUND ON AD FILE
The application is not in the application description data set. Some fields in the long-term plan report for this application will be blank.

SCHEDULED ON FREE DAY
The occurrence has been scheduled on a free day. This comment is issued when you are printing the long-term plan sorted by owner ID.

DEPENDENCY CHANGED
The dependency printed on this line has been changed manually using the panels.

ENTERED MANUALLY (ONLINE)
This occurrence has been added manually using the panels or PIF.

CHANGED MANUALLY (ONLINE)
This occurrence has been modified using the panels or PIF.

ERROR CODE = xxxx
Where xxxx is an error code. This error code was added manually using the panels when the occurrence was modified.

Creating a trial long-term plan

When you want to study schedules in detail without changing the production plans, create trial plans. Trial plans are particularly useful:
• During peak workload periods (for example, year end processing) to study the capacity problems that might occur at these times
• When you bring in a new application that could cause changes to the workload
• When overload problems are detected
Creating a trial long-term plan

Create trial long-term plans by selecting option 4 from the SELECTING LONG TERM PLAN BATCH JOB menu (Figure 115 on page 256).

![Figure 119. EQQLTEXP - Making a trial long-term plan](image)

You can produce a trial long-term plan to check the validity of any database changes that you make. This process produces a print of the long-term plan in the same format as reports on the active long-term plan, but the long-term plan file is not updated.

Depending on the day and date information that you specify on the MAKING A TRIAL LONG TERM PLAN panel, Tivoli Workload Scheduler for z/OS produces one of three possible trial plans:

- If you do not specify anything in the fields, Tivoli Workload Scheduler for z/OS produces a modify all trial plan. This simulates modifying the long-term plan, which is described in "Modifying the long-term plan in batch" on page 259.
- If you specify a start date and an end date, Tivoli Workload Scheduler for z/OS simulates a long-term plan create, which is described in "Creating the long-term plan" on page 258.
- If no start date is specified but an end date or an extension length is specified, Tivoli Workload Scheduler for z/OS simulates a long-term plan extend, which is described in "Extending the long-term plan" on page 259.

Displaying the status of the long-term plan

You can display the status of the long-term plan by selecting option 4 STATUS, from the MAINTAINING THE LONG TERM PLAN menu (Figure 114 on page 255). The STATUS OF THE LONG TERM PLAN panel, shown in Figure 120 on page 263 is displayed.
Displaying the status of the long-term plan

EQQLSTAP -------------- STATUS OF THE LONG TERM PLAN --------------
Command ==>>

View data below:

Long term plan start : 96/12/16
Earliest non-completed
occurrence in
current plan : 03/02/03
Latest update of
long term plan : 03/02/06
Current plan end : 03/02/07
Long term plan end : 03/04/30

Figure 120. EQQLSTAP - Status of the long-term plan

This panel displays key information about the long-term plan. The Earliest
non-completed occurrence in current plan field is maintained by the daily
planning process. The long-term plan contains occurrences between the dates
which are indicated by the fields Earliest non-completed occurrence in current
plan and the Long-term plan end.

Setting default successor and predecessor workstations

You can change the default workstations that Tivoli Workload Scheduler for z/OS
uses to resolve external dependencies in the LTP panel by selecting option 5 (SET
DEFAULTS) from the MAINTAINING THE LONG TERM PLAN menu. The SETTING DEFAULT FOR BROWSE panel, shown in Figure 121, is displayed.

EQQLBDWP --------------- SETTING DEFAULT FOR BROWSE ---------------
Command ==>>

Enter/Change data below:

PREDECESSOR WS ==> ____ Default predecessor work station
SUCCESSOR WS ==> ____ Default successor work station

The values set on this panel are used only by the long term plan dialog.

Figure 121. EQQLBDWP - Setting default for browse

When you create a dependency in the long-term plan using the LTP panel, the
dependency is between two application occurrences. Tivoli Workload Scheduler for
z/OS only recognizes dependencies between operations, so it must choose
operations in the predecessor and successor occurrences that the dependency will
actually be between. To do this, the LTP panel uses the information on the
SETTING DEFAULT FOR BROWSE panel. If you have not defined an operation in
the application on the default workstation, Tivoli Workload Scheduler for z/OS
uses the last operation in the application for the dependency.

The values set in the panel are used only by the LTP panel. The BATCHOPT
initialization statement specifies the corresponding defaults for the long-term plan
extend and other batch jobs. Refer to Customization and Tuning for details of
BATCHOPT.
Note: Use the OPTIONS panel to set the default calendar for online changes that you make to occurrences in the long-term plan.

Amending applications in the long-term plan online

You can modify or browse individual occurrences of an application in the long-term plan by using the ONLINE option from the MAINTAINING THE LONG TERM PLAN panel. You can:

- Add an application occurrence to the plan
- Delete an application occurrence from the plan
- Remove an occurrence from an occurrence group
- List all the occurrences of an application in the plan
- List all members of an occurrence group
- Browse individual application occurrences in the plan
- Browse the dependencies of application occurrences in the plan
- Change application occurrences in the plan
- Change the dependencies of application occurrences in the plan. You can:
  - Create or delete non-conditional dependencies.
  - Delete conditional dependencies.
- Prepare job statements for an occurrence in the plan
- Delete, complete, or add an occurrence group

When you modify the long-term plan with these panel functions, Tivoli Workload Scheduler for z/OS updates the plan online, so there is no need to run any batch jobs to carry out your changes. However, if you have made many manual changes, run a print of the long-term plan to verify that the changes are accurate.

A dependency in the long-term plan is between application occurrences. The long-term plan itself does not contain any operations or any operation-level information. When you browse dependencies on an operation level in the LTP panel, the displayed information is therefore only an estimation of how the operations will actually be connected when the current plan is extended.

Modifying occurrences in the long-term plan

Sometimes, you will need to make one-off changes to a particular occurrence of an application. For example, you might need to put a DASD space override in the job for end-of-year processing, or you might need to add some predecessors. Changes to long-term plan occurrences can be performed by selecting the ONLINE option from the MAINTAINING THE LONG TERM PLAN panel. After selecting the occurrence or list of occurrences you want to work with, you can:

- Modify operation data: change the operation text, operation level input arrival and deadline times, or prepare job statements
- Modify dependencies: delete an existing dependency, create new predecessors or successors
- Modify occurrence data: change priority, job variable tables, occurrence level input arrival and deadline times

If you change an individual occurrence and then modify the application description that gave rise to that occurrence, Tivoli Workload Scheduler for z/OS does not change the manually altered occurrence when the long-term plan is extended. Tivoli Workload Scheduler for z/OS issues a warning message to indicate that this occurrence has not been changed in line with the modified application description. Tivoli Workload Scheduler for z/OS assumes that any manual changes you have made should override any automatically generated changes.
Modifying occurrences in the long-term plan

The long-term plan does not include all operation detail. If you want to make changes to dependencies in the long-term plan, Tivoli Workload Scheduler for z/OS lets you establish the dependency to an occurrence, not to a particular operation within an occurrence. For more information see “Setting default successor and predecessor workstations” on page 263.

You can change the input arrival date or time of occurrences that are members of an occurrence group. When you modify the input arrival for an occurrence that is a member of a group, the occurrence is removed from that group. If the new input arrival corresponds to an existing occurrence group, the occurrence will become a member of that group. If there is no existing occurrence group with the new input arrival a new occurrence group will be created. Group members cannot be individually deleted from the long-term plan until they are removed from the occurrence group.

Note: If you edit a job using the LTP panel, the job is saved in the JCL repository data set EQQJSnDS, even if you make no changes, and subsequent changes to the job in EQQJBLIB will not take effect for that occurrence. So be sure to cancel the edit, or use the Browse command, unless you really mean to save the job statements for that particular occurrence in the repository data set.

Modifying dependencies in the long-term plan

Specify option 1 (ONLINE) from the main menu of the LTP panel. The LONG TERM PLAN OCCURRENCES panel, shown in Figure 122 on page 266, is displayed.

<table>
<thead>
<tr>
<th>Row</th>
<th>Application id</th>
<th>Input arrival</th>
<th>Deadline</th>
<th>P</th>
<th>C</th>
<th>Pre</th>
<th>Suc</th>
<th>Cond</th>
<th>Cond</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CP</td>
<td>03/06/05</td>
<td>12.00</td>
<td>03/06/05</td>
<td>16.00</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>MEME</td>
<td>03/06/05</td>
<td>09.00</td>
<td>03/06/05</td>
<td>10.00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>PAYBACKP</td>
<td>03/06/05</td>
<td>12.00</td>
<td>03/06/06</td>
<td>06.00</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>PAYDAILY</td>
<td>03/06/05</td>
<td>12.00</td>
<td>03/06/05</td>
<td>16.00</td>
<td>5</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>PAYW</td>
<td>03/06/05</td>
<td>12.00</td>
<td>03/06/05</td>
<td>16.00</td>
<td>5</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>CP</td>
<td>03/06/06</td>
<td>12.00</td>
<td>03/06/06</td>
<td>16.00</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>MEME</td>
<td>03/06/06</td>
<td>09.00</td>
<td>03/06/06</td>
<td>10.00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>PAYBACKP</td>
<td>03/06/06</td>
<td>12.00</td>
<td>03/06/09</td>
<td>06.00</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>PAYDAILY</td>
<td>03/06/06</td>
<td>12.00</td>
<td>03/06/06</td>
<td>16.00</td>
<td>5</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>CP</td>
<td>03/06/09</td>
<td>12.00</td>
<td>03/06/09</td>
<td>16.00</td>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>MEME</td>
<td>03/06/09</td>
<td>09.00</td>
<td>03/06/09</td>
<td>10.00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>MEME</td>
<td>03/06/09</td>
<td>09.00</td>
<td>03/06/09</td>
<td>10.00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>13</td>
<td>MEME66</td>
<td>03/06/09</td>
<td>00.01</td>
<td>03/06/09</td>
<td>10.00</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>14</td>
<td>PAYBACKP</td>
<td>03/06/09</td>
<td>12.00</td>
<td>03/06/10</td>
<td>06.00</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 122. EQQLSTOL - Long-term plan occurrences

To display the application owner ID, scroll the list right.

If you want to make the 03/02/03 occurrence of PAYW not dependent on PAYDAILY, for example, type m beside the row and press Enter. The MODIFYING AN OCCURRENCE panel, shown in Figure 123 on page 266, is displayed.
Modifying dependencies in the long-term plan

Specify option 2 and press Enter. The MODIFYING DEPENDENCIES panel, shown in Figure 124, is displayed.

To remove the dependency, type the D command beside the PAYDAILY row, and press Enter. Dependencies that you have deleted previously are shown marked with D (the deleted dependency stays in the plan to stop it being restored when the plan is extended).
Chapter 11. Producing the current plan

This chapter shows you how to maintain the current plan (CP), which is the heart of Tivoli Workload Scheduler for z/OS processing. It drives your production workload and provides feedback about the current status of batch work.

During normal processing, the current plan can be updated by job-tracking events, panel users, the program interface, the automatic recovery function, and the event-triggered tracking function. Depending on your workload, the current plan could be updated several times per second. For this reason, and because the current plan is such a critical resource, Tivoli Workload Scheduler for z/OS handles it in a different way from the other databases and data sets. The physical structure of the current plan and the backup process used to ensure integrity are explained in “Organization and integrity of the current plan” on page 280.

Daily planning is the process of producing and maintaining the current plan, which is the detailed schedule of the work that Tivoli Workload Scheduler for z/OS will carry out on your system.

When scheduling jobs in the IBM Tivoli Workload Scheduler environment, current plan processing also includes the automatic generation of the Symphony file. The Symphony file contains the necessary details for scheduled jobs on distributed agents.

You create and maintain the current plan using these options on the main menu:

**Daily Planning**

- Produce current plans, real and trial.

  This chapter describes this panel. As with long-term planning, certain functions take effect online, whereas others are performed by batch jobs submitted from the panel. In certain recovery situations, you might also need to renew the Symphony file.

**MCP**

- Modify the Current Plan.

**Information used by daily planning**

Daily planning uses data from several Tivoli Workload Scheduler for z/OS data sets:

- The long-term plan (LTP), which contains a list of application occurrences to run each day. The long-term plan details the input arrival and deadline times as well as external dependencies for every occurrence.
- The existing current plan. Uncompleted applications need to be included in the new plan, and completed applications are reported on.
- The application description database, which contains the detail of applications at operation level.
- The workstation description database, which shows the open intervals, parallel servers, and fixed resources available at each workstation.
- The resource description database, which has details of special resources.
- The script library, which is used in the creation of the Symphony file.
Information used by daily planning

- The TOPOLOGY initialization statement, which is used for the Symphony file. For more information, refer to *Customization and Tuning*.

The data is collected and used to update the relevant data sets and produce the current plan reports. The daily planning process involves:

- Update of the long-term plan for occurrences that are marked complete or have been deleted in the existing current plan.
- Creation of an updated current plan, called the new current plan (NCP), that contains:
  - Uncompleted operations from the existing current plan.
  - New occurrence selections from the long-term plan according to the specified end date. See Chapter 10, “Producing and modifying the long-term plan,” on page 255.
  - Potential predecessor records for each occurrence. The records are used to establish a list of candidates for successor resolution when an occurrence is added to the current plan using the panel, the program interface, or event-triggered tracking.
- Optional copy of the job-tracking archive log
- Creation of daily planning reports
- Creation of the Symphony file for the distributed agents

Figure 125 shows the required data.

![Diagram of daily planning process]

Figure 125. Data required by the daily planning process

The daily planning process writes messages to the EQMLOG and SYSPRINT data sets. Error and warning messages are indicated by a nonzero return code from the batch job. These messages should be investigated immediately; they can indicate a potential problem in the new plan.

In some cases, where the error is severe, the daily planning process does not create a new current plan. For example, if the daily planning process detects a loop in a chain of dependent operations, the new current plan is not created. Analyzing problems reported by daily planning” on page 277 details how daily planning analyzes dependency loops.

Creating and extending the current plan

You must create a current plan before Tivoli Workload Scheduler for z/OS can schedule work. You must also create a new plan if you have performed a refresh of the long-term plan (see “Creating the long-term plan” on page 258). Once you have created the current plan, you continually extend it.
Creating and extending the current plan

Follow these steps to create the current plan:

1. Before you can create a current plan, you must create the long-term plan that is used as input to the daily planning process. See Chapter 10, “Producing and modifying the long-term plan,” on page 255.

2. Select option 3 (DAILY PLANNING) from the main menu. The PRODUCING OPC DAILY PLANS menu, shown in Figure 126, is displayed.

   EQQDPLNP --------------- PRODUCING TWSz DAILY PLANS -----------------------
   Option ===>
   Select one of the following:
   1 REPLAN - Replan current planning period
   2 EXTEND - Extend the current planning period
   3 TRIAL - Produce a trial plan
   4 PRINT CURRENT - Print statistics for current planning period
   5 SYMPHONY RENEW - Create Symphony file starting from Current Plan

   Figure 126. EQQDPLNP - Producing daily plans

3. Select option 2 (EXTEND).

4. Specify a start date and time, and the end date and time of the planning window required, or the length, in hours and minutes. If you specify a length (extension period), you have the option of counting all days or only work days as part of the extension. For example, assume that Sunday is the only free day in your calendar, and you extend the current plan at noon on Saturday by 24 hours. If you include all days in the extension, by specifying A in the TYPE field, the plan is extended to noon on Sunday. If, however, you specify W in the TYPE field, the plan is extended to noon on the following Monday. Because Sunday is a free day, Tivoli Workload Scheduler for z/OS ignores this day when it calculates the end of the current plan.

   When creating a new plan, it is best to choose a future start date and time, so that jobs do not start running before you have time to check the messages, and so that operations do not have UNDECIDED status.

5. Create a report on the contents of the plan. See “Producing reports using daily planning” on page 273 for details of report options.

6. Inspect the plan, and use the MCP panel to manually correct any differences between the intended and actual plan contents. The first time you create the plan for a production environment, such differences are likely to occur for the applications that are planned to run at the start of the period. This is especially true if these applications are already running but are not controlled by Tivoli Workload Scheduler for z/OS. In this case, Tivoli Workload Scheduler for z/OS might mark an occurrence as completed before full control of workload submission is in place.

7. Use the MCP panel to display a list of the occurrences with UNDECIDED status.

8. Select the occurrences one by one, and either assign the proper status to them or delete them.

9. Obtain a list of all other occurrences that you suspect might have an incorrect status and correct them.

10. Submit a REPLAN job. This updates the long-term plan with the new occurrence data and lets Tivoli Workload Scheduler for z/OS recalculate the input arrival times of the remaining occurrences with the new current plan data.
Extending the current plan

The current plan should always stretch for some hours or days into the future. Extend the current plan at regular intervals, using the EXTEND option of the DAILY PLANNING menu. You can extend the current plan up to 21 days, although one day is more usual. The panel creates a batch job, which you can then submit. You can extend the current plan to a fixed date and time in the future, or you can extend it by a period of hours and minutes.

Figure 2 on page 8 shows a 48-hour current plan. The initial current plan extended 48 hours into the future: every morning the current plan is extended by a further 24 hours.

Input is taken from both the long-term plan and from the present current plan. The planning performed on Tuesday for Tuesday’s work considers the actual situation (both completed and outstanding work) as reflected in the current plan.

Normally, you extend the current plan so that it stretches 24 hours from the end of the previous plan. If you do this every 12 hours, the current plan always extends at least 12 hours into the future. For instance, you could extend the plan by 24 hours at midnight and at noon. The same rules for free days and work days that apply to the creation of the current plan also apply to its extension. See “Creating and extending the current plan” on page 268.

You can automate this process by scheduling the batch job as an operation in an application. If you specify an extension length rather than a fixed date and time in the extend job, the same JCL can be submitted repeatedly to extend the plan.

When the current plan is extended and a new plan is created, all uncompleted work is carried forward into the new plan. New work is brought into the plan from the long-term plan (that is, occurrences whose input arrival times fall within the new planning period). All these operations are competing for time on
workstations. The open intervals and resources for the workstations might have changed between the creation of the old and new plans, or operations might have finished late. Tivoli Workload Scheduler for z/OS must therefore recalculate its schedules in light of the new information now available to it. This can mean that the planned start and end times for some operations might differ in the new plan from those in the old plan.

Planned start times are system calculations of when operations will start, based on estimated duration and input arrival time. An operation might actually start before or after its planned start time, depending on conditions in the system. For example, a job might take less time to run than you estimated when setting up the operation, or your system might have been unavailable for some time. The planned start time is an estimate for your forward planning purposes; Tivoli Workload Scheduler for z/OS does not use it to schedule work.

The daily planning process also calculates a latest out time for each operation. This is the latest time an operation can start in order to meet the stated deadline. If there is contention for resources, Tivoli Workload Scheduler for z/OS allocates the resource to the operation that has the earliest latest out time. The latest out time can also be used to generate alert messages when Tivoli Workload Scheduler for z/OS determines that a deadline is in danger. Refer to Customization and Tuning for more information about the alerts that can be generated.

When the occurrences for the new plan are determined, the application database is used to generate potential predecessor records for every occurrence. Normally, external dependencies that are satisfied because the predecessor operation is completed, are removed from the current plan when the plan is extended. However, you can modify this behavior so that dependencies are maintained. This could prove useful in a case where the occurrence needs to be rerun. Refer to Customization and Tuning for information about setting the KEEPCOMPDEPS parameter in the BATCHOPT initialization statement.

EXTEND also updates the long-term plan with information about occurrences that are not carried forward into the new plan.

When you extend the current plan, you have the option of producing reports on the contents of the plan. See “Producing reports using daily planning” on page 273 for details.

Recreating the current plan

You can use the REPLAN option of the DAILY PLANNING menu to update your existing current plan. This option performs the same functions as the EXTEND option, except that the current plan continues to cover the same time interval as before. No new occurrences can be included in the current plan by a REPLAN. This is because Tivoli Workload Scheduler for z/OS does not let you add to the long-term plan any occurrences that have an input arrival time before, or within, the period covered by the current plan.

Planned start and end times and the latest out time are recalculated when a REPLAN is performed. Any changes in workstation resources or open intervals are also considered. The application database is not used by the REPLAN process to determine the structure of the occurrences; they are copied directly from the old current plan. When the list of occurrences is determined, the application database is used to generate potential predecessor records for every occurrence.
Recreating the current plan

When you REPLAN the current plan, you have the option of producing a report on the contents of the plan. See “Producing reports using daily planning” on page 273 for details.

Creating a trial current plan

Before you create, extend, or replan the current plan, you can create a trial plan to simulate the effect. A trial plan can reveal potential problems and provide the opportunity to avoid such problems before they affect your business systems. You should make a trial plan before every daily plan extend, this acts as an early-warning system.

To run a trial plan that uses VSAM copies as input, you need to follow these steps:
1. Run the EQQPCS03 sample to allocate/delete the data sets intended to contain the VSAM copies.
2. Select the TRIAL option from the DAILY PLANNING menu.
3. Specify which input data sets should be VSAM copies.

Step 1 can be performed only once. Steps 2 and 3 are repeated each time a trial plan is made.

When a trial plan with VSAM copies selected is made, the submitted job will have, in addition to the normal trial job, a first step that will invoke the EQQDPCOP routine to run the selected VSAM copies. This step will always empty and then refill the previously allocated VSAM (EQQPCS03).

If you want to run several TRIAL plans using already existing VSAM copies obtained in a first TRIAL plan, then you need to:
- Select YES on the EQQDTTRP panel in the 'Copy VSAM' field.
- Edit the trial job to be submitted.
- Manually delete the first step in the job (COPYVSM EXEC PGM=EQQBATCH, PARM='EQQDPCOP').

The skeleton member corresponding to the trial plan is EQQDPTTRZ.

When VSAM copies are no longer needed, just run EQQPCS03 and remove the comments from the delete part.

Note: EQQPCS03 has intentionally been kept separate from EQQDPTTRZ to avoid repeatedly having to allocate and delete VSAM data sets each time a trial plan is created.

When a trial plan is produced, the plan is not updated, but you get a report.

For end-to-end scheduling with fault tolerance capabilities

In an end-to-end with fault tolerance capabilities environment, you can run a trial plan to detect potential errors in the SCRIPT library members. Error messages are displayed to help you taking the appropriate actions and prevent the system from having problems.

You can perform a syntactic check of all the script library members by using the EQQSLCHK sample job. This sample runs in stand-alone mode, without interacting with the CP database. For detailed information about how to use and customize the EQQSLCHK sample Job, see the Customization and Tuning manual.
Renewing the symphony file

You can use the SYMPHONY RENEW option of the DAILY PLANNING menu to recover from error situations. In normal situations, the Symphony file is automatically generated during the daily plan processing. Some examples of error situations include the following:

- There is a non-valid job definition in the script library.
- The workstation definitions are incorrect.
- An incorrect Windows user name or password is specified.

But, there are regular operation situations when you need to renew the Symphony file:

- When you make changes to the script library or to the definitions of the TOPOLOGY statement
- When you add or change information in the current plan, such as workstation definitions

Producing reports using daily planning

When creating, extending, or replanning the current plan, you can produce reports on the results. The contents of these reports are determined by the options you select. See “Daily planning reports” on page 720 for examples of the reports. You can also produce a report on completed operations and operations in the existing plan that ended in error, by using the PRINT CURRENT option of the DAILY PLANNING menu. If you need information on the current plan, you can use the QCP option on the main menu to obtain an online view of the current plan.

Daily planning provides two printed outputs:

- Plans
- Management reports

Plan reports

These plan reports can be produced:

Workstation summary histograms
Show the planned use of the workstation and of the two workstation resources.

Daily operating plan
Show all operations and occurrences to be processed.

Special resource planned utilization report
Shows, by interval, the availability of each resource, and possible allocation problems.

Workstation plans
Can be provided for all workstations, or for non-reporting workstations only. They list all operations to be performed at each workstation, in order of planned start time.

Input arrival lists
List all operations to be performed at each workstation, in order of input arrival time.
Management reports

Management reports

Management reports can be created for the current period and for the previous period.

Current period

Current period results are obtained by specifying Y for the Current Period report option when submitting a job to REPLAN or EXTEND the current plan. You can also obtain current period results by submitting the PRINT CURRENT batch job. These reports are included:

Completed applications

This report shows all applications that have been completed or deleted in the given period. Also, each operation that has a specified input arrival or deadline is printed in the report.

The report includes any error code specified when adding an occurrence to the long-term or current plan. An application added with an error code is considered an occurrence rerun, and is reported here, whereas a rerun of one or more operations in an application is reported in the error statistics report.

Operations ended in error

This report lists all operations that have ended abnormally and that have not yet been fixed.

Previous period

You can also produce reports for a period before a REPLAN or EXTEND of the current plan. See “Reports for a previous planning period” on page 725 for examples of the reports.

The previous period is the latest complete 24-hour period starting at the hour specified by the PLANHOUR keyword of the BATCHOPT statement. The default hour is 06:00. Previous period results are stored in the current plan if the PREVRES keyword of BATCHOPT has the value YES, which is the default value. The data is kept in the current plan until the next REPLAN or EXTEND job has reported on the data, and it is then deleted. For example, if you have specified PLANHOUR as 08:00 and you are running EXTEND at 07:00 on Wednesday morning, you will get reports from the interval 08:00 Monday to 08:00 Tuesday, and this data is deleted (if you run another extend at 07:30, there is no previous period report). If you then run REPLAN at noon Wednesday, you will get the reports from 08:00 Tuesday to 08:00 Wednesday. If you wait two days before the next EXTEND or REPLAN, you get a report for the previous two 24-hour periods, because the data is kept until reported on. Refer to Customization and Tuning for details of the BATCHOPT statement.

Previous period reports include:

Summary of completed applications

The report shows the number of applications processed in the period and gives the number of applications:

- With late input arrival, showing the average input delay
- That missed their deadlines, showing the average deadline delay
- That completed before their deadlines, showing the average deadline earliness
- That were rerun
- That were deleted
Previous period

Completed applications
This report has statistics for each application.

Operations ended in error
This report shows the operations that ended in error and have not yet been fixed.

Special resource actual utilization report
Shows, by interval, the availability of each resource, the percentage of time that the resource was not being used, and the percentage of time that operations were waiting.

Error statistics on completed applications
This report (in error-code order) shows:
- Applications that have had one or more operations rerun because of an error and that have now completed successfully (these applications do not appear on the Completed Applications report).
- The error duration (time lost due to errors), if it is not zero.
- The rerun duration (time lost when rerunning completed applications), for any application that has been added to the long-term plan or current plan with a rerun (error) code.
- The total error duration for each error code.
- The total rerun duration for each error code.
- The number of errors for each error code.
- The total number of errors.

Note: An application or operation cannot appear in both the reports Completed Applications and Error Statistics on Completed Applications in any period.

Workstation histograms - actual utilization
The report shows the actual use of workstations for the given period. Completion-only and non-reporting workstations are not included in this report. These histograms can be compared with the planned utilization.

Missed feedback report
The missed feedback report is created by a REPLAN or EXTEND job only if missed feedback data has been created. The report lists all operations and occurrences where the feedback of the actual duration and deadline to the application description database was not possible.

Critical path report
The critical path report is created by a REPLAN, EXTEND, or TRIAL job only if critical job data has been created. The report lists all the critical jobs, assigning one of the following CRIT TYPE values:

ORIG The job has the CRITICAL field set to P.

PRED The job was identified as critical by running the daily planning batch job. This happens when the job is a predecessor of a critical job.
Using the track log

When you execute the daily planning process, you can create a track log data set. The track log contains the job-tracking and audit records from the previous planning period. The track log records are written to the EQQTROUT data set. The track log can be used as an audit trail, as all updates to the plans and databases can be logged in the data set.

The sample library contains an audit package, which can be used to create reports from either the track log or job-tracking data sets. Refer to the Customization and Tuning for more information about the AUDIT statement and the sample audit package.

You can use the Performance Reporter for z/OS licensed program product to create reports using the track log.

Resolving dependencies between operations

The rules used by the daily planning process for resolving external dependencies between operations are:

Case A: The successor operation has no input arrival time specified. An external dependency to an operation in the predecessor occurrence is created if the input arrival time of the predecessor occurrence is earlier than, or equal to, the input arrival time of the successor occurrence.

Case B: The successor operation has input arrival time specified. An external dependency to an operation in the predecessor occurrence is created if the input arrival time of the predecessor occurrence is earlier than, or equal to, the input arrival time of the successor operation.

When the current plan is extended, dependencies between new occurrences and existing occurrences that are carried forward from the old plan are resolved only if the dependency is in the long-term plan when the current plan is extended. An occurrence added by the daily planning process from the long-term plan will not be made a successor to an occurrence added by automatic job recovery, PIF, ETT, or the MCP panel, even if normal dependency criteria are met. For example, consider this case:

1. Application A is run daily. It has an input arrival time of 09:00 and contains one operation, A1. Application A exists in the long-term plan.
2. Application B is run daily. It has an input arrival time of 16:00 and contains one operation, B1. B1 has A1 defined as an external predecessor in the application description database. Application B exists in the long-term plan.
3. An occurrence of application A is added to the current plan from the MCP panel at 12:00. The occurrence input arrival is 12:00.
4. The daily planning batch job is run at 15:00 to extend the current plan. Application B is added to the current plan by daily planning and is given a dependency on the 09:00 occurrence of application A. The external dependency is not resolved to the 12:00 occurrence of application A.
5. If another occurrence of application B is added to the current plan at 16:15 using the MCP panel, with an input arrival time of 16:15, the external dependency is resolved on the 12:00 occurrence of application A.
If you need to change or add dependencies in the current plan, use the MODIFY CURRENT PLAN panel, which is described in “Changing and adding dependencies” on page 555.

Analyzing problems reported by daily planning

When the daily planning process detects a severe problem, it does not create a new plan, but writes messages that describe the problem and sets a nonzero return code. Messages are written to the message-log data set defined by the EQQMLOG DD statement and to the daily plan printout data set defined by the SYSPRINT DD statement. In most cases, the problem is easily solved, such as when an operation refers to a deleted workstation definition. In the case of a dependency loop, however, the problem can be complex; you must check the messages carefully and correct the problem.

A dependency loop can occur in daily planning for several reasons. The most common causes are errors when defining input arrival times or dependencies. A chain of dependent operations, called a network, must have a beginning and an end. If there is no distinct beginning or end, a dependency loop is detected. Sometimes the loop can be small and easy to fix (for example if an operation is defined as both predecessor and successor). In other cases, the loop might involve thousands of operations.

You can detect a loop using a trial plan. To help you correct a dependency loop, the daily planning program analyzes the loop and reports only those operations directly involved in the loop rather than all operations in the network. Tivoli Workload Scheduler for z/OS analyzes the loop and reports on operations that are most likely to cause the loop:

- The operation input arrival time is earlier than the predecessor operation input arrival time.
- The operation is an entry to the loop; it is not in the loop, but a successor operation is in the loop.
- Removal of a dependency has minimal impact on the network but removes the loop.

The criteria are weighted in the listed order. Any operation satisfying the first test is reported as a probable cause.

Complex loops that contain more than one looping path are reported as a single set of looping operation dependencies with more than one probable cause. Figure 128 on page 278 shows examples of the messages issued by the daily planning programs to assist dependency loop resolution.
Analyzing problems reported by daily planning

For detailed information about how daily planning detects a loop condition and provides you with suggestions on how to resolve it, see Chapter 24, “Loop detection and analysis,” on page 509.

For end-to-end scheduling with fault tolerance capabilities

If you are using the end-to-end scheduling with fault tolerance capabilities, the return code of a plan might not be 0. If the return code is 4, the current plan is created, but the Symphony file is not, as logged in EQQMLOG. To create a Symphony file, select the SYMPHONY RENEW option from the PRODUCING OPC DAILY PLANS panel.

Modifying the current plan

To construct a current plan, you must first set up the Tivoli Workload Scheduler for z/OS databases and then construct the long-term plan. Changes in the long-term plan are reflected in the current plan only after you perform a current plan EXTEND or REPLAN. If you want your changes to take effect immediately or if you want to add occurrences to the existing current plan, you must use the MCP (MODIFY CURRENT PLAN) panel.
Querying the current plan

The QCP (query current plan) option from the main menu presents you with a list of resources within the current plan that you can investigate. From here, you can obtain information on operations, workstations, and application occurrences. This information is taken directly from the current plan. The panel can also show a list of all operations that have ended in error.

For more information about the functions available from the CURRENT PLAN AND STATUS INQUIRY panel, refer to “The QCP panel” on page 535.

Current plan reference information

The remainder of this chapter describes the daily planning process in more detail. Refer to this if you have a problem, or if you need to understand how Tivoli Workload Scheduler for z/OS builds the plan.
Organization and integrity of the current plan

**Organization and integrity of the current plan**

Tivoli Workload Scheduler for z/OS is designed so that in most error situations, the current plan can be automatically recovered. If you need to recover the plan manually, refer to *Customization and Tuning*.

These data sets are used for current plan integrity:
- **EQQCKPT**: Checkpoint data set
- **EQQCP1DS**: Primary current-plan data set
- **EQQCP2DS**: Alternate current-plan data set
- **EQQCXDS**: Current plan extension data set
- **EQQDLnn**: Current and inactive dual job-tracking logs
- **EQQJTnn**: Current and inactive job-tracking logs
- **EQQNCPDS**: New current-plan data set
- **EQQNCXDS**: New current plan extension data set
- **EQQJTARC**: Job-tracking archive log
- **EQQSCPDS**: Current plan copy to produce the Symphony file
- **EQQSID**: Side information data set

However, the explanation of the current plan process uses these logical terms to describe the current plan and its associated data sets:

**Current plan**

Used when describing the current plan in general. The current plan consists of the active current-plan data set, the extension (CX) data set, and the side data set (EQQSID). These data sets are described later.

**Active current plan**

Refers to the current-plan data set that is currently in use within Tivoli Workload Scheduler for z/OS. It might be either EQQCP1DS or EQQCP2DS. Every time a current plan backup is performed, Tivoli Workload Scheduler for z/OS switches the active current plan to the other data set. The current plan backup process is described in the section *Current-plan backup process* on page 283.

**Backup current plan**

Refers to the current-plan data set that is not currently in use. This data set contains a backup copy of the current plan. It might be either EQQCP1DS or EQQCP2DS.

**Side information**

Refers to a file (EQQSID) that contains frequently referenced database information, ETT criteria, and configuration information.

**Current plan extensions**

Refers to a file that contains current special resource information. The current file refers to EQQCXDS, and the daily planning process creates EQQNCXDS.

**New current plan**

Refers to a new version of the current plan, which is created by one of the daily planning batch jobs. It always refers to EQQNCPDS.

**Current and inactive job-tracking log**

Refers to the data sets used to log updates to the current plan and to record audit information for requested files. You must use at least two job-tracking logs, referenced by ddnames EQQJT01 and EQQJT02. You can use up to 99 job-tracking logs. The job-tracking logs are used in a cyclic manner, Tivoli Workload Scheduler for z/OS automatically switches to the next available job-tracking log after a current plan backup. The data from
Organization and integrity of the current plan

the inactive data set is copied to the archive log, and the data set is considered emptied in preparation for future use. You should use at least 5 job-tracking logs. The default number specified by the JTLOGS keyword of the JTOPTS initialization statement defines 5 job-tracking logs.

Current and inactive dual job-tracking log
If the dual logging function has been requested, Tivoli Workload Scheduler for z/OS duplicates the job-tracking records in the corresponding dual job-tracking log. Dual logs are switched at the same time, and in the same sequence, as the job-tracking logs. So the number of dual job-tracking data sets is determined by the number of normal job-tracking data sets.

Job-tracking archive log
Represents the accumulated job-tracking data since the new current plan was created. When the job-tracking log is switched, the data from the inactive data set is appended to the archive log. The archive log is copied to the track-log data set referenced by the EQQTROUT ddname during the daily planning process. When Tivoli Workload Scheduler for z/OS takes over the NCP, the archive data set is emptied.

Checkpoint
Refers to the EQQCKPT data set, which contains information about the current status of the Tivoli Workload Scheduler for z/OS system, including which current plan and job-tracking data sets are currently active.

Symphony file
Represents a local plan for a set of fault-tolerant workstations and is updated according to the changes in the local and current plan.

The basic principle of current plan recovery is that if the active current plan becomes unusable for any reason, Tivoli Workload Scheduler for z/OS should always be able to re-create an up-to-date current plan from the backup current plan and the job-tracking logs. The way Tivoli Workload Scheduler for z/OS does this is described in Customization and Tuning.

Current plan during normal processing
Normal processing means that the Tivoli Workload Scheduler for z/OS subsystem is running, job tracking is active, and users have access to the Tivoli Workload Scheduler for z/OS subsystem. Job tracking is active means that an active current plan is updated as events occur in the operating system.

Figure 131 on page 282 shows the current plan with associated data sets and how they are used during normal processing.
Tivoli Workload Scheduler for z/OS can update the current plan:

- With event information from the event data sets, for example job ABC started, job XYZ ended
- With panel requests, for example from the MODIFY CURRENT PLAN panel
- With requests from the Tivoli Workload Scheduler for z/OS program interface
- As a result of a triggering event recognized by the event-triggered tracking function
- As a result of a request from Tivoli Workload Scheduler for z/OS automatic recovery statements

Every time the active current plan is updated, a record of the change is written to the active job-tracking log (JT log). The job-tracking record is also written to the dual job-tracking log if dual logging has been requested.

The remaining data sets are not used during normal processing:

- The checkpoint data set contains status information; for example, which physical data set is the active current plan.
- The backup current plan contains a copy of the current plan as it was at the last successful current plan backup. In recovery situations, this is used with the job-tracking log to re-create an up-to-date current plan. The current plan backup process is described in "The current-plan backup process" on page 283.
- The new current plan (NCP) is used by the daily planning batch jobs when extend or replan is requested. These batch jobs create the new plan in this data set. The NCP is also used to re-create the current plan in conjunction with the...
Current plan during normal processing

various job-tracking logs, if a usable current plan is not available or when you specifically request to start Tivoli Workload Scheduler for z/OS from the NCP.

• The inactive job-tracking logs and the inactive dual-logging data sets.

The current-plan backup process

Tivoli Workload Scheduler for z/OS automatically backs up the active current plan at certain stages in processing.

This is a step-by-step description of the current-plan backup process:

1. Tivoli Workload Scheduler for z/OS locks the current plan to prevent it from being updated during the backup process. During the backup, events are queued in storage. Panel users working with the current plan might experience a short delay.
2. The CX data space (EQQCXDS) is backed up to DASD.
3. The backup current plan is erased.
4. The active current plan is copied to the backup current plan. The contents of the two are now identical.
5. The data sets are switched. The backup current plan becomes the active and the active becomes the backup.
6. A current plan backup record is written to the job-tracking log, and the next available job-tracking log becomes active. The corresponding dual job-tracking log is also switched.
7. The current plan is unlocked. Normal processing continues. Events queued in storage start to update the active current plan. Panel users’ requests are processed.
8. The data from the now inactive job-tracking log is appended to the job-tracking archive log. The inactive job-tracking log is emptied for future use.

A current plan backup is performed at these times:

• During normal processing, according to the value of the BACKUP parameter of the JTOPTS initialization statement. This parameter specifies the number of current plan updates that must occur before a current plan backup is performed.
• When the BACKUP command is issued specifying the current plan resource. See "BACKUP" on page 648 for more information.
• Immediately before normal termination of the Tivoli Workload Scheduler for z/OS subsystem.
• When Tivoli Workload Scheduler for z/OS detects that a daily planning batch job has started.
• After a daily planning batch job has created a new current plan.
• After current plan recovery processing has successfully re-created an up-to-date current plan.

Creating and activating the new current plan

The creation of a new current plan is performed by the daily planning batch jobs. The two possible options are extend and replan the current plan. You also use the extend function when creating a current plan for the first time. Both the extend and replan functions cause the creation of a new current plan in the new current-plan (NCP) data set.

The steps below describe in detail how the new current plan is created and then brought into normal processing, including the use of the Symphony file:
Creating and activating the new current plan

A Daily Planning Job Starts and Is Recognized by Tivoli Workload Scheduler for z/OS.

1. A daily planning batch job starts, signals the Tivoli Workload Scheduler for z/OS subsystem using an ENQ, and then waits.

The Active Current Plan Is Backed Up

2. Tivoli Workload Scheduler for z/OS detects the ENQ from the batch job and locks the current plan, thereby preventing more updates. All events related to job streams and jobs in IBM Tivoli Workload Scheduler are queued for later processing.

3. The active current plan is updated with the latest information from in-storage control blocks representing workstations and active operations. The extension file (CX), which is held in a data space, is refreshed to DASD.

4. The inactive current plan is erased.

5. The active current plan is copied to the inactive current plan. They are now identical.

6. The inactive current plan becomes the active current plan, and the previously active becomes the inactive.

7. The job-tracking logs are switched.

8. The current plan is unlocked, and normal processing continues. The current plan is updated, processing the queued events: these updates are logged into the currently active job-tracking log.

9. The data from the inactive job-tracking log is appended to the job-tracking archive log.

10. Tivoli Workload Scheduler for z/OS signals to the batch job that backup processing is complete.

11. No more events for IBM Tivoli Workload Scheduler are processed. These events are processed later in the new plan using the usual processing for IBM Tivoli Workload Scheduler.

The daily planning job builds a NCP

12. The batch job starts executing again. The inactive current plan is used (together with the long-term plan, application description, resource database, and workstation for a current plan extend) as input, and a new current plan is created in the NCP and new current plan extension (NCX) data sets. While the batch job is building the new current plan, Tivoli Workload Scheduler for z/OS continues normal processing except that a current plan backup is not permitted because the batch job is using the inactive current plan data set.

13. In the new current plan, the daily planning job flags the jobs and job streams that will be added to the Symphony file.

14. The daily planning batch job starts updating the Symphony file. Message EQQ3133I INITIALIZING NEW SYMPHONY FILE (RUN NUMBER = \( nn \)) is issued by the DP batch job at this time.

15. The contents of the job-tracking archive data set are copied to the data set in the daily planning job referenced by the EQQTROUT ddname.

16. When the new current plan is ready, the checkpoint data set is updated to show if the new current plan was successfully created. The normal mode manager (NMM) subtask is notified that the daily planning process has completed.

17. The subsystem examines the checkpoint data set to see if a new current plan was successfully created. In this case, the scheduler sets the OPCTUNCP flag to \( H \) in the EQQCKPT header record, indicating that an NCP is being taken.
Creating and activating the new current plan

over. If the end-to-end scheduling with fault tolerance capabilities feature is not active (no TPLGYPRM parameter in BATCHOPTS), skip to step 21.

18. The scheduler sends a command to stop the local fault-tolerant end-to-end scheduling and sets a timer to 5 minutes (300 seconds).

19. If a response arrives within 5 minutes, indicating that all the local fault-tolerant end-to-end scheduling stopped, processing continues with step 21.

20. If a response does not arrive within 5 minutes, the scheduler continues as if a daily planning did not run (steps 21-34 are skipped). This might leave the LTP header record updated as if the NCP was taken over while the current plan header shows the prior value for CP end date and time.

21. The current plan is locked, preventing it from being updated. All events related to job streams and jobs for IBM Tivoli Workload Scheduler are queued for later processing.

22. The new current plan is copied to the active current plan, and the NCX is copied to the current plan extension (CX).

23. The job-tracking archive log is emptied.

24. The active job-tracking log now contains a record of the updates to the current plan that were made while the daily plan job was running. These are read, and the current plan is updated accordingly.

25. In-storage control blocks representing workstations and active operations are rebuilt from the active current plan, and a data space is created from the current-plan-extension data set.

The Newly Created Active Current Plan Is Backed Up

26. A current plan backup is performed. The inactive current plan is erased.

27. The active current plan is copied to the inactive one, and EQQSCPDS is produced. This VSAM is a copy of the current plan and will be used by data processing to add job information to the Symphony file.

28. The next available job-tracking log becomes active.

29. The current plan is unlocked and normal processing continues. All changes to jobs in the Symphony file are sent to the distributed agents, when the Symphony file is available.

30. The data from the now inactive job-tracking log is copied into the job-tracking archive log.

The Symphony File is Updated

31. Starting from EQQSCPDS that contains the updates for the job-tracking file, the job and job stream information is added to the new Symphony file. If a problem occurred during the building of the Symphony file, the Symphony file is not built. To create the Symphony file, you must perform a Symphony renew after correcting the errors.

The Symphony File is Distributed.

32. IBM Tivoli Workload Scheduler for z/OS sends the Symphony file to IBM Tivoli Workload Scheduler.

33. Before distributing the new Symphony file, all the events of the old Symphony file that are still outstanding are applied to the new Symphony file.

34. In IBM Tivoli Workload Scheduler, the new Symphony file is distributed.

Notes:

1. Consider that all long-term plan and current plan batch planning jobs have to be excluded from SMARTBATCH DA (Data Accelerator) processing. When the SMARTBATCH DATA ACCELERATOR is used with the long-term plan and
Creating and activating the new current plan

Current plan batch planning jobs, the normal I/O to EQQCKPT is delayed until END OF JOB (or at least END OF JOBSTEP). This interferes with the normal exchange of data between the batch job and the controller started task so that when the batch job signals the controller to check the EQQCKPT to determine whether a new current plan has been created, the required updates to the CKPT have not yet been made. This causes the controller to conclude that no NCP has been created, and no turnover processing is done. As a result, even if the plan jobs run successfully, the NCP is not taken into production by the controller unless a CURRPLAN(NEW) restart is performed.

2. Batch optimizer utilities, such as BMC Batch Optimizer Data Optimizer and Mainview Batch Optimizer, prevent correct communication between the scheduler's controller and CP/LTP batch planning jobs. The scheduler's logic depends on an exchange of enqueues and real-time updates of several sequential data sets to pass information back and forth between the controller's STC and the CP/LTP batch planning jobs. These optimizers hold I/O from the batch jobs until END OF STEP or END OF JOB, then preventing the required communication from taking place. When such utilities are allowed to "manage" I/O for the scheduler's CP or LTP batch planning jobs, communication between the jobs and the controller is disrupted. This causes numerous problems that are hard to diagnose. Most commonly, the CURRENT PLAN EXTEND or REPLAN jobs will run to normal completion, and an NCP data set will be successfully created, but the controller will fail to automatically take the new plan into production until it is forced to do so via a CURRPLAN(NEW) restart of the CONTROLLER. Use of BATCHPIPES with these batch planning jobs will result in the same sorts of problems.

Problems with current plan batch jobs

The recovery actions you take are crucial any time a CP batch job ends unsuccessfully, for example:

- A DP batch job abnormally ends or is cancelled.
- A system outage occurs while a DP batch job is running.
- In an end-to-end with fault tolerance capabilities environment, a problem related to UNIX system services (HFS, zFS, OMVS) occurs during DP batch processing.

Before taking any recovery action, do the following:

1. Check if the NCP you have is valid for a restart, looking for message EQQN057I in the controller message log. If you do not find it, do NOT restart the controller with JTOPTS CURRPLAN(NEW). In fact the controller issues message EQQN057I when recognizing the NCP, created by the DP process, as new version of the CP.

2. If you must restart the controller after unexpected events, such as system crash or controller abend, before starting the controller temporarily set in the in JTOPTS statement:
   - JOBSUBMIT(NO)
   - If the end-to-end scheduling with fault tolerance capabilities is active, FTWJSUB(NO)

3. With controller running, compare the current plan end value reported in LTP (panel 2.4 EQQLSTAP) with the planning period end value reported in CP (panel 6.6 EQQGSCPP). If the EQQLSTAP value is more recent than EQQGSCPP value, it is a symptom of a synchronization problem during the Symphony file creation process.
Recovery actions

Take different actions depending on whether the batch job log shows message EQQ3105I, meaning that the batch job produced the NCP:

- If the batch job produced the NCP, check if the controller message log contains message EQQN057I with the FROMDD variable set to EQQNCPDS: if you find it, then the controller recognized the NCP. If the end-to-end scheduling with fault tolerance capabilities is active, message EQQN057I is issued after the synchronization phase ended, therefore check if a synchronization process is in progress and wait for its end to understand whether the controller recognized the NCP or not. The synchronization phase lasts approximately 5 minutes, starting with EQQN117I message and ending with EQQZ195I (successful synchronization) or EQQN064E (synchronization failed).
  - If the controller recognized the NCP then, if the end-to-end scheduling with fault tolerance capabilities is active, run a Symphony renew, otherwise no further action is needed.
  - If the controller did not recognize the NCP, take the following actions:
    - Run a DP extend, with the TPLGYPRM parameter commented out in the BATCHOPTS statement.
    - Run a Symphony renew, with the TPLGYPRM parameter uncommented in the BATCHOPTS statement.

- If the batch job did not produce the NCP, rerun the batch job.

Post-recovery actions

After completing the recovery actions, consider activating the job submission, by using option 9.2 of the ISPF dialog, and restoring the original values for the JOBSUBMIT and FTWJSUB controller parameters.
Post-recovery actions
Chapter 12. Using service and optional functions

This chapter describes the service and optional functions, which let you do the following:
- Activate or deactivate job submission
- Activate or deactivate automatic job and started-task recovery
- Refresh the long-term plan
- Refresh the RACF resource profiles
- Activate or deactivate event-triggered tracking
- Produce an authorized program analysis report (APAR) tape
- Produce report of track log events

Select option 9 on the main menu to see the SERVICE FUNCTIONS menu ([Figure 132]). Select option 10 on the main menu to see the OPTIONAL FUNCTIONS menu for track logs.

### Activating and deactivating job submission

When you deactivate job submission, operations are not started on automatic reporting computer workstations and remote engine workstations. Therefore, no jobs or started tasks are submitted. Also, WTO operation messages are not issued. When you activate job submission again, the operations that were scheduled to start are started.

After a failure, when you need to check the status of jobs in the plan before anything is submitted, you can:

1. Specify JOBSUBMIT(NO) in the JTOPTS initialization statement on host systems. If you are scheduling end-to-end with fault tolerance capabilities, also specify FTWJSUB(NO) in the JTOPTS initialization statement for fault-tolerant workstations.
2. Bring up Tivoli Workload Scheduler for z/OS to recover the current plan.
3. Check that operations have the correct status.
4. Activate job submission when you are satisfied.

Activating and deactivating automatic job and started-task recovery

Automatic job recovery is normally started when Tivoli Workload Scheduler for z/OS is started. However, there are situations in which you might not want automatic job recovery, for example, when a recovery action might occupy resources required for other purposes.

You can inhibit automatic recovery in several ways. For example, you can specify this in the time parameter of the RECOVER statement and in the start-time and end-time parameters of the AROPTS (automatic job recovery options) statement. Another way is to deactivate automatic job recovery. When automatic job recovery is deactivated, ended-in-error jobs remain on the ended-in-error list, regardless of the presence of RECOVER statements.

When jobs that ended in error while automatic job recovery was deactivated are activated, they are first tested for RECOVER statements, and any necessary recovery actions are performed.

**Note:** On the HANDLING OPERATIONS ENDED IN ERROR panel in the MCP panel, you can request (with the ARC row command) automatic job recovery for jobs ending in error, even if you have deactivated automatic recovery. But you cannot activate automatic recovery here unless RECOVERY(YES) is specified on the OPCOPTS initialization statement.

For information on using automatic recovery, see Chapter 18, “Automatic recovery of jobs and started tasks,” on page 347 and “What you need for automatic recovery” on page 306.

Refreshing the long-term plan

Tivoli Workload Scheduler for z/OS has a safeguard against changing plans after they are used. Once an occurrence in the long-term plan has been scheduled in a current plan (CP), you cannot change it in the long-term plan, nor can you schedule it a second time in a current plan. There might be times, however, when you need to bypass this safeguard. The refresh function is provided for that purpose.

**Note:** Use the refresh function only when you have no other alternative because your current plan will be deleted.

You might have a situation in which you want to delete the current plan and produce a new current plan for the current period, using a new version of the long-term plan. To do this, you must go through a complete replanning cycle, including producing a new long-term plan and current plan. To do this, first run a refresh, which:

- Makes already scheduled occurrences available to both long-term and daily planning.
- Deletes the current plan so that you can make a new current plan.
Refreshing the long-term plan

If you want completely new plans, follow the refresh function with a CREATE of the long-term plan and then a daily planning EXTEND. The create long-term plan function is necessary only if the old long-term plan is no longer valid.

Refreshing RACF resource profiles

In-storage resource profiles are built for RACF-defined resources at Tivoli Workload Scheduler for z/OS start time. This is done for resources of the class that you defined in the AUTHDEF initialization statement. When you define new RACF resources after the Tivoli Workload Scheduler for z/OS start time, they are not immediately available to Tivoli Workload Scheduler for z/OS. Select the RACF RESOURCES option on the SERVICE FUNCTIONS panel to refresh the in-storage profiles and make them available to all panel users.

Activating and deactivating ETT

When Tivoli Workload Scheduler for z/OS is started, the initial status of the ETT is decided by the value of the ETT keyword on the JTOPTS initialization statement. You can change this status, while Tivoli Workload Scheduler for z/OS is active, using the Activate and Deactivate ETT functions. See "Adding occurrences by event-triggered tracking" on page 432 for more information.

Producing an APAR tape

You can use the authorized program analysis report (APAR) tape function when you are reporting a problem to Customer Support. The APAR tape function generates a batch job that collects data sets that are useful for problem determination and writes them to a tape. You might need to modify the JCL that is generated so that all your event data sets are collected.

For a detailed description of the procedure for documenting and reporting problems, refer to Diagnosis Guide and Reference.

Producing a report of track log events

In addition to the service functions, there are also additional optional functions you can use. You can create a report of track log events. To produce the report, you have the following options:
- Audit, debug, or both
- Audit created through job batch submission

When creating the report, the only required value is for the type of input for the report. If you do not provide data for the remaining fields, all the records in the input files are selected. Also, the output of the report is written to the file specified during installation.

Follow these steps to produce a report of track log events:
1. Select option 10 from the main menu. The OPTIONAL FUNCTIONS menu is displayed.
2. Select option 1 (AUDIT/DEBUG). The EXTRACT/FORMAT LOG RECORDS SPECIFY SELECTION CRITERIA panel is displayed.
3. On this panel, select the option for either real-time data (JTX) or previous period data (TRL) to be on the report.
4. Optionally, specify a string in the Search-string field to be searched for in the input records.
Producing a report of track log events

5. Optionally in the time and date fields, specify filters that limit the size and time period of the report.

6. Optionally, specify a dsname to replace the value defined at the time of installation.

For additional information, refer to Planning and Installation and Customization and Tuning.
Chapter 13. How work is selected for automatic submission

This chapter describes how Tivoli Workload Scheduler for z/OS decides the run order of operations on computer, nonreporting, and WTO workstations.

Tivoli Workload Scheduler for z/OS builds the current plan from information in the long-term plan, calendar database, application description database, workstation database, and resource database. When the current plan is created, Tivoli Workload Scheduler for z/OS assigns start times to each operation. These start times are Tivoli Workload Scheduler for z/OS’s estimates of when the operations should start. They are not the actual times that Tivoli Workload Scheduler for z/OS will start the operations, unless the operation has been designated as a time-dependent operation. Here, the start time is the time that Tivoli Workload Scheduler for z/OS will attempt to start the operation. (See “Creating time-dependent operations” on page 168.) This is because Tivoli Workload Scheduler for z/OS attempts to maximize throughput in your system by starting as many operations as soon as possible.

Identifying the order of submission

To meet occurrence deadlines, Tivoli Workload Scheduler for z/OS must assess which operations, of those that are eligible, should be processed first. Tivoli Workload Scheduler for z/OS does this in two stages:

1. It constructs a list of work that is ready to be started on each computer workstation and nonreporting workstation. This list, called a ready list, shows all operations with ready status, in the order that they should be started.
2. From each of the ready lists, it selects the most urgent operation for which resources are available. It then compares these operations and selects the single most urgent operation to be started.

Before deciding which operation to start, the daily planning function first builds queues of work that is ready to be processed at each workstation. These queues are lists of operations that are ready to be run, that is, operations with no outstanding predecessors. The operations are placed on the workstation ready list in the order that Tivoli Workload Scheduler for z/OS thinks they should be run. They are sorted in this order:

1. Operations that have the urgent flag set on.
   The urgent flag is automatically turned on by Tivoli Workload Scheduler for z/OS when an operation has been defined with priority 9, or when an operation has missed its deadline.
2. Earliest latest start time.
   The latest start time is the latest time that the operation can start in order to meet the deadline. This calculation considers the operation deadline, estimated duration, resource requirements, and successor processing. When Tivoli Workload Scheduler for z/OS creates the current plan, it calculates the latest start time for all the operations in a chain of dependencies, starting at the last one.
3. Operation priority (other than 9).
4. Shortest estimated duration.
Identifying the order of submission

Tivoli Workload Scheduler for z/OS builds these queues when there is more than one operation that is ready to be started. Operations that are time dependent, or waiting for resources to become available, are not included.

For example, if Tivoli Workload Scheduler for z/OS finds two operations in ready status, it checks if either operation is marked as urgent. If neither is urgent, the latest-start times are compared. In the unlikely event that the latest-start times are equal, Tivoli Workload Scheduler for z/OS examines the priority. If the priority is also equal, Tivoli Workload Scheduler for z/OS first starts the operation that has the shortest estimated duration.

The submission process can deal with many candidates for submission per second. It is unlikely you will ever see a significant backlog of operations to be started.

The ready list is the order that the operations should be started in. If a workstation is closed, the operations on that workstation’s ready list are not eligible for processing. To decide which operation to start, Tivoli Workload Scheduler for z/OS scans the ready lists of each computer, nonreporting, and WTO workstation that is open for operations that are eligible to start. To be eligible to start, the resources that the operation requires must be available, according to the information recorded in the Tivoli Workload Scheduler for z/OS database.

Operations that have been placed in HOLD status by a dialog user are not eligible candidates for submission until they have been set to RELEASE. For more information on holding and releasing operations in the current plan, refer to “Delaying an operation, and releasing it” on page 531.

If sufficient resources are not available, or the operation is waiting for a particular time, Tivoli Workload Scheduler for z/OS assigns an appropriate extended status. The extended status identifies which type of resource the operation is waiting for.

Operations that have been placed in waiting for Scheduling Environment (status READY, extended status W) are not eligible candidates for submission: they become eligible again after the event stating that the Scheduling Environment is again available has been received by the controller and the extended status has been reset.

If no sufficient resources are available, or if the operation is waiting for a particular one that is unavailable at the time, Tivoli Workload Scheduler for z/OS assigns an appropriate extended status. The extended status identifies which type of resource the operation is waiting for.

The submission process

For Tivoli Workload Scheduler for z/OS to start an operation, job submission must be activated for all operations. The job options of the particular operation must also have automatic submit set to yes. If this is not the case, the operation remains on the ready list in R, *, or A status. All of these rules are bypassed, however, if an operator uses the EXECUTE command, which starts the operation regardless of all normal scheduling criteria.

The status of an operation on a nonreporting workstation is automatically set to C (complete) when the operation is ready to be started. Operations that have been stopped with the NOP command by an operator are also automatically set to
The submission process

When an operation on a fault-tolerant workstation has resolved all dependencies and is ready to start, and has Automatic Submit set to YES, the controller changes the status to S with an extended status of Waiting for Submission. When the Controller is notified that the operation has started, the status remains simply as S.

On non-centralized operations, if the Automatic Submit is set to NO, the priority of the corresponding job in the Symphony file is set to 0. When the Automatic Submit is set to YES, or an EXECUTE command is issued, the Occurrence priority is reset to its real value, calculated by multiplying the Application priority by 10.

Submitting work on nonreporting workstations

The submission process is initiated when Tivoli Workload Scheduler for z/OS finds an operation on a ready list in A, R, or * status and all requested resources are available. Operations that have been manually set to HOLD by dialog users will not be considered candidates for submission until the operation is set to RELEASE or a dialog user requests EXECUTE.

This is how Tivoli Workload Scheduler for z/OS submits operations on nonreporting workstations:
- The current plan (CP) is locked to prevent other tasks or users from updating the operations.
- The best candidate for submission is chosen as described in “Identifying the order of submission” on page 293.
- Tivoli Workload Scheduler for z/OS sets the operation status to complete.
- If there are more operations eligible to start, Tivoli Workload Scheduler for z/OS will move on to the next best candidate. Up to 5 operations can be started.
- The current plan lock is released.

If the JTOPTS keyword QUEUELEN is specified, Tivoli Workload Scheduler for z/OS will start up to the number of operations defined by QUEUELEN. Refer to Customization and Tuning for more information on the QUEUELEN parameter.

Submitting work on WTO workstations

This is how Tivoli Workload Scheduler for z/OS submits operations on WTO workstations:
- The current plan is locked to prevent other tasks or users from updating the operations.
Submitting work on WTO workstations

- **The best candidate for submission** is chosen as described in “Identifying the order of submission” on page 293.
- If the operation has been marked with a NOP, Tivoli Workload Scheduler for z/OS immediately sets the operation to complete.
- Tivoli Workload Scheduler for z/OS gathers the operation text and builds the required WTO message.
- A submit request is queued, which includes the WTO and the workstation destination.
- If there are more operations eligible to start, Tivoli Workload Scheduler for z/OS will move on to the next best candidate. Up to 5 operations can be started.
- The current plan lock is released.
- Tivoli Workload Scheduler for z/OS determines the destination, which can be a submit/release data set, an XCF link, an NCF link, a TCP/IP link, or blank. If the destination is blank, the WTO is issued on the system that the controller in started on.
- The workstation reporting attribute for the WTO workstation will determine what status the operation will be set to. Operations defined on workstations with automatic, or manual start and complete reporting will be set to S status. Operations on workstations defined with manual-complete-only reporting are immediately set to C status.

If the JTOPTS keyword QUEUELEN is specified, Tivoli Workload Scheduler for z/OS will start up to the number of operations defined by QUEUELEN. Refer to *Customization and Tuning* for information on the QUEUELEN parameter.

**Submitting started tasks**

This is how Tivoli Workload Scheduler for z/OS submits operations on computer workstations with the STC attribute:

- The current plan is locked to prevent other tasks or users from updating the operations.
- **The best candidate for submission** is chosen as described in “Identifying the order of submission” on page 293.
- If the operation has been marked with a NOP, Tivoli Workload Scheduler for z/OS immediately sets the operation to complete.
- Tivoli Workload Scheduler for z/OS attempts to find the STC procedure JCL in the JS file. The JCL will be found here only if a dialog user has updated the JCL or if this operation occurrence has been previously submitted.
- If the JCL is not found on the JS file, Tivoli Workload Scheduler for z/OS attempts to find it (the member name must be the operation name) in the EQQJBLIB partitioned data set concatenation, first calling the EQQUX002 exit if it is loaded.
- If no JCL can be found, the operation is given an extended status E to indicate the error; and a message is written to the Tivoli Workload Scheduler for z/OS message log.
- If the JCL is found, either in the JS or in EQQJBLIB or returned by EQQUX002 exit, JCL substitution is performed, if required, and the job submit user exit is called.
- A copy of the JCL is written to the JS file. The JCL and workstation destination are queued. The operation status is set to S and extended status U is assigned to indicate that submission is in progress.
Submitting started tasks

- If there are more operations eligible to start, Tivoli Workload Scheduler for z/OS will move on to the next best candidate, up to 5 operations can be started.
- The current plan lock is released.
- The destination is resolved, it can be either a submit/release data set, an XCF link, an NCF link, a TCP/IP link, or blank. If the destination is blank, the JCL is directed to the system the controller is started on.
- When the JCL arrives at the destination, the procedure JCL is written into the partitioned data set referenced by ddname EQQSTC.
- Tivoli Workload Scheduler for z/OS then builds and issues the z/OS START command. Once the command is issued and processed by JES, the procedure JCL is deleted from the EQQSTC data set.
- The extended status of the operation is set to blank.
- When Tivoli Workload Scheduler for z/OS learns from JES that the started-task is executing, the extended status is set to S.

If the JTOPTS keyword QUEUELEN is specified, Tivoli Workload Scheduler for z/OS will start up to the number of operations defined by QUEUELEN. Refer to Customization and Tuning for more information on the QUEUELEN parameter.

Submitting jobs

This is how Tivoli Workload Scheduler for z/OS submits operations on computer workstations:
- The current plan is locked to prevent other tasks or users from updating the operations.
- The best candidate for submission is chosen as described in “Identifying the order of submission” on page 293.
- If the operation has been marked with a NOP, Tivoli Workload Scheduler for z/OS immediately sets the operation to complete.
- Tivoli Workload Scheduler for z/OS attempts to find the job in the JS file. The job is found here only if a dialog user has updated the job statements, or if this operation occurrence has been previously submitted.

Note: To prevent the JS file from becoming full, Tivoli Workload Scheduler for z/OS maintains the JCL records on the JS data set until the next occurrence of the same application is set to Complete. For details, see the Diagnosis Guide and Reference.
- If the job is not found on the JS file, Tivoli Workload Scheduler for z/OS attempts to find it (the member name must be the operation name) in the EQQJBLIB partitioned data set concatenation, first calling the EQQUX002 exit if it is loaded.
- If no job can be found, the operation is given an extended status E to indicate the error, and a message is written to the Tivoli Workload Scheduler for z/OS message log.
- If the job is found, either in the JS file or in EQQJBLIB or returned by the EQQUX002 exit, variable substitution is performed, if required, and the job submit user exit (EQQUX001) is called.
- A copy of the job is written to the JS file. The job and workstation destination are queued to the data router. The operation status is set to S and extended status U is assigned to indicate that submission is in progress.
- If there are more operations eligible to start, Tivoli Workload Scheduler for z/OS will move on to the next best candidate. Up to 5 operations can be started.
Submitting jobs

- The current plan lock is released.
- The destination is resolved. It can be a submit/release data set, an XCF link, an NCF link, a TCP/IP link, or blank. If the destination is blank, the job is directed to the system the controller is started on.
- (z/OS only.) When the JCL arrives at the destination, the job is submitted to JES via the EQQBRDS ddname. The EQQBRDS ddname is used to allocate a JES internal reader.
- The extended status of the operation is set to blank.
- (z/OS only.) When Tivoli Workload Scheduler for z/OS learns from JES that the job has been successfully loaded onto the internal reader, the extended status is set to Q.
- Once the job actually starts to execute, the extended status is set to S. For z/OS jobs, this happens when the initiator is available.
- If a job is routed to another NJE node for execution and in this process the related JES reader time changes by more than one minute, the job is not tracked correctly. In fact, if a job has a ROUTE EXEC statement to a specific NJE node, it is sent to the JES PLEX and can be run on any member of the JES MAS complex. To allow Tivoli Workload Scheduler for z/OS to track the job, define a tracker on every member of the JES multi-access spool complex at the destination NJE node. Instead of using ROUTE EXEC, define a CPU workstation whose destination is one of the trackers on the destination NJE node, and schedule the work on that workstation. In this way, Tivoli Workload Scheduler for z/OS, not NJE, sends the job directly to the node where it is to be run. The results is that the job goes through JES input/conversion processing only once, and the JES reader timestamp does not change.

If the JTOPTS keyword QUEUELEN is specified, Tivoli Workload Scheduler for z/OS will start up to the number of operations defined by QUEUELEN. Refer to Customization and Tuning for information on the QUEUELEN parameter.

Submitting jobs or started task on a virtual workstation

The process is the same as that described in the previous sections for jobs and started task, with the difference that the scheduler selects the submission destination among the virtual workstation destinations, according to a round-robin scheduling logic.

The availability checks related to open intervals, parallel servers, and fixed resources are done at virtual destination level.

The alternate workstation concept is not applicable to virtual workstations.
Chapter 14. Overview of job tracking on z/OS

This chapter describes briefly how Tivoli Workload Scheduler for z/OS tracks work on the host. If an operation is not tracked, there is normally a problem with the Tivoli Workload Scheduler for z/OS configuration or the initialization statements.

Tivoli Workload Scheduler for z/OS uses event records to track the workload. Most event records are generated automatically as a job progresses through the system, but they can also be generated manually. You can use Tivoli Workload Scheduler for z/OS TSO commands and Tivoli Workload Scheduler for z/OS subroutines to report event information to Tivoli Workload Scheduler for z/OS, both manually and automatically. You can:

- Request a backup of the current plan or JCL repository using the BACKUP TSO command or EQQUSINB subroutine.
- Make a special resource available or unavailable using the SRSTAT TSO command or EQQUSINS subroutine.
- Change the status of an operation using the OPSTAT TSO command or EQQUSINT subroutine.
- Update the user data field of an operation in the current plan using the OPINFO TSO command or EQQUSINO subroutine.
- Change the status of a workstation in the current plan using the WSSTAT TSO command or EQQUSINW subroutine.
- Request any of these changes using the EQQUSIN subroutine or the application programming interface (API) for Tivoli Workload Scheduler for z/OS.

This is how the event information is passed to Tivoli Workload Scheduler for z/OS:

- For z/OS jobs and started tasks, z/OS calls the SMF and JES exits at certain stages in the life of a job. For example, the job initiation exit IEFUJI is called whenever a job starts. Tivoli Workload Scheduler for z/OS code in the exit collects information about the event and passes it to the event-creation module, EQQSSCMJ, through the z/OS subsystem interface. Relevant information for a job that has started would include the job name and number, and its starting date and time.
- All Tivoli Workload Scheduler for z/OS address spaces start a submit task, which initiates work for the workstation destination that represents the system that the controller or tracker is started on. When the submit task starts work, it uses EQQSSCMJ to create initialization events, depending on the type of work to be started. Different initialization events are created for batch jobs, started tasks, write-to-operator (WTO) operations. Submit-checkpointing events are created for all work that Tivoli Workload Scheduler for z/OS submits, except operations that are routed to a user-defined destination.
- If you generate an event using the BACKUP, OPINFO, OPSTAT, WSSTAT, or SRSTAT commands from the TSO environment or from the EQQEVPGM batch program, the parameters are checked and then passed to the event-generation module, EQQSSCMJ, through the z/OS subsystem interface.
- You can generate an event if you write a program that passes parameters to the EQQUSIN, EQQUSINB, EQQUSINS, EQQUSINO, EQQUSINW, or EQQUSINT
Overview of job tracking on z/OS

subroutines. The subroutine checks the parameters and passes them to the event-generation module, EQQSSCMJ, through the z/OS subsystem interface.

When the events have been created, this is how they are passed to the controller:

1. EQQSSCMJ uses the information to build an event record and places the record into the event writer queue in ECSA. This processing can take place as soon as the z/OS subsystem interface is started. It is not necessary for Tivoli Workload Scheduler for z/OS itself to be active. If Tivoli Workload Scheduler for z/OS is not active (in particular, if the event writer subtask is not active), event records stay in the event writer queue until the event writer starts and processes them.

Event records are generated for all z/OS jobs and started tasks, even though they might not be relevant to a particular Tivoli Workload Scheduler for z/OS address space. It is not possible for the programs creating the event records to determine if a particular job is relevant to a particular Tivoli Workload Scheduler for z/OS address space. The event creation programs reside in z/OS common storage and do not belong to, or have access to, the data or resources of any Tivoli Workload Scheduler for z/OS address space that might be running on the same system or some other system.

2. The event-writer subtask of the tracker reads event records from the event writer queue and writes them to an event data set. They can be filtered at this stage by the EQQUX004 exit, if this is required for performance reasons.

3. Events are transmitted to the controller by an event reader function, which is either a function of the event writer, or is a separate event reader task. An event writer can use an XCF, NCF or TCP/IP connection to transmit events to the controller. Where a separate event reader is used, the event reader can be active at the controller, or at a tracker that is connected to the controller through XCF or NCF. If the event writer is active but there is no connection to the controller, or if the event reader is not active, events stay in the event data set until the required function is available.

4. The event manager subtask that is started at the controller processes the events, and Tivoli Workload Scheduler for z/OS takes the relevant action.

Events are never lost, if these conditions are satisfied:

- The event writer queue in ECSA is large enough to hold all the event records that might be created while the event writer is not active.
- The event data set is large enough to hold all the event records that might be created while a connection to the controller is lost, or an event reader is not active.

How to make sure that events are not lost

Use one of these methods to ensure that events are not missed between the time Tivoli Workload Scheduler for z/OS is taken down and JES (JES2 commands are given in the example) is taken down:

Method 1

1. Remove the system being stopped from the JES2 MAS by placing it into independent mode (issue $T MEM,IND=Y).
2. Allow the jobs currently running on this system to complete.
3. Stop the tracker (P 0PCx).
4. Stop JES.
5. Re-IPL.
How to make sure that events are not lost

   6. Restart JES.
   7. Restart the tracker.
   8. Resume normal work (issue $T MEM,IND=N).

Method 2
   1. Abend JES2 ($P JES2, ABEND).
   2. Stop the tracker ($P OPCx).
   3. Re-IPL.
   4. Restart JES. This will be a hot start.
   5. Restart the tracker.

Method 3
   1. Take the tracker down ($P OPCx).
   2. Bring it up again under the master scheduler ($P OPCx, SUB=MSTR).
      Remember that a tracker cannot use JES services (SYSOUT data sets,
      JCC) if it runs under the master scheduler.
   3. Bring down JES.
   4. Take the tracker down ($P OPCx).

Time zones

Tivoli Workload Scheduler for z/OS can support several z/OS systems. The tracker collects information about activity on the z/OS system it supports, such as when a job has started, or when a print has been processed. This information is recorded in an event record and sent to the controlling system. Each event contains a time stamp, which records the date and time that the event record was created.

The systems that Tivoli Workload Scheduler for z/OS supports might be in different time zones. The event record contains a field that specifies the difference between the local time of the system that collected the record and Greenwich mean time (GMT). This time difference is automatically calculated and placed in event records. If you intend to change the local clock GMT offset by using the z/OS SET CLOCK command, all Tivoli Workload Scheduler for z/OS systems running on the affected z/OS should be shut down and then restarted after the SET CLOCK command has been issued. The GMT offset is stored when the subsystem is started.

To avoid shutting down and restarting the Tivoli Workload Scheduler for z/OS systems for daylight savings time, set record 90 in member SMFPRMxx of the SYS1.PARMLIB. At daylight savings time, the time is updated automatically. See “Updating SMF Parameters” in the Planning and Installation for more information.

The controller processes event records from all the systems in your Tivoli Workload Scheduler for z/OS complex. Because all event records contain a local time difference field, the controller can convert time stamps to the local time of the controlling system. This is necessary because all time values stored in Tivoli Workload Scheduler for z/OS databases and data sets are expressed in the local time of the controlling system.

Time values appearing printed reports are also expressed in the local time of the Tivoli Workload Scheduler for z/OS controlling processor.
Chapter 15. Planning for recovery and restart

This chapter describes how to plan for a job that fails. You can restart it automatically, or examine the job log first and then restart the job from the MCP dialog.

The job log is the system data written to the SYSOUT class defined by the job message class. It is used to build information needed for restart and cleanup.

Tivoli Workload Scheduler for z/OS has a number of tools to help you restart jobs:

**Job completion checker (JCC)**
This function reads job output and can set an error code. This is useful if you cannot tell from return or abend codes alone whether or how a job must be restarted: you sometimes need to check for specific messages.

**Tracker platforms supported:** z/OS

**Job log retrieval**
This function can fetch the job log for a job (even one that has not failed) so that you can browse it.

**Tracker platforms supported:** All

**Restart and cleanup**
This function checks whether a job is restartable, and tailors the JCL, if required, to run again from one named step to another, or to the end of the job. Also, this function can back out changes to catalogs and can delete data sets that are created by the failed job. The following is an example of JCL for jobs failing on reruns:

```plaintext
//OUTDS DD DSN=NEW.DATA.SET,DISP=(NEW,CATLG,CATLG),...
```

**Tracker platforms supported:** z/OS

**Automatic recovery**
The `//*%OPC RECOVER` job statement controls automatic recovery. Parameters on this statement specify whether Tivoli Workload Scheduler for z/OS should start other occurrences, delete steps, and so on. If the data set cleanup is required, this is invoked before the rerun when immediate cleanup is specified. Otherwise, automatic recovery is not done.

**Tracker platforms supported:** All (restart and cleanup functions are available only on z/OS systems)

**History function**
This optional function lets you rerun operations that have completed and been deleted from the current plan. If this function is active, completed operations are copied to the DB2 database when the current plan is extended, and the details (including job log and job statements if available) are kept there for a period of time that you specify in initialization statements.

**RECOVERY statement**
This statement in the SCRPTLIB defines the options to run the IBM Tivoli Workload Scheduler recovery for jobs on distributed agents. The recovery actions can be followed by one of the recovery options (the OPTION parameter), stop, continue, or rerun. The RECOVERY statement is ignored if it is used with a job that runs a centralized script.
What you need to run the functions

Some of the functions need to be turned on before you can use them. This section describes the parameters that you need, and where to find more information.

What you need for the job completion checker

1. Specify JCCTASK(YES) on the OPCOPTS statement for the tracker where the job will run.
2. Code and assemble message tables that direct the job completion checker, as described in Customization and Tuning.
3. Specify options on the JCCOPTS initialization statement.

Where to find more information

On setting error codes
See “Setting error codes using completion codes” on page 309.

On the JCCOPTS and OPCOPTS statements
Refer to Customization and Tuning.

On coding message tables
Refer to Customization and Tuning.

What you need for job log retrieval

Job log retrieval for trackers on z/OS is different than retrieval for trackers on other systems. The Tivoli Workload Scheduler for z/OS data store keeps the job logs for z/OS. For distributed agents, the job log retrieval is available by request from the controller. The following subsections describe the considerations for both z/OS and non-z/OS trackers.

z/OS job logs

For z/OS job logs, the retrieval can either occur on demand or on error. That is, the job log can be requested from the panels manually, or it can be automatically retrieved when a job ends in error. Refer to the RCLOPTS and HTTPOPTS initialization statements in Customization and Tuning.

Install and customize the data store with the STOUNSD parameter set to YES. Also, specify restart and cleanup on the controller with the RCLEANUP parameter. All data stores must have the same destination (set by the SYSDEST parameter), and this must be equal to the controller RCLOPTS DSTDEST parameter.

Note: If you are running on z/OS system Version 1 Release 2 or higher, JOBLG retrieval does not function properly when the SPIN option for the JES job output is used. Therefore, use the SPIN parameter to prevent MVS JOBLG SPIN. See Customization and Tuning for details.

Job logs on systems running distributed agents

For job logs on systems running distributed agents, job log retrieval is as follows:
• The retrieval is always delayed.
• You can manually request the job log.

Where to find more information

On browsing the job log
Refer to “Selecting a history operation” on page 580.
Where to find more information

On automatic job log retrieval and customizing the HTTPOPTS and RCLOPTS statements
Refer to Customization and Tuning. For z-centric and dynamic workstations, see the JOBLOGRETRIEVAL keyword in the HTTPOPTS statement. For MVS jobs, see the JOBLOGRETRIEVAL and RESTARTINFORETRIEVAL keywords in the RCLOPTS statement.

On browsing the job log
Refer to "Selecting a history operation" on page 580.

On customizing the data store with RCLOPTS
Refer to Customization and Tuning.

What you need for restart and cleanup
This function is a combination of restart and data set cleanup functions. They can be completed together or separately depending on the operation definition and the use of the panels. To set up restart and data set cleanup, perform the following steps:

1. Install and customize the data store on each z/OS image where jobs will run. Specify the FLOPTS parameter values for the controller.
2. Specify RCLEANUP (YES) on both the OPCOPTS controller statement and the BATCHOPT daily plan batch statement.
3. Specify the RCLOPTS DSTDEST value equal to the data store SYSDEST parameter on the DSTOPTS statement.
4. Specify the MSGLEVEL (1,1), if this is not the default for the jobs.
5. If you need data set cleanup before a job rerun, specify the cleanup type to any type but NONE for the operation.
6. Make the EQQC CLEAN procedure available in all systems where the jobs can run.

   Note: If you are running on z/OS system Version 1 Release 2 or higher, restart and cleanup does not function properly when the SPIN option for the JES job output is used. Therefore, use the SPIN parameter to prevent MVS JOBLOG SPIN. See Customization and Tuning for details.
7. Customize the job-submit exit (EQQUX001) as suggested in the provided sample if you want the same user to own both the stand alone cleanup job and the original job. See "Selecting cleanup options" on page 331. Immediate option, for more information about the stand alone cleanup job. Refer to Customization and Tuning, Chapter 5, for information about the job-submit exit.

Where to find more information
On the DSTOPTS, OPCOPTS, and RCLOPTS statements
Refer to Customization and Tuning.

On step restart and cleanup
See Chapter 17, “Restart and cleanup,” on page 313.

On specifying cleanup for an operation
See "Options that apply to jobs and started tasks” on page 151.

On using the Modify Current Plan panel
Refer to Chapter 26, “Updating the current plan,” on page 547.
What you need for automatic recovery

What you need for automatic recovery

1. Make sure that automatic recovery is turned on (use the Service Functions menu, which is option 9 from the main menu).
2. Check the parameters on the controller AROPTS statement.
3. Specify RECOVERY(YES) on the controller OPCOPTS statement.
4. If you want automatic recovery to invoke the cleanup function before rerunning a job, see also “What you need for restart and cleanup” on page 305.

Note: When cleanup is required, you must set the cleanup type to IMMEDIATE for automatic recovery to run.
5. To use automatic recovery for a job running on a fault-tolerant agent, you must define it to use a centralized script. For information about defining a centralized script and about the RECOVERY statement, see the Scheduling End-to-end with Fault Tolerance Capabilities manual.

Where to find more information

On the OPCOPTS and AROPTS statements
Refer to Customization and Tuning.

On the automatic recovery function, and more examples

On the syntax of the //**OPC RECOVER statement
See “The automatic-recovery-control statement” on page 351.

On the Service Functions menu.

What you need for the history function

Specify OPERHISTORY(YES) and the DB2SYSTEM keyword on the controller BATCHOPT and OPCOPTS statements.

Where to find more information

On creating a DB2 database
Refer to Planning and Installation.

On the BATCHOPT and OPCOPTS statements
Refer to Customization and Tuning.

On using the history function
Refer to “Rerunning operations in the history database” on page 578.

What you need for recovery on distributed agents

This function applies only to operations on distributed agents using non-centralized scripts. Perform the following tasks for job recovery on distributed agents:
1. Activate the end-to-end scheduling with fault tolerance capabilities feature.
2. Use the RECOVERY statement when you define the operation in the SCRPTLIB.
3. View data about recovery, respond to the prompt, and browse the JOB log of the recovery job in one of the following ways:
   • Use the command line on the fault-tolerant workstation.
What you need for recovery on distributed agents

- Use the RI row command on the List Operations panel (EQQMOPRL) to access the EQREINP panel.

Where to find more information

On activating the end-to-end scheduling with fault tolerance capabilities feature
Refer to Planning and Installation.
Refer to Customization and Tuning.

On the RECOVERY statement
Refer to Customization and Tuning.
Where to find more information
Chapter 16. Setting error codes

This chapter describes how Tivoli Workload Scheduler for z/OS sets error codes to describe the outcome of running a computer workstation operation. You can also set error codes manually from the Ready List or from the MCP panel.

When a computer workstation operation (either a job or a started task) ends, Tivoli Workload Scheduler for z/OS determines the conditions under which it ended by inspecting the error code of the operation. Tivoli Workload Scheduler for z/OS either uses the return code from the last executed step of the job or the highest return code from any step depending on the value you specify for the EWTROPTS keyword RETCODE. Refer to Customization and Tuning for more information about EWTROPTS keywords.

You can specify whether you want certain errors for all jobs to be considered acceptable. You can also specify that particular errors for specific jobs are to be considered acceptable. Unless informed otherwise, Tivoli Workload Scheduler for z/OS will, upon identifying an error, set the operation status to E (ended-in-error).

Note: With PQ87904 APAR, when the error in a job is determined by the EQQCLEAN step, that is a step inserted into the job by the Restart and Cleanup function, Tivoli Workload Scheduler for z/OS sets the operation status to E, overriding any other specification.

How error codes are set for jobs on z/OS

Error codes can be set in these ways:
• Automatically, based either on the completion codes of the job or started-task step or on internal error detection.
• By the job completion checker. See “Setting error codes using the job completion checker” on page 310.
• Manually from the Ready List panel or from the MCP panel.
• Using the EQUSIN subroutine. Refer to Customization and Tuning.
• Using the program interface. Refer to Programming Interfaces.
• Using the OPSTAT command in batch or TSO. See Appendix A, “TSO commands,” on page 647.

In most cases, Tivoli Workload Scheduler for z/OS sets error codes using the z/OS user and system completion codes. In some special cases (for example, when there has been a JCL error), Tivoli Workload Scheduler for z/OS sets a special error code to enable it to react correctly to the situation.

Setting error codes using completion codes

Tivoli Workload Scheduler for z/OS sets error codes using completion codes when they are available. This is how Tivoli Workload Scheduler for z/OS derives the error code from the completion code:
• If the job or started task did not abend, the completion code is converted to a 4-digit number. The z/OS completion code 8, for example, is converted to the Tivoli Workload Scheduler for z/OS error code 0008. This also depends on the values specified on EWTROPTS.
Setting error codes using completion codes

- If the job or started task failed with a system completion code, the abend code (from the step-end SMF record) is set as the error code. System completion code 0C4, for example, becomes error code S0C4.

- If the job or started task failed with a user abend code, the code (from the step-end SMF record) is converted to a character string of the format Uxxx, where xxx is three hexadecimal digits. User abend 2750, for example, is converted to error code UABE. That is, the decimal value of the abend code, as displayed in the job log, is converted to hexadecimal representation.

- In certain cases, Tivoli Workload Scheduler for z/OS does not use the completion code of a job or started task to set the error code. Instead, it sets one of its own error codes. Refer to Appendix E, “Status, error, and reason codes,” on page 743 for a list of error codes.

Notes:

1. If more than one step in a job or started task has abended, the operation error code is created from the abend code of the first step that failed.
   An exception to this rule is when you set RETCODE(LAST). In this case:
   - If all the steps fail, the operation error code is created from the abend code of the last step that failed.
   - If all the steps fail, except for the last one, which flushed, the operation error code is created from the abend code of the first step that failed.

2. Tivoli Workload Scheduler for z/OS converts only the three right-most hexadecimal digits. Therefore, the highest possible return code is 4095 (X'FFF'). If you pass a return code that is greater than 4095, an invalid return code could be set or invalid return code testing performed.

Setting error codes using the job completion checker

Sometimes the success or failure of a job or started-task operation cannot be determined from step completion codes alone. For example, a job might fail because a certain data set is not available, but this particular failure might be acceptable for this operation. That is, you would not want the operation representing the job to be marked as ended-in-error. However, if the job fails because it ran out of processor time, this failure might not be acceptable. In this case, you would want the operation representing the job to be marked as ended-in-error. The job completion checker (JCC) function can be used to distinguish between operation failures of this type.

Just as you might look at the SYSOUT output of a job to find out whether a program issued a certain message, the JCC can do this automatically for you. The JCC scans the SYSOUT data set of a job or started task and sets an error code for the operation, depending on the results of the scan.

The JCC can set an error code for the operation. But the step completion or abend code is still available and can be used in RECOVER statements—the JCC only sets the job error code.

The JCC is a tracker function and is, therefore, independent of the contents of the current plan. The JCC processes all jobs and started tasks for which a termination (type 3P) event has been created in the event data set, regardless of whether the jobs or started tasks are defined in the current plan.

Note: Acceptable nonzero return codes should be defined either using the operation's HIGHRC job option or by defining statements on the NOERROR initialization statement. It is not advisable to use the JCC to reset non-zero
return codes that you consider acceptable. For more information on NOERROR and about JCC, refer to *Customization and Tuning*.

### How error codes are set for jobs on distributed agents

Error codes can be set automatically, based on the completion codes. Because IBM Tivoli Workload Scheduler return codes, with a completion code of ABND, vary from -2147483647 to 2147483647 and IBM Tivoli Workload Scheduler for z/OS return codes vary from –999 to 9999, return codes from the distributed environment that exceed four digits are truncated. For example, a return code of 123456785 becomes 1234 for IBM Tivoli Workload Scheduler for z/OS.

### Using error codes to set operations to ended-in-error

Depending on the application, you might consider that not all error codes represent true operation errors. That is, you do not want the operation representing the job or started task to be marked as ended-in-error for certain error codes. You can tell Tivoli Workload Scheduler for z/OS to ignore certain error codes, using the NOERROR statement. You specify a list of error codes that must not cause the operation to be marked as ended-in-error. If the operation ends with an error code that matches a NOERROR condition, the operation is treated as having ended normally.

A specific extended status code shows if an operation, with completed (C) as current status, ended with an error code matching a NOERROR entry.

An error code specified by the NOERROR initialization statement can apply to all computer workstation operations, or it can be specified to apply to only a subset of such operations or steps within those operations. If the completion code of the last step in the operation is used to set the error code, this is regarded as a job error code. This means that the `stepname` and `procstepname` parameters of the NOERROR statement cannot be used to establish a match with the specified code because Tivoli Workload Scheduler for z/OS does not have the relevant step information.

If the error code is numeric, after checking the NOERROR list, Tivoli Workload Scheduler for z/OS checks the error code against the HIGHRC keyword of the JTOPTS initialization statement. If HIGHRC has been specified at operation level, Tivoli Workload Scheduler for z/OS uses this value instead of the installation-defined HIGHRC. The operation is considered to have ended-in-error if the error code has a higher numeric value than the HIGHRC value specified. If the error code is numerically lower than or equal to the HIGHRC value, the operation is considered to have ended normally. In this case, the operation is marked as C, complete.

To use the NOERROR keyword for specific job or started-task steps, the event writer options, as specified in the EWTROPTS initialization statement, must be set as follows:
- The STEPEVENTS keyword must specify either ALL or NZERO.
- The RETCODE keyword must specify HIGHEST.

For more information about NOERROR, refer to Appendix D, “Supported z/OS commands,” on page 733.
Resetting operations based on error codes

Tivoli Workload Scheduler for z/OS supports an error-reset error code list. You define this list using the ERRRES keyword of the JTOPTS initialization statement.

If an error code for a job or started task is in this list, Tivoli Workload Scheduler for z/OS automatically places the job or started task on the ready list of the operation computer workstation with status A (arriving) and extended status R (error, automatically reset). Tivoli Workload Scheduler for z/OS does not restart the operation automatically.

Determining the success or failure of a job

This summarizes how Tivoli Workload Scheduler for z/OS determines the next status of an operation that ends:

1. Tivoli Workload Scheduler for z/OS creates a job-end event with the highest or last return code, depending on the RETCODE keyword of the EWTOPTS statement.
2. If the job-completion checker (JCC) is active, it gets the event. The JCC can set a new value for the return code. After JCC processing, the event passes to the controller.
3. If the return code is 0, Tivoli Workload Scheduler for z/OS sets the operation status to C. Otherwise, it continues checking.
4. If the operation definition specifies no error tracking, Tivoli Workload Scheduler for z/OS sets the operation status to C. Otherwise, it continues checking.
5. If the return code matches a NOERROR entry (a NOERROR statement or the NOERROR keyword of the JTOPTS statement), Tivoli Workload Scheduler for z/OS sets the operation status to C. Otherwise, it continues checking.
6. If the return code is less than or equal to HIGHRC (the value in the operation definition or the value on the JTOPTS statement), Tivoli Workload Scheduler for z/OS sets the operation status to C. Otherwise, it continues checking.
7. If the return code matches an entry on the ERRRES keyword of the JTOPTS statement, Tivoli Workload Scheduler for z/OS sets the operation status to A, extended status R. Otherwise, it sets the operation status to E, and recovery processing can now occur.
Chapter 17. Restart and cleanup

This chapter describes the restart and cleanup of jobs and started tasks running on OS/390® systems and z/OS environments. Unless otherwise stated, the description applies to both jobs and started tasks.

Restart and cleanup are basically two tasks:
- Restarting an operation at the job-level or the step-level
- Cleaning up the associated data sets

When you restart an operation at step-level, you select a step range: the starting step, the ending step, and any steps to exclude from the re-run. When you perform a cleanup, you take previously allocated data sets and either catalog, uncatalog, or delete them. These tasks can be completed separately or together depending on your preference and Data Processing needs. For example, you can specify the step restart and cleanup of an operation, which are completed at job run time. In other situations, you can run the data set cleanup process separately, before the job reruns. In the second case, the step restart only begins after the cleanup completes.

The OPERATION RESTART AND CLEANUP panel (see [Figure 133]) is displayed when you request the RC row command for any operation, which ran at least once (this does not include operations which were retrieved from the DB2 history file as Restart and Cleanup cannot be done in this case - message EQQM600E will be issued), from any of the following operation lists in the MODIFY CURRENT PLAN panel:
- Ended-in-error operation list
- Operation list
- Operation displayed when you request to rerun an occurrence

![Figure 133. EQQRCLSE - Operation restart and cleanup](image-url)
### Restart and cleanup

The restart and cleanup options are the following:

- Step restart
- Job restart
- Start cleanup
- Start cleanup with AR
- Display cleanup

The restart function is closely connected to the cleanup and automatic recovery functions. See [“Data set cleanup” on page 331](#) and Chapter 15, “Planning for recovery and restart,” on page 303 for an overview of these functions and a description of how to activate them.

Note that in EQQRCLSE you can also change the proposed value for the following field:

- **EDIT JCL**
  - This field is applicable only to the STEP RESTART and JOB RESTART options. DEFAULT is the last value used in the ISPF dialog. To edit the JCL that will be resubmitted via the SR or JR options, you must set this field to YES.

To execute the Restart and cleanup function, the EQQCLEAN pre-step is added as the first step in the submitted JCL. This happens also when the cleanup actions are not triggered from ISPF but are started automatically by the Controller.

Since the EQQCLEAN step has to be added before all other existing steps, when an INCLUDE statement is present before the first step in a JCL, the EQQCLEAN step is added before that INCLUDE. Therefore, you should avoid specifying the JOBLIB DD in an INCLUDE or you should list this kind of INCLUDE (containing JOBLIB or other JCL statements that need to be placed before the EXEC) using the RCLOPTS SKIPINCLUDE keyword.

**Note:** The usage of this parameter does not solve the JCL errors caused by INCLUDEs that contain both JOBLIB and EXEC statements. They must be split into two separate INCLUDEs: one for the JOBLIB and one for the EXECs.

If the INCLUDE that contains the JOBLIB statement is nested within other INCLUDEs, you must add the most external INCLUDE to the SKIPINCLUDE list, as this is the only one visible to the JCL tailoring process.

This problem does not occur with expanded JCL because it does not contain INCLUDE statements.

**Notes:**

1. With PQ87904 APAR, when the EQQCLEAN step fails with RC>=8, then all subsequent steps in the job are forced to FLUSH. This causes the operation to be marked as Ended-in-error with the ‘Status of operation on workstation’ set to CLNP, whichever completion checking logic has been implemented.
2. Same restrictions apply as for changing the operation status to ready by selecting option 6 (GENERAL) from the MODIFYING AN OPERATION IN THE CURRENT PLAN panel. In particular, a step or job restart request might imply a request to change to ready the status of an operation with conditional successors already started, completed, suppressed by condition, or ended in error. In this case the scheduler issues message EQQM208E. Only when rerunning an occurrence you might indirectly obtain this kind of change.
Restarting an operation at step or job level

This section explains how to restart an operation associated to jobs and started tasks. It also explains how to select the steps that will be included in the restarted job.

To perform a new rerun of a job, including the needed cleanup actions at job run time, select the JOB RESTART option.

To perform a step restart, including the necessary cleanup actions at job run time, select the STEP RESTART option. This option differs from JOB RESTART in that:
- The restartability of steps is checked according to information available from previous runs (see "Steps that are not restartable" on page 322 for details).
- Return code simulation is executed to restart from a specific step, that is, Tivoli Workload Scheduler for z/OS simulates how the preceding steps actually ended. The simulation works in this way:
  - Ending the steps with RC=nn, where nn is the return code obtained in their last effective run.
  - Flushing the steps, if they were never run or abended.
- You can override the suggested values in the panel and force a different simulation. The reason for running a step restart in this way is to support the IF/THEN/ELSE statements.
- If needed, GDG resolution is executed when normal JCL is used (when expanded JCL is used the GDG resolution is already applied to the JCL, see "GDG resolution" on page 325 for details)

With both JOB RESTART and STEP RESTART option, two types of JCL can be used for job rerun:
- Expanded JCL
- Normal JCL

See "JCL used for restart" for details.

JCL used for restart

The Expanded JCL option applies with the Step Restart and Job Restart functions. The option is related to the CP operation record (built from the AD application definition value) and the default is N. You are allowed to change it using MCP (Modifying the Restart and Cleanup options in the CP). The values allowed for this option are N (No) and Y (Yes). If you select N, you are establishing that the JCL that will be submitted through SR or JR is the normal JCL taken from the JS VSAM files. If you select Y, you are establishing that the JCL is expanded JCL.
- Normal JCL
  Is the normal JCL found in the JS VSAM file or in the customer libraries. It can also be edited using the J row command.
- Expanded JCL
  Is the JCL extracted from the JOBLOG of the last run and is built from the data store. When the data store builds the expanded JCL, the EXEC statements that call for procedures are commented out and the called procedures are inserted in their expanded form. The result is a linear JCL without procedure calls.
EXPANDED JCL

EXPANDED JCL
When you use expanded JCL, the same steps (together with the ones in the called procedures and INCLUDE) are re-executed exactly as they were in the previous run.

GDG resolution is performed by data store while it builds the expanded JCL from the JOBLOG: all the GDG data sets names are replaced with their equivalent expanded form (GDGRoot.GnnnnVnn). For this reason, if you rerun a JCL that contains GDGs more than once, and you do not use the expanded JCL in all the re-executions, the name substitution might not be complete.

For example, suppose you run the following JCL:

```
RUN1
```

```
//GDGSAMPL JOB
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=MYGDG.ROOT(+1),DISP=(NEW,CATLG)
//STEP2 EXEC PGM=MYPGM2
//STEP3 EXEC PGM=MYPGM3
//STEP4 EXEC PGM=IEFBR14
//DD4 DD DSN=MYGDG.ROOT(+1),DISP=OLD
```

and that the job fails in step2, which ends with RC=12:

```
STEP1 RC=0 --> allocated MYGDG.ROOT.G0001V00
STEP2 RC=12
STEP3 RC=FLUSH
STEP4 RC=FLUSH
```

The JOBLOG provides information only about the GDG allocated in STEP1. Data store is able to resolve the GDG and build the expanded JCL correctly. At this point, you fix the error in MYPGM2 and run a Step restart beginning from STEP2 and using normal JCL:

```
RUN 2
```

```
//GDGSAMPL JOB
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=MYGDG.ROOT(+1),DISP=(NEW,CATLG)
//STEP2 EXEC PGM=MYPGM2
//STEP3 EXEC PGM=MYPGM3
//STEP4 EXEC PGM=IEFBR14
//DD4 DD DSN=MYGDG.ROOT(+1),DISP=OLD
```

Now, the job fails in STEP3, which ends with RC=12:

```
STEP1 RC=0 --> simulated step
STEP2 RC=0
STEP3 RC=12
STEP4 RC=FLUSH
```

The JOBLOG contains no information about the GDG because data store was no longer able to resolve it in the expanded JCL (because the previous Step restart was run using normal JCL). You now fix the error in MYPGM3 and run a Step restart beginning from STEP3 and using expanded JCL:

```
RUN 3
```

```
//GDGSAMPL JOB
//STEP1 EXEC PGM=IEFBR14
```

IBM Tivoli Workload Scheduler for z/OS: Managing the Workload
In this way, even when the data store is unable to resolve a GDG, the necessary changes are nevertheless contained in the submitted JCL.

Expanded JCL is not stored in the JS after a Job or Step restart, because it is always built by data store from the last JOBLOG run.

To decide whether or not to use the expanded form for a specific JCL, consider that expanded JCL is built from the JOBLOG by removing all calls to procedures: as a result, all the references to steps, such as COND or IF THEN REFDD, are changed by data store if the following conditions occur:

- Stepname.Procstep is used: linear JCL cannot contain this kind of step because calls to procedures have been removed. The reference to the step and the step will usually be changed to Procstep only. This might not be enough if Procstep is not unique in the JCL (see the next bullet).
- When Stepname is used and the Stepname is not unique in the JCL. For example, if STEP2 in a procedure has a COND=(0,NE,STEP1), and STEP1 is the first step in the procedure and a previous step of the original JCL, the COND is resolved incorrectly by JES if the step name from STEP1 is not made unique. When these conditions exist, data store generates the required unique step name in the format JJJnnnnn (where nnnnn is a number) and adds a comment line just before the changed EXEC statement in the format:

```
//OLDStep=(xxx) OLDProcStep=(yyy) NEWStep=zzz
```

where xxx.yyy is the old step name, and zzz is the new one.

To see which steps have changed name in the expanded JCL use the STEP command on the EQQMERSL panel.

The following limitations also occur when you use expanded JCL:

- The change of proc/step names might affect external products that identify with these values.
- JJJnnnnn step names cannot be used in the JCLs, because this might cause incorrect step name substitution.
- Procedures submitted by Tivoli Workload Scheduler for z/OS must contain the PEND statement.
- Nested external procedures must contain the PEND statement.
- Nested inline procedures cannot be ambiguous. For example, the following nesting is clear:

```
step1 exec pgm=PGM1
step2 exec PRO1
    step1 exec pgm=PGM2
    step2 exec PRO2
        step1 exec pgm=PGM3
        step2 exec PRO23
    step1 exec pgm=PGM4
step3 exec pgm=PGM5
```

On the other hand, the following nesting is ambiguous and might not work properly:
In addition, consecutive inline procedures can cause problems, for example:

To avoid problems, insert a dummy step as follows:

Note: Using expanded JCL does affect how the StepName and ProcStep fields are displayed in the step lists or data set lists shown by the dialog. For example, consider running Restart and Cleanup on the following job:

If you select Expanded JCL=N, you will see the following values for the StepName and ProcStep fields in the step list of EQQMERSL panel:

If you select Expanded JCL=Y, you will see:

This is consistent with the nature of expanded JCL. As mentioned before, an expanded JCL is a linear JCL, in which all the procedure calls are expanded and made ‘flat’. Therefore, if you want Expanded JCL=Y, the step name (which is the procedure invocation step name) will always be blank and the procedure name will always be evaluated with the step name in the JCL procedure (or the job step name, if the step does not invoke a procedure), possibly changed to JJJxxxxxx to univocally identify the step tested by the COND parameter.
NORMAL JCL

When you use normal JCL, you can get procedure changes that were applied after the last job run.

If GDG are present, they are resolved transparently, as the JCL is left unchanged. GDG resolution is accomplished at job run time by the substitution of the GDG data set names with their equivalent expanded form (GDGRoot.GnnnnVnn) in the JES control block, using the previous run information. This is accomplished by the EQQCLEAN pre-step.

Additionally, you can use the EQQUXGDG exit to check the GDG name substitution just before EQQCLEAN executes. The exit lets you exclude a GDG from substitution in the same way you can exclude a data set from cleanup with exit EQQUXCAT.

GDG resolution is performed only within a Step Restart path and not in a Job Restart path. A normal JCL will be stored in the JS library at each job or step restart.

In the JES3 environment, the use of the normal JCL has a limitation when a JCL that includes nested procedures, where the steps that call the procedures are not the last ones inside the procedure, ends with a JCL error leading to NORUN steps. In this case, Restart and Cleanup might produce an incorrect step list. Users should modify the failing JCL by adding the PEND statement in the called cataloged PROCs. For example, a nesting like the following, might not work properly:

```
step1 exec pgm=PGM1
  step2 exec PRO1
  step1 exec pgm=PGM2
  step2 exec PRO2
  step1 exec pgm=PGM3
  step2 exec PRO23
  step1 exec pgm=PGM4
  step3 exec pgm=PGM5
  step3 exec pgm=PGM6
  step3 exec pgm=PGM7
```

Step restart

When you rerun a job, you select the range of the restart. The range of the restart is defined by the first and last step to be included in the rerun. You select the Step Restart option in the OPERATION RESTART AND CLEANUP panel.

Step restart allows you to restart the job or the started task at the step level and performs the appropriate cleanup actions. When you request a step restart, the scheduler shows you which steps are restartable and which is the best step. In any case, you can modify the selection made by the scheduler.

The way in which step restart works is based on the simulation of return codes and on the use of data store. The scheduler adds a pre-step, EQQCLEAN, which performs the simulation from the history of the previous runs. This step also performs the cleanup actions and the needed GDG resolution.

**Note:** Return code simulation, GDG resolution, and cleanup actions are based on the previous run history and used JCL structure. For this reason, changes to the submitted JCL structure (like adding a step, deleting a step, and changing DD names) might cause unexpected results. For example, the EQQCLEAN program uses the list of step names, and the return codes associated with them, that are provided by the scheduler, so that, if a
simulated step no longer exists in the submitted JCL, EQQCLEAN fails with
a message about step mismatching. For the same reason, it is preferable not
to switch from normal JCL to expanded JCL, and vice versa, because the JCL
structure could be different (for example when procedures have been
changed in libraries).

To perform a step restart with the appropriate data set cleanup actions, Tivoli
Workload Scheduler for z/OS needs the JCL extracted from the JOBLOG. This JCL
must contain the expanded procedures and INCLUDEs because GDG relative
numbers must be substituted with the data set names that are actually allocated.
This cannot be done using the JCL OVERRIDE statement if GDGs are inside an
INCLUDE. This kind of extracted JCL is called expanded JCL. The only source that
can build it is the JOBLOG.

Return code simulation allows for support of IF/THEN/ELSE statements as shown
in the following example, where you run the following JCL:

```plaintext
//JOBSAMP JOB (07A2,D07),MSGLEVEL=(1,1)
//S1 EXEC PGM=MYPROG
//S2 EXEC PGM=IEFBR14
//S3 EXEC PGM=IEFBR14
//S4 EXEC PGM=IEFBR15
//SIF IF S1.RC EQ 4 THEN
//S5 EXEC PGM=IEFBR14
//EIF ENDIF
//*
```

In this example, steps S1, S2, and S3 are executed and are ended with RC=4, RC=0,
RC=0; S4 is abended with S806, and S5 is flushed.

If you select a step restart from step S4 (after fixing the error causing the abend),
steps S1, S2, and S3 will be simulated with return code 4, 0, 0, while S4 and S5 will
be executed. Return code simulation guarantees that the //SIF statement check
will be correctly evaluated by JES, allowing execution of S5. You have to select the
step range from the EQQMERSL panel (STEP RESTART SELECTION LIST panel),
as shown in Figure 134 on page 321 where the JCL used is the one of the previous
example.
Figure 134 shows that step S4 is selected as the restart point (with the S row command), and the restarted job will end at step S5 (the end step defaults to the last step). The scope of the restart are steps 4 to 5.

To change the completion code values for the steps that will be simulated, do the following after setting the code value:

- For steps outside the restart range, enter the I or F row command.
- For steps inside the restart range, enter the F row command.

For example, to set code value 0008 for step S4, you must use row command F, while for step S2 you might use row command I.

The ‘Rest’ column shows if a step is restartable or not. Steps S4 and S5 are marked as not restartable because they follow the abended step S3. The logic used to decide this is described in the “Steps that are not restartable” on page 322 section. Usually you cannot override suggested logic (that is, you can start from step S1, S2, and S3, but you cannot start from step S5) unless a specific keyword was specified in the Controller Initial statement file (see the RCLOPTS STEPRESCHEK keyword for details).

The ‘Step Type’ column shows if this is a simulated, normal, or EQQCLEAN step. As this is the first run of the job, all steps are marked as Normal. To display the step name change table, enter the primary command STEP on the EQQMERSL panel. The EQQMERSI panel is displayed:
Return code simulation logic

Return code simulation logic
The Step Restart of a job is performed by return code simulation. The Scheduler obtains a history of the previous runs, the step execution, and the return codes from the Operinfo record. Then it adds a pre-step named EQQCLEAN to the JCL. The list of simulated steps and return codes serves as input for EQQCLEAN.

At run time, EQQCLEAN checks the JES control blocks and alters them to force the simulated return codes so that IF THEN ELSE and COND can follow the correct logic.

Abended steps are simulated as flushed for the following reasons:
- IF THEN ELSE or COND related to abended steps are supposed to be run immediately at abend time. The flushed simulation avoids the duplicate execution of these steps.
- If the abended steps were not simulated as flushed, JES would flush all the steps following the abended steps again.

Steps that are not restartable
Cataloging, re-cataloging, and uncataloging operations cannot, by themselves, hinder the capability to restart, because it is possible to use EQQCLEAN. However, there are some cases where a step is not restartable and the logic applied by Tivoli Workload Scheduler for z/OS is the following:
- The step must be re-executable (see the “Re-executing steps” on page 323 section)
- The step must not satisfy any of the following conditions:
  - The step follows the abended step.
  - The step includes a DDNAME that is listed in the parameter DDNOREST (in the RCLOPTS initialization statement).
  - The step includes a DDNAME that is listed in the parameter DDNEVER (in the RCLOPTS initialization statement). In this case the preceding steps are also not restartable.
  - The step is a cleanup step.
Steps that are not restartable

- The step is flushed or not run, and the step is not simulated. The only exception is when the step is the first to be flushed in the JCL, all the following steps are flushed, and the job did not abend.
- The data set is not available, and the disposition is different from NEW.
- The data set is available, but all the following conditions exist:
  - The disposition type is OLD or SHR.
  - The normal disposition is different from UNCT.
  - The data set has the disposition NEW before this step (the data set is allocated by this JCL).
  - The data set has been cataloged at the end of the previous run and a CAT action is performed in one of the steps that follow.
- The step refers to a data set with DISP=MOD, unless the step never ran in the previous job runs (flushed or NORUN).
- Restarting the job from this step entails the running of a step that cannot be rerun.

Data set availability
The scheduler determines the availability of a data set by using information from the previous job runs. In some cases, the scheduler is not able to determine availability where there have been no actions against the data set in the previous runs. Thus, the scheduler establishes availability as follows:

- The data set is available and cataloged if there are no JCL statements that allocate it in the job.
- The data set is not available if there are JCL statements that allocate it in the job.

For example, a step refers to an existing data set with the disposition SHR. This step has never run, thus the data set is available.

Re-executing steps
This section refers only to a Step Restart; it does not apply to a Job Restart.

The idea of a re-executable step is different from the one of a restartable step, as it can be considered a subset of a restartable step:

- A restartable step is always a re-executable step.
- A re-executable step might be restartable or not.

A step can be re-executed if it does not refer to any data sets or if it includes a DDNAME that is listed in the parameter DDALWAYS (in the RCLOPTS initialization statement). Otherwise, if the step does refer to a data set, it can be re-executed if the data set meets one of the three following conditions:

- The disposition type is NEW.
- The disposition type is MOD, and the data set is allocated before running the step, unless the step never ran in the previous job runs (flushed or NORUN).
- The disposition type is OLD or SHR, and the data set is either of the following:
  - Allocated before running this step.
  - Available and has one of the following characteristics:
    - The normal disposition is UNCATLG.
    - The data set is not allocated in the JCL before this step.
    - The data set is cataloged before running this step.
    - The data set has been cataloged at the end of the previous run and no catalog action is taken in any of the steps that follow.
Best restart step

Best restart step
The scheduler suggests the best restart step in the job, depending on how the job ended in the previous run. Table 19 summarizes how the scheduler selects the best restart step.

Table 19. Determining the best restart step

<table>
<thead>
<tr>
<th>How the Job Ended</th>
<th>Best Restart Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>The job terminated in error with an abend or JCL error for non-syntactical reasons.</td>
<td>Last restartable step</td>
</tr>
<tr>
<td>There was a sequence of consecutive, flushed steps.</td>
<td>Last restartable step</td>
</tr>
<tr>
<td>The last run was a stand-alone cleanup action performed according to the IMMEDLOGIC(FIRSTSTEP) parameter.</td>
<td>First restartable step</td>
</tr>
<tr>
<td>All other situations.</td>
<td>First restartable step</td>
</tr>
</tbody>
</table>

Example of step restart
Continuing with the JCL example on page 320, you accept the suggested best S3 and enter the GO command to confirm the selection. At this point, having selected in panel EQQRCLSE the option Edit JCL= YES, the edit JCL panel is shown. You fix the error causing the abend (IEFBR15) and confirm the restart by entering first GO, and then Y, in the final confirmation panel.

The job is run, adding another step named EQQCLEAN that will execute the return code simulation for steps S1, S2, and S3. Then, EQQCLEAN also causes the re-execution of step S4 and the execution of step S5.

The following messages are logged to the JES message log:

```
EQQCN001 START CLEANUP AND/OR RET-CODE SIMULATION PROCESS(ES).
EQQCN181 SNUM STEPTIME PROCNAME RC
EQQCN021 002 S1 0004
EQQCN021 003 S2 0000
EQQCN021 004 S3 0000
EQQCN99I CLEANUP AND/OR RET-CODE SIMULATION PROCESS(ES)ENDED.
```

You select the step restart again, and the following panel is displayed:
From the panel you can see that the EQQCLEAN step has been added. Steps S1, S2, and S3 are shown as simulated and steps S4, and S5 as successfully executed.

**GDG resolution**

GDG resolution is the possibility to re-execute a job using exactly the same GDG data set used in the previous run. This is needed in a step restart mainly to avoid JCL errors, but also to use the same GDG data set as if the job had been executed in one run only.

GDG resolution will be performed in different ways depending on whether you are using normal or expanded JCL.

With a normal JCL, GDG resolution is performed by overwriting the GDG names with their expanded form (GDGRoot.GnnnnVnn) in the JES control block, just before the job run.

With expanded JCL, GDG resolution is performed by overwriting the GDG names with their expanded form (GDGRoot.GnnnnVnn) in the expanded JCL as it is produced. See “**JCL used for restart**” on page 315 for more details.

For example, assume that you run the following JCL:

```plaintext
//GDG00001 JOB (9805,SS), 'D-JOB', MSGLEVEL=(1,1)
//STEP0 EXEC PGM=IEFBR14
//STEP1 EXEC PGM=IEFBR14
//DD01 DD DSN=MYGDG.ROOT(+1),UNIT=3390,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),SPACE=(TRK,(1,1)),
// VOL=SER=TWS01,DISP=(NEW,CATLG,DELETE)
//STEP2 EXEC PGM=IEFBR15
//DD21 DD DSN=MYGDG.ROOT(+1),DISP=SHR
```

The resulting JOBLOG will be the following:
**GDG resolution**

JES2 JOB LOG -- SYSTEM ES33 -- NODE

--- MONDAY, 09 JUN 2009 ---

JOB00107 IRR0301 USERID OPCSTC IS ASSIGNED TO THIS JOB.

JOB00107 ICH70001 OPCSTC LAST ACCESS AT 11:25:40 ON MONDAY, JUNE 9,

JOB00107 $HASP373 GDG00001 STARTED - INIT 1 - CLASS A - SYS ES33

JOB00107 IEF403I GDG00001 - STARTED - TIME=11.52.19

---TIMINGS (MINS.)---

<table>
<thead>
<tr>
<th>JOBNAME</th>
<th>STEPNAME</th>
<th>PROCSTEP</th>
<th>RC</th>
<th>EXCP</th>
<th>CPU</th>
<th>SRB</th>
<th>CLOCK</th>
<th>SERV</th>
<th>PG</th>
<th>PAGE</th>
<th>SWAP</th>
<th>SWAPS</th>
<th>STEPNO</th>
</tr>
</thead>
<tbody>
<tr>
<td>GDG0001</td>
<td>STEP0</td>
<td>00 8</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>279</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>GDG0001</td>
<td>STEP1</td>
<td>00 8</td>
<td>00</td>
<td>0</td>
<td>0</td>
<td>287</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

CSV003I REQUESTED MODULE IEFBR15 NOT FOUND

CSV028I ABEND806-04 JOBNAME=GDG00001 STEPNAME=STEP2

IEF450I GDG00001 STEP2 - ABEND=S806 U0000 REASON=00000004

TIME=11.52.19

CSV003I REQUESTED MODULE IEFBR15 NOT FOUND

CSV028I ABEND806-04 JOBNAME=GDG00001 STEPNAME=STEP2

IEA995I SYMPTOM DUMP OUTPUT

SYSTEM COMPLETION CODE=806 REASON CODE=00000004

TIME=11.52.19 SEQ=00394 CPU=00000000 ASID=0016

PSW AT TIME OF ERROR 070C1000 811C020A ILC 2 INTC OD

NO ACTIVE MODULE FOUND

NAME=UNKNOWN

DATA AT PSW 011C0204 - 9FB0181C 0A0D18FB 180C181D

GPR 0-3 0001C008 84086000 00FC566F 00000010

GPR 4-7 000000FF 006E7568 00000004 0000000C

GPR 8-11 811BF750 811C074F 00000000

GPR 12-15 84086000 006D5450 811C018C 00000004

END OF SYMPTOM DUMP

--- JES2 JOB STATISTICS ---

09 JUN 2009 JOB EXECUTION DATE

10 CARDS READ

82 SYSOUT PRINT RECORDS

0 SYSOUT PUNCH RECORDS

5 SYSOUT SPOOL KBYTES

0.00 MINUTES EXECUTION TIME

1 //GDG00001 JOB (9805,SS),'D-JOB',MSGLEVEL=(1,1)

2 //TIVDST00 OUTPUT JESDS=ALL,DEST=TWSFDEST inserted by scheduler

3 //TIVDSTAL OUTPUT JESDS=ALL inserted by scheduler

4 //STEP0 EXEC PGM=IEFBR14

5 //STEP1 EXEC PGM=IEFBR14

6 //DD01 DD DSN=MYGDG.ROOT(+1),UNIT=3390,

   // DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),SPACE=(TRK,(1,1)),

   // VOL=SER=TWS01,DISP=(NEW,CATLG,DELETE)

7 //STEP2 EXEC PGM=IEFBR15

8 //DD21 DD DSN=MYGDG.ROOT(+1),DISP=SHR

ICH70001I OPCSTC LAST ACCESS AT 11:25:40 ON MONDAY, JUNE 9, 2009

IEF142I GDG00001 STEP0 - STEP WAS EXECUTED - COND CODE 0000

IEF373I STEP/STEP0 /START 2009160.1152

IEF374I STEP0 /STOP 2009160.1152 CPU 0MIN 00.00SEC SRB 0MIN 00.00S

IEF375I STEP/STEP0 /START 2009160.1152

IEF236I ALLOC. FOR GDG00001 STEP1

IGD100I 0118 ALLOCATED TO DDNAME DD01 DATACLAS ( )

IEF142I GDG00001 STEP1 - STEP WAS EXECUTED - COND CODE 0000

IEF375I STEP/STEP1 /START 2009160.1152

IEF374I STEP1 /STOP 2009160.1152 CPU 0MIN 00.00SEC SRB 0MIN 00.00S

IEF236I ALLOC. FOR GDG00001 STEP2

IEF375I STEP/STEP2 /START 2009160.1152

IEF374I STEP2 /STOP 2009160.1152 CPU 0MIN 00.00SEC SRB 0MIN 00.00S

IEF236I ALLOC. FOR GDG00001 STEP3

IEF237I 0118 ALLOCATED TO DD21

CSV003I REQUESTED MODULE IEFBR15 NOT FOUND

CSV028I ABEND806-04 JOBNAME=GDG00001 STEPNAME=STEP2

IEA995I SYMPTOM DUMP OUTPUT

---
From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:

From this JOBLOG you can see that STEP0 has been executed, STEP1 has been executed and allocated GDG data set MYGDG.ROOT.G0126V00, while STEP2 abended and used the just allocated GDG data set MYGDG.ROOT.G0126V00 as input.

Now, you decide to fix the error causing the abend and restart from STEP2, re-using the previously allocated GDG MYGDG.ROOT.G0126V00. To obtain this, you can use normal JCL (set Expanded JCL to NO).

When you execute the step restart, the job completes successfully and the JOBLOG is the following:
GDG resolution

2 //TIVDST00 OUTPUT JESDS=ALL,DEST=TWSFDEST inserted by scheduler
3 //TIVDSTAL OUTPUT JESDS=ALL inserted by scheduler
4 //EQQCLEAN EXEC EQQCLEAN,EQQPASS='RMM=N'
5 XEQQQCLEAN PROC
6 XEQQQCLEAN EXEC PGM=EQQCLEAN,REGION=0M,TIME=1440,PARM='&EQQPASS'
7 //STEP1IB DD DISP=SHR,DSN=TWS81.SERVICE.APFLIB1
8 // DD DISP=SHR,DSN=TWS81.SERVICE.APFLIB
9 XEQQQCLNDD DD DUMMY
10 //EQQSIMDD DD *
11 X/EQQSIMDD DD DUMMY
12 //EQQQGDGDD DD *
13 X/EQQGDGDD DD DUMMY
14 //SYSUDUMP DD SYSOUT=*  
15 X/SYSUDUMP DD SYSOUT=*  
16 //SYSUDUMP DD SYSOUT=*  
17 XX PEND
18 //STEP0 EXEC PGM=IEFBR14
19 //STEP1 EXEC PGM=IEFBR14
20 //DD01 DD DSN=MYGDG.ROOT(+1),UNIT=3390,  
 // DCB=(RECFM=FB,LRECL=80,BLKSIZE=800),SPACE=(TRK,(1,1)),  
 // VOL=SER=TWS01,DISP=(NEW,CATLG,DELETE)
21 //STEP2 EXEC PGM=IEFBR14
22 //DD21 DD DSN=MYGDG.ROOT(+1),DISP=SHR

STMT NO. MESSAGE
4 IEC001I PROCEDURE EQQCLEAN WAS EXPANDED USING SYSTEM LIBRARY USER.PRO
ICH70001I OPCSTC LAST ACCESS AT 12:14:32 ON MONDAY, JUNE 9, 2009
IEF236I ALLOC. FOR GDG00001 EQQCLEAN EQQCLEAN
IEF237I 0118 ALLOCATED TO STEPLIB
IEF237I 0121 ALLOCATED TO
IEF237I DMY ALLOCATED TO EQQCLNDD
IEF237I JES2 ALLOCATED TO EQQSIMDD
IEF237I JES2 ALLOCATED TO EQQQGDGDD
IEF237I JES2 ALLOCATED TO EQQQDUMP
IEF237I JES2 ALLOCATED TO SYSSPRINT
IEF237I JES2 ALLOCATED TO SYSSDUMP
IEF237I JES2 ALLOCATED TO SYSSPRINT
IEF237I JES2 ALLOCATED TO SYSSDUMP
IEF237I JES2 ALLOCATED TO SYSSPRINT
IEF237I JES2 ALLOCATED TO SYSSDUMP
IEF142I GDG00001 EQQCLEAN EQQCLEAN - STEP WAS EXECUTED - COND CODE 0000
IEF205I TWS81.SERVICE.APFLIB1 KEEP
IEF205I TWS81.SERVICE.APFLIB KEEP
IEF205I TWS81.SERVICE.APFLIB KEEP
IEF205I OPCSTC.GDG00001.JOB00010.0000101.7 SYSIN
IEF205I OPCSTC.GDG00001.JOB00010.0000102.7 SYSIN
IEF205I OPCSTC.GDG00001.JOB00010.0000103.7 SYSIN
IEF205I OPCSTC.GDG00001.JOB00010.0000104.7 SYSOUT
IEF205I OPCSTC.GDG00001.JOB00010.0000105.7 SYSOUT
IEF205I OPCSTC.GDG00001.JOB00010.0000106.7 SYSOUT
IEF205I OPCSTC.GDG00001.JOB00010.0000107.7 SYSOUT
IEF373I STEP/EQQCLEAN/START 2009160.1215
IEF373I STEP/EQQCLEAN/STOP 2009160.1215 CPU 0MIN 00.01SEC SRB 0MIN 00.00S
IEF202I GDG00001 STEP0 - STEP WAS NOT RUN BECAUSE OF CONDITION CODES
IEF272I GDG00001 STEP0 - STEP WAS NOT EXECUTED.
IEF202I GDG00001 STEP1 - STEP WAS NOT RUN BECAUSE OF CONDITION CODES
IEF272I GDG00001 STEP1 - STEP WAS NOT EXECUTED.
IEF236I ALLOC. FOR GDG00001 STEP2
From this JOBLOG you can see that:

- STEP0 and STEP1 were correctly simulated (see EQQC02I messages).
- STEP2 was executed correctly using as input MYGDG.ROOT.G0126V00.
- The submitted JCL was run with the relative numbers unchanged.
- The GDG MYGDG.ROOT(+1) was correctly overwritten (see message EQQC25I).

To obtain the same result, you could also have used the following expanded JCL:

```plaintext
//GDG00001 JOB (9805,SS), 'D-JOB', MSGLEVEL=(1,1)
//TIVDST00 OUTPUT JESDS=ALL, DEST=TWSFDEST
//TIVDSTAL OUTPUT JESDS=ALL
//STEP0 EXEC PGM=IEFBR14
//STEP1 EXEC PGM=IEFBR14
//DD01 DD DSN=MYGDG.ROOT.ROXGDG.G0126V00,UNIT=3390,
// DCB=(RECFM=FB, LRECL=80, BLKSIZE=800), SPACE=(TRK,(1,1)),
// VOL=SER=TWS01, DISP=(NEW, CATLG, DELETE)
//STEP2 EXEC PGM=IEFBR14
//DD21 DD DSN=MYGDG.ROOT.ROXGDG.G0126V00, DISP=SHR
```

With the normal JCL, you could also execute a step restart while retaining previously allocated GDG data set, while this is not possible with expanded JCL. For example, if you wish to execute a step restart from STEP2, while retaining the previously allocated GDG data set MYGDG.ROOT.G0126V00, and allocating a new generation GDG, you can:

- Select normal JCL.
- Customize user exit EQQUXGDG to retain GDG root MYGDG.ROOT data sets for GDG00001 JOB.
- Exclude the scratch of MYGDG.ROOT.G0126V00 from the action list.
- Set STEP1 as the restart step.

For details about EQQUXGDG, see Customization and Tuning.

## GDG resolution considerations

GDG resolution is slightly different depending on the type of JCL:

- Expanded JCL:
  
  Data store gets the information about GDG data sets from the JOBLOG of the last run only. All the GDG data sets for which information is directly available (SYSMGS contains a GnnnnVxx expansion message), are resolved immediately with the use of such information. The GDG data sets for which information is not directly available (such as GDG in flushed steps) are resolved with the following DIFF mechanism: If a resolved GDG data set exists with the same GDG root data store, calculate the difference between the two relative numbers to obtain the absolute value. For example, if data set GDGRoot(+1) was resolved as GDGRoot.G0025V00, the unresolved GDG data set GDGRoot(+2) will be resolved as GDGRoot.G0026V00.
  
  This might cause numbers to wrap around. For example, if the resolved GDG data set is GDGRoot(+1)=GDGRoot.G9999V00, the unresolved GDG data set
GDG resolution considerations

GDGRoot(+2) will be resolved as GDGRoot.G0001V00. As a consequence, if there are skipped GDG generation numbers, the resolution might be incorrect. The DIFF mechanism implies also that different versions (such as V00 and V01) could lead to incorrect resolutions: GDG resolution is not supported for this kind of JCL. For example, suppose you run the following JCL having GDGROOT(0) = GDGROOT.G0004V00 before the first job run:

```jcl
//JOB XXX
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=GDGROOT(+1),DISP=(NEW,CATLG)
//STEP2 EXEC PGM=IEFBR14
//DD2 DD DSN=GDGROOT.G0005V01,DISP=(NEW,CATLG)
//STEP3 EXEC PGM=IEFBR14
//DD3 DD DSN=GDGROOT(+1),DISP=OLD
//STEP4 EXEC PGM=IEFBR14
//DD4 DD DSN=GDGROOT(+2),DISP=OLD
```

The result of the first run will be:
- STEP1 ==> allocates GDGROOT.G0005V00
- STEP2 ==> allocates GDGROOT.G0005V01
- STEP3 ==> refers GDGROOT.G0005V01
- STEP4 ==> refers GDGROOT.G0006V00

Now execute a second run, restarting from STEP3 and excluding STEP4. The result will be:
- STEP3 ==> refers via GDG simulation GDGROOT.G0005V01

Now execute a third run, restarting from STEP1. The result will be a JCL error due to incorrect GDG resolution. This happens because the JOBLOG of the last run had information only about:
- STEP3 GDGROOT(+1) = GDGROOT.G0005V01

and the DIFF mechanism is not able to distinguish when to use V01 and when to use V00.

- Normal JCL:
  Same as in expanded JCL with an additional iteration of the resolution process using information from all the previous job runs whenever unresolved GDGs still exist. Resolution is performed by the Controller and not by Data store.

Job restart

Job restart lets you restart the entire job and performs the appropriate cleanup actions. For more information about data set cleanup, see "Data set cleanup" on page 331. As you are restarting the job from the beginning, there is no simulation of preceding steps. There will also be no resolution of GDG relative numbers when normal JCLs are used. On the contrary, if expanded JCL is used, GDG resolution will always be applied (see "JCL used for restart" on page 315 and "GDG resolution" on page 325 for details.)

Start cleanup

To perform cleanup actions only (no restart), select the START CLEANUP option in the OPERATION RESTART AND CLEANUP panel. All the data sets, starting from the first step, are considered for the cleanup actions. The type of cleanup action depends on the following:
- The status of the data set when cleanup is requested.
- The DISP parameter specified in the JCL.
Start cleanup with AR

For more information about data set cleanup, see “Data set cleanup” on page 331.

Start cleanup with AR

Start cleanup with AR behaves as the Start cleanup option, with the difference that the list of data sets is created considering the restart step detected by the automatic recovery function. If the cleanup type of an operation is set to manual and the automatic recovery function detects a restart step for that operation, the automatic recovery actions are postponed until the appropriate cleanup actions are complete. To perform the cleanup actions, you need to request them by selecting the Start Cleanup with AR option in the panel that is displayed.

The Start cleanup with AR action is performed only if the cleanup type is set to manual and the automatic recovery function detects that a job restart is required. If a job restart is not required and in the AROPTS statement CHKRESTART=NO, use the Start cleanup function to confirm or discard any pending cleanup action.

Display cleanup

To see the results of a data set cleanup, select the DISPLAY CLEANUP option on the OPERATION RESTART AND CLEANUP panel. The results refer only to the last cleanup completed.

Data set cleanup

This section shows how Tivoli Workload Scheduler for z/OS can help with the recovery and restart of z/OS jobs and started tasks by:
- Deleting data sets that were created in a job
- Uncataloging data sets that were cataloged in a job
- Cataloging data sets that were uncataloged in a job.

In this section, the term job refers both to a z/OS job and to a started task, unless otherwise stated. Data set cleanup is available for jobs that are tracked by the scheduler on the z/OS operating system and are in the current plan.

There are two types of data set cleanup:
- Cleanup that is based on the history of a failed job run. This section describes this type in detail.

Note: After an operation is run with CLEAN UP TYPE = NONE, any subsequent attempt to restart that occurrence might fail since history information is not available. GDG resolution will not be performed if CLEAN UP TYPE = NONE.

- Preventive cleanup that you can perform before a job runs. This type of cleanup runs a general cleanup of data sets, which is useful if you do not know the exact state of the data sets. If you prefer to cleanup before a job first runs, look at the EQQDELDI member in the supplied sample library. This describes a program that deletes data sets with the NEW disposition that are not referenced with a different disposition (OLD or SHR) in previous job steps. EQQDELD is an optional part of the product, which can be used to avoid “not catlg 2” situations.

Selecting cleanup options

See “What you need for restart and cleanup” on page 305 for the initialization options that you need for cleanup. When you have specified the initialization options, your cleanup options are: Automatic, Immediate, or Manual.
Automatic, immediate, and manual cleanup

You can specify the type of cleanup you require from any of the following:

- APPLICATION DESCRIPTION panel (see "Options that apply to jobs and started tasks" on page 151 for more information)
- JOB DESCRIPTION panel
- Batch loader
- Program interface
- MODIFY CURRENT PLAN (MCP) panel

To perform the type of cleanup you require, specify either the Automatic, Immediate, or Manual option for each operation. If you require no cleanup, specify None.

Because in some cases EQQCLEAN might delete a data set by mistake, it is recommended that you protect critical data sets from deletion by using either the RCLOPTS parameters (DDPROT, DDPRMEM, DSNPROT, DSNPRMEM) or the EQQUXCAT exit.

Automatic

Cleanup actions are started either by a pre-defined trigger or from the panels. The controller automatically determines the cleanup actions to be taken and also inserts them as the first step in the JCL of the restarted job. With this option the cleanup will automatically rerun an occurrence when the job is ready and no other conditions keep the job from rerunning.

Immediate

The job is cleaned up as soon as it fails. If a tracked job ends in error, and you have specified Immediate for the operation, the controller checks for data set information in the current plan, according to the following criteria:

If the automatic recovery action is to restart the operation

Checks for data set information starting from the step indicated by the automatic recovery function.

If the automatic recovery action is not to restart the operation

- If RCLOPTS IMMEDLOGIC(FIRSTSTEP) was specified (this is also the default), checks for data set information starting from the first step up to the last step of the job (included).
- If RCLOPTS IMMEDLOGIC(BESTSTEP) was specified, from the best step suggested.

If data set information is found, the controller submits a stand-alone cleanup job, otherwise sets the cleanup status to Complete.

Manual

Cleanup is controlled manually. If a tracked job ends in error and manual cleanup actions have been defined for the operation, or when a rerun is requested for an operation that specifies Manual, you must initiate the cleanup action from the MODIFYING CURRENT PLAN panel. The panel shows what action the scheduler will take for each of the affected data sets. You can exclude the action for any or all of the data sets, if required. When you initiate the cleanup action, the controller submits a standalone cleanup job. In each case, actions for data sets belonging to flushed steps are not performed. Actions are also not performed for data sets referenced in previous steps that have a disposition of OLD, SHR, or MOD, whenever these steps are included in the new run.
Automatic, immediate, and manual cleanup

Note that the manual cleanup option has the same effect as the automatic cleanup option, if you run a Fast Step Restart or a Fast Job Restart function. In fact, in these cases, you run the Step Restart or the Job Restart process with a unique command, and you are not allowed to change by panel the actions that the scheduler will take for each of the affected data sets.

Job and step-level cleanup

With step-level cleanup, you can perform cleanup actions for a range of steps. On the other hand, job-level cleanup is for all steps up to the last step executed.

Cleanup logic selects all data sets eligible for deletion depending on the range of selected steps and the options specified in the RCLOPTS initialization statement.

Cleanup marks all data sets that are defined with DISP=NEW as eligible for potential deletion. If they belong to flushed steps, or are referenced in previous steps with a disposition of OLD, SHR, or MOD, then they are not marked for action.

To start cleanup actions from the MCP panel, use the RC row command from one of the following:

- Ended-in-error-list.
- Operation list when you modify an operation.
- Operation list when you rerun an occurrence.
- Dependency status change list.

In each case, the OPERATION RESTART AND CLEANUP main menu is displayed. You can choose a step-level restart, job restart, a cleanup without restart, or cleanup results for a specific operation.

The operator can reject the deletion if it is not appropriate in the context of the job stream to be restarted.

Accessing the restart and cleanup data sets, the controller reads the historical data set information and decides the required cleanup actions depending on the step range. Consider this example of the events for restart and cleanup:

```
//S1 EXEC PGM=P1
//S2 EXEC PGM=P2
//S3 EXEC PGM=P3
//DD1 DD DSN=TEST.FILE.ONE,DISP=(NEW,CATLG,DELETE),
//VOL=SER=TSOL05,SPACE=(TRK,(1,1)),UNIT=3390,
//DCB=(BLKSIZE=3120,LRECL=80,RECFM=FB)
//S4 EXEC PGM=P4
//DD1 DD DSN=TEST.FILE.TWO,DISP=(SHR,UNCATLG)
//S5 EXEC PGM=P5
//S6 EXEC PGM=P6
//S7 EXEC PGM=P7
//S8 EXEC PGM=P8
```

This is the sequence of events for restart and cleanup:
1. The job fails at step S6.
2. From the error list, you restart the job from step S5.
3. The scheduler does not take any cleanup actions, because steps S3 and S4 are outside the scope of the restart.
4. The job fails again.
5. You restart it from step S2.
6. The scheduler now deletes the data set that was created in the first run in step S3. This does not happen for release 2.1 or earlier of the scheduler.
7. The scheduler catalogs the data set that was uncataloged in step S4. This does not happen for release 2.3 or earlier of scheduler.

**When are cleanup actions performed?**

If you specified RCLEANUP=YES in the OPCOPTS initialization statement, the scheduler performs cleanup actions depending on the cleanup option defined at operation level.

If you define an operation with the automatic cleanup option, cleanup actions can be started internally or from the panels. Usually, cleanup actions are executed as a first step at run time. If you define an operation with immediate cleanup type, the scheduler initiates the actions as soon as the job fails, with these exceptions:

- The job fails with an error code that is set before, or during, job submission:
  - OSEQ
  - OSUB
  - OSUF
  - OSUP
  - OJCV
  - JCLI
  - LOOP

- The job fails with one of these codes:
  - MCP
  - OSSQ
  - OSSS
  - OFSQ
  - OFSS
  - OFSC

These error codes indicate that workload restart has failed, or that the operation has been manually set to error. The scheduler automatically changes the cleanup action from Immediate to Manual.

If you define an operation with the manual cleanup option, the scheduler starts cleanup actions only when you initiate it from Modify Current Plan. In any case, the cleanup process can be launched for a selected operation without restarting it (unless the cleanup option is set to none).

If both automatic recovery and cleanup actions are defined for an operation, the scheduler performs the cleanup actions first.

**What actions are taken for affected data sets?**

Cleanup actions are designed to restore the catalog entries that were modified by the job to the state that they were in before the job started. Cleanup either deletes or uncatalogs data sets.

When the scheduler performs cleanup processing for a job, data sets and generation data groups (GDG) generations that have been created in that job are either deleted or uncataloged. Uncataloged data sets are cataloged. Table 20 describes the action that the scheduler takes for each data set disposition.
### What actions are taken for affected data sets?

Table 20. Cleanup actions for data set dispositions in a job

<table>
<thead>
<tr>
<th>DISP Reference to the data set</th>
<th>Data Set Allocated and Cataloged at Job Start</th>
<th>Reference to a Previous Step</th>
<th>SMS in the Environment</th>
<th>Protected by RCLOPTS</th>
<th>Cleanup Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEW, CATLG</td>
<td>YES</td>
<td>YES</td>
<td>No impact</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td>NEW, KEEP</td>
<td>YES</td>
<td>YES</td>
<td>No impact</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td>OLD, UNCATLG</td>
<td>YES</td>
<td>No impact</td>
<td>YES</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>YES</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>YES</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>YES</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td>SHR, UNCATLG</td>
<td>YES</td>
<td>No impact</td>
<td>YES</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>YES</td>
<td>No impact</td>
<td>YES</td>
<td>None</td>
</tr>
<tr>
<td>MOD, CATLG</td>
<td>YES</td>
<td>Yes</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>Yes</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>Yes</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>MOD, UNCATLG</td>
<td>YES</td>
<td>No impact</td>
<td>Yes</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td>, , CATLG (abnormal step end)</td>
<td>YES</td>
<td>Yes</td>
<td>No impact</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td>, , UNCATLG (abnormal step end)</td>
<td>YES</td>
<td>Yes</td>
<td>No impact</td>
<td>No impact</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
<tr>
<td></td>
<td>NO</td>
<td>No impact</td>
<td>No impact</td>
<td>Yes</td>
<td>None</td>
</tr>
</tbody>
</table>
What actions are taken for affected data sets?

Table 20. Cleanup actions for data set dispositions in a job (continued)

<table>
<thead>
<tr>
<th>DISP Reference to the data set</th>
<th>Data Set Allocated and Cataloged at Job Start</th>
<th>Reference to a Previous Step</th>
<th>SMS in the Environment</th>
<th>Protected by RCLOPTS</th>
<th>Cleanup Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 The Cleanup action depends on the value specified in the SMSMODDELETE parameter in DSTOPTS:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>If SMSMODDELETE(NO)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>If SMSMODDELETE(YES)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 The Cleanup action depends on where the data set is allocated:</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>None</td>
<td>If the data set is allocated on a tape</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delete</td>
<td>If the data set is allocated on a DASD</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

If you restart a job from a specified step, the scheduler does not try to restore a data set catalog status that was changed before the restart step.

For example, the job in Figure 137 was tracked by the scheduler, failed at the third step, and had immediate cleanup defined. The cleanup actions are the following:

1. This data set would be uncataloged.
2. This data set would be cataloged.
3. This data set would be cataloged.

```
//STEP1 EXEC PGM=FIV
//DDX DD DSN=IM.TEST.SHIP,DISP=(NEW,CATLG),UNIT=TAPE, VOLUME=SER=T2,LABEL=(3,NSL),RETPD=188
//STEP2 EXEC PGM=EQQYCOM
//DDY DD DSN=IM.TEST.PANELS,DISP=(OLD,UNCATLG),UNIT=3400, VOLUME=SER=10222,LABEL=4
//STEP3 EXEC PGM=NOPGM
//DDZ DD DSN=IM.TEST.HELP,DISP=(SHR,UNCATLG),UNIT=3400, VOLUME=SER=10222,LABEL=4
```

Figure 137. Example of cleanup action for a job

Actions for data sets with DISP=NEW are not executed when they belong to flushed steps or when they were referenced in previous steps with DISP=SHR, OLD, or MOD, which are included in the new run range.

**SMS**

SMS does not allow UNCATALOG without DELETE, so the only valid action performed by the scheduler is DELETE.

Depending on the value specified in the SMSMODDELETE parameter in DSTOPTS, the SMS data sets allocated with DISP=(MOD, CATLG, CATLG) are deleted at restart. For more details, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*

**Migrated data sets**

If a data set is migrated by DFHSM (VOLSER=MIGRAT) between creation and the time when cleanup attempts to take some action for the job, the scheduler uses the HDELETE macro to complete the action.
Generation data groups

In a non-SMS environment, the entry for the data set will be removed from the GDG base, leaving the generation data set uncataloged. In either environment, DELETE causes the data set to be scratched, and its entry in the catalog to be removed. The absolute generation name is used during cleanup processing (GDGRoot.GnnnnVnn).

The definition of the GDG base determines whether a generation data set is removed from the VTOC and the catalog, when the maximum number of generations for a GDG base is reached in a job. The scheduler does not try to override these rules. You cannot restore GDGs that have rolled off the group as a result of new data sets created in the failing job. Cleanup can only reset the catalog; it cannot recover data that has been deleted.

When specifying operation details for a job that has GDG, you can decide whether or not to use expanded JCL. At job or step restart you will be able to confirm or override your choice. To decide which kind of JCL to use, see "JCL used for restart" on page 315 and "GDG resolution" on page 325.

Protecting data sets from unintentional deletion

Tivoli Workload Scheduler for z/OS provides different ways to protect data set from unintentional deletion. They are:

- Using a DD name that matches one of the names specified in the RCLOPTS DDPROT keyword.
- Adding the data set name to the list of protected data set in the RCLOPTS DSNPROT keyword.
- Implementing the EQQUXCAT user exit that is called just before the data set is deleted.

Recovering from workstation failures

The scheduler can transfer operations from one workstation to another in the event of a failure or of the unavailability of a workstation. If you specify that operations be automatically rerouted to alternate workstations, ensure that your systems are symmetrical as far as data set and catalog references are concerned.

This is the effect of the first parameters of the WSFAILURE and the WSOFFLINE keywords of the JTOPTS statement on cleanup processing:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ERROR</td>
<td>All operations with the immediate cleanup type are changed to manual cleanup. This means that you must initiate or discard the action from the MODIFY CURRENT PLAN panel.</td>
</tr>
<tr>
<td>RESTART</td>
<td>The cleanup action is discarded.</td>
</tr>
<tr>
<td>LEAVE</td>
<td>The cleanup action is determined by the succeeding status of the operation.</td>
</tr>
</tbody>
</table>

How does cleanup work?

The scheduler cleans up the data set of a job in one of the following ways:

- At the time the job is submitted by the controller, a step is added to the JCL before the existing steps. The name of the added step is EQQCLEAN. It uses the list of data sets to be deleted, cataloged, or uncataloged as input. This list is obtained from the history of previous runs. To get the history, the controller requests the job log from the data store and then merges the history in the local repository of the controller.
How does cleanup work?

- A single-step job, created by the controller, is executed before the job run.

If automatic job recovery is used, cleanup and automatic recovery cooperate to ensure that cleanup actions are either completed, or revoked before any automatic recovery action is started for the operation. For information about automatic job recovery, see Chapter 18, “Automatic recovery of jobs and started tasks,” on page 347.

Note that a user exit, EQQUXCAT, is available to be called by EQQCLEAN to discard specific cleanup actions at job run time. For details, see the Customization and Tuning manual.

Fast step restart

When you invoke the step restart function, you are required to go through several panels before the step restart is actually run. For example, the STEP RESTART SELECTION LIST panel (EQQMERSL) is displayed for you to select the restart range. The CLEANUP ACTION LIST panel (EQQMCDML) might be displayed to let you change the cleanup action list. The editing panel (EQQMMJCL) might be displayed to let you edit the JCL during restart. Also, confirmation panels (EQQMERMJP, EQQMERTP) are shown. To skip these panels, leaving all choices as defaults, you can invoke the Fast Step Restart function.

Like the Step Restart function, Fast Step Restart lets you restart a job or a started task at the step level and performs the appropriate cleanup actions. Unlike Step Restart, it lets you do it with a single command. Fast Step Restart is based on the same simulation mechanism from the history of the previous runs as the Step Restart function. The EQQCLEAN step is added and performs the simulation of return codes, cleanup actions, and resolution of GDGs, but defaults are used where selections were allowed with the Step Restart. See “Return code simulation logic” on page 322, “Data set cleanup” on page 331, and “GDG resolution” on page 325 for details.

The Fast Step Restart function can be requested for any operation, which ran at least once, by issuing the FSR command from any of the following operation lists in the MODIFY CURRENT PLAN panels:
- Ended-in-error operation list
- Operation list
- Operation displayed when you request to rerun an occurrence

Note: The first time you issue the FSR command, message EQQM601I (Scheduler has requested needed JOBLOG information to process the command) is displayed if the JOBLOG information is not yet available to data store. In this case, you must issue the FSR command again. Message EQQM676I (Fast step restart is being processed) is then displayed and, unless other warnings are issued, the operation is finally completed.

Defaults
- Expanded JCL
  - The Expanded JCL value used during Fast Step Restart function is the value found in the related CP operation record (built from the AD application definition value unless later modified by the user via MCP).
- Restart Range
  - When you request a Fast Step Restart, you take defaults for the first and last step to be included in the rerun. The first restart step is defaulted to the best
Fast step restart

step identified by the scheduler, if any. If there are no restartable steps and therefore there is no best step, the process is stopped and an appropriate message is issued.

Fast job restart

The Fast Job Restart function allows you to restart a job entirely and to perform the appropriate cleanup actions. Unlike the Job Restart function, Fast Job Restart lets you process a Job Restart with a single command. Because you are restarting the job from the beginning, there is no simulation of preceding steps. There is also no GDG relative numbers resolution, if normal JCL is used. If Expanded JCL is used, instead, GDG resolution is applied, based on the history of the previous runs. See “Data set cleanup” on page 331 “JCL used for restart” on page 315 and “GDG resolution” on page 325 for details.

The Fast Job Restart function can be requested for any operation, which ran at least once, by issuing the FJR command from any of the following operation lists in the MODIFY CURRENT PLAN panels:

- Ended-in-error operation list.
- Operation list.
- Operation displayed when you request to rerun an occurrence.

Note: The first time you issue the FJR command, message EQQM601I (Scheduler has requested needed JOBLOG information to process the command) is displayed if the JOBLOG information is not yet available to data store. In this case, you must issue the FJR command again. Message EQQM677I (Fast job restart is being processed) is then displayed and, unless other warnings are issued, the operation is finally completed.

Defaults

- Expanded JCL
  - The Expanded JCL value used by the Fast Job Restart function is the value found in the related CP operation record (built from the AD application definition value, unless later modified by the user via MCP).

Fast start cleanup

The Fast Start Cleanup function allows you to perform the appropriate cleanup actions for a job and, unlike the Start Cleanup function, Fast Start Cleanup lets you process a Start Cleanup with a single row command.

Because you are issuing a fast path command, all the data sets that are not protected, starting from the first step, will be considered for the cleanup actions; that is, there is no possibility to exclude data sets from the cleanup list. See “Data set cleanup” on page 331 for further details.

This function can be requested for any operation, which ran at least once, by issuing the FSC command from any of the following operation lists in the MODIFY CURRENT PLAN panels:

- Ended-in-error operation list
- Operation list
- Operation displayed when you request to rerun an occurrence
- List dependencies that will be changed at rerun

Note: The first time you issue the FSC command, message EQQM601I (OPC has requested needed JOBLOG info to process the command.) is displayed if the
Fast start cleanup

JOBL06 information is not yet available to Data store. In this case, you must issue the FSC command again. Message EQQM692I (Fast Start Cleanup is being processed) is then displayed and, unless other warnings are issued, the controller submits a standalone cleanup job.

Considerations for event-triggered tracking

In some cases, a job is submitted from outside Tivoli Workload Scheduler for z/OS by using event-triggered tracking (ETT). When you set the ETT criteria for job-name replace tracking to YES, consider the following:

- You must add additional JCL to enable data set cleanup. To make a copy of the SYSOUT, which is then captured by the data store, add the following to the JESDS statement:

  ```
  //TIVDST00 OUTPUT JESDS=ALL,DEST=OPCA
  //TIVDST08 OUTPUT JESDS=ALL
  //TIVDSTUS OUTPUT DEST=OPCA
  ```

  The result provides a second output from the job taken from the data store to enable the jobs or steps to be restarted.

- A job added using ETT, with job-name replace option set to YES, might produce a stand-alone cleanup job. In this case, the job-submit exit (EQQUX001) cannot affect the owner ID of the stand-alone cleanup job.

For more information about ETT, see “Adding occurrences by event-triggered tracking” on page 432.

Generating copies for data store

Both the Joblog retrieval and the Restart and cleanup functions require for job output that an "extra" copy of the JESDS spool data sets be generated for processing by the Tivoli Workload Scheduler for z/OS DATASTORE started task on its reserved destination queue. This data is captured into the DATASTORE SDF and UDF data sets (STRUCTURED and UNSTRUCTURED DATA FILES) and used to create restart JCL and determine what catalog cleanup might be required to restart a specific job at the user-selected restart point.

To generate job output copies, the controller automatically adds statements to each JCL before it submits it, for example:

```
//TIVDST00 OUTPUT JESDS=ALL,DEST=TWSFDEST INSERTED BY TWS
//TIVDSTAL OUTPUT JESDS=ALL INSERTED BY TWS
//TIVDSTUS OUTPUT DEST=TWSFDEST INSERTED BY TWS
```

The type and number of statements vary depending on the characteristics of the JCL, for example:

- If a user SYSOUT is set to YES at the operation level.
- If the JESDS keyword is already used by the JCL.
- If a DEFAULT output is defined in the JCL.

The TIVDSTxx OUTPUT JCL statements are inserted into each job as its JCL is prepared for submission by the PRE-SUBMIT subtask in the Tivoli Workload Scheduler for z/OS Controller. Use the EQQUX013 user exit to select the jobs for which you do not need to generate additional copies, because you do not need to retrieve the joblog and use the restart and cleanup function. The EQQUX013 user exit allows the Work Station Analyzer to return an error code that prevents the jobs that are being submitted from being tailored with the //TIVDSTxx OUTPUT statements. For additional details, refer to Customization and Tuning.
Generating copies for data store

The TIVDST00 statement is inserted into ALL scheduled jobs and started tasks. It generates the extra copy of the JESDS=ALL output required by the Tivoli Workload Scheduler for z/OS Data Store.

If the job did not already contain at least one JESDS=ALL OUTPUT statement, and only the TIVDST00 statement was added, that statement would “hi-jack” the one-and-only “normal” copy of the joblog output to the data store destination. So, in this case, the TIVDSTAL statement is added to make sure there are two copies of this output, one for normal processing and a second one for the data store.

The TIVDSTUS statement is added on a job-by-job basis, depending on whether the operation has cleanup option “USER SYSOUT ==> YES”. Its purpose is to generate an extra copy of any user-allocated SYSOUT datasets (/xxxxxxxx DD SYSOUT=*) for the DATASTORE to capture and process. USER SYSOUT is not needed for any Tivoli Workload Scheduler for z/OS R&C processing but users might wish to be able to browse it through the “L” row command in the Tivoli Workload Scheduler for z/OS dialogs.

Notes:

1. OUTPUT statements with JESDS=ALL must be inserted after the job card, but prior to any EXEC statements. Because an INCLUDE member might contain an EXEC statement, the default processing is for the Tivoli Workload Scheduler for z/OS pre-submit code to stop scanning the JCL for OUTPUT statements when it encounters the first INCLUDE. Thus, if a given job has OUTPUT statements with JESDS=ALL after an INCLUDE, that statement will not be found by Tivoli Workload Scheduler for z/OS and a TIVDSTAL OUTPUT statement will be inserted when it is not needed. This causes an unwanted extra copy of the JESDS=ALL output to be generated.

2. OUTPUT statements with JESDS=ALL must be inserted AFTER any JOBLIB or JOBCAT statements. So, if a job has INCLUDES which contain these statements, the TIVDSTxx OUTPUT statement is inserted prior to the JOBLIB/JOBCATs, and the job will fail with a JCL ERROR.

Both of the above reported problems are easily resolved.

If the TIVDSTxx OUTPUT statements are to be inserted AFTER an INCLUDE, put the name of the INCLUDE MEMBER in the RCLSKIP INCLNAME() list in the parmlib member pointed to by the CONTROLLER RCLOPTS SKIPINCLUDE() keyword.

Note:

If a specific INCLUDE member contains both JOBLIB and JOBCAT statements and EXEC statements, the TIVDSTxx OUTPUT statement cannot be validly placed either before, or after that INCLUDE. Thus, the INCLUDE must be broken into two or more separate members and the JCL adjusted appropriately in the job to be scheduled and submitted by Tivoli Workload Scheduler for z/OS.

If, on the other hand, you submit a JCL externally to Tivoli Workload Scheduler for z/OS and not from a controller (for example, via ETT with the jobname replacement option set to YES), you need to add similar statements to the JCL yourself. Otherwise, data store cannot archive the output.
Generating copies for data store

The purpose of the //TIVDST.xx cards, that are added by the controller, is only to create extra copies for data store, without altering the behaviour of the previous job. The //TIVDST.xx cards are reserved and customers should not use this type of names.

This automatic copy generation does not work in the following cases:
- For a user SYSOUT that is defined in expanded procedures. Data store cannot archive this kind of user SYSOUT.
- For jobs that end in JCL error when the error type prevents JES from processing the JESDS statement. No copies are generated, neither of the MVS joblog nor of the user SYSOUT. Data store cannot archive this type of joblog.

Automatic copy generation requires some additional JCL tailoring by the controller in the following cases:
- When the submitted JCL refers to a started task. The controller adds a JOB card statement, because the JESDS statement works only if the JOB card is present.
- Whenever the started task is run by only one procedure, adding the JOB card statement is not enough. The controller also adds a step statement that calls the procedure at the end of the JCL. For example, the controller changes the following JCL:
  
  //MYSTC PROC
  //MYSTC EXEC PGM=IEFBR14

  to:

  //MYSTC JOB MSGLEVEL=(1,1)
  //TIVDST00 OUTPUT JESDS=ALL,DEST=TWSZDEST INSERTED BY TWS
  //TIVDSTAL OUTPUT JESDS=ALL INSERTED BY TWS
  //MYSTC PROC
  //MYSTC EXEC PGM=IEFBR14
  //PEND
  //TIVDSPRO EXEC MYSTC

  where the reserved destination is TWSZDEST. The statements in bold are added to the JCL.

**Note:** When the TIVDSPRO step is added to tailor the JCL, the referback of type STARTING does not work. The STARTING referback can be used when the started task invokes procedures having COND statements that point to original started task JCL. For example, the following started task:

  //FRED PROC
  //S1 EXEC PGM=IEFBR14
  //S2 EXEC XXXSPRO
  //PEND

  where XXXSPRO is:

  //XXXSPRO PROC
  //S1 EXEC PGM=IEFBR14,COND=(0,NE,STARTING.S1)
  //PEND

  will be tailored before submission as:

  //FRED JOB MSGLEVEL=(1,1)
  //FRED PROC
  //S1 EXEC PGM=IEFBR14
  //S2 EXEC XXXSPRO
Generating copies for data store

and this will cause the following JCL error:
10 IEF645I INVALID REFERBACK IN THE COND FIELD

To avoid this type of problem, you should either not use the STARTING referback, or update the JCL by changing STARTING to TIVDSPRO.

JCL changes considerations

The Restart and Cleanup function is based on information from the previous run. In particular, for step restart, the step list obtained from the previous run is used to run the return code simulation. The same considerations apply also to GDG simulation. To accomplish this, the EQQCLEAN pre-step (the one running data set cleanup, step RC simulation, and GDG data set simulation) identifies and locates the steps and data set to be handled using the list of steps and data set built from the controller, based on step names, DD names, and data set names taken from the previous run.

As a consequence, Restart and Cleanup might fail if either of the following occurs:
• The JCL structure is changed (via exit01, ISPF, directives like FETCH, BEGIN, or END) in a way that makes the list passed by the controller no longer valid (for example, by changing the name of a step to be simulated).
• Any data sets obtained from the previous run is renamed or deleted in a way that makes the list passed by the controller no longer valid.

As a consequence, if the JCL structure is changed (via exit01, ISPF, directives like FETCH, BEGIN, or END) in a way that makes the list passed by the controller no longer valid (for example, by changing the name of a step to be simulated), Restart and Cleanup might fail.

If the JCL is changed via user exit EQQUX001, the exit will know if it is called in a restart and cleanup path by checking the CALTYP parameter.

Usually, EQQUX001 is used to add pre-step EQQDELDS. If this happens, it should be done every time the exit is called, so that the step list used by the RC function is consistent with the real JCL step list. To prevent EQQDELDS from being re-executed in the following run, add a DD DUMMY with the DDNAME listed in the RCOPTS DDNEVER parameter: this makes the EQQDELDS step never re-executable on Step Restart.

Also, to prevent EQQDELDS from being re-executed at Job Restart, customize the EQQUX001 exit to invoke the EQQDELDS program with NOREEX set to YES in the following runs of the job (as shown in the sample provided with Tivoli Workload Scheduler for z/OS). In this way, the execution of EQQDELDS is simulated.

Remember that if EQQCLEAN is present, it must always appear as the first step in the sequence. If you decide to add EQQDELDS, place it following EQQCLEAN.

In general, you should not:
• Remove reserved TWSZ statements such as:
JCL changes considerations

- TIVDSTxx OUTPUT statements
- //OLDSTEP=--
- //......$SUBJCL

- Change the name of an existing step.
- Change the name or the characteristics of an existing DD.
- Change the name or the characteristics of an existing data set name.
- Delete a step (unless done via EQQUX001 at each call, so that the submitted JCL will never contain that step).
- Insert a step (unless done via EQQUX001 at each call, so that the submitted JCL will always contain this step).
- Delete a DD that is not a DUMMY DD (unless done via EQQUX001 at each call, so that the submitted JCL will never contain that DD).
- Delete or rename a data set that was obtained from a previous run and that is needed to perform step restart actions.
- Insert a DD that is not a DUMMY DD (unless done via EQQUX001 at each call, so that the submitted JCL will always contain that DD).

But you can:
- Alter SYSIN data if needed.
- Alter Job Card to set a password or user.
- Insert a control statement such as /*JOBFROM.
- Add steps at the end of the JCL.

If the JCL is refreshed or changed in a Restart and Cleanup function, new and changed variables will be replaced with the current value in AD variable tables.

See also “Limitation on the number of job steps” on page 345.

EQQCLEAN Pre-step insertion considerations

The IBM Tivoli Workload Scheduler for z/OS controller can insert the pre-step EQQCLEAN into the JCL before submitting it, to perform:
- The Cleanup together with the job, and/or
- A Step Restart simulating the Restart and Cleanup and GDG, when you requested in the appropriate panel

This means that the EQQCLEAN pre-step can be inserted when requested by:
- A user command in a panel, when Restart and Cleanup Step Restart was selected
- A rerun of a job with the cleanup type set to Automatic

In both these situations, you cannot see or modify the EQQCLEAN pre-step when editing the JCL, because the submitted JCL is stored in JS VSAM without the pre-step. Only when you edit the JCL through the Restart and Cleanup, Step Restart or Job Restart, and the expanded JCL type was requested, you can see the EQQCLEAN pre-step, though the changes cannot be seen. The JCL of expanded type is never stored in JS VSAM.

When you insert the EQQCLEAN pre-step in a JCL, consider that:
- It must be inserted after the JOB statement.
- It must be inserted immediately before the first EXEC statement.
**EQQCLEAN Pre-step insertion considerations**

- If there are INCLUDE statements before the first EXEC statement, insert the pre-step before the INCLUDE statements. However, if the INCLUDE statements are listed in the RCLOPTS SKIPINCLUDE keyword, this rule does not apply. In this case, insert the pre-step immediately before the first EXEC statement.

Because the Tivoli Workload Scheduler for z/OS Controller inserts the EQQCLEAN pre-step without knowing the contents of the INCLUDE statements (handled by JES), these type of statements in a JCL could cause, in particular conditions, a wrong positioning of the EQQCLEAN pre-step. To prevent EQQCLEAN pre-step from wrong positioning, use the RCLOPTS SKIPINCLUDE keyword. For details about RCLOPTS SKIPINCLUDE, refer to the Customization and Tuning.

See also "Limitation on the number of job steps."

---

**Limitation on the number of job steps**

The default maximum number of steps that can be included in each job is 255. If this number is exceeded, the following message is issued:

IEFC602I EXCESSIVE NUMBER OF EXECUTE STATEMENTS

The allowed maximum decreases by one when each of the following steps is added:

- An EQQDELDS step via EQQUX001
- An EQQCLEAN step as part of restart and cleanup processing
- A dummy last step if RCLPASS(NO) is used and fix PK50941 fix is installed

If you add two of the steps above, it goes down to 253. If you add all three steps, it becomes 252.
Chapter 18. Automatic recovery of jobs and started tasks

Tivoli Workload Scheduler for z/OS supports the automatic recovery of failed jobs or started tasks if the job or started task fails. You can specify that Tivoli Workload Scheduler for z/OS should also recover jobs or started tasks automatically in the event of a system failure within the sysplex. A similar function is available without XCF. In this case, however, you must issue the command to recover operations manually, using the panels.

Specifying recovery criteria

You specify the recovery criteria as part of the JCL for the operation in the form of special control statements. In the case of jobs, the recovery instructions must be inserted between the JOB statement and the first execution step (after the JOBLIB or JOBCAT statement and in-stream procedures, if present). Tivoli Workload Scheduler for z/OS ignores recovery instructions within in-stream procedures. So for started tasks (where the whole JCL is an in-stream procedure) place any recovery statements before the PROC statement.

Using these statements, you specify the type of error for which the recovery is attempted and how the recovery is achieved. If the error type is not one that you have covered in your specification, the failed operation remains in the Operations Ended-in-Error list.

Tivoli Workload Scheduler for z/OS retrieves the JCL for automatic recovery from the JCL repository (JS) data set. This means that automatic recovery can take place only for jobs or started tasks submitted by Tivoli Workload Scheduler for z/OS.

The automatic recovery function takes over when a job or started task ends in error. At that time, this information is available:
- The error code for the operation. This can be:
  - The abend code of an abending step
  - The return code of the last step
  - An error code set by Tivoli Workload Scheduler for z/OS, such as JCLI, CCUN, JCL, CLNO, CLNA, CLNC, CAN, PCAN, CLNP, OFxx, or OSxx
  - An error code set by the job completion checker

  **Note:** Automatic recovery is not applicable for error codes, such as OSUP, that refer to jobs that have not reached the job queue.

- The name of the abending step, if the error is associated with a step.
- Step completion codes and step names for all steps executed. The step completion code is either an abend code or a return code.

If the error occurs in the initialization phase or in the completion phase of the job or started task, no step information is available. Statements that specify recovery actions for certain steps are not applicable to such errors.

Tivoli Workload Scheduler for z/OS begins the automatic recovery process by scanning the job for the first /*%OPC RECOVER statement where:
- The step name matches the name of the failing step from the operating system.
Specifying recovery criteria

- The error code matches the error code from the job and started-task tracking function.
- The return code matches the step return codes or abend codes from the job and started-task tracking function.
- The RECOVER statement is unconditional (it specifies no step name, error code, return code, or abend code).

This means you should place the RECOVER statements with the most restrictive matching conditions before the RECOVER statements that deal with more general cases.

For example, assume there are three recovery procedures for a job. R1 is set up to handle errors of type E. R2 is set up to handle errors of type T, which includes error type E. R3 is a general recovery procedure for all errors in the job. The RECOVER statements should be placed in this order:

```plaintext
//%OPC RECOVER if error E - actions R1
//%OPC RECOVER if error type T - actions R2
//%OPC RECOVER unconditionally - actions R3
```

This ensures that errors are handled by the RECOVER statement that is designed to handle it best.

When a matching statement is found, the recovery actions are controlled by the parameters on that statement. The RECOVER statement can specify these actions:

- Restart the current occurrence at the failed operation, with or without JCL changes.
- Restart the current occurrence at another operation.
- Add occurrences of special recovery applications. Make the restart of the failed occurrence dependent on the completion of the recovery occurrences. This action lets you, for example, perform a data set recovery before restarting your main application. See “Adding predecessor recovery occurrences to the current plan” on page 365.
- Release a dependent occurrence.
- Restart the current occurrence at the failed step or at another step, with any of these JCL modifications:
  - Delete steps
  - Add recovery steps
  - Change JCL statements in a program exit module
- Remain in error status.

The rule that controls how Tivoli Workload Scheduler for z/OS selects the failed step is described in “Deciding which step of an operation has failed” on page 364. For example, the error selection criteria if error E might match more than one failed step. In this case, Tivoli Workload Scheduler for z/OS selects the first step in the job that meets the criteria specified in the RECOVER statement. If this is not correct, you must change your RECOVER statements so that the correct step is chosen.

You can disable and enable the recovery function from the SERVICE FUNCTIONS panel. Also, you can set defaults so that the recovery function does not react to certain errors or reacts only at certain times unless specific requests are made.
Specifying recovery criteria

When a RECOVER statement is matched against an error condition, the statement is changed to a JCL comment. If the job is rerun, it no longer functions as a RECOVER statement. Other RECOVER statements in the JCL remain active.

Automatic recovery processing and data set cleanup

If a failing job operation has a cleanup action defined and it contains automatic recovery statements requiring a job restart, cleanup action is performed before automatic recovery. The process is the following:

1. The automatic recovery function detects the recovery statements and checks the type of cleanup.
2. If the cleanup type is set to None, the recovery actions are immediately performed. If the cleanup type is set to Immediate or Manual, the recovery actions are postponed.
3. For the immediate cleanup type, the cleanup actions are started automatically by the scheduler. For the manual cleanup type, you need to request the cleanup actions by selecting the Start Cleanup with AR option.
4. Automatic recovery processing waits until after Tivoli Workload Scheduler for z/OS finishes the data set cleanup. When the cleanup status is set to Complete, the recovery actions are performed.

   If the cleanup type is set to Automatic, no recovery action is taken and an error message is issued.

If you specify the other cleanup types, manual or automatic, the recovery action does not occur and an error message is issued.

If you defined RESSTEP as an automatic recovery statement in the JCL, Tivoli Workload Scheduler for z/OS performs the cleanup action according to the RESSTEP instructions. This means that the RESSTEP is considered as the starting step and the last step is considered as the ending step. If, after all the automatic recovery actions are performed, the operation fails again, a stand-alone cleanup action is started based on the value set in RCLOPTS IMMEDLOGIC:

- If IMMEDLOGIC(BESTSTEP), the action starts from the best step
- If IMMEDLOGIC(FIRSTSTEP), the action starts from the first step

For detailed information about the RCLOPTS statement, see the Customization and Tuning.

In some situations, you might decide to create a logic for job runs that bypasses the need for data set cleanup. You do not have to wait for the cleanup, because automatic recovery adds a new occurrence, which does not depend on the cleanup of the data set. When you specify in the AROPTS initialization statement the CHKRESTART(Y) parameter, the recovery actions are executed immediately whether or not data set cleanup is required.

The example in Figure 138 on page 350 shows how data set cleanup and automatic recovery can work together to ensure that a job is cleaned up and restarted.
Automatic recovery processing and data set cleanup

```c
// * OPC RECOVER ERRSTEP=STEPFAIL,JOBCODE=JCL,DELSSTEP=STEPFAIL
// * OPC SCAN
// *STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=EID.EID4R2.CATTEST,DISP=(CS,CS),
// UNIT=SYSDA,SPACE=(CYL,(1,1),RLSE),
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=12100)
//STEPFAIL EXEC PGM=IEFBR14
//DD3 DD DSN=DSN.NOT.CATLG,DISP=(OLD,DELETE,DELETE)
//STEP2 EXEC PGM=IEFBR14
//DD2 DD DSN=EID.EID4R2.CATTEST,DISP=(OLD,DELETE,DELETE)
//
```

Figure 138. Example of data set Cleanup with Automatic Job Recovery

In Figure 138, the job is designed to fail in STEPFAIL with a JCL error. Automatic recovery statements specify that when this error occurs, the step is to be deleted and the job rerun. Without data set cleanup, the job fails immediately in STEP1 with a JCL error: duplicate data set. When cleanup is active for the operation, the automatic recovery action is postponed until the cleanup action is completed or discarded.

For more information, see Chapter 17, “Restart and cleanup,” on page 313

Automatic recovery and restart

When automatic recovery (AR) actions require that an operation is rerun, consider that the following occurs. This is important to define the appropriate recovery statement (for example, using unique step names).

If the operation is a job

The operation is rerun by inserting the parameter RESTART=stepname in the JOB statement, where stepname is the value set in the RESSTEP parameter. When a job restart is requested, the RESTART keyword can be omitted.

If the operation is a started task

The operation is rerun by inserting the parameter COND=ONLY in all the steps that precede the restart step, unless a specific stepname is not identified (job restart).

Consider that the kind of tailoring required by the AR tasks (for example, add a step, delete a step, add the RESTART keyword, and so on) might cause the following limitations:

- Nested procedures
- INCLUDE statements
- IF/THEN/ELSE/ENDIF statements
- GDG data sets

Recovery of operations when a workstation becomes inactive

In the event of a system failure, the computer workstation representing that system can be automatically set to inactive if XCF is available. If XCF is not available, you can manually set the workstation to inactive using the panels. See “Specifying workstation destinations” on page 73. Operations that have not yet started on an inactive workstation can be rerouted to an alternate workstation. Refer to “Redirecting work to alternate workstations” on page 586 for more information about redirecting work to alternate workstations.
Recovery of operations when a workstation becomes inactive

The status Tivoli Workload Scheduler for z/OS assigns to operations that were started when the workstation became inactive is decided by the WSFAILURE and WSOFFLINE keywords of the JTOPTS statement. Refer to Customization and Tuning. If you decide to mark these operations as ended-in-error, they are given an error code of OSxx or OFxx (where xx is the status and extended status of the operation).

These operations are handled the same way as other operations that have ended in error. If a RECOVER statement has been defined to cover the situation, as defined by the job code and step code, it is invoked in the same way as for other operation failures.

The automatic-recovery-control statement

Each automatic-recovery-control statement describes an error situation and the recovery actions for it.

Use these rules to code RECOVER statements:

- Each statement must begin on a new 80-byte logical record.
  The symbols //**OPC must appear in bytes 1 to 7 and be followed by at least one blank.
  //**OPC RECOVER
    Identifies a RECOVER statement
  //**OPC
    Identifies a RECOVER continuation statement
  The automatic recovery function also inserts informational statements in the JCL:
  /// OPC
    Identifies a message statement
  ///=>OPC
    Identifies a comment statement
  See "Message and comment statements" on page 368 for more information.
- You cannot use variables anywhere in the RECOVER statement.
- The parameters are optional; you can code them in any sequence.
- Each parameter consists of a keyword followed by an equals sign and variable information.
- Parameters are separated by commas.
- You cannot code the same keyword more than once on the same statement.
- If you code only one parameter value, you do not need to enclose it in parentheses; for example, JOBCODE=PCHK.
- Bytes 72 to 80 are ignored by the automatic recovery function.
- When the total length of fields on a control statement exceeds 71 bytes, continue the statement using the following continuation conventions:
  1. Interrupt the field after a complete or partial parameter, including the comma that follows it, before byte 72.
  2. Code the identifying continuation characters //**OPC followed by at least one blank in bytes 1 to 7 of the statement that follows.
  3. Continue the interrupted operand in any position from bytes 9 to 16.
- When automatic recovery is related to non-z/OS trackers, limit the RECOVER statement to one card (avoid continuation).
RECOVER statement syntax

The syntax of the RECOVER statement follows. If you are unfamiliar with syntax diagrams, see "How to read the syntax diagrams" on page xxiv.

Statement parameters

All RECOVER statement parameters are optional. There are three RECOVER statement categories: selection, JCL rebuild, and recovery action. Tivoli Workload Scheduler for z/OS supports parameters that refer to job steps only for jobs on
host systems or jobs that run on fault-tolerant workstations and use a centralized script. For jobs that run on fault-tolerant workstations that do not use a centralized script, you must use the RECOVERY statement in the job definition to perform actions (refer to Customization and Tuning for instructions).

Selection parameters
These selection parameters specify the error situations that the RECOVER statement handles:

ERRSTEP
Restricts the RECOVER statement to be valid only for those steps specified.

Note: The automatic recovery function acts on information received about a job or started task failing step, but, because a job or started task might have several potential failing steps, it is the RETCODE keyword of the EWTROPTS statement that determines which of these steps should be passed to the automatic recovery function. If this failing step corresponds to a step specified by ERRSTEP, automatic recovery occurs.

Tivoli Workload Scheduler for z/OS supports this parameter only for jobs on host systems. This parameter is not supported when the EWTROPTS STEPEVENTS(ABEND) option is used.

JOBCODE
Restricts the RECOVER statement to be valid only for those job completion codes and return codes specified.

STEPCODE
Restricts the RECOVER statement to be valid only for those step return codes specified. Tivoli Workload Scheduler for z/OS supports this parameter only for jobs on host systems. This parameter is not supported when the EWTROPTS STEPEVENTS(ABEND) option is used.

TIME
Restricts the RECOVER statement to be valid only in the time range specified.

Note: Review the EXCLUDECC and EXCLUDERC keywords of the AROPTS statement, which specify codes for which no automatic recovery is done. For information about the AROPTS statement, refer to Customization and Tuning.

JCL rebuild parameters
These JCL rebuild parameters control the rebuilding of the JCL:

DELSTEP
Specifies a step or a list of steps that should be deleted from the inline JCL before rerunning the failed operation. You can also specify a range of step names to delete. Tivoli Workload Scheduler for z/OS supports this parameter only for jobs on host systems.

ADDPROC
Specifies the name, or a list of names, of JCL procedure library members to be added to the inline JCL before rerunning the failed operation.

RESSTEP
Specifies the name of the job or started-task step at which the operation should be restarted. Tivoli Workload Scheduler for z/OS supports this parameter only for jobs on host systems. If the operation specifies a data set cleanup action, Tivoli Workload Scheduler for z/OS performs the cleanup actions for all steps from the RESSTEP value up to and including the last step executed.
JCL rebuild parameters

Note: RESSTEP on the RECOVER statement is a different, and simpler, function than the step-restart function in the MODIFYING THE CURRENT PLAN panel. MCP step restart retrieves the job log and uses it to rebuild the JCL. RESSTEP does not require the job log, unless data set cleanup is involved.

CALLEXIT  Specifies the name of a program exit module that should be called before the restart.

Recovery action parameters
These recovery action parameters specify the actions to take for recovery:

RESTART  Specifies if the occurrence is to be restarted.

RESJOB  Specifies the name of the job or started task from which the occurrence must be rerun.

ADDAPPL  Specifies an application or a list of applications to be added as occurrences in the current plan.

RELSUCC  Specifies the application ID of a successor occurrence, or a list of IDs.

ALTWS  Specifies the name of an alternate workstation on which to run the operation.

ALTJOB  Specifies an alternate job or started-task name to use when the job is restarted. This is used in a MAS complex to allow the restart of a job that has not yet ended as far as JES is concerned.

This action parameter does not apply to operations that are run on z-centric agent workstations.

Selection parameters
When a job or started task ends in error, this information is available for all executed steps:
• Step name
• Abend code, if step abended
• Return code, if step did not abend

The selection parameters use this information:

ERRSTEP  If the recovery specification is for a specific step within a JCL procedure, specify ERRSTEP=stepname.procedurestepname.

If the recovery specification is for specific step within standard JCL, specify ERRSTEP=stepname.

Each step name you specify must correspond to a step in the job or started task. All steps in the JCL should have a name; these names should be unique.

Default: The recovery specification is for all steps.

JOBCODE  The code can be an abend code, an error code set by Tivoli Workload Scheduler for z/OS or JCC, a case code, a return code, or a return code range. (For more information on case codes, see "Case code lists" on page 369.) The values are those given as the error code on the HANDLING OPERATIONS ENDED IN ERROR panel in the MODIFYING THE CURRENT PLAN panel.

The JOBCODE keyword values are:
Selection parameters

Sxxx  Specifies a system abend code.
Uxxx  Specifies a user abend code.
xxxx  Specifies a case code or an error code set by Tivoli Workload Scheduler for z/OS, either directly or by using the job completion checker.
n    Specifies a return code.
x-y  Specifies a return code range, where x and y represent positive decimal values.

The codes specified in the JOBCODE parameter are tested against the return code of the first abending step or against the job code as set by Tivoli Workload Scheduler for z/OS when the job or started task ends. This step is called the terminating step.

Note: When specifying the JOBCODE keyword values, you must use the formats of the system abend code, user abend code, and job return code described in Chapter 16, “Setting error codes,” on page 309.

You can also specify abend codes in generic form. Therefore, an asterisk (*) can represent any character, or any group of characters, in those positions where it is placed. A code can contain more than one asterisk only if each asterisk is separated from the next by another character.

Note: The JOBCODE=* notation covers all possible user and system abend codes; it does not cover return codes. To cover all possible return codes, specify a return code range in the form:

   JOBCODE=x-y

   where x and y represent positive decimal values.

A return code cannot be greater than 4095. To specify a range of values, set x to the lower value and y to the higher value. To specify all return codes greater than or equal to a certain value, set x to that value and y to 4095.

If you specify the ERRSTEP parameter as well as JOBCODE, the step that terminated the job or started task must be specified by the ERRSTEP parameter; otherwise, the RECOVER statement criteria are not met, and the statement is ignored.

Default: The default depends on whether the STEPCODE parameter is specified:

- STEPCODE specified: the JOBCODE default is no recovery, and the recovery is controlled by the STEPCODE parameter.
- STEPCODE not specified: the JOBCODE default is recovery for all codes except those for which no automatic recovery is specified by the EXCLUDECC and EXCLUDERC keywords of the AROPTS statement. Refer to Customization and Tuning for details.

STEPCODE  The STEPCODE keyword value can be:
Sxxx  Specifies a system abend code.
Selection parameters

Uxxx  Specifies a user abend code.
xxxx  Specifies a case code.
  n     Specifies a return code.
x-y  Specifies a return code range, where x and y represent positive decimal values.

'FLSH'  Specifies that the step was flushed.

The return codes of the executed steps are tested against the parameter values and, if there is a match, the RECOVER statement is eligible for processing. The STEPCODE parameter can be used to test the result of all executed steps, whereas JOBCODE tests only the result of the terminating step.

Note: When specifying the STEPCODE keyword values, you must use the formats of the system abend code, user abend code, and job return code described in Chapter 16, "Setting error codes," on page 309.

Abend codes can also be specified in generic form. Therefore, an asterisk (*) can represent any character, or any group of characters, in those positions where it is placed. Only the first character of the abend code must be necessarily specified, any asterisk must follow that character. A code can contain more than one asterisk only if each one is separated by another character.

If ERRSTEP is specified, the STEPCODE values are tested only against the return codes of the steps specified in the ERRSTEP parameter. If no ERRSTEP parameter is specified, the STEPCODE value is tested against the return codes of all executed and flushed steps.

A return code cannot be greater than 4095. To specify all return codes greater than or equal to a certain value, set x to that value and y to 4095.

When using this parameter, all steps in the JCL should have a name; these names should be unique within the job or started task.

Tivoli Workload Scheduler for z/OS treats the first step that abended as the terminating one. The JOBCODE values are tested against this step. With COND=EVEN specified on the EXEC statement, steps that follow an abending step might still be executed. So you can use the STEPCODE parameter to test for an abend in such a step.

Default: Return codes from steps other than the terminating one do not cause automatic recovery. The return code from the terminating step is handled as described for the JOBCODE parameter.

TIME  The time is specified in the form hhmm, where hh is the hour from 00 to 24, and mm is the minute from 00 to 60. This is the time when the recovery is automatic.

For example:

TIME=0700-1600  No recovery actions occurs between 4 p.m. and 7 a.m.
Selection parameters

TIME=2200-0800
Tivoli Workload Scheduler for z/OS does automatic recovery only between 22.00 and 8.00.

TIME=0000-2400
Recovery can be automatic at any time.

TIME=0000-0000
Recovery is not started unless there is manual intervention.

The recovery actions for a job or started task that remains in the ended-in-error list can be manually started up later. (See “Recovery actions from the Modifying Current Plan panel” on page 367.) Such requests override any TIME value specified.

Default: The recovery specification is for the time range specified by the STARTIME and ENDTIME keywords of the AROPTS automatic recovery options statement. Refer to Customization and Tuning for more details.

JCL rebuild parameters

When a recovery statement is selected, Tivoli Workload Scheduler for z/OS uses these parameters to build the JCL stream that is used to restart the operation.

DELSTEP
DELSTEP specifies job or started-task steps to be deleted if the RECOVER statement is activated. Only one step of each name is deleted, so ensure that all your step names are unique.

Notes:
1. Steps within in-stream procedures and started tasks cannot be deleted.
2. IF-THEN-ELSE-ENDIF construct statements are recognized.
   IF-THEN-ELSE-ENDIF statements are deleted when a sequence of delete lines, one or more steps as specified by DELSTEP, contains both the IF-THEN statement and the matching ENDIF statement, and possibly an ELSE statement.
   IF-THEN-ELSE-ENDIF statements which have their counterparts outside the delete sequence are kept.
   The delete sequence begins with an EXEC statement and continues to the line immediately before the EXEC statement of the next step to be retained in the JCL record. However, IF-THEN-ELSE-ENDIF statements immediately preceding this delete sequence are taken into account. That is, statements separated from the delete sequence by comment statements or subsequent IF-THEN-ELSE-ENDIF statements. In this sequence of lines preceding the delete sequence IF-THEN statements are converted to comment lines if the delete sequence contains the matching ELSE and/or ENDIF statements.
   In the delete sequence IF-THEN-ELSE-ENDIF statements are retained if the matching statement is not within the immediately preceding lines or not in the delete sequence. If an ELSE and the matching ENDIF is found but the corresponding IF-THEN statement is not, then only the ELSE statement is deleted, or converted to a comment statement. The ENDIF is retained for the IF statement in a preceding section of the JCL.
To summarize the IF-THEN-ELSE-ENDIF handling:
JCL rebuild parameters

- IF-THEN statements might be changed to comment statements by inserting *> in columns 3 and 4. This is done when the IF-ENDIF is in front of a sequence of JCL lines to be deleted and the matching ENDIF is within the delete sequence.

- Statements are left in the JCL when required because the matching statements are not available in the sequence of JCL lines to be deleted.

There is no check that the result is a valid job. The result is not valid if all steps are deleted between an IF-THEN statement and a corresponding ELSE statement. The result is also not valid if an IF-THEN statement remaining in the JCL references the name of a step to be deleted.

Default: No steps are deleted.

ADDPROC
This parameter specifies the names of JCL procedures to include in the JCL if recovery occurs. The JCL procedure library members are in the procedure library with ddname EQQPRLIB.

**Note:** This is not a JES procedure library but an internal procedure library specific to Tivoli Workload Scheduler for z/OS.

The added JCL is placed after the RECOVER statements in the JCL in the order the names appear in the ADDPROC parameter. This makes it necessary to have the RECOVER statements after in-stream procedures in the JCL file.

A useful technique is to have any procedures you want to add included in the JCL. The JCL is then invoked by including an EXEC statement, which calls the procedure. The EXEC statement itself is within a procedure invoked by ADDPROC. This reduces the risk of calling a procedure that is not available in EQQPRLIB.

**Note:** If you include an in-stream procedure, remember that it must always begin with a PROC statement and end with a PEND statement.

Default: No JCL is added.

RESSTEP
For a job, this specifies a value for the RESTART parameter on the JOB statement of the failing job.

For a started task, a COND=ONLY parameter is added to all EXEC statements that precede the specified step. If a COND parameter already exists in a step, it is commented out before the COND=ONLY is added.

The value of the RESSTEP parameter is enclosed in parentheses and is used either as the RESTART parameter or as the first step that COND=ONLY is not added to. If the JOB statement already contains a RESTART value, it is replaced by the RESSTEP value. However, if RESTART=Y is not specified in the recovery statement, RESSTEP is not active, meaning that the job is not rerun. If the restart step name is within a JCL procedure, specify `stepname.proctype`.name.

* (asterisk)  Specifies that the job or started task should be restarted at the first step (possibly a step of a cataloged procedure).
JCL rebuild parameters

% (percent) Specifies that the job or started task should be restarted at the failing step. “Deciding which step of an operation has failed” on page 364 describes which step is selected if more than one step has failed, and also if the error cannot be associated with a particular step.

If % is specified, make sure that all steps in the in-stream JCL have a unique name. Do not specify % for a started task if the JCL file includes procedure calls.

A checkpoint ID can also be specified. The SYSCHK DD statement must then be present in the step JCL. The checkpoint name must not contain special characters, such as commas, blanks, or parentheses.

Default: The JOB statement is unchanged.

CALLEXIT The CALLEXIT parameter specifies an exit routine to be called if this RECOVER statement is activated. The exit is called for each JCL line, and the exit can decide to accept the line without any changes, modify it, insert one or several JCL lines, or delete it. The exit also has the option of stopping the restart and recovery of the failing job or started task.

For more details, refer to Customization and Tuning.

Default: No exit is called.

Action parameters

These parameters specify the action that Tivoli Workload Scheduler for z/OS should take when the recovery statement is invoked.

RESTART RESTART=Y causes the job or started task to be rerun, either from the failing operation or, if you specified RESJOB, from an earlier operation within the occurrence.

RESTART=N prevents the restart. It can be used with ADDAPPL when the recovery actions are to be done by a separate application. It can also be used to select cases for which no recovery should be performed or when testing the recovery procedure. The operation remains in ended-in-error status. JCL rebuild and other requested actions are performed, however. This includes also the cleanup actions that are performed according to the RESSTEP specified. If no value is specified for RESSTEP, the default (Restart from begin) is used and no JOB card tailoring is requested.

Default: RESTART=Y.

RESJOB The RESJOB parameter handles failures at an occurrence level, not within the failing job or started task itself.

The occurrence of the application is rerun from the first preceding computer workstation operation, whose name matches the job name specified in the RESJOB parameter. If the job name specified cannot be found in a computer operation preceding the failed operation in the same occurrence, no automatic recovery occurs and the job or started task remains in the error handling list with an extended status code indicating an automatic recovery error.
Action parameters

The indicated operation must be a predecessor to the failed operation or be the failed operation itself.

**Note:** External successors cannot be handled automatically. Therefore the set of operations selected for rerun that can be completed at failure time must not have any external successors.

The rerun operation must be defined on an automatically reporting computer workstation.

The parameter is ignored if RESTART=N is specified.

Default: Rerun from the failed operation.

**ADDAPPL**

Here you specify a list of applications. Occurrences of these applications are added to the current plan if this RECOVER statement is invoked.

- If RESTART=N is specified, the applications are independent of the failed occurrence.
- If RESTART=Y is specified, the recovery applications are added to the current plan as predecessors to the failing operation or, if RESJOB is specified, to the operation where restart is being attempted. How a predecessor operation is selected is described in “Adding predecessor recovery occurrences to the current plan” on page 365.

Added applications are independent of each other. A maximum of 40 application occurrences can be added.

For example, suppose a database update fails. A rerun of the failed job is necessary but must be postponed to a later time. However, a database restore job must be run to repair the database before the online users require the database. This recovery situation can be specified as:

```plaintext
//+%OPC RECOVER JOBCODE=SCHK,RESTART=N,ADDAPPL=A301RORG
```

This means that, on case code SCHK, do not rerun the failing operation. (For information on case codes, see “Case code lists” on page 369.) However, add an occurrence of the application A301RORG to the current plan, without a dependency to the failing operation and leave the failing operation in ended-in-error status.

Default: No application occurrences are added.

**Notes:**

1. When automatic recovery adds an occurrence to the current plan, input arrival and deadline times are not taken from the application description. Instead the occurrence is given an input arrival of the time the add is run, according to the time on the z/OS system where the controller is started and the deadline is set for 8 hours after input arrival. If an occurrence of that application already exists with this input arrival time, then one minute is added to the time until a time is reached when the occurrence can be included. If the added occurrence includes time-dependent operations with specific input arrival times, then the operations are started at the specified time.
2. An occurrence that has been added to the current plan by automatic job recovery does not become the predecessor to an occurrence that is added later by daily planning, even if normal dependency criteria are met.

**RELSUCC** This parameter specifies which external successors to the failing operation are allowed to run even if their predecessor operation has ended in error.

The external successors to the failing operation are checked, and the dependencies between the failed operation and the specified successors are deleted at recovery time.

The effect is that this predecessor (the failed operation) is reported as complete to the external successor, and the successor-predecessor chaining is removed. The external successor becomes ready if its other predecessors are completed. The dependency does not exist when the failed occurrence is rerun.

Even if one successor is released, other successors might be waiting for the failed occurrence to complete. These might be successors not yet in the current plan. Assume that W is a weekly application and D is a daily application that is dependent on W. If W fails and there is a RECOVER statement causing the release of that day’s D, the occurrence of D the next day also waits for W to complete, but without any automatic release.

You can specify a maximum of 40 application IDs.

Default: None.

**ALTWS** Specifies the name of an alternate workstation on which to run the operation. The ALTWS parameter overrides the alternate workstation defined in the workstation description. You can use this parameter, for example, with the TIME parameter to specify alternate workstations for an operation, depending on the time of day. This parameter is not supported for automatic recovery of jobs using a centralized script.

Default: None.

**ALTJOB** Specifies an alternate job or started-task name to use when the job is restarted. The name applies only to this particular occurrence. This parameter is not supported for automatic recovery of jobs using a centralized script.

Default: The job or started-task name is not changed.
Recovery coding examples

This section contains five examples of JCL for automatic job or started-task recovery.

Example 1

```jcl
//A103001 JOB ....
//SALARY PROC GENER=+1
//S190 EXEC PGM=PAX
//POS DD DSN=WATSON,DISP=SHR
//PRELPOS DD DSN=T191(&GENER),DISP=(NEW,CATLG,DELETE)
//CARDS DD DSN=T830(&GENER),DISP=OLD
//STAT DD DSN=T192(&GENER),DISP=(NEW,CATLG,DELETE)
//PEND
//%OPC RECOVER ERRSTEP=SAX91,JOBCODE=SALR,TIME=0000-2400,
//*%OPC DELSTEP=SAX80-SAX91,ADDPROC=SALRECOV
//*%OPC RECOVER ERRSTEP=SAX92... (for steps after SAX91)
//SAX80 EXEC PGM=IEBISAM
//SAX91 EXEC SALARY
//P194237A JOB ....
//*%OPC RECOVER JOBCODE=S*37,ADDPROC=SPACECHG
//*%OPC RECOVER JOBCODE=(*,16-4095),RESTART=N,ADDAPPL=REORG
//*%OPC RECOVER JOBCODE=12
//STEP01 EXEC PGM=ATTACH8A,REGION=256K
//SYSOUT DD SYSOUT=A
//TSTIN DD DSN=T194237.IN.DATA,DISP=SHR
//TESTWK1 DD UNIT=3380,SPACE=(CYL,(1,1))
//TESTWK2 DD UNIT=3380,SPACE=(CYL,(1,1))
//TESTWK3 DD UNIT=3380,SPACE=(CYL,(1,1))
//TSOUT DD DSN=T194237.OUT.DATA,DISP=SHR
```

In example 1:

The RECOVER statement specifies that, if there is an error in step SAX91 with an error code in the SALR case code list, all steps from SAX80 up to and including SAX91 should be deleted. (For more information on case codes, see "Case code lists" on page 369.) They are replaced by the member SALRECOV from the Tivoli Workload Scheduler for z/OS procedure library. SALRECOV contains RECOVER statements for the steps after the failed one and an EXEC statement to run the internal procedure, now with a different value on the symbolic parameter to modify the failed step.

Example 2

```jcl
//A103001 JOB ....
//SALARY PROC GENER=+1
//S190 EXEC PGM=PAX
//POS DD DSN=WATSON,DISP=SHR
//PRELPOS DD DSN=T191(&GENER),DISP=(NEW,CATLG,DELETE)
//CARDS DD DSN=T830(&GENER),DISP=OLD
//STAT DD DSN=T192(&GENER),DISP=(NEW,CATLG,DELETE)
//PEND
//%OPC RECOVER ERRSTEP=SAX91,JOBCODE=SALR,TIME=0000-2400,
//*%OPC DELSTEP=SAX80-SAX91,ADDPROC=SALRECOV
//*%OPC RECOVER ERRSTEP=SAX92... (for steps after SAX91)
//SAX80 EXEC PGM=IEBISAM
//SAX91 EXEC SALARY
```

In example 2:
The first RECOVER statement specifies that, for space problems in any of the steps, the member SPACECHG should be added and the failed job should be restarted (rerun the failed Tivoli Workload Scheduler for z/OS operation).

The second RECOVER statement specifies that, for any other error code (not in the EXCLUDECC list) and return code 16 or higher, the failed job (Tivoli Workload Scheduler for z/OS operation) should not be restarted, but the application called REORG should be started.

The third RECOVER statement specifies that, for return code 12, restart should occur at the failed job (Tivoli Workload Scheduler for z/OS operation).

The preceding actions apply to the time range specified by the STARTTIME and ENDTIME parameters of the AROPTS automatic recovery initialization statement. If the job fails at another time, it gets the ended-in-error status and remain on the ended-in-error list. The scan for recovery statements can be repeated at a later time; see “Recovery actions from the Modifying Current Plan panel” on page 367.

Example 3

```c
//AP301A66 JOB ....
//**OPC RECOVER JOBCODE=NRC4,RESTART=N
//**OPC RECOVER JOBCODE=(U046,S*37),ERRSTEP=STEP01,
//**OPC ADDPROC=(A66RECOV,SPACECHG)
//**OPC RECOVER JOBCODE=S*37,ERRSTEP=STEP02,
//**OPC ADDAPPL=AP301ARA,RESSTEP=%
//**OPC RECOVER ERRSTEP=STEP02
//STEP01 EXEC PGM=SORT
//SYSOUT DD SYSOUT=A
//SORTIN DD DSN=AP301A.SALES.DATA,DISP=OLD,UNIT=3400-6,
// LABEL=(1,SL)
//SORTOUT DD DSN=&sortout,DISP=(NEW,PASS),UNIT=SYSDA,
// SPACE=(CYL,(5,1))
//SORTWK01 DD UNIT=3380,SPACE=(CYL,(1,1))
//SORTWK02 DD UNIT=3380,SPACE=(CYL,(1,1))
//SORTWK03 DD UNIT=3380,SPACE=(CYL,(1,1))
//SYSIN DD *
// SORT FIELDS=(40,36,CH,A)
// RECORD TYPE=F,LENGTH=80
END
//STEP02 EXEC PGM=REPORT
//SYSPRINT DD SYSOUT=A
//REPORTIN DD DSN=&sortout,DISP=(OLD,DELETE)
//XXXX DD DSN=&XXX,DISP=NEW,UNIT=SYSDA,SPACE=(CYL,(5,1))
```

In example 3:
The first RECOVER statement specifies that, for error code NRC4 (which is a case code set by the job completion checker), no recovery actions should be taken. Assume that NRC4 represents system completion codes such as 0C4 for which no automatic recovery should be done.

The second RECOVER statement specifies that, for space problems in step STEP01, JCL from the Tivoli Workload Scheduler for z/OS procedure library (which might be a copy of the RECOVER statements for step STEP02 and a space allocation step) should be included, and the failing job (Tivoli Workload Scheduler for z/OS operation) should be restarted from the beginning.

The third RECOVER statement specifies that, if there are space problems in step STEP02, a new application named AP301ARA is to be added, and the failing job (Tivoli Workload Scheduler for z/OS operation) must be restarted at step STEP02. The failing job will be a successor operation to AP301ARA and will be started when AP301ARA is completed.
Recovery coding examples

The fourth RECOVER statement specifies that, for STEPO2, all the remaining error codes not in the EXCLUDECC list should be restarted from the beginning without any JCL changes or other recovery actions.

If the error occurs outside the time range specified by the STARTTIME and ENDTIME parameters of the AROPTS automatic recovery initialization statement, the job remains in ended-in-error status.

Example 4

```plaintext
//FSF007 JOB ...
//%OPC RECOVER ERRSTEP=UPDATE,ADDAPPL=RECOV1,
//%OPC RESTART=N,TIME=0800-1600
//%OPC RECOVER ERRSTEP=UPDATE,ADDAPPL=RECOV1,TIME=0000-2400
//%OPC RECOVER TIME=0000-2400
// EXEC ...
.
.
.
```

In example 4:
The application RECOV1 is added if an error occurs in step UPDATE. During normal working hours, the failed operation remains on the ended-in-error list. At other hours, the operation is restarted from the beginning. For errors in other steps, the operation is restarted from the beginning. In all three cases, an error code in the INCLUDECC list means no recovery.

Example 5

```plaintext
//TI94237A JOB ....
//%OPC RECOVER JOBCODE=S*37,ADDPROC=SPACECHG,
//%OPC RESTART=N
//%OPC RECOVER JOBCODE=(+,16-4095),RESTART=N,ADDPPL=REORG
//%OPC RECOVER JOBCODE=8,
//%OPC RESTART=N
//STEP01 EXEC PGM=ATTACH8A,REGION=256K
//SYSOUT DD SYSOUT=A
//TSTIN DD DSN=TI94237.IN.DATA,DISP=SHR
//TESTWK1 DD UNIT=3380,SPACE=(CYL,(1,1))
//TESTWK2 DD UNIT=3380,SPACE=(CYL,(1,1))
//TESTWK3 DD UNIT=3380,SPACE=(CYL,(1,1))
//TSTOUT DD DSN=TI94237.OUT.DATA,DISP=SHR
```

In example 5:
This is the same as in example 2, except that this time it is set up for testing the recovery procedure. If the job fails, it has ended-in-error status independent of the error type. The JCL can be inspected and the recovery actions can be started manually via the MODIFYING THE CURRENT PLAN panel.

Deciding which step of an operation has failed

More than one step might meet the criteria set by the selection parameters. If you specify the STEPCODE parameter but not the ERRSTEP parameter, more than one step can have a return code that would cause recovery. If both the STEPCODE and JOBCODE parameters are given but not the ERRSTEP parameter, two or more
Deciding which step of an operation has failed

steps might meet the criteria, one as determined by the JOBCODE, and the others as determined by the STEPCODE. In these cases, the first step that fulfills the selection criteria is selected as the failed step for recovery purposes; for example, when determining the restart step.

A failure can also occur at job or started-task initialization time before a step is entered. Such failures have the error code JCLI or CAN. Assume that a RECOVER statement is specified that has RESSTEP=,% and no ERRSTEP parameter, indicating restart at failing step for any errors in the job. For JCLI and CAN errors, the restart is from the start of the job.

When a job or started task is transferred from one workstation to another and recovery occurs, Tivoli Workload Scheduler for z/OS assumes that the step that was executing at the time of failure was the step that would have followed the last step that Tivoli Workload Scheduler for z/OS knows completed (that is, for which it has a step-end event).

JCCE errors mean that the job completion checker (JCC) has terminated its processing of the SYSOUT file. Such errors are not associated with a specific step. If RESSTEP=,% is specified but the step return codes show that the executed steps have completed successfully, the job or started task remains on the ended-in-error list. The same is true for other error codes set by JCC.

Adding predecessor recovery occurrences to the current plan

When application occurrences are added to the current plan, as the result of the ADDAPPL automatic recovery statement, and RESTART=Y is specified for the failed occurrence, the failed occurrence is made a successor of the newly added application occurrences. This is to allow for the common situation in which, for example, a data set restore must take place before a job can be rerun. The rerun is dependent on the successful completion of the restore job.

Because external dependency relationships are between particular operations within occurrences, Tivoli Workload Scheduler for z/OS must work out which operation in the successor occurrence depends on which operation in the predecessor occurrence. The operation Tivoli Workload Scheduler for z/OS designates (in the failed occurrence) as the restart operation is either:

- The failing operation itself
- The operation given by the RESJOB parameter, if RESJOB is specified

The predecessor operation, in the application occurrences added by recovery, is chosen using the following rules:

1. If the failed occurrence has a defined predecessor in the application added by recovery, this predefined predecessor-successor linkage is maintained.
2. If no such dependency is defined, the default predecessor workstation is used to select the predecessor operation. The default predecessor workstation is defined in the PREDWS keyword of the AROPTS automatic recovery initialization statement (refer to Customization and Tuning). The last operation in the application on that workstation is selected as the predecessor operation.
3. If no dependency is defined and no PREDWS name is defined, the last operation, is selected as the predecessor operation.

The last operation mentioned above is the last operation the user specifies when creating the application description, or when the application description was last updated via panels or the PIF.
Adding predecessor recovery occurrences to the current plan

However, it is recommended that the application description of the failing occurrence has a predecessor defined to the recovery application for the potential restart operation.

When you are defining applications, consider setting up any recovery applications that might be required. If you do not define run cycles for the recovery applications, the required recovery occurrences are included in the current plan only when the error condition is matched and the recovery action is invoked.

You require a recovery application for any operation in your main application that might cause a RESTART, and for which some “cleaning up” must be done before a rerun.

Status changes

Automatic recovery causes a change in the status and extended status of the operations that are recovered.

When a job or started task that is controlled by Tivoli Workload Scheduler for z/OS ends abnormally, the corresponding operation appears on the ended-in-error list. The job or started task remains on the ended-in-error list in the following cases:

- There are no recovery specifications for the error.
- RESTART=N is specified in the RECOVER statement.
- An error occurred in the automatic recovery processing.
- Automatic recovery is inactive. You can specify this in the AROPTS initialization statement or using the panels.

In all three cases, the status remains E.

Status

The automatic recovery function gets control for an operation when the operation is set to status E (ended-in-error). If RESTART=Y is specified in the recovery statement that is invoked, an automatic restart is performed, and the status changes. This is the same procedure that would be followed if the operation were rerun using the MODIFYING THE CURRENT PLAN panel. This means that the operation that is selected as the restart point is set to status R, and succeeding operations are set to status W.

Note: This is not the case if you specify the RESTARTABLE attribute when you define an operation. In this case, the operation is replaced on the ready queue of its workstation with status code R.

Extended status

The extended status is also set by the automatic recovery function. Extended status E shows that an error was detected while attempting automatic recovery; extended status R means that automatic recovery has been performed and RESTART was requested.

If no RECOVER statement is invoked for the error, the extended status field is left blank.

If RESTART=N is specified, the extended status field is set to R to show that recovery has taken place.
Extended status

If an error is detected while processing a RECOVER statement, the extended status field is set to E.

Security in automatic recovery

The current plan is protected from unauthorized update attempts. The level of protection is controlled by use of the AROPTS and AUTHDEF initialization statements. Refer to Customization and Tuning for details. Certain automatic recovery functions, such as the ADDAPPL and RELSUCC functions, perform updates to the current plan as part of the recovery procedure.

There are two parameters in the AROPTS statement that you can use to control the authority checking. The USERREQ keyword specifies whether it is possible for authority to be granted if the selected USERID is not known or if no USERID information is available. Using the AUTHUSER keyword you can specify the source from which Tivoli Workload Scheduler for z/OS should determine the USERID. You can specify:

- The last user to update the JCL
- The authority group of the failing occurrence
- The owner ID of the failing occurrence

When you specify JCLUSER, the last user who updated the JCL is assumed to be responsible for the JCL including its RECOVER statements. Tivoli Workload Scheduler for z/OS checks that this user has the level of authority required to perform the updates requested by the RECOVER statements.

Tivoli Workload Scheduler for z/OS finds the user ID of the last user to update the JCL from one of three sources, depending on where the JCL is obtained:

- When the JCL is obtained from the EQQJBLIB data set, the ID of the last user to update the JCL is retrieved from the ISPF statistics in the directory entry for the JCL member. If there is no user ID recorded in the statistics, no authorization checking can be performed.
- When the JCL is entered from the EQQUX002 exit, the updating user ID is passed as a parameter to Tivoli Workload Scheduler for z/OS so that authorization checking can be performed.
- When the JCL is updated via a panel, the ID of the last user who updated the JCL is stored. This is then used for authorization checking.

Recovery actions from the Modifying Current Plan panel

A job or started task that ends in error is automatically scanned for a RECOVER statement that corresponds to the reported error.

It is possible to invoke this recovery scan manually from the HANDLING OPERATIONS ENDED IN ERROR panel in the MODIFYING THE CURRENT PLAN panel. Selecting an operation with the row command ARC starts the scan.

If you invoke recovery from the MODIFYING THE CURRENT PLAN panel, any time restrictions on the RECOVER statements are overridden. That is, recovery occurs immediately, regardless of the TIME parameter that is specified on the RECOVER statement. The RECOVER statement that is invoked matches the original error condition of the operation, as would be the case if the scan were invoked automatically.
Recovery actions from the Modifying Current Plan panel

It is also possible to specify a RESTART=N parameter on a RECOVER statement and then cause a RECOVER statement scan to take place. This lets you see the effect that invoking the RECOVER statement would have, without causing the operation to be resubmitted. The JCL created in this way will be the JCL that Tivoli Workload Scheduler for z/OS uses if the operation is rerun. To get back to the original JCL, you must manually edit the JCL to undo the effects of the recovery statement. This includes amending the RECOVER statement because it will have been automatically changed to a comment statement. See "Message and comment statements." If you want the original job from EQQJBLIB, an easier method is to delete all the job statements and press PF3 (End) to save, forcing Tivoli Workload Scheduler for z/OS to take a fresh copy from the library.

Message and comment statements

Automatic recovery can insert message and comment statements in your JCL.

Message statement

The automatic recovery function updates the JCL of the failing operation with information in the form of message statements about the recovery action taken. Also, if an error occurs during the automatic recovery function, Tivoli Workload Scheduler for z/OS uses message statements to insert an error description in the JCL file saved on the JS data set. The format of the message statement is:

```//* OPC message-text```

Comment statement

In addition to making any JCL modifications requested in the RECOVER statement and adding message statements, the selected RECOVER statement is changed to a comment statement. Therefore, the RECOVER statement is treated as such only the first time. You could, however, add the RECOVER statement again using the ADDPROC parameter.

Note: An automatic recovery statement is commented only when the statement has matched a reported error, and the recovery actions have been initiated successfully. Statements that do not match the error remain in the JCL as candidates for subsequent failures.

Logging and failure reporting

Automatic recovery actions are logged in three places:

- On the job-tracking log, for Tivoli Workload Scheduler for z/OS restart purposes
- On the Tivoli Workload Scheduler for z/OS message log
- In the Tivoli Workload Scheduler for z/OS JS data set (in the JCL record of the failed job or started task) in the form of message statements

Successful recovery start

The successful start of recovery actions is recorded in all three places, as described previously. Message statements are added to the JCL to let you reconstruct the recovery activities performed. These messages are usually inserted after the RECOVER statement that they relate to, telling you that recovery has been performed using a preceding RECOVER statement. In addition, all selected RECOVER statements are changed to comment statements.
Unsuccessful recovery start

If an error is detected in a RECOVER statement, messages describing the error are inserted in the JCL, and the incorrect statement (and possibly the position within the statement where the error was detected) is identified. The position is identified by the character A in the inserted message, as indicated in the following example:

```
//*>OPC SCAN
//SAMPLEA JOB (885002,NODB),SAMPLE,NOTIFY=XMAWS,MSGCLASS=Q,
// CLASS=B,MSGLEVEL=(1,1),PRTY=1
//*%OPC RECOVER JOBCODE=JCL,RESTART=N,ADAPPL=RECOV
//* OPC MSG: A
//* OPC MSG: E INCORRECT PARAMETER
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=XMAWS.NOT.THHERE,DISP=(OLD,DELETE,DELETE)
//
```

If an exit requests termination of the recovery attempt or if it requests saving of the JCL but no restart, this request is also logged in the JCL. Message statements are added to the JCL close to the RECOVER statement.

If an error cannot be associated with a particular JCL record, message statements are added at the start of the JCL.

Note: RECOVER statements are checked only when a job fails.

Case code lists

You can group return codes, Tivoli Workload Scheduler for z/OS job codes, and abend codes for the purposes of recovery. Each group is assigned a name, called a case code. In the RECOVER statement, you can specify the recovery action for a number of abend situations, using one RECOVER statement and the case codes. The case code definition list is kept in a load module, EQQCASEM. A macro, EQQCASEC, is available to help you set up the case code lists.

With automatic recovery, the case codes are used in the following way: Tivoli Workload Scheduler for z/OS tries to match the error code from the failing operation with a value specified in the JOBCLASS and STEPCODE parameter. If no direct match is found and a case code list exists, Tivoli Workload Scheduler for z/OS assumes that the codes defined in the JOBCLASS and STEPCODE parameters are case codes. It searches the case code list for each parameter value. If it finds a match, implying that the value in the parameter is a true case code, it then seeks to match the error code of the failing operation with the list of return and abend codes associated with the case code. If a match is found and the other RECOVER criteria are met, recovery occurs.

For a description of how you define case codes, refer to Customization and Tuning.

Deactivating and activating automatic recovery

Automatic recovery is performed by a subtask of the Tivoli Workload Scheduler for z/OS subsystem. You request that this subtask should be started by specifying RECOVERY(YES) on the OPCOPTS initialization statement. If this is specified, automatic recovery is started and stopped when the Tivoli Workload Scheduler for z/OS subsystem is started and stopped.

Automatic recovery can be deactivated and activated using the SERVICE FUNCTIONS option from the main menu. When automatic recovery is deactivated,
Deactivating and activating automatic recovery

The ended-in-error jobs or started tasks remain on the ended-in-error list, independent of RECOVER statements. Requests from the MODIFYING THE CURRENT PLAN panel HANDLING OPERATIONS ENDED IN ERROR panel are, however, honored, regardless of whether automatic recovery is activated or deactivated. See "Recovery actions from the Modifying Current Plan panel" on page 367.

Restarting work on different workstations

When jobs or started tasks are restarted on workstations other than the workstation that they were initially defined on, you should consider:

- Data set availability. All the data sets required by the job or started task should be available on the system associated with the alternate workstation.
- The RACF environment on the system associated with the alternate workstation.
- The initiator structure on the alternate system.

The following points apply to automatic recovery in general:

- Jobs and started tasks should be made (as far as possible) restartable from the failed step. A major problem is the handling of work files conveying information from one step to another. One way of dealing with this is:
  1. At the beginning of the job or started task, add a special step referring to all work files (IEFBR14) with DISP=(OLD,DELETE,DELETE).
  2. When you create the file, code DISP=(NEW,CATLG,DELETE).
  3. When you receive the file and you must pass it to the next step, code DISP=(OLD,PASS,KEEP).
  4. At the end of the job or started task, add an extra step executing IEFBR14 with a DD statement for each work file specifying DISP=(OLD,DELETE,KEEP).

- Files passed across job or started-task steps, and between jobs or started tasks, must be permanent or intermediate files.
- DISP=MOD should be used with care, because it can cause problems with restart.
- It is better to have small, uncomplicated jobs or started tasks with few steps rather than large, complicated jobs or started tasks with many steps. A complicated process should be broken into several smaller jobs or started tasks.
- Cataloging should be performed in a separate job. This is especially important when using generation data group (GDG) data sets. Usually, a job runs with input as generation 0 and output as generation +1. On rerun, input should be referenced as generation -1 and output as generation 0, which would require JCL changes. An alternative method is to catalog the new generation data group in a previous job, as shown in the following example:

```plaintext
//A105C01 JOB ....
//STEP01 EXEC PGM=IEFBR14
//A105CTLG DD DSN=A105.INVOICE.BASE(+1),
// UNIT=DISK,
// SPACE=(CYL,(5,1))
//* DCB PARMS ARE AVAILABLE IN THE MODEL data set.
```

The following example refers to the generation data group with the former generation 0 as generation -1 and the new generation as generation 0:
Restarting work on different workstations

- Do not use backward references. Backward references to data sets in previous steps make restarts more complicated.
- Avoid the use of return codes to control the execution of successive steps under normal conditions because this can lead to restart problems. Use return codes only to bypass step execution after failure, for example, COND=(0,NE).
- Always code a step name for every step (for example, STEPnn, where nn is the step sequence number).
Restarting work on different workstations
Chapter 19. Conditioning the processing of operations

This chapter describes the key concepts about *conditioning your operations* and the guidelines for deciding if and how to create conditions upon which your operations depend.

It contains the following sections:

- “Using conditional dependencies”
- “Evaluating conditions and conditional successor status” on page 374
- “Making an operation dependent on the status or return code of another operation” on page 377
- “Making an operation dependent on another operation step return code” on page 383
- “How to condition the status of operations with predecessors in X status” on page 390
- “Handling recovery using conditional dependencies” on page 392
- “Interactions with other functions” on page 395
- “MCP consistency rules” on page 395
- “Monitoring conditions in the current plan” on page 397
- “Daily plan batch handling of conditional dependencies” on page 402
- “Automatic conditional dependencies resolution” on page 402

Using conditional dependencies

Using conditional dependencies is a powerful way to manage scheduling workflow processing. It allows you to dynamically define alternative branches, for example, to run a recovery alternative flow. These alternative branches, when their processing completes, are automatically removed from the current plan when the daily plan batch is run.

For example, an application developer can use conditional logic to reduce the Total Cost of Ownership (TCO) and increase the automation of the business processes managed using Tivoli Workload Scheduler for z/OS. By using conditional dependencies while implementing workflow, he can automate the flow to follow alternative branches when a combination of specific criteria on operations occurs.

You can define conditional dependencies at the following levels:

**Job level**

By conditioning the start of the conditional successor to the check on job return code or status of the conditional predecessor. See “Making an operation dependent on the status or return code of another operation” on page 377 for more details.

**Step level**

By conditioning the start of the conditional successor to a specific step return code of the conditional predecessor. See “Making an operation dependent on another operation step return code” on page 383 for more details.
Evaluating conditions and conditional successor status

Evaluating conditions and conditional successor status

This section describes how the status of a conditional dependency is evaluated. These concepts apply to all conditional dependencies, both those defined at job level and those defined at step level.

A conditional dependency relationship is set up by using a condition.

A condition is a set of one or more condition dependencies aggregated by a rule. You can set the rule to be satisfied, and so the condition to be true, when one of the following occurs:

- Either all condition dependencies are true (AND)
- Or at least \( n \) condition dependencies are true (OR)

A condition dependency is a specific check of the status or return code of a predecessor operation or of the return code of an operation step.

The operations processing flow is affected by the conditions set and their final status.

The status of a condition is set based on the rule defined and on the statuses of its condition dependencies.

The following sections explain in more detail how to define a condition and its statuses transitions:

- “How to define a rule in a condition”
- “When the scheduler evaluates the status of a condition dependency” on page 375
- “How the scheduler evaluates the status of a condition dependency” on page 375
- “How the scheduler evaluates the status of a condition” on page 376
- “How the scheduler evaluates the status of the conditional successor operation” on page 376

This information can help you to understand how to customize conditions when defining your workflow to satisfy your requirements.

How to define a rule in a condition

As a condition rule you can require either of the following:

- All the condition dependencies must be true. This corresponds to the AND operator in the Boolean logic.
- At least \( n \) out of all the condition dependencies must be true. This corresponds to the OR operator in the Boolean logic.

Within each condition dependency contained in the condition, depending on the type of check that you require on the conditional predecessor, you can specify one of the following types of checks:

- **RC**: To check the conditional predecessor return code or abend. The return code can be either numeric or a string, for example OSUF. An abend can be either a system abend, represented by a string starting with S, or user abend, represented by a string starting with U.
- **ST**: To check the conditional predecessor status.
How to define a rule in a condition

In the required check you can use one of the following logical operators:

- **GE** Greater than or equal to. Valid only for RC condition type.
- **GT** Greater than. Valid only for RC condition type.
- **LE** Less than or equal to. Valid only for RC condition type.
- **LT** Less than. Valid only for RC condition type.
- **EQ** Equal to.
- **NE** Not equal to. Use it to specify conditions on final statuses only C (Completed), E (Ended-in-error), or X (Suppressed by condition).
- **RG** Range. Valid only for RC condition type.

**Note:** If you use the NOERROR or highest return code capabilities, the original return code is used to evaluate a condition dependency.

When the scheduler evaluates the status of a condition dependency

The status of condition dependencies and, consequently, the status of the condition to which they belong, is evaluated when the conditional predecessor has one of the following statuses:

- **S** Started
- **E** Ended in Error
- **C** Completed
- **X** Suppressed by Condition

or when a step-end event is received.

The condition dependency is evaluated only when a path in the plan exists, otherwise the condition dependency remains Undefined until a manual intervention or a Rerun is done.

A possible path for the conditional predecessor exists when at least one of the following occurs:

- The job is set to C and a normal successor exists.
- There is at least one conditional successor that has all the subsets of conditions referencing that conditional predecessor true according to the condition rules.

How the scheduler evaluates the status of a condition dependency

Before evaluating a condition dependency, the scheduler considers the current existence of at least one possible path for progression.

Based on these evaluations, the condition dependency status can be:

**U** (Undefined)

- When the check cannot be performed or when a possible path for progression does not exist in the plan.

**T** (True)

- When both the following occur:
  - A possible path for progression exists in the plan.
How the scheduler evaluates the status of a condition dependency

- When the evaluation was made and the result matches the expected value.
  
  F (False)
  - When both the following occur:
    - A possible path for progression exists in the plan.
    - The evaluation was made and the result does not match the expected value.

How the scheduler evaluates the status of a condition

The condition status can be any of the following values:

U (Undefined)
- When the rule cannot be evaluated yet.

T (True)
- If the rule is set to AND
  - When ALL condition dependencies are true.
- If the rule is set to OR (at least n condition dependencies must be true)
  - When at least n condition dependencies are true.

F (False)
- If the rule is set to AND
  - When at least one condition dependency is false.
- If the rule is set to OR (at least n condition dependencies must be true)
  - When at least n condition dependencies cannot be true.

How the scheduler evaluates the status of the conditional successor operation

Because the status of an operation depends on the status of its conditions and on the status of its normal predecessors, the status of an operation is set to one of the following values:

X (Suppressed by Condition)
- When at least one condition is F (False).

Note: When evaluating conditional successors status, predecessor operations in status X are considered equal to predecessor operations in status C.

R (Ready)
- When its normal predecessors are C or in X status and all its conditions are T (True).

W (Waiting)
- When at least one normal predecessor is not C or X or at least one condition is U (Undefined).

Note: If you define both conditional and normal dependencies on an operation, the normal dependency is managed as a conditional dependency that is true when the status of the predecessor operation is C. For this reason, when conditions are defined, to have the predecessor operation completed successfully is not sufficient to let the successor operation start, but instead the following considerations are applied:
- The conditions can be evaluated because a path for progression exists in the plan.
- The status of all conditions is considered to calculate the conditional successor operation status.
Making an operation dependent on the status or return code of another operation

Use a job-level conditional dependency when you want a successor job to start depending on a combination of one or more return codes or statuses of predecessor jobs.

Figure 139 shows the two different types of job level conditions, one based on the predecessor return code and the other based on the predecessor job status.

For example, using the return code as condition type, you can define that operation OP2 is dependent on operation OP1, specifying that OP2 must run when OP1 ends with a return code in the range from 1 to 3.

For example, using the status as condition type, you can define that operation OP4 is dependent on operation OP3, specifying that OP4 must run when OP3 status is E (Ended-in-error):

![Diagram of job conditional dependencies](image)

**Examples of job conditional dependencies evaluation**

This section contains an example showing how the operation processing flow is affected when using job-level conditions.

The workflow is required to run according to the following conditions:

- JOB1 ends successfully only when RC=0, RC=4, or RC=8 is returned. Any different result causes the job to end in error.
- JOB2 must run if JOB1 ends successfully with RC=4.
- JOB3 must run if JOB1 ends successfully with RC=8.
- JOB2 and JOB3 are skipped if JOB1 ends successfully with RC=0.
- JOB5 runs unconditionally when both JOB1 completes successfully and the two jobs, JOB2 and JOB3, are either completed or skipped.

To implement these requirements for the workflow, you can set the following conditions:

- Use NOERROR to force the JOB1 status to C (Completed) for RC=4 and RC=8.
- Make JOB5 a normal successor of JOB1, JOB2, and JOB3.
Examples of job conditional dependencies evaluation

- Define a condition on JOB2, the conditional successor, to run it when JOB1 status is C and JOB1 RC=4.
- Define a condition on JOB3, the conditional successor, to run it when JOB1 status is C and JOB1 RC=8.

An alternative way to implement these requirements, without using the NOERROR function, is described in “An example of using recovery jobs to implement workflow” on page 393.

In [Figure 140] dotted lines represent conditional dependencies and straight lines represent normal dependencies.

![Figure 140. Example of conditioning operations.](image)

The operations processing proceeds based on the possible JOB1 status and return code:

**JOB1 ends with status C and RC=0**

[Figure 141 on page 379] shows the operation processing flow:
A possible path exists because JOB1 has completed successfully and a normal successor, JOB5, exists. Because a path exists, the successor conditions can be evaluated.

- JOB1 condition is false because JOB1 return code is not 4. JOB2 status is set to X.
- JOB3 condition is false because JOB1 return code is not 8. JOB3 status is set to X.
- JOB5 has three predecessors with statuses C, X, and X so JOB5 status becomes R (Ready).

**JOB1 ends with status C and RC=4; JOB2 runs successfully**

Figure 142 shows the operation processing flow:

**Figure 141. Successors status evaluation if JOB1 ends with status C and return code 0**

**Figure 142. Successors status evaluation if JOB1 ends with status C and return code 4 and JOB2 runs successfully**
**Examples of job conditional dependencies evaluation**

A possible path exists because JOB1 has completed successfully and a normal successor, JOB5, exists. Because a path exists, the successor conditions can be evaluated.

- JOB2 condition is true because JOB1 status is C and RC=4. JOB2 runs and then successfully completes.
- JOB3 condition is false because JOB1 return code is not 8. JOB3 status is set to X.
- JOB5 has three predecessors with statuses C, C, and X so JOB5 status becomes R.

**JOB1 ends with status C and RC=4; JOB2 ends in error**

Figure 143 shows the operation processing flow:

![Graph showing job conditional dependencies evaluation](image)

**Figure 143. Successors status evaluation if JOB1 ends with status C and return code 4 and JOB2 ends in error**

A possible path exists because JOB1 has completed successfully and a normal successor, JOB5, exists. Because a path exists, the successor conditions can be evaluated.

- JOB2 condition is true because JOB1 status is C and RC=4. JOB2 runs, but ends in error
- JOB3 condition is false because JOB1 return code is not 8. JOB3 status is set to X.
- JOB5 has three predecessors with statuses C, E, and X so JOB5 status is set to W(Wait).

Now the operator can examine the reason why JOB2 failed and then either rerun the occurrence or fix the error and continue the processing.

**JOB1 ends with status E and RC=U2345**

Figure 144 on page 381 shows the operation processing flow:
Examples of job conditional dependencies evaluation

In this case no path for progression exists in the plan, therefore the conditions evaluations cannot be finalized.

- The condition on JOB2 is left U (Undefined), even though both condition dependencies are false. JOB2 status is set to W.
- The condition on JOB2 is left undefined, even though both condition dependencies are false. JOB3 status is set to W.
- JOB5 has three predecessors with statuses E, W, and W. JOB5 status is set to W.
- JOB1 is marked with the Unexpected RC flag and the warning message EEQ141W is logged in the Controller MLOG and in the system log.

Now the operator can examine the reason why JOB1 failed, check if the condition definitions are correct, and then either rerun the occurrence or fix the error and continue the processing.

### Job conditional dependencies key concepts and restrictions

Consider the following key concepts when using conditional dependencies:

- You cannot have mixed combinations of AND and OR operators within the same condition, but you can define more than one conditional dependency on a single operation and link them using dummy operations.
- Because internal and external dependencies are managed differently in Tivoli Workload Scheduler for z/OS, it is important to know that conditional dependencies are considered to be external dependencies, even if they link operations belonging to the same application occurrence in the plan. This means the following:
  - Because all operations belonging to the same application must be chained using normal dependencies, in some cases it might be necessary to provide a common normal predecessor or successor to the conditional predecessor and conditional successor operations to join the operations together. For more information, see “How to ensure application consistency using FOP” on page 382.
  - Within the same application you cannot define an operation both as conditional predecessor and as a normal predecessor for a specific operation.

![Diagram](image-url)  

**Figure 144. Successors status evaluation if JOB1 ends with status E and return code U2345**
Job conditional dependencies key concepts and restrictions

- When adding an occurrence to the plan, set Dependency Resolution to YES to add conditions even if they involve only internal predecessors. To set the Dependency Resolution to YES when the condition addition is the result of a match in the ETT criteria, you must set the value in the DR column to Y in the definition of the ETT Tracking Criteria.

- You cannot change the status of an operation, out of rerun path, if you have a condition on it which is not True. This is because a condition whose status is not T (True) is equivalent to an external predecessor that is not C (Completed).

- You cannot change the status of an operation, out of Rerun path, if you have a conditional successor in status C, X, E, or S. This is because you cannot have a successor that is already processed, successfully or not.

- You cannot use conditional dependencies on a non-centralized end-to-end operation because the Tivoli Workload Scheduler Symphony file does not support conditional dependency. However, you can specify an end-to-end operation as a conditional predecessor.

How to ensure application consistency using FOP

The scheduler informs you, by issuing an "APPLICATION INCONSISTENT" message when trying to save your application, if a common dependency might be needed to link either side of a condition relationship. You can do this by using a dummy operation on a non-reporting general workstation or by using an actual predecessor of the conditional predecessor operation, if one already exists within the same application.

For example, Figure 145 shows an example that would cause an "APPLICATION INCONSISTENT" message to be displayed because JOB4 has only one predecessor, which is conditional.

![Diagram](image.png)

**Figure 145. An example of APPLICATION INCONSISTENT problem**

To avoid the APPLICATION INCONSISTENT error from occurring, without modifying the job processing flow and logic within the application, it is sufficient to make the following changes:

1. Add a dummy operation JOB1 as First Operation (FOP).
How to ensure application consistency using FOP

2. Set JOB1 as normal predecessor for JOB2, so that JOB1 is completed as soon as the occurrence starts to run.

3. Set JOB1 as normal predecessor for JOB4, and also for all operations that have only conditional predecessors.

For more information about using a FOP operation, see Chapter 24, “Loop detection and analysis,” on page 509. Figure 146 shows the result of these changes:

![Diagram](image)

Figure 146. An example of how to fix an APPLICATION INCONSISTENT problem

Making an operation dependent on another operation step return code

In Tivoli Workload Scheduler for z/OS, jobs can comprise several steps whose return codes might condition the start of successor jobs.

In some instances the business logic might mean that a successor to a long running job could, in theory, actually start on the completion of a step part way through the long running job, instead of waiting for the entire predecessor job to complete. Traditional dependencies force you to wait for the entire job to complete, thereby incurring extra time that could be saved if the successor could start in parallel part way through the long running job.

With job-level condition dependencies, it is possible to split the long running JCL into smaller jobs, thereby allowing more parallel processing, and a more granular visibility of the workflow. In some cases though, it is not possible to split a large job into smaller components.

If the long running job cannot be split, a successor job can be made dependent on a step within a long running job using step-level condition dependencies. This means...
Making an operation dependent on another operation step return code

that the successor is allowed to start, if the named step completes with stated return codes, before the entire long running job completes, shortening overall processing time.

Step-level condition dependencies apply only to operation step-end return codes and, if a possible path for progression exists in the plan, are evaluated as soon as that step-end return code is generated.

As for job-level condition dependencies, step-level conditional dependencies can be added in the plan at run time and monitored from the ISPF panels and from the Dynamic Workload Console.

Note: By default Tivoli Workload Scheduler for z/OS tracks job processing using appropriate events about job start and job end. If you want to use the step dependency capability, configure Tivoli Workload Scheduler for z/OS to track step-end events. See "How to activate step dependency evaluation" for more information.

How to identify a step

A step has the following identifiers:

STEPNAME
   It is the procedure invocation step name. It is blank if the step is not in a procedure.

PROCSTEP
   It is the step name in the JCL procedure or the job step name if the step is not in a procedure.

The following rules apply to these step identifiers:

• If you specify blank for both STEPNAME and PROCSTEP, the return code dependency applies to the entire job.
• The combination of STEPNAME blank and PROCSTEP not blank is not supported.

When you define or change a step-level dependency in the database or in the current plan:

• Do not specify a step that is duplicated in the corresponding JCL.
• Consider that blank PROCSTEP and STEPNAME delays the check at job completion time.

Note: The scheduler cannot check the actual step names in the JCL, which might be located in any user library of the tracker system and updated independently of the database definitions.

How to activate step dependency evaluation

If you want the scheduler to check step dependencies, you must ensure that the event writer options, defined by using the EWTROPTS initialization statement, contain one of the following parameters:

• STEPEVENTS(ALL)
• SDEPFILTER

If you want to use the SDEPFILTER parameter, you must define the programmer name keyword in the JOB card of any JCL corresponding to a condition dependency at step level. The SDEPFILTER parameter must specify a selected
How to activate step dependency evaluation

subset of the programmer name keyword. For more information about how to set the SDEPFILTER parameter, see *Customization and Tuning*.

**Note:** Use the SDEPFILTER parameter as an alternative to the STEPEVENTS(ALL) if you want to reduce the load on scheduling performance.

**How a step dependency is evaluated**

The scheduler considers the step entries segment of the current plan job name table to obtain updated results about step-end events.

The step dependency is evaluated at step-end time or at job-end time, including jobs manually set to C (Complete) or E (Ended-in-error), whenever no step-end information is available.

The criteria used to evaluate a step dependency are the same as those described in “Evaluating conditions and conditional successor status” on page 374, with the following two exceptions:

- The return code (RC) is the only valid check for step dependency.
- If at job-end time the step dependency is not yet evaluated the evaluation is made as described in the following section "How the evaluation is done when no step-end event is received."

**How the evaluation is done when no step-end event is received**

It might happen that, at job-end time, a step dependency is not yet evaluated, because the event writer did not receive the step-end event for one of the following reasons:

**Configuration error**

The EWTRLOPTS initialization statement, for the tracker event-writer task, does not activate the step-end event generation.

**Definition error**

There is an error in the step dependency definition. For example, the step name specified in the dependency definition does not match any step in the JCL that actually ran.

In both cases the scheduler does the following actions:

- Sets the condition to U (Undefined) status.
- Marks the condition with the missing step-end flag.
- Logs the EQQE127W message in the controller MLOG and in the system log.
- Keeps the status of the conditional successor as W (Wait).

From the ISPF panels you can do the following actions to manage your workload if a missing step-end occurs:

**Monitor missing step-end information**

A column in the condition dependency list identifies the single dependencies that are undefined because of missing step-end information. For details, see the description of the BROWSING CONDITION DEPENDENCIES panel in Figure 161 on page 401.

**Reset the step dependency status**

You are allowed to reset the U (Unresolved) step dependency to T (True) or F (False) in the current plan using a row command. For more information, see “Changing and adding conditional dependencies” on page 558.
Examples of step-level conditional dependencies evaluation

Examples of step-level conditional dependencies evaluation

If you configured Tivoli Workload Scheduler for z/OS to track step-end events, then the step dependencies are checked at step-end time when the return code value is available.

This section contains an example showing how operation processing flow is affected when using step-level conditions.

Figure 147 shows an example of conditional dependency logic at job step level. In the figure, dotted lines represent conditional dependencies and straight lines represent normal dependencies.

In this example:

- JOBB can start if STEP100, belonging to JOBA, ends with RC=4.
- JOBC is a normal successor of JOBA and therefore starts if JOBA status is C (Completed).

The operation processing proceeds based on possible STEP100 return codes and JOBA statuses:

**STEP100 ends with RC=4; JOBA status is not yet completed**

Figure 148 on page 387 shows the operation processing flow:
A possible path exists because the step dependency is true and therefore a conditional successor exists. The successor conditions can be evaluated.

JOBB can start processing, if free from other dependencies, even if JOBA has not yet ended processing.

**STEP100 ends with RC=8; JOBA status is not yet completed**

Figure 149 shows the operation processing flow:

Even though STEP100 ended with a return code different from 4, the successor conditions cannot be evaluated yet because JOBA is not completed and so an existing path in the plan does not exist.

- The step-level condition is set to U (Undefined).
- JOBB status is set to W (Wait).

The condition is evaluated again as soon as JOBA ends processing.
Examples of step-level conditional dependencies evaluation

**STEP100 ends with RC=8; JOBA ends successfully**

Figure 150 shows the operation processing flow:

- A possible path exists because JOBA has completed successfully and a normal successor, JOBC, exists. The step-level condition is evaluated again.
  - JOBC can start processing, if free from other dependencies.
  - The step-level condition status is set to F (False).
  - JOBB status is set to X (Suppressed by condition).

**STEP100 ends with RC=8; JOBA ends in error**

Figure 151 shows the operation processing flow:

- In this case a possible path in the plan does not exist.
  - JOBA ended in error, therefore its normal successor, JOBC is set to W (Wait).
  - The step-level condition is not evaluated again and so its status remains U (Undefined) and JOBB status remains W (Wait).
Examples of step-level conditional dependencies evaluation

- JOBA is marked with the Unexpected RC flag and the warning message EQQE142W is logged in the controller MLOG and in the system log.

Now the operator can examine the reason why JOBA failed and then either rerun the occurrence or fix the error and continue the processing.

No step-end event is received for STEP100; JOBA ends successfully

Figure 152 shows the operation processing flow:

![Diagram showing the operation processing flow]

A possible path exists because JOBA has completed successfully and a normal successor, JOBC, exists.
- JOBC can start processing, if free from other dependencies.
- The step-level condition is left U (Undefined) and the warning message EQQE127W is logged in the controller MLOG and in the system log.
- JOBB status remains W.

No step-end event is received for STEP100; JOBA ends in error

Figure 153 on page 390 shows the operation processing flow:
Examples of step-level conditional dependencies evaluation

In this case a possible path in the plan does not exist:
- JOBC status remains W (Wait).
- The step-level condition is left as U (Undefined) and the warning message EQQE127W is logged in the controller MLOG and in the system log.
- JOBB status remains W.
- JOBA is marked with the Unexpected RC flag and the EQQE141W message is logged.

How to condition the status of operations with predecessors in X status

An operation in X status is equivalent to a NOPED operation; it does not run but its normal successors can start.

This section describes two possible ways to condition the status of operations with predecessor operations in X status.

How to propagate the X status in a branch

You might want to modify how the X (Suppressed by condition) status is managed, for example, by allowing the X status to be propagated along a whole branch. To do this you must link operation successors using condition on predecessor status C (Completed), as displayed in Figure 154 on page 391.
How to propagate the X status in a branch

A possible path exists because JOB1 is completed and a normal successor, JOB3, exists. The successor conditions can be evaluated.

- JOB2 condition is false because JOB1 return code is 0. JOB2 status is set to X.
- JOB4 condition is false because JOB2 status is not C. JOB4 status is set to X.
- JOB6 condition is false because JOB4 status is not C. JOB6 status is set to X.

In this way you can propagate the X status in the branch.

How to set to W the status of an operation with predecessors in X status

You might want to modify how the X (Suppressed by condition) status is managed, for example, by allowing the status of the successor to become W (Wait) if the status of the predecessor is X. Figure 155 on page 392 shows how you do this:
How to set to W the status of an operation with predecessors in X status

This example uses as conditional successor a non-existing NOJOB operation so that, when the scheduler tries to run, it fails and the NOJOB operation status is set to E (Ended-in-error).
- JOB2 condition is true because JOB1 status is X. NOJOB can start.
- NOJOB status is set to E because the operation does not exist in the database.
- JOB2 status is set to W.

Note: To accomplish the same result, avoiding to collect an error message because of the non-existing job, you can use a General Completion Only workstation.

Handling recovery using conditional dependencies

Using conditional dependencies, the error status of a job can be used as a criteria for starting a successor, when this successor is used as a recovery job.

By setting to Y the COND RECOVERY JOB option for an operation, you define that the operation is used as the recovery job for a conditional predecessor.

Any conditional predecessor that Ended-in-Error, with a status or error code matching a condition dependency defined for the job, does not prevent the daily plan process from removing the occurrence to which the predecessor belongs. To check if the status E (Ended-in-error) can be ignored at occurrence removal phase, the daily plan process uses a field automatically set by the scheduler, corresponding to the Recovered by COND output field in the browsing operation flow.

Note: As soon as a recovery job becomes ready, the scheduler checks the predecessors in error status at that time. Any predecessor that ends in error after the recovery job runs cannot be flagged as Recovered by COND.

The daily plan process removes the occurrence in the following cases:
- The occurrence status is C (Completed).
Handling recovery using conditional dependencies

- The occurrence status is E (Ended-in-error), and includes only operations in one of the following statuses:
  - C
  - X (Suppressed by condition)
  - E, with the Recovered by COND field set to Y.

For example, suppose that either JOBR1 or JOBR2 must run when JOBB ends with an error. You can specify JOBB as their conditional predecessor, according to Figure 156:

When defining JOBR1 and JOBR2 and specifying JOBB as their conditional predecessor, you can also set the COND RECOVERY JOB option to Y in the AUTOMATIC OPTIONS of JOBR1 and JOBR2, so that the daily plan process removes the occurrence containing JOBB, because it ended with an error code matching one of the defined condition dependencies.

An example of using recovery jobs to implement workflow

This section describes how you can implement the scenario described in “Examples of job conditional dependencies evaluation” on page 377 by using recovery jobs instead of the NOERROR function.

These are the requirements that the workflow must satisfy:
- JOB1 ends successfully only when RC=0, RC=4, or RC=8 is returned. Any different result causes the job to end in error.
- JOB2 must run if JOB1 ends successfully with RC=4.
- JOB3 must run if JOB1 ends successfully with RC=8.
- JOB2 and JOB3 are skipped if JOB1 ends successfully with RC=0.
- JOB53 runs unconditionally when JOB1 completes successfully and the two jobs, JOB2 and JOB3, are either completed or skipped.

Figure 157 on page 394 shows how you can customize the workflow to implement its requirements using recovery jobs. In the figure dotted lines represent conditional dependencies.
An example of using recovery jobs to implement workflow

Define the workflow in the following way:

- Set JOB2 as recovery job for JOB1, specifying that JOB2 must run if JOB1 RC=4.
- Set JOB3 as recovery job for JOB1, specifying that JOB3 must run if JOB1 RC=8.
- Define a condition on JOB5 specifying that JOB5 must run if the status of one among JOB1, JOB2, and JOB3 is C.

Figure 157. Example of conditioning operations using recovery jobs

An example of handling recovery in an end-to-end environment

If an operation has a conditional predecessor to be run on a fault-tolerant workstation, the scheduler takes into account any RECOVERY statement defined in the SCRPTLIB member related to this predecessor. If there is a RECOVERY statement defined, the scheduler postpones the evaluation of the condition according to the following logic:

- If the conditional dependency check is based on the status value C (Completed) or E (Ended-in-error):
  - If the recovery option is STOP or CONTINUE, the scheduler evaluates the condition when the recovery job, specified through the JOBCMD or JOBSCR parameter, ends. If there is no recovery job specified, the scheduler evaluates the condition after the reply to the prompt.
  - If the recovery option is RERUN, the scheduler evaluates the condition only if the rerun succeeds.

- If the conditional dependency check is based on the status value S (Started), the scheduler evaluates the condition immediately. If the recovery option is RERUN, the condition status is not reset after the rerun.

For details about the RECOVERY statement, see *Scheduling End-to-end with Fault Tolerance Capabilities*, Chapter 3 *Customizing*, section *Defining centralized and non-centralized scripts*, topic *Configuring the SCRPTLIB*. 
Interactions with other functions

This section contains useful information if you use any of the following functions:

**Automatic Recovery**
If an operation has conditional successors defined, check that the related JCL does not contain automatic recovery statements that cause the job to be restarted.

**Restart and cleanup**
The same restrictions apply as for changing the operation status to ready by selecting option 6 (GENERAL) from the MODIFYING AN OPERATION IN THE CURRENT PLAN panel. See also "MCP consistency rules." In particular, a step or job restart request might result in a request to change to ready the status of an operation with conditional successors already started, completed, suppressed by condition, or ended in error. In this case the scheduler issues message EQQM208E. Only when rerunning an occurrence might you indirectly have this kind of change.

**NOERROR or highest return code**
If you use the NOERROR or highest return code capabilities, the scheduler saves the original return code when setting an operation to complete. It then uses the original return code to evaluate a condition dependency.

You can display the original return code when listing operations interactively, by using the ISPF dialog or Dynamic Workload Console, or through batch programs, by using a PIF application or BCIT.

**Critical path handling**
When processing a critical path, the scheduler considers that a job, with any conditional predecessor defined, breaks the critical path. Therefore avoid using critical jobs with conditional predecessors.

However, given a critical job, the scheduler updates the related hot list with any critical predecessor ended in status X (Suppressed by condition).

---

**MCP consistency rules**

When you add work to the plan on request, the scheduler typically handles operation conditions as external objects. Some checks and restrictions, that are valid for external predecessors, are valid also for conditions. In particular, this applies for:

**Status change consistency:**

- You cannot directly change the status of an operation with undefined or false conditions (the scheduler applies the same checks as it does for not completed external predecessors). Only when rerunning an occurrence might you indirectly have this kind of change.

- You cannot directly change the status of an operation with external or conditional successors that are already S (Started), C (Completed), X (Suppressed by condition), or E (Ended in error). Only when rerunning an occurrence might you indirectly have this kind of change.

**Dependency resolution:**
When adding an occurrence to the current plan, you must require the dependency resolution if you want to add also the related conditions, even for conditions that specify only internal predecessors. For details, see "Including dependencies defined in the database" on page 553.
The following sections highlight other impacts of condition definitions on some main MCP functions.

**Rerunning an occurrence**

By rerunning an occurrence, you can reset conditional successors that have already completed.

If you selected an operation as restart point, the scheduler automatically perform the following actions:

- Modifies the status of conditional successors to W (Wait). Their conditions, if any, are forced to U (Undefined). The condition extended status is set to *Forced by rerun*.
- Modifies the status of conditional predecessors to C (Completed) until a conditional predecessor with status C is found. Their conditions, if any, are forced to T (True). The condition extended status is set to Forced by rerun.
- Because the X (Suppressed by condition) status is a final status equivalent to C status, the modification of conditional predecessors status stops also when a conditional predecessor with status X is found. In this case, the condition is left as F (False) and the extended status of the condition is set to Forced by rerun.

For more details, see "Rerunning an occurrence in the current plan from a specific operation" on page 563.

**Setting an occurrence to waiting**

The same rules apply as described in "MCP consistency rules" on page 395.

If the selected occurrence has conditional successors, the scheduler might reset the related conditions to undefined.

For more details, see "Restarting an occurrence from the beginning" on page 562.

**Setting an occurrence to complete**

Setting an occurrence to C (Completed) is equivalent to setting to C each single operation in the occurrence. All possible incongruences in both normal and conditional dependencies are adjusted.

The successors conditions that are already evaluated are not evaluated again to avoid running alternative branches out of rerun path. The conditions that must be evaluated again are marked with extended status *Frozen by Complete*.

For example, suppose that you have the following two occurrences, A and B , each containing only one operation, A-001 and B-002. You define a condition on B-002 saying that B-002 runs if A-001 ends in error. If A-001 ends in error: the condition is evaluated as true and the B-002 status is set to ready.

Then, if you decide to manually set to complete the job occurrence A-001 the condition on B-002 remains true, but its extended status is set to *Frozen by Complete*.

For more details, see "Completing an application occurrence" on page 566.
Deleting an operation or an occurrence

Unlike the behavior for normal dependencies, when you delete an operation or an occurrence containing operations referenced in a condition as conditional predecessor or conditional successor jobs, the scheduler does not redefine the rule criteria to avoid workflow propagation tree from breaking.

This is how the scheduler manages condition dependencies for which the deleted operation is the conditional predecessor:

If the condition status has already been evaluated T (True) or F (False)

All condition dependencies are kept and the condition dependencies whose predecessor is the deleted job are marked as removed.

If the condition status is undefined (some condition dependencies have not been evaluated yet)

- Evaluated condition dependencies whose predecessor is the deleted job are marked as removed.
- Non-evaluated condition dependencies whose predecessor is the deleted job are deleted from the condition.

If the deleted job is the conditional successor job the condition is deleted.

Monitoring conditions in the current plan

Condition dependencies introduce a different logic in how job processing is run across the plan. For example, it allows you to start an operation even if its conditional predecessor has just started or ended in error.

The following sections provide you with some examples of how to use ISPF panels both to check if job processing is blocked and to see if a job that ended in error is preventing job processing from continuing.

How to look for operations that ended in error

The ERROR LIST panel (EQQMEP1L / EQQMEP2L) lists jobs that ended in error and provides you with other important information to understand if that error is preventing job processing from proceeding.

Figure 158 shows you how the information about the jobs that ended in error is displayed if you use the default layout TWSZOS:

<table>
<thead>
<tr>
<th>Cmd</th>
<th>Jobname</th>
<th>no.</th>
<th>date</th>
<th>time</th>
<th>Errc</th>
<th>xRC</th>
<th>CoRj</th>
<th>RbyCo</th>
<th>CPR</th>
<th>PRE</th>
<th>CSUE</th>
<th>SUE</th>
<th>SRE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROXJOB1</td>
<td>1 00/03/22 00.00</td>
<td>2134</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROXJOB2</td>
<td>3 02/03/19 12.02</td>
<td>5134</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROXJOB3</td>
<td>1 03/03/22 01.00</td>
<td>5134</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>N</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 158. EQQMEP1L - Left side of the Error List panel

The error code is displayed in the ERRC column.

The right side of the panel, accessible by typing RIGHT, provides you with additional information such as the occurrence the job belongs to, the workstation, and the execution destination. By typing LEFT you see again the left side of the panel.
How to see if an operation that ended in error prevents job processing from proceeding

In the ERROR LIST panel you can see if the job listed in the JOBNAME column might prevent successor jobs from running by looking at the values in the CSU£ and SU£ columns.

How to check if conditional successors are prevented from running

In the CSU£ column you see the number of conditional successors for the job displayed in the JOBNAME column. If this number is 0, then that job has no conditional successors. If the number is greater than 0, then the job has conditional successors and their status depends on how the rule is set in the condition.

The Unexpected RC field XRC indicates if something unexpected occurred and leaves some conditions in Undefined status preventing conditional successors from running.

The job processing is blocked if CSU£ is greater than 0 and the value listed under the XRC column is set to Y.

You can filter by XRC (Unexpected RC) both on ERROR LIST (ISPF 5.4 or 6.4) and OPERATION LIST (ISPF 5.3 or 6.3) panels. For each operation listed, you can check conditional successors and their conditions by using the row command I from error list or B from operation list. The OPERATION DETAILS panel EQQSPSP is displayed and gives you additional information about what was incorrect or unexpected in the path.

To investigate in more detail you can also look for warning messages EQQE141W and EQQE142W issued in EQQML06 and in the console log.

Note: When messages EQQE127W and EQQE141W are issued a manual intervention is required to allow job processing to proceed.

If you find, the message EQQE142W in the log, it means that, when the step-end event, identified in the message text by STEPNAME PROCSTEP, was received, the scheduler evaluated that existing conditional and normal successors would never have run due to conditions definitions. For example, see the example in “Examples of step-level conditional dependencies evaluation” on page 386.

If you find the message EQQE141W in the log, one of the following events occurred:

• When the status of the job indicated in the message became ST, the scheduler evaluated that existing conditional and normal successors would never have run due to conditions definitions. For example, see the example in “Examples of job conditional dependencies evaluation” on page 377.

• You defined a step dependency for a step that does not exist in the referenced job. The scheduler recognizes that there is a missing step and issues the warning message EQQE127W, when the job ends, it evaluates that successors will never run and issues the warning message EQQE141W, even if the origin of the problem is in the step definition. For example, see the example in “Examples of step-level conditional dependencies evaluation” on page 386.
How to check if conditional successors are not prevented from running

How to check if conditional successors are not prevented from running
A conditional predecessor job that ended in error does not block job processing from proceeding when at least one of the following occurs:

- Its entry under the JOBNAME has the value listed under the XRC (Unexpected RC) column set to N.
- Its entry under the JOBNAME column has the value listed under the RBYCO (Recovered by Condition) column set to Y.

A job that ended in error and flagged as Recovered by Condition has the recovery action already activated by its conditional successor, the conditional recovery job, and so it does not prevent any other job from processing. Therefore, it does not stop processing and is removed by the daily plan batch if it belongs to an occurrence with all operations in X, C, or Error Recovered by Condition status. For more information, see “Handling recovery using conditional dependencies” on page 392.

You can filter by XRC and RBYCO both on the ERROR LIST and OPERATION LIST panels.

How to check if normal successors are prevented from running
The SU£ column shows if the job displayed under the JOBNAME column has a normal successor. If the number displayed in the SU£ column is 0 then that job has no normal successor, and, even if the operation ended in error, no successor job is prevented from running. If the number displayed under SU£ is greater than 0, then the job has a normal successor that is prevented from running because its status can be set to Ready (R) only if the predecessor job status is Complete (C) or Suppressed by condition (X).

How to check if an operation is waiting due to conditions
An operation is in W (Wait) status due to condition when its condition is in U (Undefined) status.

You can list these operations by filtering on Status=W and Undefined Cond=Y in the SELECTING OPERATIONS filter panel as displayed in Figure 159 on page 400.
How to search for skipped operations in a conditional flow

The operations in status X (Suppressed by condition) are skipped in the current plan, meaning that they are not run, but, like NOPED operations, their normal successors can run if free from other constraints. To list skipped operations you can filter on status X in the operation list filter panel.

When you define an application you can decide to allow the X status to propagate through a branch using conditional dependencies as described in “How to propagate the X status in a branch” on page 390.

How to display information about conditions

You can monitor the conditions defined in the current plan by selecting option 4 (Dependecies) from the SELECTING APPLICATION OCCURRENCE AND OPERATION INFORMATION panel and then entering the COND command. The panel in Figure 160 is displayed.

For a list of possible values in the Status field, see “How the scheduler evaluates the status of a condition” on page 376.
The possible values of the Ext. Status field are:

**C** Frozen by complete, meaning that one of the operations included in the conditional dependencies belongs to an occurrence that was set to complete, after an MCP request.

**F** Forced by rerun, meaning that the condition status was forced by a rerun action. For more information see "Rerunning an occurrence" on page 396.

After selecting a row, the panel shown in Figure 161 is displayed:

![EQSOCCP - Browsing condition dependencies](image)

**Figure 161. EQSOCCP - Browsing condition dependencies**

The R (Removed) column set to Y indicates that the corresponding operation was either removed by daily planning or manually deleted from the current plan.

The last two columns, SD and MI, provide information about step-level dependencies. In particular:

- The SD (Step Dependency) flag indicates if the dependency is at step level.
- The MI (Missing step completion Information) flag indicates if the dependency is undefined because of missing step-end information.

For such operations, the event writer issues message EQQE127W in the message log (EQQMLOG), unless the operation did not actually run (status manually set).

If the dependency is undefined because of missing step-end information and DP or MCP processes removed the corresponding operation, the last three columns, R, SD and MI, are set to Y.

You can check any step name definition by scrolling the list to the right, to display the panel shown in Figure 162 on page 402.
Daily plan batch handling of conditional dependencies

The daily plan batch removes the following entries from the new plan:

- All the occurrences that are completed
- All the conditions that were evaluated (with status True or False)
- All the conditions defined on operations in X status

This behavior can lead, in some particular scenarios, to the setting of the Unexpected RC flag after a daily plan batch run. An example of one of these scenarios is when a true path (Condition and conditional successors) is removed and the conditional predecessor has another conditional successor still in the plan in waiting status due to an undefined condition caused by a missing step event.

If the daily plan batch needs to reconsider the undefined conditions referring to predecessor operations that have been removed from the plan, then the condition dependencies are:

- Left in the Condition with flag Removed set to on if they were already evaluated (True or False)
- Dropped from the Condition if they were still Undefined. In this case the Condition rule is readjusted if needed.

If all operations mentioned in a condition are removed from the plan, the condition is removed from the plan too.

Automatic conditional dependencies resolution

When you add new occurrences to the current plan, the scheduler applies some criteria to identify potential conditional successors:

1. Searches, among all the potential conditional successors defined in the database, for those existing in the plan.
2. Selects the closest following conditional predecessor.
3. Determines whether to add a new condition dependency to the existing condition or to create a new condition ad hoc as follows:
Automatic conditional dependencies resolution

- If the condition dependency referring to the new predecessor occurrence is not present in the plan for the conditional successor, a new COND ID is created.
- If the condition dependency referring to the new predecessor occurrence is already present in the plan for the conditional successor, then:
  - If that condition dependency has not yet been filled in the plan, the condition dependency is added to the existing COND ID.
  - If the condition dependency has already been filled in the plan, then a new condition containing that condition dependency is created and marked as a *clone*. A different COND ID is generated, but the original one is also saved and displayed in the description of the cloned condition. The new COND ID is the first available number in the range from 1 to 999.
Automatic conditional dependencies resolution
Chapter 20. Defining and managing cross dependencies

Tivoli Workload Scheduler for z/OS cross dependencies help you to integrate and automate job processing when:

- The workload is spread across different scheduling environments, because some of the activities run at different sites or involve different organizational units or require different skills to be run.
- Even if most of the batch workload is managed locally, none of these environments is completely isolated from the others because they frequently interoperate to exchange or synchronize data and activities.

More specifically, the cross dependency feature is key when you need to synchronize activities between different scheduling environments in an easy way so that you can:

- Define in one scheduling environment dependencies on batch activities that are managed by another scheduling environment.
- Monitor the status of the remote predecessor jobs as if they were running in your local environment.

Additionally, you can control the status of these dependencies by navigating from a unique user interface across the different scheduling environments.

This chapter describes how you define and use cross dependencies.

It contains the following sections:

- “An introduction to cross dependencies”
- “Defining a cross dependency in the database” on page 406
- “Daily plan consistency checks on shadow jobs and remote engine workstations?” on page 409
- “Monitoring a cross dependency resolution in the current plan” on page 410

An introduction to cross dependencies

A cross dependency is, from a logical point of view, a dependency for a local job on a job instance that is scheduled to run on a remote engine plan.

Use cross dependencies to integrate the workload running on different engines, which can be Tivoli Workload Scheduler for z/OS engines (controller) or Tivoli Workload Scheduler engines (master domain manager and backup master domain manager).

The following objects and terms are used to describe and implement cross dependencies:

**Remote engine workstation**

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

**Remote job**

A job scheduled to run on a remote Tivoli Workload Scheduler engine.
An introduction to cross dependencies

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

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

From a logical point of view in the local environment:
- The remote engine workstation is used to map the remote Tivoli Workload Scheduler engine.
- The shadow job, defined on that remote engine workstation, is used to map a remote job instance scheduled to run on that remote Tivoli Workload Scheduler engine.

To implement a cross dependency, you must define a normal dependency for your local job on a shadow job that:
- Points to the remote job on which you want to create the cross dependency.
- Is defined locally on the remote engine workstation that points to the engine where the remote job is scheduled to run.

Figure 163 shows the logical flow implementing cross dependencies:
1. In the bind process, the shadow job is associated to the remote job instance.
2. After the bind is established, the shadow job status is updated according to the remote job status transition.
3. When the shadow job status becomes COMPLETE (C) the normal dependency of the local job is released, and so also the cross dependency of that local job on the remote job is also released.

Defining a cross dependency in the database
The following sections describe the steps to define a cross dependency between a job defined in your environment and another job defined on a different Tivoli Workload Scheduler engine:
- “Step 1. Set up destinations” on page 407
- “Step 2. Create a remote engine workstation” on page 408
- “Step 3. Define a shadow job on the remote engine workstation” on page 408
- “Step 4. Add a dependency on the shadow job” on page 409
Step 1. Set up destinations

An existing HTTP or HTTPS destination to communicate with the remote engine is a prerequisite:

- When you want to define a remote engine workstation to send bind requests to the remote engine referenced in the destination. The name of the destination must be specified in the remote engine workstation definition, as the DESTINATION field.
- When you expect to receive incoming bind requests from the remote engine referenced in the destination.

Depending on whether the remote engine communicates using a secure or non-secure HTTP protocol, you define the destination using either ROUTOPTS HTTPS or ROUTOPTS HTTP.

The ROUTOPTS statement is defined in the member of the EQQPARM library that is specified by the PARM parameter in the JCL EXEC statement of the controller. The syntax for defining an HTTP or HTTPS destination is the following:

```
ROUTOPTS HTTP|HTTPS(dest name='IP address or hostname'/port/remote engine type)
```

Where:

- **dest name**
  Name of the destination, up to 8 alphanumeric characters.

- **IP address or hostname**
  Fully qualified IP address or hostname used to communicate with the remote engine. For more information about how to set this value, see "How to get the correct values to specify in the destination" on page 408.

- **port**
  Port number used to communicate with the remote engine. For more information about how to set this value, see "How to get the correct values to specify in the destination" on page 408.

- **type**
  Type of the remote engine:
  - **D** Distributed remote engine
  - **Z** z/OS remote engine

Destinations for which a remote engine type has been specified are named remote engine destinations.

To ensure connection availability, the remote engine destinations are checked on a timely basis, even if there is no HTTP pending request for that destination. As a best practice, avoid defining unnecessary remote engine destinations, to prevent unnecessary connection checks. If a remote engine destination is unreachable the following occurs:

- The error message EQQHT15E with a specific TCP/IP error is issued in the MLOG.
- The remote engine workstation that uses an unreachable destination is set to OFFLINE. The status of the workstation is set to ACTIVE as soon as the connection check to the remote engine succeeds.

For more information about the ROUTOPTS statement, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning.*
How to get the correct values to specify in the destination

When defining the destination to communicate to the remote engine make sure that the values specified for IP address or hostname and port are the same as those set locally on the remote engine.

Do the following to verify which values are set on the remote engine:

If the remote engine is z/OS based

The settings are optional. If specified, they are set in the HTTPOPTS options HOSTNAME and HTTPPORTNUMBER. If not specified their default values are displayed in the MLOG in the EQQHT19I message.

If the remote engine is distributed

The settings are mandatory and are specified at installation time. First add a destination using the information specified in the file JobDispatcherConfig.properties, then add destinations using the values specified in the output of the following command:

On Windows:

TWS_HOME\TDWB\bin\exportserverdata.bat

On UNIX:

TWS_HOME/TDWB/bin/exportserverdata.sh

Step 2. Create a remote engine workstation

Create a remote engine workstation for a specific remote engine when you need to define dependencies on job instances defined on that remote engine. You can run only shadow jobs on a remote engine workstation.

Optionally, you can specify when the remote engine workstation is available.

You can create a remote engine workstation either from the CREATING GENERAL INFORMATION ABOUT A WORKSTATION panel, as shown in Figure 35 on page 63 or from the Dynamic Workload Console by selecting Scheduling Environment → Design.

For more information about the settings that define a remote engine workstation, see “Specifying remote engine workstations” on page 71.

Step 3. Define a shadow job on the remote engine workstation

Create a shadow job pointing to a specific job instance defined on a remote engine when you want to track in your local environment the status of that remote job and define cross dependencies on that remote job.

The closest preceding matching criteria is used to map the shadow job to a job instance in the remote engine plan.

The input arrival (IA) of the shadow job is used to find the matching with the job instance to bind in the remote engine plan.

The shadow job status transition is derived from the remote job instance status transition.

You can create a shadow job from the OPERATIONS panel, as shown in Figure 68 on page 138 Choose a remote engine workstation as the workstation for the job.
Step 3. Define a shadow job on the remote engine workstation

operation, and select option 13 (REMOTE JOB INFO) in the OPERATION DETAILS panel to specify the information to identify the job instance in the remote engine plan.

To create a shadow job from the Dynamic Workload Console, select Workload → Design → Create Workload Definition.

On Tivoli Workload Scheduler distributed environments, you can use alias for job stream names and job names. If you are defining in your Tivoli Workload Scheduler for z/OS environment a distributed shadow job, make sure that:

- The remote job stream name specified, contains the job stream name as it is defined in the database.
- The remote job name specified, contains the alias, if defined, of the remote job to bind.

If you do not follow these guidelines, the bind fails and the status of the shadow job becomes ERROR.

For more details about the settings that define a shadow job, see “Specifying remote job information in shadow jobs” on page 164.

Step 4. Add a dependency on the shadow job

After defining the shadow job you can add the cross dependency.

Add a cross dependency on a job instance scheduled in a remote engine plan, by defining a normal dependency on the shadow job that:

- Points to the remote job on which you want to create the cross dependency.
- Is defined on a local remote engine workstation that points to the engine where the remote job is defined.

Because the remote job instance status transition is mapped into the shadow job status transition, the status of the cross dependency is represented by the status of the normal dependency.

Daily plan consistency checks on shadow jobs and remote engine workstations

Depending on the type of workstation selected, some fields in the job definition become mandatory. To avoid incongruences, the daily plan batch program, at plan creation or extension, performs consistency checks to ensure that no mismatch has occurred in between the definition of jobs and workstations in the database and their inclusion in the current plan.

Based on the results of the checks, the daily plan batch program can:

- Complete successfully.
- Generate the plan, providing a return code and a warning message.
- Fail to generate the plan, issuing an error message documenting the exception that occurred.

The possible results of the checks run by the daily plan batch against shadow jobs and remote engine workstations are defined in the following sections.
Daily plan consistency checks on shadow jobs and remote engine workstations

The daily plan batch program generates the plan providing a return code 4 and logging a warning message in the report, if the workstation involved was not yet present in the plan and one of the following situations occurred:

A job, defined on a non-remote engine workstation, contains remote job information.

This situation might occur, for example, if the type of workstation changed from remote engine into another type after the job was defined. In this case the job is no longer considered as a shadow job, but its definition still includes the information that applies to a shadow job.

The daily plan completes with return code 4, message EQQ0396W is logged in the DP batch report, the job is added to the plan, and its remote engine information settings are ignored.

A job, defined on a remote engine workstation, does not contain remote job information.

This situation might occur, for example, if the type of workstation changed to remote engine type after the job was defined. In this case, the jobs defined on that workstation are automatically considered as shadow jobs and so they lack the remote job information.

The daily plan completes with return code 4, message EQQ0391W is logged in the DP batch report, and the occurrence is not added to the plan.

The remote engine type of a workstation was changed causing an inconsistency in the remote job information set in the shadow job.

Because the remote engine type determines the set of information that is necessary to map a shadow job to a remote job instance, if you change the remote engine type you cause an inconsistency in the shadow job definition.

In this case, the daily plan completes with return code 4, message EQQ0392W is logged in the DP batch report, and the occurrence is not added to the plan.

The daily plan batch program fails, at plan extension or replan time, if the workstation is already present in the plan and one of the following situations occurs:

The remote engine type of a workstation existing in the plan was changed.

The daily plan fails and message EQQ0397E is logged in the DP batch report.

The type of a workstation existing in the plan was changed.

The daily plan fails and message EQQ0316E is logged in the DP batch report.

Monitoring a cross dependency resolution in the current plan

Shadow jobs are added to the plan in the same way as any other job:

- At run time, if a shadow job is added to an occurrence existing in the current plan or if a new occurrence containing the shadow job is added to the current plan.
- When the plan is extended or created again and its time frame includes the shadow job input arrival time.

As soon as a shadow job instance is added to the plan, you can start monitoring its status.
How the shadow job status changes until the bind is established

Figure 164 summarizes how the shadow job status changes until the bind is established. The bind status of the shadow job is represented as STATUS - Extended status. You can see the extended status of a job shown in the SELECTING APPLICATION OCCURRENCE AND OPERATION INFORMATION panel.

The initial status of the shadow job is WAITING (W). It changes to READY (R) as soon as the shadow job becomes free from dependencies and is ready to start.

The scheduler then sends an HTTP request to the remote engine asking to bind that shadow job instance with the instance of the remote job that closest precedes the shadow job input arrival in the remote engine plan. The HTTP request contains both the information to identify the shadow job in the current plan and the information to uniquely identify the remote job instance in the remote engine plan. The scheduler requires, as well, to be notified about the status of the remote job instance bound.
As soon as the remote engine receives the HTTP request it tries to identify in its plan the job stream instance to bind. The definition of that job stream must contain the definition of the remote job to bind.

**Note:** On a distributed remote engine it can be specified which job can be bound by assigning specific authorization to the user ID that runs the bind in the plan. If that user ID has not the needed authorization to run the bind, the bind fails. For more information, see the *Tivoli Workload Scheduler: User's guide and reference*.

Depending on whether the shadow job is a z/OS or a distributed job, the way to associate it with a remote job instance is slightly different. The following sections describe how the job instance is matched in the remote engine plan.

**How a distributed shadow job is bound with a remote job instance**

On a Tivoli Workload Scheduler distributed remote engine, the bind is established in the preproduction plan. Distributed remote job instances belonging to the JOBS job stream and submitted at runtime are not written into the preproduction plan and so they are not involved in the bind process.

If the preproduction plan does not exist on the remote engine when the bind request is received, the distributed shadow job status remains WAIT until the preproduction plan generation is completed and the bind request is processed. This might happen, for example, when the preproduction plan is created again from scratch on the remote engine.

The preproduction plan does not contain the instances of the jobs belonging to the job streams. The job instances are added later on to the production plan when the plan is extended or created.

This means that the bind is established with the job stream instance included in the preproduction plan and that is scheduled to run on the specified workstation. The definition of that job stream must contain the definition of the remote job to bind.

The following sections describe the scenarios that can occur when binding a distributed shadow job having:

- Input arrival (IA): 18:00
- Remote job information:
  - Job stream name: JS2
  - Job name: JOB2
  - Job stream workstation: WKST2

In the figures:
- The white box indicates the time interval covered by the preproduction plan.
- The light grey box indicates the time interval covered by the production plan.
- The dark grey box indicates the interval in the remote engine plan during which the job instance to bind must be searched.
- The JS2 instances shown are scheduled to run on workstation WKST2.
- The JS2 occurrence highlighted in bold is the instance selected for the bind.

**Scenario 1: The production plan contains the instance of job stream JS2 that closest precedes the shadow job input arrival.**

Both, in the case where the shadow job input arrival is included in the
How a distributed shadow job is bound with a remote job instance

production plan interval, as shown in Figure 165.

Figure 165. Instance to be bound if the shadow job input arrival is included in the production plan interval

and in the case where the bind interval extends outside the production plan interval, as shown in Figure 166.

Figure 166. Instance to be bound if the bind interval extends outside the production plan interval

the JS2 instance that closest precedes the shadow job input arrival is selected for the bind. This means that the shadow job and the remote job stream instance are now associated. If, at a later time, a new instance of JS2 that more closely precedes the shadow job input arrival is submitted ad hoc in the remote engine plan, the match with the JS2 instance selected for the bind is not modified.

One of the following situations can now occur:

The selected JS2 instance contains JOB2

The bind is established and a notification containing:

- The remote job information identifying the JOB2 instance in the remote engine plan
- The current status of that JOB2 instance in the remote engine plan

is sent back to the Tivoli Workload Scheduler for z/OS controller, the shadow job instance is updated with the remote job information, and its status is updated accordingly.

The selected JS2 instance no longer contains JOB2 because it was either canceled or completed, and was removed from the production plan.

The bind is established and a notification informing about the
How a distributed shadow job is bound with a remote job instance

Successful execution status is sent back to the Tivoli Workload Scheduler for z/OS controller. The shadow job instance is marked as completed. Its successors can start.

**The selected JS2 instance contains JOB2, but JOB2 is in canceled status**

The selected JS2 instance contains JOB2, but JOB2 is in canceled status. The bind is established and a notification informing about the successful execution status is sent back to the Tivoli Workload Scheduler for z/OS controller. The shadow job instance is marked as completed. Its successors can start.

**Scenario 2: The production plan interval contains the input arrival of the shadow job, but no instances of job stream JS2 exist.**

Figure 167 shows the case where a JS2 instance that closest precedes the shadow job input arrival does not exist.

In this case, the bind with JOB2 fails. A notification informing that the bind failed is sent back to the Tivoli Workload Scheduler for z/OS controller, and the shadow job status is updated according to what you set in the Complete if bind fails field.

**Scenario 3: The preproduction plan interval contains the input arrival of the shadow job and the production plan does not yet include the closest preceding JS2 instance.**

Figure 168 shows the JS2 instance that can be associated to the shadow job, even though the job JOB2 is not yet in the production plan.

A notification informing that the bind is established is sent back to the Tivoli Workload Scheduler for z/OS controller and the status of the shadow job is set to "READY-The bind was established".
How a distributed shadow job is bound with a remote job instance

Scenario 4: The preproduction plan interval still does not contain the input arrival of the shadow job.

Figure 169 shows that no JS2 instance can be associated with the shadow job, because, until the preproduction plan includes the shadow job input arrival, closer preceding JS2 instances can still be added.

In this case, the bind request is put on hold until the preproduction plan is extended to include the shadow job input arrival. Until then the status of the shadow job remains "READY-Waiting for bind result".

How a z/OS shadow job is bound with a remote job instance

If the remote engine is a Tivoli Workload Scheduler for z/OS controller, the search for the remote instance to bind is done as follows:

- First, the instance is searched in the Long Term Plan (LTP) in the part of the bind interval that follows the Current Plan (CP) end time and precedes the shadow job input arrival.
- If no instance is found, the instance is searched in the CP in the part of the bind interval that precedes the current plan end.

Note: If the remote controller receives a bind request with a client notify URI that is not defined among the HTTP destinations, the bind request is discarded and the message EQQHT62E is logged in the MLOG.

The following sections describe the scenarios that can occur when binding a z/OS shadow job having:

- Input arrival (IA): 18:00
- Remote job information:
  - Application ID: JS2
  - Operation number: OP2

In the figures:

- The white box indicates the time interval covered by the LTP.
- The light grey box indicates the time interval covered by the CP.
- The dark grey box indicates the interval in the remote engine plan during which the job instance to bind must be searched.
- The JS2 occurrence highlighted in bold is the instance selected for the bind.

Scenario 1: The CP interval contains the input arrival of the shadow job and JS2 occurrences exist.
How a z/OS shadow job is bound with a remote job instance

Figure 170 shows, highlighted in bold, the JS2 instance that more closely precedes the shadow job input arrival. This instance is selected for the bind because the input arrival is contained in the CP. The shadow job and the remote job instance are associated. If, at a later time, a new instance of JS2 that closest precedes the shadow job input arrival is submitted ad hoc in the remote engine plan, the match with the JS2 instance selected for the bind is not modified.

At this point, one of the following situations can occur:

The selected JS2 instance contains OP2

The bind with OP2 belonging to JS2 is established and a notification containing:

- The information necessary to identify the OP2 instance in the remote engine plan
- The current status of that OP2 instance

is sent back to the Tivoli Workload Scheduler for z/OS controller, the shadow job instance is updated with the remote job information, and its status is updated accordingly.

The selected JS2 instance no longer contains OP2 because either it was deleted and a daily plan removed it from the CP, or it was never contained in JS2.

The bind fails. A notification informing that the bind failed is sent back to the Tivoli Workload Scheduler for z/OS controller, and the shadow job status is updated according to what you set in the Complete if bind fails field.

The selected JS2 instance contains OP2 that was deleted but not yet removed from the CP

The bind is established and a notification informing about the successful execution status is sent back to the Tivoli Workload Scheduler for z/OS controller. The shadow job instance is marked as completed. Its successors can start.

Scenario 2: The current plan interval contains the shadow job input arrival, the JS2 instance that most closely precedes the shadow job input arrival exist in the LTP but it was canceled from the CP.
How a z/OS shadow job is bound with a remote job instance

---

**Figure 171.** Instance to be bound if the instance that most closely precedes the shadow job input arrival exist in the LTP but it was canceled from the CP

The bind with OP2 is established and a notification containing:

- The information necessary to identify the OP2 instance in the remote engine plan
- The current status of that OP2 instance

is sent back to the Tivoli Workload Scheduler for z/OS controller, the shadow job instance is updated with the remote job information, and its status is updated accordingly.

**Scenario 3: The CP interval contains the input arrival of the shadow job but no JS2 instance exists**

**Figure 172** shows that a JS2 instance that closely precedes the shadow job input arrival does not exist.

The bind fails. A notification informing that the bind failed is sent back to the Tivoli Workload Scheduler for z/OS controller, and the shadow job status is updated according to what you set in the Complete if bind fails field.

**Scenario 4: The LTP interval contains the input arrival of the shadow job and the CP does not yet include the closest preceding JS2 instance.**
How a z/OS shadow job is bound with a remote job instance

A notification informing that the bind is established is sent back to the Tivoli Workload Scheduler for z/OS controller and the status of the shadow job is set to READY-The bind was established.

Scenario 5: The LTP interval still does not contain the input arrival of the shadow job.

A notification informing that the bind is established is sent back to the Tivoli Workload Scheduler for z/OS controller and the status of the shadow job is set to READY-The bind was established.

Monitoring the shadow job status changes after the bind is established

As soon as the bind is established, the remote engine sends back an HTTP notification containing the status of the bind and, if the bind was successful, the information to identify the remote job instance bound. This information is saved in the shadow job instance details.

To see these details, choose option 14 (REMOTE JOB INFO) in the SELECTING APPLICATION OCCURRENCE AND OPERATION INFORMATION panel. A panel containing the following information is displayed:

**Shadow job data:**

- **IA**
  - 18:00
- **JS2**
  - 16:00
  - 17:00
  - 19:00
- **Bind interval**
- **Current plan interval**
  - IA
  - 18:00
- **Long term plan interval**
Monitoring the shadow job status changes after the bind is established

- Application name
- Input arrival
- Operation
- Job name

**Complete if bind fails**
It determines if the operation must be forced to COMPLETE status when the bind process fails.

**Bind status**
It can be one of the following:
- Sending bind request
- Waiting for bind result
- The bind was established
- The bind failed

Depending on the type of remote engine, the following information about the remote job instance is shown:

**The remote engine type is z/OS**
- Application ID
- Input arrival
- Operation number
- Workstation
- Job name

**The remote engine type is distributed**
- Job stream name
- Input arrival
- Job stream workstation
- Job name

Notifications about job status changes are sent asynchronously from the remote engine. These notifications are used to map remote job status transition into shadow job status transition. Store and forward mechanism ensures the delivery of the messages and the recovery in case of failures. Figure 175 on page 420 shows how the status of a distributed shadow job changes, from when the bind is established until the shadow job status becomes COMPLETE (C).
Monitoring the shadow job status changes after the bind is established

Figure 175. Distributed shadow job status transition chain after the bind was established

- Started - Executing
  - A match with a job instance is found in the remote engine plan. An event is sent back with remote job instance details.
  - The remote job instance:
    - was deleted in the preproduction plan
    - was cancelled in the production plan
    - does not exist in the selected job stream
  - The bind was established
  - The remote job failed.
- Error
  - The remote job: completed successfully
  - was cancelled
  - The submission of the remote job failed.
  - The remote job is rerun.
- Complete
  - If you set the shadow job to Complete if bind fails=N.
  - The error code is FBND.
  - If you set the shadow job to Complete if bind fails=Y.

Figure 176 on page 421 shows how the status of the z/OS shadow job changes, from when the bind is established until the shadow job status becomes COMPLETE (C).
If the remote job instance has already completed successfully when the match is done, the shadow job status becomes C immediately.

If the shadow job ends in ERROR, an error code explains why the job failed. For example, if the job failed because the bind failed then the error code is FBND.

ERROR status is not considered a final status for the shadow job, because the remote job instance bound can be rerun at a later time. Only status C is considered as the final status for a shadow job.

As soon as the shadow job status satisfies the dependency rule, the dependency of the local job from the shadow job is resolved, and the cross dependency for the local job on the remote job is also resolved.

**Modifying shadow job instances in the current plan**

In a shadow job instance you can modify the following settings:

**The remote job data that is used to bind the shadow job to a job instance in the remote engine plan**

To modify the remote job data, select option 11 - Remote job info in the MODIFYING AN OPERATION IN THE CURRENT PLAN (EQQMMODP) panel. Depending on whether the shadow job is distributed or z/OS, one the following panels opens:

**MODIFY DISTRIBUTED JOB INFORMATION IN THE CURRENT PLAN (EQQMMRDP)**

In this panel you can modify any of the following:
Modifying shadow job instances in the current plan

- Job stream name
- Job stream workstation
- Job name

MODIFY Z/OS JOB INFORMATION IN THE CURRENT PLAN (EQQMMRZP)
In this panel you can modify any of the following:
- Application ID
- Operation number

Changes applied to these settings affect bind requests not yet triggered or shadow job reruns, they neither affect bind requests already in progress or established, nor update the shadow job definition in the database.

The workstation name
To modify the workstation name, select the job instance in the MODIFYING OPERATIONS IN THE CURRENT PLAN (EQQMOPRL) panel and update the workstation name field. Depending on the type of the workstations, both the original and the new one, the following situations can occur:

If you change from a non remote engine workstation to a remote engine workstation:
Because only shadow jobs are supported on remote engine workstations, you are requested to add remote job data to the job instance definition before saving the change.

If you change from a remote engine workstation to a non remote engine workstation:
Because shadow jobs are not supported on non remote engine workstations, when you save the changes the following occurs:
- All information contained in the remote job data fields are deleted.
- If a bind request was in progress, notifications are discarded.
- If a remote job instance was already bound the asynchronous notifications about remote job status are discarded.

If you change from a remote engine workstation to another remote engine workstation:
When you save the changes the following occurs:
- If a bind request was in progress, notifications are discarded.
- If a remote job instance was already bound, the information about the job instance matched in the remote engine plan are deleted and the asynchronous notifications about remote job status are discarded.

If the types of the remote engine workstations are different, the information necessary to bind the remote job are deleted and you are requested to add new remote job data before saving the change.

Shadow job status transition during the remote job recovery or rerun
If the shadow job is used to map locally the status transition of an remote job with recovery job, prompt or options defined, the following behavior occurs:
- The shadow job status remains STARTED with extended status RECOVERY_IN_PROGRESS while the remote job recovery is in progress.
The shadow job status is updated only when the remote job reaches a final state. The shadow job status will become:

**ERROR**
When the remote job final status notified is FAILED_EXECUTION or FAILED_SUBMISSION..

**COMPLETE**
When the remote job final status notified is SUCCEEDED_EXECUTION.

It might happen that the remote job is rerun, either manually or as result of an automatic recovery. If the remote job was already completed successfully then the rerun has no impact on the shadow job status because it was already SUCC. If the remote job rerun requires the change of the shadow job status from ERROR to STARTED, Tivoli Workload Scheduler for z/OS applies the change if:

- The shadow job has no conditional successors.
- The shadow job is a conditional predecessor and its status change does not cause inconsistencies in the condition evaluation.

If the shadow job status change from ERROR to STARTED causes inconsistencies in evaluating conditions where the shadow job is a conditional predecessor, Tivoli Workload Scheduler for z/OS automatically sets the status of the shadow job to ERROR with error code OSEQ.

You can see the information about the remote job status:
- In the extended status of the shadow job displayed on the SELECTING APPLICATION OCCURRENCE AN OPERATION INFORMATION panel.
- In the Extra Information fields in the shadow job properties panel on the Dynamic Workload Console.

### Modifying remote engine workstations in the current plan

You can modify the status of remote engine workstation instances in the current plan from the MODIFYING A WORKSTATION IN THE CURRENT PLAN (EQQMWSRP) panel.

If you set this type of workstation to OFFLINE, the outgoing bind requests to the remote engine referenced in the remote engine workstation are queued until the workstation status becomes ACTIVE. The status of a remote engine workstation does not affect the outgoing bind requests already triggered or the incoming bind requests.

### Contacting the remote engine in a failover scenario

On the Tivoli Workload Scheduler distributed system, the management can be switched from the current master domain manager to a backup master domain manager. In this case, when your distributed remote engine workstation tries to contact the master domain manager its status becomes OFFLINE. To prevent the bind process from failing:

1. Define a distributed remote engine workstation for the master domain manager and for each backup master domain manager.
2. Define each remote engine workstation as the alternate workstation for the previous one.

For example, in your z/OS scheduling environment you have defined a distributed remote engine workstation RENG that points to the master domain manager MDM on the distributed Tivoli Workload Scheduler. You have also defined the remote
Contacting the remote engine in a failover scenario

Engine workstations REW1 and REW2, respectively pointing to the backup master domain managers BMDM1 and BMDM2. If the master domain manager is switched to a backup master domain manager, to prevent the bind process from failing, define REW1 as the alternative workstation for RENG, and REW2 as the alternative workstation for REW1.

<table>
<thead>
<tr>
<th>Local TWS for z/OS environment</th>
<th>Remote TWS distributed environment</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Diagram" /></td>
<td><img src="image" alt="Diagram" /></td>
</tr>
</tbody>
</table>

The routing of workload between primary and alternate remote engine workstations works also when the Tivoli Workload Scheduler engine on the remote master domain manager that you switched off remains active.

If all the primary and alternate remote engine workstations are OFFLINE, the requests are queued and will be managed by the first of these workstations that will become ACTIVE again. If more than one of these workstations becomes ACTIVE simultaneously, the requests will be managed by the highest remote engine workstation in the cascade.

When the remote engine is a z/OS controller the failover scenario is managed automatically by the SYSPLEX so you do not need to define any alternate remote engine workstation. In fact, a single HTTP address is sufficient to manage in a transparent way failover scenarios because the configuration with a hot standby controller is supported through Dynamic Virtual Internet Protocol Addressing - VIPA and controller and standby controller listen on the same hostname and port.
Chapter 21. Running event-driven workload automation

Event-driven workload automation (EDWA) addresses on-demand workload automation in addition to plan-based job scheduling.

Using event-driven workload automation you can carry out a predefined set of actions in response to events that occur in environments where the scheduler runs.

This process is based on the following concepts:

**Event**

The event that the scheduler can detect for action triggering. It can be:
- Activity impacting a data set, when SMF writes a record that traces the closure of one of the following types of file:
  - Data set
  - GDG file
  - VSAM cluster
- HFS or ZFS file changes.
- Job submission outside Tivoli Workload Scheduler for z/OS.
- Special resource status change, requested by an operator command.

**Action**

The action that the scheduler runs when the defined condition is satisfied. It can be:
- Add a special resource to the current plan or modify an existing resource, after the tracker generated an event of special resource status change.
- Releasing a dependency on an operation that runs file watching.
- Add an occurrence to the current plan.

**Condition**

The event condition that is to be monitored, based on specified filtering criteria.

---

**Business scenario**

The business analyst of a Bank Information Center requires that Tivoli Workload Scheduler for z/OS produces periodic reports on daily data, provided by a bank employee using FTP to update a data set located on a tracker system.

To automate this process, the scheduling administrator creates an application where the job producing the report depends on a special resource that corresponds to the data set to be updated by FTP.

When the update process ends, the scheduler submits the job to produce a formatted report.

---

**The event-driven process**

Event-driven workload automation is an ongoing process and can be reduced to the following steps:
The event-driven process

1. The administrator defines the event rule, that is the criteria to correlate event conditions and actions.
2. The administrator builds and activates triggering selection data based on the previous criteria, starting the event-driven process: on each system where the scheduler runs, specific components act as event or action provider. The action provider access the data correlating event conditions and actions, according to the following schema:

<table>
<thead>
<tr>
<th>Event Provider</th>
<th>Action Provider</th>
<th>Correlation data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data set triggering</td>
<td>SMF</td>
<td>Tracker, requesting to change special resource availability</td>
</tr>
<tr>
<td>HFS or ZFS file triggering</td>
<td>Tracker, handling jobs that run file watching</td>
<td>Controller, releasing dependencies on operations that run file watching</td>
</tr>
<tr>
<td>Event-triggered tracking (ETT)</td>
<td>JES • user request to change special resource availability</td>
<td>Controller, adding an occurrence to the current plan</td>
</tr>
</tbody>
</table>

The following sections describe how you can set up this process. In particular:
- To trigger an action on a special resource, refer to “Data set triggering.”
- To trigger an action based on HFS or ZFS file changes, refer to “HFS or ZFS file triggering” on page 429.
- To trigger workload submission (adding an occurrence), define the correlation as described in “Adding occurrences by event-triggered tracking” on page 432.

Data set triggering

To build the EQQJCLIB member, you can:
- Write a series of EQQSELECT macros as described in “Invoking the EQQSELECT macro” on page 760, compile them using the EQQLSJCL member of the SEQQSAMP library, name the output file EQQDSLST and place it in the partitioned data set specified by the EQQJCLIB DD statement in the tracker start process. You have to manually populate the EQQJCLIB data set for each system where data set monitoring must be active.
- Using a single job, that you can submit from the product dialog, customize the triggering criteria for all your trackers, build configuration files based on that criteria and deploy them at tracker side. These configuration files are automatically placed as EQQEVLST members in the EQQJCLIB data set. For a description of these tasks, refer to the following sections:
  - “Defining event rules” on page 427
  - “Producing the configuration files” on page 427
They require preliminary steps for the system where the controller runs, as described in Planning and Installation, section Activating support for the Java utilities.

Your EQQJCLIB data set can contain EQQEVLST data only, EQQDSLST data only, or both EQQEVLST and EQQDSLST data.

To decide which kind of selection table is best for your purpose, consider that customizing EQQEVLST provides the following capabilities:

- Centralized deploy process.
- The process that monitors data set closure can distinguish if a data set is closed after it was opened in read or write mode (by checking the ReadCompleted or ModificationCompleted values specified in the rule as eventType parameter, described in Appendix G, “Defining event rules and invoking EQQLSENT macro,” on page 753).

### Defining event rules

A scheduling event rule defines a set of actions that are to run upon the occurrence of specific event conditions. The definition of an event rule correlates events and triggers actions. An event rule definition is identified by a rule name and by a set of attributes. It includes information related to the specific events (eventCondition) that the rule must detect and the specific actions it is to trigger upon their detection (action).

The SEQQXML01 member contains EQQXML01 member as sample of event rule definition. For details about the XML syntax, refer to Appendix G, “Defining event rules and invoking EQQLSENT macro,” on page 753.

### Producing the configuration files

Based on the current routing options (ROUTOPTS initialization statement), the controller creates your configuration files in the EQQEVLIB data set, that is a partitioned data set where each member name correspond to a tracker destination.

When the controller starts, it updates the EQQEVLIB data set by creating a member for each active destination. The controller leaves unchanged any existing member with the same name as an active destination, and empties any member that does not correspond to an active destination.

To populate the configuration files for all the defined tracker destinations, perform the following steps:

1. Select option 1.7.3 from the main menu.
2. Press Enter to submit or edit a JCL to run the EQQRXTRG program, that performs the following:
   a. Reads and parses the defined event rules.
   b. Updates EQQEVLIB data set, according to the following criteria:
      - If the controller is not active when you run the JCL, EQQRXTRG updates the EQQEVLIB data set based on the defined event rules, producing one member for each Destination value.

      Existing members, with the same name as a specified Destination value, are replaced.
      Existing members, with names that do not correspond to any specified Destination value, remain unchanged.
Producing the configuration files

- If the controller is active when you run the JCL, EQQRXTRG updates the EQQEVLIB data set based on the defined event rules, replacing any existing member corresponding to a specified Destination value.

  Existing members, with names that do not correspond to any Destination value, remain unchanged.

  When detecting a Destination value that does not correspond to any member in the EQQEVLIB data set, EQQRXTRG returns an error message without creating a new member.

The process formats each EQQEVLIB member data according to a trigger table layout mapped by the TRG data area. It includes a header with a Cyclic Redundancy Check (CRC) number, to univocally identify the configuration file: changing the input to the EQQRXTRG program produces a different CRC number.

Any comment inserted in the XML source file is not stored in the formatted output.

For details about the TRG data area, refer to Diagnosis Guide and Reference.

c. Optionally, issues a modify command with the DEPLOYCF option, to immediately start the deploy process for the configuration files. You can add the modify command in the JCL source before submitting it, or manually issue the same command later. For details about the deploy process, refer to “Deploying configuration files.”

Deploying configuration files

When starting the deploy process, the controller loads the CRC numbers and uses them to check whether or not any connected tracker needs a new copy of the configuration file.

As result of the deploy process, each tracker that:

- Is connected to the controller
- Is at the same level as the controller
- Has its destination associated with a workstation present in the current plan

receives an event configuration file updated according to the current content of the controller EQQEVLIB data set. The refreshed configuration file is stored in the EQQEVLST member of the tracker EQQJCLIB data set.

Note: If EQQJCLIB contains also the EQQDSLST member, the resulting triggering selection table is the union of EQQEVLST and EQQDSLST.

Then, each tracker monitors for the events that match the refreshed data, based on an in-storage copy of the configuration file.

The deploy process starts in the following cases:

- At tracker connection time.
- Periodically, depending on the frequency of handshakes initiated by the trackers, specified by the PULSE parameter of the ROUTOPTS initialization statement.
- When processing a modify command with the DEPLOYCF option.

If a tracker destination is referenced in the ROUTOPTS initialization statement, but it is not currently connected, the tracker acquires the configuration file data when the connection is restored.
**Effects on special resource availability**

When the deploy process is completed, an automatic special resource availability handling starts and goes on, as follows:

1. Whenever any file is affected by an action that corresponds to a possible triggering event type, the proper SMF write exit is called to examine the generated SMF record.

2. The exit invokes the tracker subsystem, which compares the file name and other specifics from the SMF record against the entries in the configuration file.

3. If a match is made, the tracker subsystem creates a special resource status change event record and places it on the writer queue (WTRQ) in the extended common service area (ECSA). Following rules apply:
   - For a partitioned data set, the member name is not part of the resource name in the event that the tracker generates.
   - Generation Data Group data sets are specified by the group name. For example, when a GDG data set with the name 'DSN.OPCSUBS.GDG.G0001V00' is closed the special resource event contains resource name 'DSN.OPCSUBS.GDG'.
   - For VSAM data sets the resource name in the generated event is the cluster name (without the DATA or INDEX suffix).
   - If you used wild cards in the FileName parameter values of your event rule definition, the resource name in the generated event is the full name of the matching data set.

   **Note:** This record is an EXS event. It begins with the characters $YY. When browsing the tracker event data set, $YY appears in columns 21-23. The data set name from the SMF record begins in column 53 (x'20'). For more information about the EXS event, refer to *Diagnosis Guide and Reference*.

4. The tracker event WRITER task removes the record from the writer queue, writes it to the event data set, and passes it to the controller.

---

**HFS or ZFS file triggering**

The product package includes a file watching utility. After installing the product, you find this utility as load-module EQQFLWAT in the SEQQLMD0 library. You can use it to check for file system changes of HFS or ZFS files and directories, for example when you want to make sure that a file exists before running a job that processes that file. By defining a job that runs this utility, you can implement file dependency, that is a relationship between a file and an operation in which specific activity on the file determines the starting of the operation. For example, you can define the job that runs EQQFLWAT as predecessor of the operation depending on the file.

The user running EQQFLWAT must have execute (x) access to the directory path that contains files to be monitored.

TheSEQQSAMP library contains EQQFLWAT member as sample JCL to call the file watching utility.

**Syntax**

As shown in the EQQFLWAT member of the SEQQSAMP library, the following format is supported for the parameter of the EXEC statement that specifies PGM=EQQFLWAT:
HFS or ZFS file triggering

```
//FLWATCH EXEC PGM=EQQFLWAT,PARM='ENVAR()/-c condval -dea deadline
//   -fi file_path -i interval -r rc -t trace_level'
```

Consider that:
- No continuation character is used. To continue the PARM field, interrupt it at column 72 and continue in column 16 of the next card.
- Separate arguments and values by using a blank character.
- Use ENVAR() to pass environment variables to EQQFLWAT.
- EQQFLWAT returns messages and trace information (if required) in the log of the job used to run the utility. If you use the job log retrieval function, set to Y the User Sysout field in the cleanup options at operation level, both in the database and in the current plan.
- The arguments are not positional.
- You can use an abbreviated format for all the arguments. Generally you can truncate the arguments to any position following the first character, except for deadline that requires at least three characters.

A detailed description follows.

**Arguments**

- **-condition**
  The condition to be checked. Valid values are:

  - **wcr | waitCreated**
    Waits until the file exists. If the file already exists, filewatch exits immediately. If -filename argument specifies a directory, the process waits until the directory exists and contains a new file.

  - **wmr | waitModificationRunning**
    Waits until the file size or modification time changes. If -filename argument specifies a directory, the process waits until the size or earlier file modification time changes, when a file is created, modified, or deleted.

  - **wmc | waitModificationCompleted**
    Checks that the file size or modification time stopped changing, meaning that filewatch waits for three search intervals without any change. If -filename argument specifies a directory, the process checks the size or the earlier file modification time change, for example if the number of directory files and the earlier file modification time does not change within three search intervals.

  - **wmrc | waitModificationRunningCompleted**
    Waits until the file size or modification time changes and stops changing, meaning that, after the first change, filewatch waits for three search intervals without any further change. If -filename argument specifies a directory, the process checks the size or the earlier file modification time change, for example if the number of directory files and the earlier file modification time does not change within three search intervals.

  - **wdl | waitDelete**
    Stops running when the file is deleted. If -filename argument specifies a directory, the process waits until a file is deleted from the directory.
HFS or ZFS file triggering

-filename
The file path to be processed. You can embed blank or special characters, by using double quotes. Wildcard characters are not supported. To include more files in the monitoring process, you can store those files in a specific directory and use a file path specifying that directory.

-deadline
The deadline period, expressed in seconds. The allowed formats are:
- An integer in the range 0 to 31536000 (the upper value corresponds to one year). To have filewatch performing an indefinite loop, specify 0.
- hh:mm:ss, in the range 00:00:01 to 24:00:00, to select a time within the same day when filewatch started.

It corresponds to the GMT value. To specify a value different from the GMT, use the ENVAR parameter as shown in "Example." For details about setting the TZ environment variable, refer to the UNIX System Services Command Reference.

-interval
The file search interval, expressed in seconds. Specify an integer in the range:
- 5–3600 when specifying wcr or wdl as condition value.
- 30–3600 otherwise.

The default is 60.

-returncode
The exit return code, if the file is not found by the deadline. Specify an integer in the range 0 to 256. The returncode value is ignored if you specify 0 as deadline value. The default is 4.

-trace
Trace level for internal logging and traces. Possible values are:

0 To receive error messages only.
1 Indicates the fine level, to receive the most important messages with the lowest volume.
2 Indicates the finer level, to activate entry and exit traces.
3 Indicates the finest level, to receive the most detailed tracing output.

The default value is 0.

You find the trace output in the log of the job that run filewatch.

Example

```
//FLWATCH EXEC PGM=EQQFLWAT,PARM='ENVAR(TZ=GMT-1CET)/-co wcr -dead 30
// -fi /u/falsi/prova -int 30 -t 2'
```

In the previous example ENVAR(GMT-1CET) is used to change the time zone to Central Europe Time. GMT-1 is the time zone value and CET is the selected daylight saving time.
Adding occurrences by event-triggered tracking

Adding occurrences by event-triggered tracking

Event-triggered tracking adds occurrences to the current plan based on a triggering event. You can use ETT to track work that has been submitted outside Tivoli Workload Scheduler for z/OS control, or simply to respond to an on-demand request for processing.

The following sections follow:

- "Triggering event types"
- "Activating ETT" on page 433
- "Defining your ETT criteria" on page 433
- "Automatically adding an occurrence to the current plan" on page 435
- "How you can use ETT to automate tasks" on page 437
- "Using ETT occurrence-related variables" on page 437

Triggering event types

Tivoli Workload Scheduler for z/OS generates events when certain incidents occur on the system. The events generated when a resource is set available, or when a job or started task arrives in JES, can be used to communicate to ETT that a particular occurrence should be added to the current plan.

When you define ETT criteria, you specify which event type is triggering Tivoli Workload Scheduler for z/OS to add an occurrence of a application to the current plan.

J-type

These occur when a job or started task arrives in JES which cannot be matched with an operation in the current plan. The job that has entered the reader can be the actual job that you want Tivoli Workload Scheduler for z/OS to track, or an IEFBR14 that is being used only as a trigger for additional processing. J-events can also be generated when a STATUS(Q) is requested using the OPSTAT command issued either in native TSO, or in a CLIST or REXX EXEC, the EQQEVPGM using OPSTAT as input, or by using the EQQUSIN subroutine.

The first operation of the added application must be defined on the same type of computer workstation as the triggering event. In other words, if the triggering event is a batch job, the first operation of the added application must be defined on a job computer workstation. If the triggering event is a started task, the first operation of the added application must be defined on a started-task (STC) computer workstation. If the workstation types do not match, no application will be added.

R-type

These occur when the availability status of a resource is set to YES. The availability of a resource can be set to YES by the SRSTAT command issued either in native TSO, or in a CLIST or REXX EXEC, the EQQEVPGM batch program using SRSTAT as input, or by the EQQUSIN subroutine. The availability of a resource can also be set automatically when a data set is closed if your installation implements the data set triggering support.

The resource might be one that is also used by operations in the current plan, or it can be a resource dedicated as an ETT trigger.

Note: If you change the availability of a resource on the MODIFYING RESOURCES panel, an R-event is not created.
Tivoli Workload Scheduler for z/OS does not recognize this as an ETT-triggering action. This also applies to resources that are created dynamically either from the Dynamic Workload Console or the PIF. For more information about creating and using resources, see Chapter 5, “Creating special resources,” on page 81.

Activating ETT
To activate ETT automatically when Tivoli Workload Scheduler for z/OS is started, specify ETT(YES) on the JTOPTS statement (see Customization and Tuning).

To activate or deactivate ETT while Tivoli Workload Scheduler for z/OS is active, use the SERVICE FUNCTIONS panel. See Chapter 12, “Using service and optional functions,” on page 289.

Defining your ETT criteria
Follow these steps to specify ETT criteria:
1. Specify option 1 (DATABASE) on the main menu.
2. Specify option 7 (EDWA) on the MAINTAINING TWSz DATA BASES menu.
3. Specify option 2 (MODIFY) on the MAINTAINING EVENT TRIGGERED TRACKING menu.

<table>
<thead>
<tr>
<th>Row</th>
<th>Name of triggering event</th>
<th>Id of associated application</th>
<th>E</th>
<th>J</th>
<th>D</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DAP*</td>
<td>SAMPLEA</td>
<td>J</td>
<td>N</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>2</td>
<td>TESTJOBB</td>
<td>SAMPLEB</td>
<td>J</td>
<td>Y</td>
<td>N</td>
<td>N</td>
</tr>
<tr>
<td>3</td>
<td>SPECIAL RESOURCE 1</td>
<td>SAMPLED</td>
<td>R</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>

---

4. Specify these fields for the criteria:

   **Name of triggering event**
   This is:
   - For a J-event, the name of the job that will act as the trigger
   - For an R-event, the name of the resource whose availability will act as the trigger

   You can specify the triggering-event name generically by using the generic search characters * (asterisk) or % (percent). If there is more than one match for generic filters, the most specific match is selected. For a J-event, for example, you could specify a job name of DAP* as the name of the triggering event. Whenever a job that is outside Tivoli Workload Scheduler for z/OS control with a job name beginning with DAP enters the system, Tivoli Workload Scheduler for z/OS adds an occurrence of SAMPLEA to the current plan.

   **ID of associated application**
   Specify the ID of the application to be triggered.
Defining your ETT criteria

| ET | The type of event (J or R). See “Triggering event types” on page 432. |
| JR | This specifies job-name replace tracking. Specify Y if Tivoli Workload Scheduler for z/OS will track the job that is being used as the trigger. If you specify N, Tivoli Workload Scheduler for z/OS will not track the trigger, and the job serves only as the means for automatically adding an occurrence to the current plan. Use of job-name replace can give you a single point for monitoring all tasks. Tracking the actual job also means that powerful error detection capabilities and dependency control will be available to you for that operation. |

For example, the JCL for IMS recoveries can be built by DBRC. If you want to track the recoveries and use Tivoli Workload Scheduler for z/OS to detect errors and establish dependencies, define the tasks to ETT with job-name replace tracking. You would need only a single application description in the AD for all DBRC-generated recoveries. When the occurrence is added to the current plan, the job name of the first operation, which must be defined on a z/OS computer workstation, is changed to match the actual name of the triggering job.

**Note:** You can specify Y only for J-type triggering events.

| DR | Specify dependency resolution for the occurrence to be added. As with the MCP panel and the program interface, you can choose: |
| Y | Resolve both predecessor and successor dependencies. |
| N | Ignore all dependencies. |
| P | Resolve only predecessor dependencies. |
| S | Resolve only successor dependencies. |

| AS | If this is N, ETT adds an occurrence each time the availability status is set to YES on a R-event. If you want an occurrence added only in the case of a true availability status switch for a resource from available=no to available=yes, set the availability status switch to Y. The availability status switch is valid only if the event type is R. If the event type is J, this field should have the value N, or blank. |

**Note:** Be aware that if an SRSTAT command (or equivalent) is issued for a resource that does not exist in either the resource definition database (panel option 1.6) or in the special resource monitor (panel option 5.7), Tivoli Workload Scheduler for z/OS reacts differently depending on whether the SRSTAT create option is set to YES or NO:

- If you issue an SRSTAT command with CREATE(NO) AVAIL(YES), no resource is created in the plan and the resource status changes from NO to YES, then immediately back to NO. Each time the SRSTAT command is repeated, another availability switch to YES occurs, and ETT processing adds another occurrence, even if the ETT criteria has AS set to Y.
- If you issue an SRSTAT command with CREATE(YES) AVAIL(YES), the resource is created in the special resource monitor, but Tivoli Workload Scheduler for z/OS does not count this creation as a real status change, and if AS in the ETT definition is set to Y, then no ETT triggering occurs.

Thus for predictable processing, if you wish to use the ETT ACTUAL STATUS option, all special resources, for which this
option is to be set, should be predefined in the MAINTAINING
SPECIAL RESOURCES panel (option 1.6)

Automatically adding an occurrence to the current plan

You use a J-event or an R-event to trigger the addition of an application occurrence
to the current plan. This is equivalent to adding an occurrence using the
MODIFYING CURRENT PLAN panel.

When the J or R triggering event occurs, the application occurrence is added to the
current plan. If possible, its input arrival time is set to the time of the triggering
event. If this is not possible, because an occurrence of the application already exists
with that input arrival time, the next closest time available is used. If Tivoli
Workload Scheduler for z/OS cannot allocate an input arrival time before the end
of the current plan, the occurrence will not be added.

The application description of the occurrence to be added must exist in the AD
database. If the application has run cycles specified in the AD database, the
deadline date and time of the occurrence are calculated based on the first run cycle
that is found. Otherwise, they are set to the input arrival time of the triggering
event, plus 8 hours. The input arrival time of operations within the occurrence is
calculated using the operation IATs in the AD database.

To control the calculation of the deadline date and time plan, you can define a
special period, ETTRCY2, and create a run cycle in the application description that
refers to that period. The period must be called ETTRCY2. Define the period as
noncyclic, with an origin date of 71/12/31, 31 December 2071, which is the Tivoli
Workload Scheduler for z/OS high date. Such a date satisfies requirements for
noncyclic period definitions but ensures that applications that refer to the period
definition are not accidentally included in the long-term plan. Set the run cycle
input arrival time as 00.00, and the deadline day and time as the offset you want
to add to the actual input arrival time. If an ETT-added occurrence contains a run
cycle that uses the period ETTRCY2, Tivoli Workload Scheduler for z/OS uses the
deadline day and time defined on that run cycle to calculate the deadline from the
actual input arrival time, which by default is the triggering-event time.

If the application has external dependencies, you can specify if these dependencies
should be automatically resolved when the occurrence is added to the current plan.
An occurrence that has been added to the current plan by ETT will not become the
predecessor to an occurrence that is added later by daily planning, even if normal
dependency criteria are met.

If the ETT criteria specifies job-name replace, the job name of the first operation is
changed to that of the triggering job name. If the first operation has a successor
print operation, the job name of the print operation is also changed. The first
operation is the one with the lowest operation number. The operation cannot have
internal predecessors and must be defined on a computer workstation. Tivoli
Workload Scheduler for z/OS ignores the TRACK keyword of the JTOPTS
statement for jobname replace. The operation's SUBMIT option is set to NO.

Regardless of whether you use job-name replace, the application occurrence added
to the plan can contain up to 255 operations. If you need to trigger the addition of
multiple applications from a single triggering event, define the ETT trigger for the
first application. The first application can then contain a job to set the availability
of a resource to YES, which in turn matches another ETT definition, and the
second application is added to the current plan.
Automatically adding an occurrence to the current plan

If you use ETT to trigger a job when a resource becomes available, you might get into an occurrence-add loop under certain situations. For example, if you want Tivoli Workload Scheduler for z/OS to submit job A when ETT detects the availability of resource X, and X originally has an availability of NO, the following can occur:

1. Special resource X becomes available (that is, its availability status changes to YES), and ETT detects the change.
2. In response to the R-event, ETT causes job A to be submitted.
3. Job A uses resource X, changes its availability status back to YES, and then terminates.
4. ETT detects the changed availability of X to YES and submits job A again.

In this situation, job A is continually submitted.

Note: When you set the availability of a resource to YES, Tivoli Workload Scheduler for z/OS generates an R-event regardless of the initial status of the resource. If a resource is currently available and you issue an SRSTAT specifying AVAIL(YES), an R-event is generated.

Normally, when an occurrence is added to the plan and resolution of successor dependencies is requested, the input arrival time of the occurrence is used to identify the best candidate operations. ETT-added occurrences are, by their nature, not very predictable. The input arrival time of the occurrence is equally unpredictable.

To control the input arrival time that should be used when establishing predecessor dependencies in the current plan, you can specify an explicit start time on the operation level of the application that is to be added by ETT.

To control the time used when establishing successor dependencies in the current plan, you can define a special period, ETTRCY1 and create a run cycle in the application description that refers to that period. The period must be called ETTRCY1. Define the period as noncyclic, with an origin date of 71/12/31, 31 December 2071, which is the Tivoli Workload Scheduler for z/OS high date. Such a date satisfies requirements for non-cyclic period definitions but ensures that applications that refer to the period definition are not accidentally included in the long-term plan.

If an ETT-added occurrence contains a run cycle that uses the period ETTRCY1, Tivoli Workload Scheduler for z/OS adds the occurrence to the current plan with an input arrival time that corresponds to the current time. As a consequence, to establish successor dependencies Tivoli Workload Scheduler for z/OS uses the time supplied by ETTRCY1, not the occurrence input arrival time. Use the JTOPTS ETTNEWDEP parameter to customize the criteria used by the scheduler when selecting candidate successors. For details, refer to Customization and Tuning.

Note: If you define an application that both uses the period ETTRCY1 and has a dependency on itself (with the same input arrival day), you can add it only twice. A further attempted addition will result in a message being displayed describing the ETT failure.
How you can use ETT to automate tasks

ETT provides a method for automating some tasks currently performed manually. Using ETT, you can reduce processing-request telephone calls or run-request sheets from your operations department.

You can also use the ETT to trigger batch processing from online systems, particularly from on-demand transaction programs. These are ways you might use ETT in your environment:

- Start batch processing after the receipt of unsolicited file transfer.
- Request some processing via SRSTAT, which triggers an occurrence to be added to the plan.
- Start a batch process when NetView intercepts a particular message.
- Provide IF-THEN logic in your batch streams: if a step returns CC4, execute a step to trigger APPLA; if the step returns CC8, execute a different step to trigger APPLB.
- Dynamically add the next occurrence in a sequence, for processing that cannot be determined in advance (database backups that are run only when the database has been changed, for example).
- Start batch processing initiated by an online transaction.

ETT can be used in many different situations to streamline your operation.

Using ETT occurrence-related variables

To determine the type of event that has caused the automatic addition of a new occurrence to the plan, use the variables provided by ETT. For example, consider that a company is asked to produce customer-oriented reports according to an external demand. The on-demand request is created by a job name that identifies the customer and is contained in a new generation of an input GDG data set. You can resolve the on-demand process by using the ETT definition and ETT variables in a single job, as follows:

1. In the CREATING A SPECIAL RESOURCE panel (option 1.6.2 on the main menu), define the special resource that identifies the input GDG data set root.
Using ETT occurrence-related variables

**EQQDCRP -------------- CREATING A SPECIAL RESOURCE --------------**

Select one of the following:

1. INTERVALS - Specify intervals
2. WS - Modify default connected work stations

**SPECIAL RESOURCE** -> TWS.REQUESTS.GDG

**TEXT** ->

**SPECRES GROUP ID** ->

**Hiperbatch** -> Y DLF object Y or N

**USED FOR** -> B Planning and control C, P, B or N

**ON ERROR** -> On error action F, FS, FX, K or blank

**ON COMPLETE** -> On complete action Y, N, R or blank

**MAX USAGE LIMIT** -> Max number of allocations before usage reset

**MAX USAGE TYPE** -> Y Status change type Y, N or R

**Defaults**

**QUANTITY** -> 1 Number available 1-99999

**AVAILABLE** -> Y Available Y or N

Figure 179. EQQDCRP - Creating a special resource

2. In the MODIFYING ETT TRACKING CRITERIA panel, specify the ETT criteria for the GDG data set root.

**EQQMTCL ------------- MODIFYING ETT TRACKING CRITERIA ------------**

Command -> Scroll -> PAGE

Change data in the rows, and/or enter any of the following row commands:

I(nn) - Insert, R(nn), RR(nn) - repeat, D(nn), DD - Delete

<table>
<thead>
<tr>
<th>Row</th>
<th>Name of triggering event</th>
<th>ID of associated E J D A application</th>
<th>T R R S</th>
</tr>
</thead>
<tbody>
<tr>
<td>'&quot;'</td>
<td>SETT*</td>
<td>ETTAPPL1</td>
<td>J N Y N</td>
</tr>
<tr>
<td>'&quot;'</td>
<td>TWS.CWSA.CLIST</td>
<td>ETTAPPL2</td>
<td>R N Y N</td>
</tr>
<tr>
<td>'&quot;'</td>
<td>TWS.CWSA.JCL*</td>
<td>ETTAPPL3</td>
<td>R N Y N</td>
</tr>
<tr>
<td>'&quot;'</td>
<td>TWS.REQUESTS.GDG</td>
<td>ETTAPPL4</td>
<td>R N Y N</td>
</tr>
</tbody>
</table>

Figure 180. EQQMTCL - Modifying ETT tracking criteria

3. Update the selection table for the tracker to identify the ETT criteria for the GDG data set root. The file to be edited depends on which kind of selection table the tracker uses:
   - If the tracker uses EQQDSLST as selection table, update the EQQLSJCL member of the sample job JCL library as follows:
If the tracker uses EQQEVLST as selection table, updated it as described in "Data set triggering" on page 426.

4. Write a sample JCL that includes the ETT occurrence-related variables.

5. When you create a request, a new generation of TWS.REQUESTS.GDG is created. The ETT criteria schedule a new occurrence and the variable is resolved.
Using ETT occurrence-related variables

For a detailed description of the ETT occurrence-related variables, see Table 22 on page 449.

```
//TESTJCL JOB (876903.D07), 'AAAAAAA',MSGLEVEL=(1,1),
//MSGCLASS=A,CLASS=A
/*
/*%OPC SCAN
/*******************************************************************************/
/* - READ AND ELABORATE THE NEW REQUEST
/* - CREATE THE OUTPUT REPORT FOR CUSTOMER
/*******************************************************************************/
//STEP1 EXEC PGM=READREQ
//REQUEST DD DISP=SHR,DSN=TWS.REQUEST.GDG..00002V00
//REPORT DD DISP=(,CATLG),DSN=REPORT.USER1,
// SPACE=(TRK,1,1),UNIT=SYSDA,
// DCB=(RECFM=VBA,BLKSIZE=136,LRECL=1360)
/*
```

Figure 183. JCL with variables resolved

For a detailed description of the ETT occurrence-related variables, see Table 22 on page 449.

Combining data set triggering and ETT

After deploying configuration files, you can combine event-triggered resource handling and event-triggered tracking by specifying the FileName data, that you defined in the event rule, as the special resource name in the MODIFYING ETT TRACKING CRITERIA panel.

**Note:** The MODIFYING ETT TRACKING CRITERIA panel does not support ? as generic search character.

Here is a summary of the overall process:

1. **Produce the configuration files.** Same steps as in “Deploying configuration files” on page 428 and “Producing the configuration files” on page 427 are valid.

2. **Deploy your configuration files.** Then, same automatic process as in “Effects on special resource availability” on page 429 starts.

3. After the tracker passed the special resource event to the controller, the controller processes the event exactly the same as it would one created by means of an SRSTAT batch job. The resource name is compared to the ETT trigger definitions, and if a match is made, an occurrence of the specified application is added to the current plan.
Chapter 22. Job tailoring

This chapter describes the job tailoring, which enable jobs to be automatically edited using information that is known only at job setup or submit. This can reduce your dependency on time-consuming and error-prone manual editing of jobs. Job tailoring provides:

- Automatic variable substitution
- Dynamic inclusion or exclusion of inline job statements
- Dynamic inclusion of job statements from the EQQJBLIB data set or from an exit

For jobs to be submitted on a z/OS system, these job statements will be z/OS JCL, but Tivoli Workload Scheduler for z/OS JCL tailoring directives can be included in jobs to be submitted on other supported operating systems. The directives have the same format for all supported operating systems.

An alternative method of substituting variables is to use the Tivoli Workload Scheduler for z/OS program interface (PIF). The PIF lets user-programs issue requests to the Tivoli Workload Scheduler for z/OS subsystem, thereby automating actions that you would otherwise perform with the panels. The PIF resource, JCLPREPA, can simulate variable substitution; in other words, you can perform trial substitution of your variables without actually updating the job.

Note: You must specify values when using a JCLPREPA-call to PIF, otherwise the variable will not be substituted, leaving $, ?, and $ signs in the job.

For more information about PIF, refer to Programming Interfaces.

Variable substitution

Tivoli Workload Scheduler for z/OS supports automatic substitution of variables during job setup and at job submit. Tivoli Workload Scheduler for z/OS also supplies several standard variables, which you can use in your job. You can create your own variables using the OPC JCL VARIABLE TABLES panel; these variables are stored in variable tables in the Tivoli Workload Scheduler for z/OS database. By using the same variable name in different variable tables and associating the tables with different applications, you can make the value the variable takes dependent on the application that it is being used in.

Tivoli Workload Scheduler for z/OS lets you create variables in job statements, in comment statements, and in any in-stream data within the job. You cannot use Tivoli Workload Scheduler for z/OS variables within cataloged or in-stream procedures. Variables occurring in comments are substituted (including comments to the right of job statements). When you create a variable using the JCL VARIABLE DEFINITION panel, you specify whether Tivoli Workload Scheduler for z/OS should substitute it at job setup, at job (or started task) submission, or both.

You can specify that the variable values be supplied in any of these ways:

- Manually
- By a user-written exit
- Automatically from information in the Tivoli Workload Scheduler for z/OS database

A variable that must be supplied by the operator is called a promptable variable.
Variable substitution

The same variables can be used by several applications. The default value of a variable is stored in a variable table. If you have the same variable in different variable tables, make sure that the correct concatenation is in effect when substitution occurs. The order of concatenation can change the value assigned to the variable at substitution (see the example on page 479). The variable table contains the default value for a variable and the rules for evaluating the variable.

**Note:** If a variable is given a value of ∅ (blank), the variable is not substituted if the variable definition states that no value is required. This lets these variables undergo normal operating system substitution after tailoring. If a variable is set to ¬ (logical not) or has this as a default setting, the variable is deleted from the line.

Associating variable tables with applications

Tivoli Workload Scheduler for z/OS searches for values to variables in this sequence:

1. In tables specified for the operation:
   - This varies depending on whether or not you are using JCL or a centralized script.
   - When using JCL or a centralized script:
     a. In the variable table identified in the TABLE directive in the job. See "TABLE directive" on page 479.
     b. In the variable tables identified in the SEARCH directive in the job. See "SEARCH directive" on page 470.
   - When not using JCL or a centralized script, use the VARSUB statement in the script definition in the SCRPTLIB. "Variable substitution for jobs running on fault-tolerant workstations" on page 490 provides information.

2. In tables specified for the application:
   a. Specified using the MODIFYING CURRENT PLAN (MCP) panel.
   b. Specified using the LONG TERM PLAN panel.
   c. In the variable table associated with the run cycle, if any, for the application occurrence. See "Creating run cycles with rules" on page 125.
   d. In the variable table associated with the period, if any, for the application occurrence (for offset-based run cycles only). See "Creating periods" on page 106.

3. In the global variable table. See "Global variable table" on page 463.

Figure 184 on page 444 shows how user-defined variables are matched to individual units of work. User-defined variables are specified in the JCL VARIABLE TABLES panel (A) and are stored in the Tivoli Workload Scheduler for z/OS database.

In the PERIOD panel (B), you can associate a variable table name with a period, and this table is then used if the period is used in an offset-based run cycle.

When creating an application in the APPLICATION DESCRIPTION panel (C), you can specify run cycles to be used when creating occurrences of the application in the plan. You can associate a table with a run cycle. If you use offset-based run cycles, you can associate a table with a period.
Associating variable tables with applications

JCL variable table names for individual occurrences can be modified or added using the LONG TERM PLAN panel (D) or the MODIFYING CURRENT PLAN panel (E).

A JCL variable table name (F) can be referenced by statements within the job by using table directives (see "JCL tailoring directives" on page 466).
Invoking and avoiding variable substitution

To invoke variable substitution, take one of the following actions:

- Set the VARSUB keyword of the OPCOPTS statement to YES. This means that variable substitution occurs from the beginning of the job for all operations defined on setup or computer workstations.

---

**Figure 184. JCL variable handling**
**Invoking and avoiding variable substitution**

- Set the VARSUB keyword of the OPCOPTS statement to SCAN (this is the default) and specify the directive /*%OPC SCAN in your job. Substitution in the job starts where the SCAN directive is found. See “SCAN directive” on page 469 for more information.

In the JCL VARIABLE TABLES panel, specify whether variables will be substituted during the setup of the job or when the job is submitted. This is called the phase of the substitution.

If you want to avoid having Tivoli Workload Scheduler for z/OS scan the job for variables, you must set the VARSUB parameter on the OPCOPTS initialization statement to NO. If you do this, job setup is bypassed on the READY LIST panel-setting next logical status sets the status of a job setup operation to Complete.

If you want to bypass variable substitution errors, then set the VARFAIL keyword. If you want to apply variable substitution also to inline procedures, then use the VARPROC statement, as described.

See Chapter 1 of the *Customization and Tuning* manual for details.

**Coding variables in JCL**

A variable, either user-defined or supplied by Tivoli Workload Scheduler for z/OS, consists of up to 8 alphanumeric characters, the first of which must be alphabetic. When using a variable in a job, precede it with an ampersand (&), a percent sign (%), or a question mark (?). The symbol preceding the variable determines how Tivoli Workload Scheduler for z/OS carries out the variable substitution. A period can denote the end of a variable name. You can use a variable repeatedly within the job using different prefix symbols.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>,</td>
<td>Comma</td>
</tr>
<tr>
<td>/</td>
<td>Forward slash</td>
</tr>
<tr>
<td>'</td>
<td>Single quote</td>
</tr>
<tr>
<td>(</td>
<td>Left parenthesis</td>
</tr>
<tr>
<td>)</td>
<td>Right parenthesis</td>
</tr>
<tr>
<td>*</td>
<td>Asterisk</td>
</tr>
<tr>
<td>&amp;</td>
<td>Ampersand</td>
</tr>
<tr>
<td>&amp;&amp;</td>
<td>Double ampersand</td>
</tr>
<tr>
<td>%</td>
<td>Percent sign</td>
</tr>
<tr>
<td>+</td>
<td>Plus sign</td>
</tr>
<tr>
<td>-</td>
<td>Dash</td>
</tr>
<tr>
<td>=</td>
<td>Equals sign</td>
</tr>
<tr>
<td></td>
<td>Blank ( )</td>
</tr>
<tr>
<td>?</td>
<td>Question mark</td>
</tr>
</tbody>
</table>

To maintain compatibility with variable substitution within z/OS JCL procedures, Tivoli Workload Scheduler for z/OS will assume that a variable has ended (even if
the completing period is missing) if the variable is followed by one of the symbols listed in Table 21 on page 445. An ampersand or percent variable can be assigned a value that is itself a variable.

For example, if LIBRARY is given the value LINKLIB for the following statement:

```
//STEPLIB DD DSN=MY.&LIBRARY.(IEFBR14),DISP=SHR
```

or the following statement (without the completing period):

```
//STEPLIB DD DSN=MY.&LIBRARY(IEFBR14),DISP=SHR
```

The JCL line becomes as follows:

```
//STEPLIB DD DSN=MY.LINKLIB(IEFBR14),DISP=SHR
```

The product assumes that the variable LIBRARY ends when it detects the left parenthesis '('.

**Note:** The completing period is discarded when a variable is substituted. Other termination symbols are left in place.

**Note:** When coding JCL, do not insert all the messages that Tivoli Workload Scheduler for z/OS uses during the variable substitution process (that is, VARIABLE SUBSTITUTION FAILED with OPCSMSG at column 73).

### Ampersand, percent, and question mark variables

The three variable types, ampersand (&), percent sign (%), and question mark (?), cause Tivoli Workload Scheduler for z/OS to perform variable substitution in different ways. You can use a variable multiple times with a different prefix symbol within the same job.

**Note:** If JCL variable substitution is on, the ampersand (&), percent sign (%), and question mark (?) are considered by Tivoli Workload Scheduler for z/OS as symbols that precede a JCL variable. Therefore, if you want to use these variables as comment characters within a JCL procedure, specify ACTION=NOSCAN to prevent the JCL from ending with errors.

#### Ampersand variables

These variables are substituted from left to right within the line. Ampersand variables correspond to the standard variables in z/OS JCL procedures and behave accordingly. Refer to JCL Reference.

If an &-variable is immediately followed by a % variable (that is, there is no intervening termination character), a compound variable is formed. See “Compound variables” on page 447. A compound variable is also formed if an &-variable immediately follows a ?-variable.

Any string that begins with && is not substituted. This is because the double ampersand within JCL is usually used to denote a temporary data set. Any such strings are unaffected by the variable substitution.

If the JCL member contains symbolic parameters in the format &SYMBOLIC and you do not want Tivoli Workload Scheduler for z/OS to attempt resolution, define the variable in a Tivoli Workload Scheduler for z/OS JCL variable table. Define the variable with a blank default value, and specify that a value is not required. See “User-defined variables and variable tables” on page 454 for more information.
Percent variables
Percent variables can be used to form simple variables and compound variables.

Simple variables: If the variable is preceded by a % and ended by a period or any termination character other than %, a value is assigned to the variable, and substitution, for this variable, completes. This is called a simple variable.

Compound variables: Using JCL substitution, you can form compound variables. A compound variable is made up of a concatenation of:
- A variable (of any type) followed by a percent variable with no intervening periods or other termination symbols
- A question mark variable followed by an ampersand variable with no intervening periods or other termination symbols

The values of the percent variables making up a compound variable are not substituted directly. Instead, these values are used to form new variables, which have their own values assigned. Tivoli Workload Scheduler for z/OS attempts to resolve such variables in a series of passes. The individual variables making up the compound variable are resolved, moving from right to left.

For example, consider the following line of JCL from a job:
//STEPLIB DD DSN=MY.&DATA%SET,DISP=OLD

Assume that SET has been given a value of LIB. After the first pass, the variable DATA%SET becomes variable DATALIB because the right-most percent variable is resolved on the first pass. This first pass has now formed a new variable, DATALIB, which Tivoli Workload Scheduler for z/OS will try to resolve on its next pass across this line of JCL.

The new variable DATALIB is treated the same as any other JCL variable. That is, it must be predefined in a variable table and have a value assigned to it. The value can be assigned automatically by Tivoli Workload Scheduler for z/OS, a substitution exit, or manually, using the panels.

Compound variables can be made up of a sequence of many %-variables. Consider the following:
//DDNAME1 DD DSN=MY.%VAR1%VAR2%VAR3....DATA,DISP=OLD

Assume that VAR3 has value SIX and VAR2SIX has value JUNE. On the first pass over this line of JCL, the variable %VAR1%VAR2%VAR3....DATA becomes %VAR1%VAR2SIX....DATA. On the second pass, the variable %VAR1%VAR2SIX. becomes %VAR1JUNE..DATA. The value assigned to %VAR1JUNE. determines the final value that is substituted.

At every substitution, a period was discarded when the variable was substituted. You must specify the correct number of periods to ensure that the substitution is performed correctly. In the preceding example, an extra period was required to denote the beginning of the second-level data set qualifier.

In the next example, you need only one parenthesis to complete the compound variable. This is because the parenthesis is not discarded at substitution.
//DDNAME1 DD DSN=MY.%VAR1%VAR2%VAR3(MEMBER),DISP=OLD
**Compound variables**

If you specify a compound variable `%VAR1%VAR2...DATA`, where `VAR1` is a promptable variable and `VAR2` is a (not promptable) setup variable, the substitution fails, because Tivoli Workload Scheduler for z/OS resolves the compound variable right-to-left.

**Question mark variables**

Question mark variables are *tabular*; that is, you can specify in which column on the line the variable value should begin when the variable is substituted. The position at which the value is placed can be defined in the Tivoli Workload Scheduler for z/OS database when the variable is defined, or can be specified in the job where the variable is used. You will probably use these variables primarily within in-stream data, although they can be used anywhere within your job. For example:

```plaintext
?VAR1.
```

will cause the value of `VAR1` to be placed on the line that the variable appears on, starting at the column number defined in the Tivoli Workload Scheduler for z/OS database.

```plaintext
?nnVAR1.
```

will cause the value of `VAR1` to be placed on the line that the variable appears on, starting at the column number specified by `nn`. Any column value specified for this variable in the Tivoli Workload Scheduler for z/OS database is overridden.

More than one `?`-variable can appear on a JCL line. The positions of the variables themselves have no influence on the positions of the variable values. These positions are decided by the column number specified for the variable. For example:

```plaintext
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7...
?20VAR1.?9VAR2.
```

where `VAR1` is APRIL and `VAR2` is MAY (the scale line has been included only for example purposes), the result after variable substitution would be:

```plaintext
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7...
MAY     APRIL
```

The value of `?`-variables is evaluated in the same way as for `&`- and `%`-variables, and in the same sequence (see “[Making one variable dependent on another” on page 457](#)). However, `?`-variables are substituted only after all percent and ampersand variables have been substituted. This is because the value of the `?`-variable can be placed only in areas of the line that are blank. Tivoli Workload Scheduler for z/OS can only know which areas of a line will be blank after ampersand and percent substitution has occurred.

Tabular variables cannot overlap. That is, the values of two different variables cannot be defined to occupy the same space on a line. The space that the variables themselves originally take up is ignored when substitution occurs. For example:

```plaintext
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7...
?20VAR1.?21VAR2.
/*
```
Question mark variables

where VAR1 is APRIL and VAR2 is MAY, the substitution would be invalid because the two variables are attempting to use columns 21, 22, and 23.

Tivoli Workload Scheduler for z/OS changes the space occupied by the variable to spaces, if it is not covered by the substituted value. For example:

```
//SYSIN DD *
.....+.....1....+.....2....+.....3....+.....4....+.....5....+.....6....+.....7..
THIS IS?40VAR1. THE STANDARD DATA.     IS A WET MONTH.
```

VAR1 is APRIL. After substitution, the line becomes:

```
//SYSIN DD *
.....+.....1....+.....2....+.....3....+.....4....+.....5....+.....6....+.....7..
THIS IS     THE STANDARD DATA.     APRIL IS A WET MONTH.
```

Tivoli Workload Scheduler for z/OS has changed the space occupied by the variable to spaces. The other data in the line does not move.

**Note:** Variables supplied by Tivoli Workload Scheduler for z/OS do not have an implied position. When these variables are specified as tabular variables, you must include the column number. For example, ?OADID will not be accepted; however, ?20OADID is valid: the application ID is substituted at column 20.

Supplied variables

Tivoli Workload Scheduler for z/OS supplies variables that you can use in your business. These variables are reserved, in the sense that Tivoli Workload Scheduler for z/OS will never try to read variable definitions for these variables from a variable table. The variables are split into four groups:

- Occurrence-related variables
- Operation-related variables
- Date-related variables
- Dynamic-format variables

Occurrence-related supplied variables

The first group of variables, described in Table 22, are related to information about the occurrence. Occurrence-related variables can be used as setup or submit variables.

```
Table 22. Occurrence-related supplied variables

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Length (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OADID</td>
<td>16</td>
<td>Application ID.</td>
</tr>
<tr>
<td>OADOWNER</td>
<td>16</td>
<td>Occurrence owner.</td>
</tr>
<tr>
<td>OAUGROUP</td>
<td>8</td>
<td>Authority group.</td>
</tr>
<tr>
<td>OCALID</td>
<td>16</td>
<td>Calendar name.</td>
</tr>
<tr>
<td>ODAY</td>
<td>1</td>
<td>Occurrence input arrival day of the week (1-7); 1 represents Mon., 7 represents Sun.</td>
</tr>
<tr>
<td>ODD</td>
<td>2</td>
<td>Occurrence input arrival day of month, in DD format.</td>
</tr>
<tr>
<td>ODDD</td>
<td>3</td>
<td>Occurrence input arrival day of the year, in DDD format.</td>
</tr>
<tr>
<td>ODMY1</td>
<td>6</td>
<td>Occurrence input arrival date in DDMMYY format.</td>
</tr>
<tr>
<td>ODMY2</td>
<td>8</td>
<td>Occurrence input arrival date in DD/MM/YY format.</td>
</tr>
</tbody>
</table>
```
## Occurrence-related supplied variables

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Length (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OETCRIT</td>
<td>44</td>
<td>Event triggering policy name from the ETT table. <strong>Note:</strong> This variable can be used only by the ETT added occurrence.</td>
</tr>
</tbody>
</table>
| OETEVNM       | 44               | Complete ETT event name:  
  **Event type J**  
  Contains the same value of OETJOBN.  
  **Event type R**  
  Contains the complete resource name that triggered the event. **Note:** This variable can be used only by the ETT added occurrence. |
| OETGGEN       | 8                | GDG data set generation number (GnnnVnn). For the ETT event type R, generated by a data set triggering for GDG. **Note:** This variable can be used only by the ETT added occurrence. |
| OETGROOT      | 35               | GDG data set root. For the ETT event type R, generated by a data set triggering for GDG. **Note:** This variable can be used only by the ETT added occurrence. |
| OETJNUM       | 8                | Job number associated with the OETJOBN. It is set only for the event type J. **Note:** This variable can be used only by the ETT added occurrence. |
| OETJOBN       | 8                | The complete job name that triggered the ETT event:  
  **Event type J**  
  Contains the job name of the triggering job  
  **Event type R**  
  Contains the job name or TSO user ID that closed the ETT data set triggered  
  **Note:** This variable can be used only by the ETT added occurrence. |
| OETTYPE       | 1                | Event type of the ETT table entry (J=Job, R=Resource). **Note:** This variable can be used only by the ETT added occurrence. |
| OFREEDAY      | 1                | Denotes whether the occurrence input arrival date is a free day (F) or a workday (W). |
| OHH           | 2                | Occurrence input arrival hour in HH format. |
| OHHMM         | 4                | Occurrence input arrival hour and minute in HHMM format. |
| OMM           | 2                | Occurrence input arrival month in MM format. |
| OMMYY         | 4                | Occurrence input arrival month and year in MMYY format. |
| OWW           | 2                | Occurrence input arrival week of the year in WW format. |
| OWWD          | 3                | Occurrence input arrival week, and day within week, in WWD format, where WW is the week number within the year, and D is the day within the week. |
| OWWLAST       | 1                | A value, Y (yes) or N (no), that indicates whether the occurrence input arrival date is in the last week of the month. |
### Occurrence-related supplied variables

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Length (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OWWMONTH</td>
<td>1</td>
<td>A value between 1 and 6 that indicates the occurrence input arrival week-in-month, where each new week begins on a Monday. For example, consider these occurrence input arrival dates for the month of March in 1997:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Date</td>
</tr>
<tr>
<td></td>
<td></td>
<td>OWWMONTH</td>
</tr>
<tr>
<td></td>
<td>Saturday 1st</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Monday 3rd</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Monday 31</td>
<td>6</td>
</tr>
<tr>
<td>OYMD</td>
<td>8</td>
<td>Occurrence input arrival date in YYYYMMDD format.</td>
</tr>
<tr>
<td>OYM</td>
<td>6</td>
<td>Occurrence input arrival month within year in YYYYMM format.</td>
</tr>
<tr>
<td>OYMD1</td>
<td>6</td>
<td>Occurrence input arrival date in YYMMDD format.</td>
</tr>
<tr>
<td>OYMD2</td>
<td>8</td>
<td>Occurrence input arrival date in YY/MM/DD format.</td>
</tr>
<tr>
<td>OYMD3</td>
<td>10</td>
<td>Occurrence input arrival date in YYYY/MM/DD format.</td>
</tr>
<tr>
<td>OYY</td>
<td>2</td>
<td>Occurrence input arrival year in YY format.</td>
</tr>
<tr>
<td>OYYDDD</td>
<td>5</td>
<td>Occurrence input arrival date as a Julian date in YYDDD format.</td>
</tr>
<tr>
<td>OYYMM</td>
<td>4</td>
<td>Occurrence input arrival month within year in YYMM format.</td>
</tr>
<tr>
<td>OYYYY</td>
<td>4</td>
<td>Occurrence input arrival year in YYYY format, for example, 1997.</td>
</tr>
</tbody>
</table>

### Operation-related supplied variables

The second group of variables are related to information about the operation that is to be submitted. Operation-related variables can be used only as submit variables. Table 23 describes these variables.

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Length (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OJOBNAME</td>
<td>8</td>
<td>Operation job name.</td>
</tr>
<tr>
<td>OLDD</td>
<td>2</td>
<td>Operation latest start day (day in the month).</td>
</tr>
<tr>
<td>OLMM</td>
<td>2</td>
<td>Operation latest start month, in MM format.</td>
</tr>
<tr>
<td>OLHH</td>
<td>2</td>
<td>Operation latest start hour.</td>
</tr>
<tr>
<td>OLHMM</td>
<td>4</td>
<td>Operation latest start in hours and minutes.</td>
</tr>
<tr>
<td>OLMD</td>
<td>4</td>
<td>Operation latest start time (month and day), in MMDD format.</td>
</tr>
<tr>
<td>OLWK</td>
<td>2</td>
<td>Operation latest start week (week in the year) in WW format.</td>
</tr>
<tr>
<td>OLYMD</td>
<td>6</td>
<td>Operation latest start date in YYMMDD format.</td>
</tr>
<tr>
<td>OLYYDDD</td>
<td>5</td>
<td>Operation latest start in Julian date format (YYDDD).</td>
</tr>
</tbody>
</table>
Operation-related supplied variables

Table 23. Operation-related supplied variables (continued)

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Length (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OOPNO</td>
<td>3</td>
<td>Operation number within the occurrence, right-justified and padded with zeros.</td>
</tr>
<tr>
<td>OWSID</td>
<td>4</td>
<td>Workstation ID for current operation.</td>
</tr>
<tr>
<td>OXJOBNAM</td>
<td>54</td>
<td>Extended job name set in the current plan at the operation level. If you enter blanks in the Extended Job Name field, the OXJOBNAM value is truncated at the first blank.</td>
</tr>
</tbody>
</table>

Date-related supplied variables

The third group of variables are related to the current date and time; that is, the time and date on which the job or started task was submitted. Because they relate to the submission time of the operation, these variables can be used only at submit time. They cannot be used during job or started-task setup. Table 24 describes these variables.

Table 24. Date-related supplied variables

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Length (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDAY</td>
<td>1</td>
<td>Current day of the week; 1 represents Monday, 7 represents Sunday.</td>
</tr>
<tr>
<td>CDD</td>
<td>2</td>
<td>Current day of month in DD format.</td>
</tr>
<tr>
<td>CDDD</td>
<td>3</td>
<td>Day number in the current year.</td>
</tr>
<tr>
<td>CDDMMYY</td>
<td>6</td>
<td>Current date in DDMMYY format.</td>
</tr>
<tr>
<td>CFREEDAY</td>
<td>1</td>
<td>Denotes whether the job actual run date and time is a free day (F) or workday (W).</td>
</tr>
<tr>
<td>CHH</td>
<td>2</td>
<td>Current time in HH format.</td>
</tr>
<tr>
<td>CHHMM</td>
<td>4</td>
<td>Current hour and minute in HHMM format.</td>
</tr>
<tr>
<td>CHHMMSS</td>
<td>6</td>
<td>Current hour, minute, and second in HHMMSS format.</td>
</tr>
<tr>
<td>CHHMMSSX</td>
<td>8</td>
<td>Current hour, minute, second, and hundredths of seconds in HHMMSSXX format.</td>
</tr>
<tr>
<td>CMM</td>
<td>2</td>
<td>Current month in MM format.</td>
</tr>
<tr>
<td>CMMYY</td>
<td>4</td>
<td>Current month within year in MMYY format.</td>
</tr>
<tr>
<td>CWW</td>
<td>2</td>
<td>Week number in the current year.</td>
</tr>
<tr>
<td>CWWD</td>
<td>3</td>
<td>Current day within week in WWD format, where WW is the week number within the year, and D is the day within the week.</td>
</tr>
<tr>
<td>CYMD</td>
<td>8</td>
<td>Current date in YYYYMMDD format.</td>
</tr>
<tr>
<td>CYY</td>
<td>2</td>
<td>Current year in YY format.</td>
</tr>
<tr>
<td>CYYDDD</td>
<td>5</td>
<td>Current Julian date in YYDDDD format.</td>
</tr>
<tr>
<td>CYYMM</td>
<td>4</td>
<td>Current month within year in YYMM format.</td>
</tr>
<tr>
<td>CYYMMDD</td>
<td>6</td>
<td>Current date in YYMMDD format.</td>
</tr>
<tr>
<td>CYYYY</td>
<td>4</td>
<td>Current year in YYYY format, for example, 1997.</td>
</tr>
<tr>
<td>CYYYYMM</td>
<td>6</td>
<td>Current month within year in YYYYMM format.</td>
</tr>
</tbody>
</table>
Dynamic-format supplied variables

The fourth group are time-and-date-related supplied variables. You define the format you require for these variables using the SETFORM directive described in “SETFORM directive” on page 472. For example, if you want to substitute the occurrence input arrival date with the format MM:DD:YY, you define the dynamic variable OCDATE:

Example

```//%OPC SETFORM OCDATE=(MM:DD:YY)```

Table 25 describes these variables.

**Table 25. Dynamic-format date-related supplied variables**

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CDATE</td>
<td>Current date.</td>
</tr>
<tr>
<td>CTIME</td>
<td>Current time.</td>
</tr>
<tr>
<td>OCDATE</td>
<td>Occurrence input arrival date.</td>
</tr>
<tr>
<td>OCFRSTC</td>
<td>First calendar day in month of the occurrence input arrival date.</td>
</tr>
<tr>
<td>OCFRSTTW</td>
<td>First work day in the month of the occurrence input arrival date.</td>
</tr>
<tr>
<td>OCFRSTWY</td>
<td>First work day in the year of the occurrence input arrival date.</td>
</tr>
<tr>
<td>OCLASTC</td>
<td>Last calendar day in the month of the occurrence input arrival date.</td>
</tr>
<tr>
<td>OCLASTW</td>
<td>Last work day in the month of the occurrence input arrival date.</td>
</tr>
<tr>
<td>OCLASTWY</td>
<td>Last work day in the year of the occurrence input arrival date.</td>
</tr>
<tr>
<td>OCTIME</td>
<td>Occurrence input arrival time (hours and minutes).</td>
</tr>
<tr>
<td>OPIADATE</td>
<td>Operation input arrival date (if blank, this takes the value of the occurrence input arrival date).</td>
</tr>
<tr>
<td>OPIATIME</td>
<td>Operation input arrival time (if blank, this takes the value of the occurrence input arrival time).</td>
</tr>
<tr>
<td>OPLSDATE</td>
<td>Operation latest start date.</td>
</tr>
<tr>
<td>OPLSTIME</td>
<td>Operation latest start time.</td>
</tr>
</tbody>
</table>

You can use the variables in Table 22 on page 449, Table 23 on page 451, Table 24 on page 452, and Table 25 in the same way as any other variables. Note that dynamic-format variables are treated as the supplied, predefined variables rather than user-defined variables.

When you have defined the format of a dynamic-format variable by using the SETFORM directive, you can use a different format later in the job by redefining the same variable with another SETFORM directive.

Variables beginning with C are substituted only at submit phase. Variables beginning with O are substituted at setup phase, if present, and otherwise at submit. An exception to this is when a C variable is used in a SETVAR expression; it is substituted at setup phase if the SETVAR directive specifies this.

**Temporary variables**

You can create temporary variable using one of the following:

- An arithmetic expression on the date-related or time-related variables.
- A substring of another variable.
Temporary variables

- The result of an arithmetic addition or subtraction.
- Concatenated strings or variables set to an alphanumeric value.

For example, you might want to refer to the first workday after the occurrence input arrival date in the format YY/MM/DD. You do this by creating a temporary variable from the supplied variable, OYMD2, using the SETVAR directive. The temporary variable is assigned the value (date) of the first workday after the occurrence input arrival date like this:

Example

```plaintext
//*%OPC SCAN
//*%OPC SETVAR TVAR=(OYMD2+1WD)
```

If the occurrence input arrival date is 97/02/07 (Friday), and the next working day is Monday 97/02/10, TVAR will be assigned the value 97/02/10. You can now refer to TVAR as a normal variable through the rest of the job: you can also give it a new value later in the job.

For details, see “SETVAR directive” on page 474.

User-defined variables and variable tables

Beside the supplied variables, you can define variables in a variable table by using the JCL VARIABLE panel.

1. Select option 9 (JCLVAR) from the MAINTAINING TWSz DATABASES panel, or enter the fast path 1.9 from the main menu. You see the MAINTAINING TWSz JCL VARIABLE TABLES panel, shown in Figure 185.

2. (MODIFY). You then see a filter panel that lets you select which JCL variable tables you want to look at. If you leave all the fields blank when you press Enter, you see a list of all the JCL variable tables defined to your Tivoli Workload Scheduler for z/OS system. See Figure 186 on page 455.
3. From this panel, you can copy, modify, browse, delete, and create a variable table. Enter the CREATE command to display the CREATING A JCL VARIABLE TABLE panel, where you are prompted for a table name, the names of the variables you want to create, and other details. See Figure 187 for an example of variable creation.

The SETUP option specifies when the variable is substituted. See “When variables are substituted” on page 456 for more information.

You can specify that an exit, rather than Tivoli Workload Scheduler for z/OS, should substitute the variable. For more details about this exit, refer to Customization and Tuning.

When you select a variable from panel EQQJVCL, you can make detailed modifications to it using this panel:
User-defined variables and variable tables

---

**EQQJVMP --------------- MODIFYING A JCL VARIABLE -------------------------------**

Command ==> INSERT DATA BELOW, OR DEP COMMAND TO CREATE/MODIFY DEPENDENCY,
Enter VER command to define verification rules.

Variable table :MONTHLY Monthly variables
Variable name :PXM0001
DESCRIPTION ===> Descriptive text
DEFAULT VALUE ===> Y - Yes, N - No
UPPER CASE ===> Y - Yes, N - No
SETUP ===> P - Prompt
SUBSTITUTION EXIT ===> A load module name
VALUE REQUIRED ===> Y - Yes, N - No
DEFAULT POSITION ===> 01 - 80 A position in a JCL instream data line
DIALOG TEXT ===> 

---

**Figure 188. EQQJVMP - Modifying a JCL variable**

Set the UPPER CASE field to Y, if you want lower case characters of the
DEFAULT VALUE field to be translated to upper case.

The VALUE REQUIRED option specifies whether the variable can be blank. If
you set this field to Y, the value cannot be assigned a blank value.

4. When you have selected and defined a variable, you can specify:
   - Dependencies on the variable (see "Making one variable dependent on
   another" on page 457)
   - Validation criteria for the variable (see "Variable validation" on page 461)
by entering the DEP or VER primary command.

**When variables are substituted**

Table 26 shows how to specify the SETUP option in the panel shown in Figure 188.

**Table 26. Specifying the SETUP option**

| SETUP | Tivoli Workload Scheduler for z/OS substitutes the variable ...
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Y</td>
<td>Automatically at job setup, or at submission if there is no setup operation.</td>
</tr>
<tr>
<td>N</td>
<td>Automatically at job submission. The variable is stored with the job in the JCL repository (JS) data set, and is then substituted automatically before the job is routed to the required destination.</td>
</tr>
<tr>
<td>P</td>
<td>After substituting any SETUP=Y variables and prompting the operator. Tivoli Workload Scheduler for z/OS first substitutes SETUP=Y variables. Then it scans the job for promptable variables and displays them. The operator enters values for each variable. If there is no setup phase, the job fails with error code OJCV.</td>
</tr>
</tbody>
</table>

The operator initiates setup by setting the operation to next logical status using the
workstation READY LIST panel. Variables can also be substituted by a
user-supplied exit rather than by Tivoli Workload Scheduler for z/OS.
Customizing System Automation commands

The System Automation command tailoring function automatically edits System Automation commands through predefined information. This will help you to save time and reduce the possibility of editing errors. System Automation command tailoring provides automatic variable substitution before the command text is submitted to System Automation for processing. Note that:

- No support for directives is provided. Variables in the command text are searched in the table specified for the application, if any. If no table is defined for the application, variables are searched in the global variable table.
- Variable substitution is enabled only by setting the VARSUB parameter of the OPCOPTS statement.
- Only the variable types ampersand (&) and percent sign (%) are supported. The question mark (?) is not valid.
- The SAVARFAIL parameter in the OPCOPTS statement supports any combination of the variable types (& and %), but at least one type must be specified. Note that by setting the SAVARFAIL parameter you prevent the System Automation command operation from failing, in case of unresolved variables. Unresolved variables are left unchanged. If you do not set SAVARFAIL and a variable is not resolved, an error message is issued.
- If errors occur during variable substitution, messages are issued in EQQMLOG. The maximum length supported for command text is 255 bytes. Any text exceeding this length after variable substitution, produces a truncation error and the operation failure (OJCV). Text truncation is not allowed.
- The variables supplied by Tivoli Workload Scheduler for z/OS are the same as those supplied for JCL. For details, refer to “Supplied variables” on page 449.
- Variable substitution exits are supported.

To use variable substitution in System Automation commands, the load module EQQZVGSU must have been successfully loaded at controller startup. Any variable contained in a System Automation command text is resolved when the operation starts, before it is submitted to System Automation. After the variables in System Automation commands are successfully resolved, the corresponding CP33 record in current plan (CPLREC33) is updated with the resulting values. If there are no variables to resolve or the variable substitution is not successful, the record in the current plan is left unchanged. The System Automation command text supports mixed case, but the string that identifies a variable must be uppercase, otherwise an error message about variable substitution is issued.

Making one variable dependent on another

You can have variables that depend on other variables or on a substring of other variables. For example, the message class and print destination might both depend on the z/OS system that a job runs on. If you make the message class (MSGC) and destination (DEST) variables dependent on the system (SYSTEM) variable, you need only change the value of the system variable, and the others are given the correct value.

Follow these steps to specify a dependent variable (MSGC in this example) based on the value of another variable (SYSTEM in this example):

1. Enter the DEP command from the MODIFYING A JCL VARIABLE panel. You see the JCL VARIABLE DEPENDENCY VALUE LIST panel, shown in Figure 189 on page 458.
Making one variable dependent on another

2. Specify the system as the independent variable.

3. For each value of the independent variable (SYSTEM), specify the associated value of the dependent variable (MSGC). Make sure that you specify the independent variable in the correct case, it is case-sensitive, and any in Figure 189 does not give the same result as ANY.

   If there are many values of SYSTEM with the same value of MSGC, make that value of MSGC the default, and do not specify those values of SYSTEM: if there is no match, MSGC takes its default value.

4. When you create the JCL, make sure that the independent variable occurs first, or the result is unpredictable.

   /*%OPC SCAN
   //JOB6 JOB (&ACCT.,NOBO),'SAMPLE',
   // MSGCLASS=&MSGC.,NOTIFY=XRAYNER,CLASS=A
   /*JOBPARM SYSAFF=&SYSTEM.
   /*MAIN SYSTEM=&SYSTEM.
   //OUTPUT1 OUTPUT DEST=&DEST,DEFAULT=NO

   In this example, the job will fail if MSGC depends on SYSTEM, because the MSGC variable occurs first.

   Change the JCL by inserting a comment card before the first dependent variable:

   /*%OPC SCAN
   /* THE VALUE OF SYSTEM IS &SYSTEM.
   //JOB6 JOB (&ACCT.,NOBO),'SAMPLE',
   // MSGCLASS=&MSGC.,NOTIFY=XRAYNER,CLASS=A
   /*JOBPARM SYSAFF=&SYSTEM.
   /*MAIN SYSTEM=&SYSTEM.
   //OUTPUT1 OUTPUT DEST=&DEST,DEFAULT=NO

   To specify a dependent variable based on a substring of the independent variable, use the SUBSTRING fields. For example, suppose that:
   - The SYSTEM variable is set to ZOS18P1 or ZOS18T1, depending on your JCL runs in a production or a test system.
   - The input for your JCL must be TWSZ.PROD.IN or TWSZ.TEST.IN, depending on your JCL runs in a production or a test system.
You can define DSNQL as dependent variable as follows:

```plaintext
DSNQL

TWSZ.&DSNQL..IN

JCL and run in both your systems.
```

**Restrictions**

Do not:

- Specify both dependent and independent variables as setup variables, if the independent variable is part of a compound variable that cannot be resolved at the setup phase.
- Specify a setup or promptable dependent variable if the independent variable is resolved at submit.
- Cause a loop by making a variable directly or indirectly dependent on itself.
- Give a dependent variable a blank value, if VALUE REQUIRED=Y is specified for that variable.
- Use a variable substitution exit to set the value of the independent variable.

The dependent variable is given its value on the first pass of variable substitution, and you should be careful if the independent variable needs several passes to resolve. For example, if you gave SYSTEM a value &OWSID. (which is substituted further, because it is a scheduler-defined variable), MSGC will not be substituted correctly, unless you specify the literal &OWSID. as the value for the independent variable, which is probably not what you want. If you need to make MSGC dependent on the workstation name, the correct method is to make OWSID the independent variable. See "Using supplied variables" on page 460 for a description of this method.

**Using dependent variables for migration**

Dependent variables can be useful when you are changing the release of some software. One way is to have two copies of each job, but in many cases it is just data set qualifiers that change (for example, from CICS320.LOADLIB to CICS330.LOADLIB. Making the data set qualifier a variable lets you have just one set of jobs, which is important if you make other changes unconnected to the change.
Using dependent variables for migration

in release. Your effort in coding variables is rewarded when you next upgrade the software, because the variables are already in the jobs.

Another advantage is that fallback is easier, because you do not have to remember what happened to the old jobs, and what changes must be retroactively made to them.

Using supplied variables

The example using SYSTEM needs some manual intervention when the value of the SYSTEM variable must be changed (either at a setup workstation or using the variable table panel). But you can use dependent variables even in a fully automatic system, if the independent variable is a supplied variable.

This example shows how to change variables automatically, depending on the input arrival day of the occurrence. The supplied variable is ODAY, which has values from 1 (Monday) to 7.

This is how the dependent variable (HLQ1) is defined:

```
Command ===> MODIFYING A JCL VARIABLE
Variable table : SPECIALONMONDAY
Variable name : HLQ1
DESCRIPTION ===> high-level-qualifier
DEFAULT VALUE ===> OTHER.DAYS
UPPER CASE ===> Y
SETUP ===> Y
SUBSTITUTION EXIT ===> ________ A load module name
VALUE REQUIRED ===> Y
```

Figure 191. Specifying the variable HLQ1

The default value, OTHER.DAYS, is used when there is no match on the independent variable values.
Using supplied variables

Tivoli Workload Scheduler for z/OS substitutes DAY1 for the HLQ1 variable when ODAY=1. On other days, it uses the default value OTHER.DAYS. ODAY is a setup or submit variable, so HLQ1 can also be specified as a setup or submit variable.

Variable validation

Tivoli Workload Scheduler for z/OS can check input entered by the operator at job setup or by a substitution exit. You can specify, for example, whether the input will be numeric or alphabetic, or whether it should match a predefined picture. You can also specify whether the entry should be within a range of values or be one of the values in a given list. Logical comparisons can also be performed; for instance, you can specify that the entered value will be less than, equal to, or not greater than a given value. Define your validation criteria on the SPECIFYING VERIFICATION CRITERIA panel (see Figure 193). This panel is displayed when you enter the VER command from the MODIFYING A JCL VARIABLE panel (see Figure 188 on page 456).

Figure 192. Specifying a special value for Monday (ODAY=1)

Tivoli Workload Scheduler for z/OS substitutes DAY1 for the HLQ1 variable when ODAY=1. On other days, it uses the default value OTHER.DAYS. ODAY is a setup or submit variable, so HLQ1 can also be specified as a setup or submit variable.

Variable validation

Tivoli Workload Scheduler for z/OS can check input entered by the operator at job setup or by a substitution exit. You can specify, for example, whether the input will be numeric or alphabetic, or whether it should match a predefined picture. You can also specify whether the entry should be within a range of values or be one of the values in a given list. Logical comparisons can also be performed; for instance, you can specify that the entered value will be less than, equal to, or not greater than a given value. Define your validation criteria on the SPECIFYING VERIFICATION CRITERIA panel (see Figure 193). This panel is displayed when you enter the VER command from the MODIFYING A JCL VARIABLE panel (see Figure 188 on page 456).

Figure 193. EQQJVVEP - Specifying verification criteria
Variable validation

You can specify these values in the fields on the SPECIFYING VERIFICATION CRITERIA panel:

**VERIFICATION TYPE**

**ALPHA**  The input must be alphabetic.
**NUM**  The input must be numeric, in the range \(-2^{31}\) to \(2^{31}-1\).
**ENUM**  The input must be numeric but can also contain the plus sign (+), negative number indicator, delimiter symbols, decimal symbol (.), or certain national character decimal symbols, such as a comma (,).
**HEX**  The input must be valid hexadecimal characters.
**BIT**  The input can have only the digits 0 and 1.
**PICT**  The input must match a certain pattern.
**NAME**  The input must be a valid partitioned data set member name.
**DSNAME**  The input must be a valid data set name. (Lowercase letters are not allowed.)
**RANGE**  The input must be within the range specified in the Valid values/ranges field.
**LIST**  The input variable must be a member of the list given in the Valid values/ranges field.

**LENGTH COMPARISON and VALUE**

These fields are for validating the length of the entered value. The possible comparisons are:

- **= or EQ**  Equal to
- **< or LT**  Less than
- **> or GT**  Greater than
- **<= or LE**  Less than or equal to
- **>= or GE**  Greater than or equal to
- **!=**  Not equal to
- **!<**  Not less than
- **!>**  Not greater than.

For example, to specify that the value of the variable must be less than 6 characters, specify LT 6.

**PICT VALIDATION**

Specify this only for validation type PICT. The variable must contain characters that match the corresponding characters in the validation pattern. The pattern can have these characters:

- **C**  Any character
- **A**  Any alphabetic character (A–Z, a–z, ?, $, @)
- **N**  Any numeric character (0–9)
- **9**  Any numeric character (same as N)
- **X**  Any hexadecimal character (0–9, A–F, a–f).

For example, if a six-byte field must have its first three characters in the range 0–9, and the last three characters can have any value, specify 999CCC.

**LIST/RANGE NUMERIC**

This option applies only for types RANGE or LIST. If you specify Y (Yes),
Variable validation

Tivoli Workload Scheduler for z/OS performs a numeric comparison with the LIST/RANGE VALUES. If you specify N (No), a character-by-character comparison is performed.

LIST/RANGE VALUES
This field is used for validation types LIST and RANGE. For type LIST, specify the values separated by commas (,) or spaces. For type RANGE, each pair of values in the range is separated by commas, and each range is separated by commas or spaces.

<table>
<thead>
<tr>
<th>Variable</th>
<th>TYPE</th>
<th>NUMERIC</th>
<th>VALUES</th>
</tr>
</thead>
<tbody>
<tr>
<td>DAYS-IN-MONTH</td>
<td>LIST</td>
<td>Y</td>
<td>28,29,30,31</td>
</tr>
<tr>
<td>DAYS-IN-MONTH</td>
<td>RANGE</td>
<td>Y</td>
<td>28,31</td>
</tr>
<tr>
<td>DATA-CENTERS</td>
<td>LIST</td>
<td>N</td>
<td>LONDON, NEW YORK</td>
</tr>
<tr>
<td>CREDIT-RATING</td>
<td>RANGE</td>
<td>N</td>
<td>A1, A9, B1, B9</td>
</tr>
</tbody>
</table>

Notes:
1. Embedded blanks are allowed in character values (NEW YORK).
2. Tivoli Workload Scheduler for z/OS checks that the value for CREDIT-RATING is in the range A1-A9 or B1-B9. Because this is a character comparison, a value A12 is allowed, unless you also specify a length of 2.

Global variable table
You can define a global variable table. The name of this variable table is specified in the GTABLE keyword of the initialization statement OPCOPTS (refer to Customization and Tuning). If Tivoli Workload Scheduler for z/OS cannot find a variable in the variable tables specified for the operation or in the operation job, it searches the global variable table.

Where variables can be used
Tivoli Workload Scheduler for z/OS variables can be used anywhere in your job and as many times as you want. Variables used in comment and in-stream data will also be substituted.

Substitution takes place depending on the setting of the VARSUB initialization statement. If VARSUB(YES) is specified, variable scanning is performed for all operations. If VARSUB(SCAN) is specified, Tivoli Workload Scheduler for z/OS searches for variables only if the //*%OPC SCAN directive has been encountered in the job.

Tivoli Workload Scheduler for z/OS does not substitute variables within procedures, with the exception of inline procedures when VARPROC(YES) has been specified in the OPCOPTS statement. However, variables are substituted in the calling EXEC statement. See "Variable substitution in z/OS JCL procedures" on page 464.

Errors in variable substitution
If an error occurs in variable substitution, the operation is put on the error list with the error code OJCV.

For more information about a variable substitution error:
• Examine the message log, EQQMLOG.
Errors in variable substitution

- Look at the job for the operation that ended in error. Use the J row command from the error list.

Variable substitution error messages are inserted into the job in the lines immediately preceding the erring variable. These messages are displayed in the form of ISPF note lines:

Examples of notes lines

```
000001 //XCALLINN JOB (885002,J09), 'BOC', MSGLEVEL=(1,1).
000002 // NOTIFY=XCALLIN, MSGCLASS=Q, CLASS=B
000003 //STEP001 EXEC PGM=IEFBR14
000004 //DD DD DSN=&DS1..DATASET, DISP=SHR
NOTE=> ///EQQJ535E 03/17 08.57.00
NOTE=> /// UNDEFINED VARIABLE DS2 LINE 00005 OF ORIG JCL
000005 //DD DD DSN=&DS2..DATASET, DISP=SHR
000006 // More JCL
```

To find variable substitution error message note lines in the job, enter the primary command LOCATE SPECIAL. To save the messages in the job, convert these temporary note lines into data lines, by entering the row command MD (make data).

You can avoid the variable substitution errors by using the VARFAIL keyword. Refer to Customization and Tuning for details.

Note: Tivoli Workload Scheduler for z/OS scans the job only up to the first detected error.

Suppressing variable substitution

You can specify that variable substitution should be suppressed in jobs that would normally be eligible for variable substitution, using the statement //**%OPC BEGIN ACTION=NOSCAN in your job. Any lines after this statement will not be eligible for variable substitution. If you want to turn variable substitution on again, you can do so using the //**%OPC END ACTION=NOSCAN statement. For example, in the following job, Tivoli Workload Scheduler for z/OS would substitute DS1, would leave DS2 and DS3 as they were, and would substitute DS4.

```
//DD1 DD DSN=MY.&DS1, DISP=SHR
//**%OPC BEGIN ACTION=NOSCAN
//DD2 DD DSN=MY.&DS2, DISP=SHR
//DD3 DD DSN=MY.&DS3, DISP=SHR
//**%OPC END ACTION=NOSCAN
//DD4 DD DSN=MY.&DS4, DISP=SHR
```

A //**%OPC BEGIN ACTION=NOSCAN statement must have an associated //**%OPC END ACTION=NOSCAN statement, or an error will result. See “JCL tailoring directives” on page 466 for a complete explanation of the directives.

Variable substitution in z/OS JCL procedures

Variable substitution does not take place within JCL procedures. If you have z/OS JCL variables within such procedures, their substitution is independent of any substitution made by Tivoli Workload Scheduler for z/OS. Even if the z/OS JCL
Variable substitution in z/OS JCL procedures

variable name is the same as an Tivoli Workload Scheduler for z/OS variable name, the values assigned to the two variables will be unrelated. However, if you want the variable reference outside the procedure to be substituted by z/OS, you can define the Tivoli Workload Scheduler for z/OS-variable with the attribute DEF= (value not required).

If a variable is given a value of b (blank), the variable will not be substituted if the variable definition states that no value is required. This lets these variables undergo normal JCL substitution after the tailoring performed by Tivoli Workload Scheduler for z/OS. If a variable is given a default value of ¬ (logical not), the variable is deleted from the line.

This means that z/OS substitutes the variables that have a b, or not-required, value. Tivoli Workload Scheduler for z/OS substitutes the variables that have values that are not blanks, as well as variables that require values.

Because procedures are called after Tivoli Workload Scheduler for z/OS substitution has taken place, a situation in which a procedure variable is assigned a Tivoli Workload Scheduler for z/OS variable value can occur. Consider the JCL in Figure 194 and the procedure in Figure 195.

```jcl
//EXEC PROC=MYPROC
//DD2 DD DSN=MY.&DS2,DISP=SHR
//EXEC PROC=MYPROC
//DD2 DD DSN=MY.&DS2,DISP=SHR
```

Figure 194. Substituting a variable in a procedure: job JCL

If DS2 is defined as a Tivoli Workload Scheduler for z/OS variable, it is substituted in the usual way.

```jcl
//MYPROC EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=4096K
//DD2 DD DSN=MY.&DS2,DISP=SHR
//MYPROC EXEC PGM=IKJEFT01,DYNAMNBR=50,REGION=4096K
//DD2 DD DSN=MY.&DS2,DISP=SHR
```

Figure 195. Substituting a variable in a procedure: procedure JCL

If, as in this example, the ddname DD2 occurs in the procedure MYPROC, the entire JCL line will be substituted in the procedure, including the Tivoli Workload Scheduler for z/OS substituted value of DS2:

```jcl
//DD2 DD DSN=MY.&DS2,DISP=SHR
```

Any value that the variable DS2 is assigned within the procedure is not substituted in this line because, as far as z/OS is concerned, the variable DS2 does not occur. If you want the variable to take the value assigned by the procedure rather than the Tivoli Workload Scheduler for z/OS value, you can give the variable a default value of b (blank) and set the VALUE REQUIRED attribute to 'N' in the Tivoli Workload Scheduler for z/OS database using the JCL VARIABLE DEFINITION panel. This causes Tivoli Workload Scheduler for z/OS to ignore the variable when performing variable substitution.

Note: For inline procedures it is possible to apply variable substitution when VARPROC=YES has been specified in the OPCOPTS statement.
Substituting variables with embedded blanks

Tivoli Workload Scheduler for z/OS allows embedded blanks and trailing blanks in variables. For example, the string 25 Dec 1997 can be specified as a variable value. If you specify variable verification, the type of variable must be LIST or blank. When you define a default value with embedded or trailing blanks, a warning message is issued to ensure that these blanks are intentional.

Leading blanks, however, are not supported. If you enter leading blanks, the string is left-justified (the blanks are removed) with no warning or error message.

To get trailing blanks, you must specify a length for the variable—this is the only way Tivoli Workload Scheduler for z/OS can tell if there should be trailing blanks. Use the LENGTH VALUE field on the SPECIFYING VERIFICATION CRITERIA panel (Figure 193 on page 461). If you do not specify a length, trailing blanks are not included in the substitution.

JCL tailoring directives

Tivoli Workload Scheduler for z/OS can dynamically include or exclude job statements during job setup and job submit. Tivoli Workload Scheduler for z/OS excludes lines in the job by skipping them at job setup or at job submit. You can include the job statements from a member in the EQQJBLIB library or supply them through a user-defined JCL-embed exit.

Tivoli Workload Scheduler for z/OS uses special comment statements, called directives, to manage the inclusion and exclusion of lines and to control aspects of variable substitution. The directives are:

- NOP
- SCAN
- SEARCH
- SETFORM
- SETVAR
- TABLE
- BEGIN and END
- FETCH

The general syntax of the directives is:
- Each directive must begin on a new 80-byte line.
- All directives begin with //*OPC in columns 1 to 7 followed by at least one space. The only exception is the NOP directive, for which only one space is allowed.
- Directive parameters can be coded in any order.
- Directive parameters can occur more than once in the same directive.
- Directive parameters are separated by commas with no embedded blanks between parameters on the same line.
- If more than one parameter value is specified, parentheses are required. For example, this is correct:
  NAME=TABLE1

But this is incorrect:
NAME=TABLE1,TABLE2
JCL tailoring directives

It should be defined:
NAME=(TABLE1,TABLE2)

• A directive specification cannot exceed 71 characters. It can be continued on a
new line if the directive is split by a comma after a complete or partial
parameter.
• Positions 72 to 80 are ignored.
• Each continuation line must begin with //,%OPC in columns 1 to 7 followed by a
least one space.
• Except for the NOP directive, if the directive is executed successfully, the
//,%OPC is changed to //,%OPC.

If a line begins with //,%OPC and none of the known directives is found, the Tivoli
Workload Scheduler for z/OS job substitution routines treat any other directives
that it finds as “unknown,” and will take no action.

Notes:
1. When a variable has been substituted with a value from the current variable
table concatenation, that value remains valid for the entire substitution phase. It
will not be changed when a different table is declared by a subsequent
SEARCH or TABLE directive (see "JCL tailoring directives" on page 466), even
if the same variable name is found later. Therefore avoid using several variable
tables for the same job, especially if they contain variables with the same name.

2. If you set VARSUB(SCAN) in the OPCOPTS statement, the SCAN directive
must be present in the JCL, in order to process all the subsequent directives.

Note:

A description of the directives follows. If you are unfamiliar with syntax diagrams,
see "How to read the syntax diagrams" on page xxiv.
NOP directive

**NOP directive**

**Purpose**
This directive determines that the job does not run. Like the existing NOP command issued against an operation in the current plan, it produces the following results:

- All triggers and dependencies are maintained.
- Final operation status is set to complete.

Unlike the NOP command, this directive does not update the NOP indicator in the current plan (mapped by the CPOPNOP field of the CPOP data area or the CPLNOP field of the CPLREC3P record).

**Syntax**

```plaintext
/SM590000/SM590000
//*%OPC NOP
/SM590000/SM630000
```

**Usage Notes**
Unlike the behavior for other directives, after the successful processing for this directive, the scheduler does not change the string //*%OPC to //*>OPC. Therefore, the NOP directive remains active after the processing. You must manually remove it when you no longer need it.

Special attention must be used when setting VARSUB (SCAN) in OPCOPTS. In this case, the NOP directive can be ineffective even if left active, for example, in the case the SCAN directive is not active (*OPC SCAN instead of *%OPC SCAN) or, in the case it is not at all preceded by the SCAN directive.
SCAN directive

Purpose
If the VARSUB keyword of the OPCOPTS statement is set to SCAN, this directive, when found in the JCL of a computer workstation operation, informs Tivoli Workload Scheduler for z/OS that variable substitution should start from this line. This applies also for processing the directives, meaning that the processing of the directives starts from the line where SCAN is specified.

Syntax

```/*%OPC SCAN
```

Usage Notes
The SCAN directive is honored only if the VARSUB parameter of the OPCOPTS statement is set to SCAN.

Assuming that VARSUB(SCAN) is specified, in the following example, MODULE will not be substituted because it is before the SCAN directive. The variable LIBRARY, occurring after the SCAN directive, is substituted.

Example

```//OPSTATUS JOB (ACCOUNT),'Set completed',CLASS=A
//STEP1 EXEC PGM=&MODULE.
//%OPC SCAN
//STPLIB DD DSN=OPC.LOAD.&LIBRARY.,DISP=SHR
//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY,DISP=SHR
//EQQMLIB DD SYSOUT=A
//SYSIN DD *
/*```
SEARCH directive

Purpose
This directive defines the variable tables that are searched when attempting to assign a variable a value.

Syntax

```
/*%OPC SEARCH NAME=( )
```

Parameters

```
NAME( table name, ..., APPL|NOAPPL, GLOBAL|NOGLOBAL)
```

Identifies the variable tables you want searched, and in what order.

Usage Notes
Use the SEARCH directive to specify the variable tables you want searched. Up to 16 tables, including the application and global tables, can be specified. By default, the variable tables specified using `table name` are searched first, in the order specified. If Tivoli Workload Scheduler for z/OS does not find a variable in these tables, the application variable table, if it exists, is then searched, followed by the global variable table, if the variable is not found in the application table. A SEARCH directive cannot contain any Tivoli Workload Scheduler for z/OS variables. The following example shows how you can change the search order of the tables using the SEARCH directive.

Example 1
```
/*%OPC SEARCH NAME=(GLOBAL,TABLE1,NOAPPL)
```

The search order for variables in JCL containing the SEARCH directive given would be:
1. GLOBAL
2. TABLE1

The NOAPPL keyword specifies that the application variable table will not be searched.

Example 2
```
/*%OPC SEARCH NAME=(TABLE1,TABLE2,TABLE3)
```

In this example, the search order for variables in a job containing the SEARCH directive given would be:
1. TABLE1
2. TABLE2
3. TABLE3
4. The application variable table (if it exists)
5. The global variable table (if it exists)
SEARCH directive

A SEARCH directive causes all variables in the job before the directive to be resolved. If the SEARCH directive is used more than once, the last specified search order is used to resolve any new variables found. Variables that have already been assigned values retain those values, even if they are found again after a SEARCH or TABLE directive that would have caused a change in their values.
SETFORM directive

**Purpose**
This directive defines the format of dynamic-format supplied variables. After Tivoli Workload Scheduler for z/OS processes the SETFORM directive, you can refer to the variable and perform arithmetic calculations using the variable. You can redefine the variable many times within the job, if you need to.

**Syntax**
```
/*%OPC SETFORM dynamic-variable-name=(format expression)
```

**Parameters**
`dynamic-variable-name=(format expression)`
The dynamic variable uses the format defined in the format expression.

**Usage Notes**
The dynamic variable name must be one of the supplied dynamic variables (see Table 25 on page 453).

The format expression can contain a combination of time-related keywords, date-related keywords, and delimiters.

The date-related keywords are:
- **CC** Represents the century. This is used in combination with YY to define the format of a full year, such as 1997.
- **YY** Represents the last two figures in the year.
- **MM** Represents the month.
- **DDD** Represents day-in-year. This is substituted before DD: the character string DDDDDDD is understood as two DDD keywords, not three DD keywords.
- **DD** Represents the day in the month.

The time-related keywords are:
- **HH** Represents the hour.
- **MM** Represents the minutes.

Any other characters in the format expression are regarded as delimiters. These delimiters can be alphabetic, numeric, or any symbol except the Tivoli Workload Scheduler for z/OS variable substitution characters $, %, ?, =, and the parentheses ()

For the time-related dynamic variables, OCTIME, OPIATIME, OPLSTIME, and CTIME, only HH and MM are recognized. YY, for example, is not substituted. MM is substituted by the minutes part of the time.

For date-related dynamic variables, only CCYY, YY, MM, DD, and DDD are recognized. CC without YY is not recognized. HH is not substituted. MM is substituted by the month part of the date.

You can use more than one delimiter between keywords.
SETFORM directive

For example, MM/DD--YY is a valid format expression.

Delimiters are optional; that is, you can define consecutive keywords with no delimiters, such as DDMYY.

In the following examples, assume that the occurrence input arrival time is at 4:10 PM on December 31st 1997.

Example

```c
//*%OPC SCAN
//*%OPC SETFORM OCDATE=(YY/MM/DDD)
```

The resulting &OCDATE variable would be: 97/12/365

The examples in Table 27 use the same occurrence input arrival date.

<table>
<thead>
<tr>
<th>Dynamic format variable</th>
<th>Format expression</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>OCDATE</td>
<td>YY-MM-DDABC</td>
<td>97-12-31ABC</td>
</tr>
<tr>
<td>OCTIME</td>
<td>HH MM</td>
<td>16 10 (Note the MM substitutes as minutes for time variables and substitutes as month for date variables.)</td>
</tr>
<tr>
<td>OCDATE</td>
<td>DDDDD</td>
<td>36531. DDD is the 365th day of the year, and DD is the day of the month.</td>
</tr>
<tr>
<td>OCDATE</td>
<td>DDDD</td>
<td>365D. DDD is the 365th day of the year, but no match was found for the last D.</td>
</tr>
<tr>
<td>OCDATE</td>
<td>YYMMHHMMSS</td>
<td>9712HH12SS. This is a date variable, so HH is not substituted.</td>
</tr>
</tbody>
</table>
SETVAR directive

**Purpose**
This directive creates a temporary variable using one of the following:
- An arithmetic expression together with supplied date or time variables.
- A substring of another variable.
- The result of an arithmetic addition or subtraction.
- Concatenated strings or variables set to an alphanumeric value.

After Tivoli Workload Scheduler for z/OS processes the SETVAR directive, you can use the temporary variable in the same way as you use other variables. You can redefine a temporary variable later in the job.

**Syntax**

```plaintext
Tname= (date time variable +/- nnnTT1 nnnTT2 nnnTT3)
```

**Parameters**

- `Tname` The name of the temporary variable, beginning with the letter `T`.  
- `date time variable` One of the following defined formats:
  - **Date formats:**
    - See Table 28 on page 478
  - **Day-in-year formats:**
    - See Table 29 on page 478
  - **Day-in-month formats:**
    - ODD, OLDD, CDD
  - **Day-in-week formats:**
    - OWWD, CWWD
  - **Day-of-week formats:**
    - ODAY, OLDAY, CDAY
  - **Week formats:**
    - OWW, OLWK, CWW
  - **Month formats:**
    - See Table 30 on page 478
  - **Year formats:**
    - OYY, OYYYY, CYY, CYYYY
  - **Time formats:**
    - See Table 31 on page 478
  - **Hour formats:**
    - OHH, OLHH, CHH, CYYY
- `nnn` A number in the range 0 to 999.
- `TT1` The first possible type. You can specify the following values:
For date-related variables

WD Work days, as defined for the calendar used by the occurrence.
CD Calendar days.
WK Weeks. Weeks are converted to days before the calculation is performed.
MO Months. Performing calculation on the month portion affects only the month, and possibly the year. The calculation always generates valid results, handling actual months durations and leap years. For example, adding one month to 080131 gives 080229 (considering the leap year).
YR Years.

For time-related variables

HH Hours.
MM Minutes.
SS Seconds.

TT2 The second possible type, valid only for time-related variables. You can specify the following values:

MM Minutes.
SS Seconds.

TT3 The third possible type, valid only for time-related variables. You can specify only the value SS (seconds).

You can use the format nnnTT1 nnnTT2 nnnTT3 only for time-related variables, to add or subtract hours, minutes, and seconds to or from a given time. Specify this triple format only if you want to specify hours, minutes, and seconds.

Using duplicated types, as for example in 6HH, 5MM, 7MM, is not allowed.

PHASE=SETUP|SUBMIT
Specifies whether the SETVAR calculation should take place during the setup or submit phase of the operation. This parameter is optional, and if nothing is specified, the default SUBMIT is assumed.

Note: If you use setup phase, you can still use submit-phase date or time variables (those beginning with the letter C) in the expression for the temporary variable.

variable
The name of the source string variable. Use an existing variable name properly defined and accessible. You can specify any supplied or user-defined variable. The length of the variable that is replaced is limited to the remaining JCL row length that is not used by the statement.

n An integer in the range 1 to 60. It defines the starting position, in variable, of the substring variable. If it exceeds the length of variable, the resulting substring is padded with blank.

length An integer in the range 1 to 60. It defines the length of the substring variable. If it exceeds the length of variable, the resulting substring is padded with blank.

numeric value
An integer in the range 0 to 99999, defining the value of the temporary variable.

alphanumeric value
String of alphanumeric characters, defining the value of the temporary variable. Enclose the string in single quotes. It can be up to 48 characters.
If the length of the whole statement in the JCL reaches column 72, the error message EQQJ585E is issued.

`variable1`
An integer in the range 0 to 99999, defining the first operand of an arithmetic addition or subtraction, whose result defines the value of the temporary variable. The arithmetic expression cannot include blank characters.

`variable2`
An integer in the range 0 to 99999, defining the first operand of an arithmetic addition or subtraction, whose result defines the value of the temporary variable. The arithmetic expression cannot include blank characters.

`concatenation item`
One of the following:

- A variable previously set to an alphanumeric value.
- A string of alphanumeric characters.

Enclose the item list in single quotes. The result cannot exceed 48 characters.

**Usage Notes**
These examples show how to use temporary variables created through SETVAR:

**Example 1 (with an arithmetic expression)**

If the occurrence input arrival date is 97/12/26, the expression is substituted as follows:

```
TVAR=(360+4)
TVAR=364
```

If the occurrence input arrival date is 97/12/30, the expression is substituted as follows:

```
TVAR=(364+4)
TVAR=003
```

**Example 2 (with dynamic-format variable)**

```
//*OPC SCAN
//*OPC SETFORM CDATE=(ACCURATE DATE CCYY MM DD)
//*OPC SETVAR TDATE=(CDATE + 1CD)
```

If the occurrence input arrival date is 97/12/26, the expressions are substituted as follows:

```
CDATE = 'ACCURATE DATE 1997 12 26'
TDATE = 'ACCURATE DATE 1997 12 27'
```

**Note:** If the date calculation results in dates earlier than 1 January 1984 or later than 31 December 2071, the substitution fails with an error message.

If the expression includes dynamic-format supplied variables containing the first or the last day in the month or in the year of the occurrence IA, the calculated date must fall within the range of four years earlier and seven years later than the current year. If the current year is 1997, the resulting date of the temporary variable must be later than 92/12/31 and earlier than 05/01/01.

**Example 3 (SUBSTR usage)**

- Using a variable defined in a JCL variable table:

```
//*OPC SETVAR TVAR=SUBSTR(&VAR1,2,4)
```
VAR1 is a variable defined in a JCL variable table.

TVAR is a substring of VAR1 value, starting from position 2 for a length of 4 characters.

- Using a predefined variable:

```
//*%OPC SETFORM ODATE=(YYMMDD)
//*%OPC SETVAR TVAR1=('&OCDATE')
//*%OPC SETVAR TVAR2=SUBSTR(&TVAR1,3,2)
//*%OPC SETVAR TVAR3=(OCDATE + 1MO)
//*%OPC SETVAR TVAR4=SUBSTR(&TVAR3,3,2)
```

If the occurrence input arrival date is 08/06/16, the expressions are substituted as follows:

TVAR1 = 080616
TVAR2 = 06
TVAR3 = 080716
TVAR4 = 07

In fact the SUBSTR parameter identifies a substring of TVAR1 and TVAR3 values, starting from position 3 for a length of 2 characters. According to the format set by the SETFORM directive, it identifies the MM part of the date value.

**Example 4 (arithmetic with temporary variables)**

```
//*%OPC SETVAR TX=(1)
//*%OPC SETVAR TY=(2)
//*%OPC SETVAR TZ=(&TX+&TY)
```

TZ is a temporary variable set to the result of the arithmetic addition.

**Example 5 (concatenating temporary variables)**

```
//*%OPC SETVAR T001=('STRING1')
//*%OPC SETVAR T002=('STRING2')
//*%OPC SETVAR T003=('&T001 &T002 CONCATENATED STRINGS')
```

T003 is a temporary variable set to the following value: STRING1 STRING2 CONCATENATED STRINGS
### SETVAR directive

**Table 28. Date formats allowed in the SETVAR directive**

<table>
<thead>
<tr>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODMY1</td>
<td>ODMY2</td>
<td>OYMD1</td>
<td>OYMD2</td>
<td>OYMD3</td>
<td></td>
</tr>
<tr>
<td>OLYMD</td>
<td>OLMD</td>
<td>CDDMMYY</td>
<td>CYMD</td>
<td>CYYMDD</td>
<td>OCDATE</td>
</tr>
<tr>
<td>OCFRSTW</td>
<td>OCFRSTWY</td>
<td>OCFRSTC</td>
<td>OCLASTW</td>
<td>OCLASTWY</td>
<td>OCLASTC</td>
</tr>
<tr>
<td>CDATE</td>
<td>OPIADATE</td>
<td>OPLSDATE</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 29. Day-in-year formats allowed in the SETVAR directive**

<table>
<thead>
<tr>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>ODDD</td>
<td>OYYDDD</td>
<td>OLYYDDD</td>
<td>CDDD</td>
</tr>
<tr>
<td>CYYDDD</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 30. Month formats allowed in the SETVAR directive**

<table>
<thead>
<tr>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>OMM</td>
<td>OMMYY</td>
<td>OYM</td>
<td>OYYMM</td>
<td>CMM</td>
</tr>
<tr>
<td>CYYMM</td>
<td>CYYYYMM</td>
<td>OLMM</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Table 31. Time formats allowed in the SETVAR directive**

<table>
<thead>
<tr>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>OHHMM</td>
<td>OLHHMM</td>
<td>CHHMM</td>
<td>CHHMSSX</td>
<td>CTIME</td>
</tr>
<tr>
<td>OPIATIME</td>
<td>OPLSTIME</td>
<td></td>
<td></td>
<td>OCTIME</td>
</tr>
</tbody>
</table>
TABLE directive

Purpose
This directive defines a variable table that will be searched before the variable tables in any existing concatenation when resolving JCL variables.

Syntax
```
//***OPC TABLE NAME=(table name)
```

Parameters
NAME=(table name)
Identifies the variable table that you want to precede any existing table concatenation.

Usage Notes
The TABLE directive is used to include a variable table at the front of any existing table concatenation. The table specified by table name must have already been created using the OPC JCL VARIABLE TABLES panel (see “User-defined variables and variable tables” on page 454). The table name specified can be the name of a table or a & or % variable. This variable follows the standard Tivoli Workload Scheduler for z/OS format and can be promptable or not promptable.

More than one TABLE directive can be used within the operation JCL. As each table statement is found, the table name it specifies is added to the front of the table search list. Up to 16 tables can be concatenated in this way, including application and global variable tables.

When a variable is assigned a value, this value is unaffected by any later table concatenations.

In the following example, variables HIONE and data set2 are defined in both TABLE1 and TABLE2.

Example
```
//***OPC SCAN
//***OPC SEARCH NAME=(TABLE2)
//DDNAME1 DD DSN=HIONE..FINANCE,DISP=SHR
//***OPC TABLE NAME=(TABLE1)
//DDNAME2 DD DSN=HIONE.&DATASET2.,DISP=SHR
```

The search order for variables found after the TABLE directive is reached would be:
1. TABLE1
2. TABLE2
3. The application variable table (if it exists)
4. The global variable table.

The variable HIONE was set before the TABLE directive was reached. The value it is assigned, therefore, is taken from TABLE2, not from TABLE1. The variable data set2, however, because it was not found before the TABLE directive, is assigned a value from TABLE1.
BEGIN and END directives

Purpose
These directives, used in pairs, denote the following, depending on the value of the ACTION keyword:
- The start and end of Tivoli Workload Scheduler for z/OS variable substitution
- The start and end of the lines to be included in the tailored job
- The start and end of the lines to be excluded from the tailored job

Syntax

```
BEGIN
ACTION=EXCLUDE
PHASE=SETUP
COMP=(<comparison expression>)

INCLUDE
NOSCAN
SUBMIT

//%OPC END ACTION=EXCLUDE
INCLU
NOSCAN
```

Parameters

**ACTION=(EXCLUDE | INCLUDE | NOSCAN)**
- Specifies which BEGIN/END action is required.
  - **EXCLUDE** This specifies that the lines following this BEGIN directive up to the next END ACTION=EXCLUDE directive should be excluded from the job that is submitted for this operation.
  - **INCLUDE** This specifies that the lines following this BEGIN directive up to the next END ACTION=INCLUDE directive should be included as part of the job that is submitted for this operation.
  - **NOSCAN** This specifies that any variables following this BEGIN directive up to the next END ACTION=NOSCAN directive should not be substituted. When you specify NOSCAN, you do not specify a PHASE parameter, because NOSCAN is valid at both submit and at setup.

**PHASE=SETUP | SUBMIT**
- Specifies whether the BEGIN/END pair should take effect during the setup or submit phase of the operation. This parameter is required only if ACTION=EXCLUDE or INCLUDE.

**COMP=(<comparison expression>, <comparison expression>, ...)**
- Specifies comparison expressions that are used to decide whether the BEGIN directive should be acted on. If the comparison expression is true, the BEGIN directive is honored. For details on defining comparison expressions, see "The COMP keyword on BEGIN and FETCH directives" on page 485.
BEGIN and END directives

Usage Notes
In a job, every BEGIN directive must have a matching END directive specifying the same ACTION. For example, the directive:

```plaintext
//*%OPC BEGIN ACTION=EXCLUDE,PHASE=SUBMIT
```

requires the following matching END directive:

```plaintext
//*%OPC END ACTION=EXCLUDE
```

If Tivoli Workload Scheduler for z/OS detects an unpaired BEGIN or END, the processing ends in error. Even a BEGIN statement that is not honored because its comparison expression is not true requires a matching END statement.

Only the following directives can lie within the domain of a BEGIN ACTION=NOSCAN directive and an END ACTION=NOSCAN directive:

- SEARCH
- SETFORM
- SETVAR
- TABLE

When these directives are in the range of a NOSCAN directive, they are always acted upon even if there is a comparison condition that is false.

BEGIN and END directives that specify ACTION=INCLUDE or ACTION=EXCLUDE cannot be nested and cannot overlap. They can, however, completely contain a nested NOSCAN domain.

Consider the following examples:

**Example 1**

```plaintext
//*%OPC SCAN
//*%OPC BEGIN PHASE=SETUP,ACTION=INCLUDE
/DDNAME1 DD DSN=&HIONE..&DATASET1.,DISP=SHR
/DDNAME2 DD DSN=&HIONE.&DATASET2,DISP=SHR
//*%OPC END ACTION=INCLUDE
```

Example 1 is valid. Lines 2 and 3 will be included in the job for the operation.

**Example 2**

```plaintext
//*%OPC SCAN
//*%OPC BEGIN PHASE=SUBMIT,ACTION=EXCLUDE
//EXEC PGM=MYPROG
//*%OPC BEGIN PHASE=SETUP ACTION=INCLUDE
/DDNAME1 DD DSN=&HIONE.&DATASET1,DISP=SHR
//SYSOUT DD SYSOUT=A
//*%OPC END ACTION=EXCLUDE
/DDNAME2 DD DSN=&HIONE.&DATASET2,DISP=SHR
//*%OPC END ACTION=INCLUDE
```

Example 2 is invalid. An EXCLUDE action (lines 2 and 7) overlaps an INCLUDE action (lines 4 and 9). Error message EQQJ533 will be issued.
BEGIN and END directives

<table>
<thead>
<tr>
<th>Example 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>//**OPC SCAN</td>
</tr>
<tr>
<td>//**OPC BEGIN PHASE=SUBMIT,ACTION=EXCLUDE</td>
</tr>
<tr>
<td>//EXEC PGM=MYPROG</td>
</tr>
<tr>
<td>//**OPC BEGIN ACTION=NOSCAN</td>
</tr>
<tr>
<td>//DDNAME1 DD DSN=&amp;HIONE.&amp;DATASET1,DISP=SHR</td>
</tr>
<tr>
<td>//SYSOUT DD SYSOUT=A</td>
</tr>
<tr>
<td>//**OPC END ACTION=NOSCAN</td>
</tr>
<tr>
<td>//DDNAME2 DD DSN=&amp;HIONE.&amp;DATASET2,DISP=SHR</td>
</tr>
<tr>
<td>//**OPC END ACTION=EXCLUDE</td>
</tr>
</tbody>
</table>

Example 3 is valid. The NOSCAN domain defined by lines 4 and 7 is completely contained within the BEGIN and END ACTION=EXCLUDE (lines 2 and 9).

Note also that the variable HIONE on line 3 will be substituted, but the variable HIONE on line 5 will not be substituted because it is within a NOSCAN domain.

BEGIN and END directives that specify ACTION=INCLUDE or ACTION=EXCLUDE can modify the structure of JCL from one run to another. For this reason, to apply again the same directives in a Restart and Cleanup path, ensure that they do not alter the structure of the JCL. For details, refer to "JCL changes considerations" on page 343.
FETCH directive

Purpose
This directive lets you include lines, fetched from a partitioned data set member or supplied by an exit, in your job.

Syntax

```
//*%OPC FETCH

member name

EXIT=exit name

SUBMIT

PHASE=SETUP

COMP=(comparison expression)
```

Parameters

**MEMBER=(member name)**
Specifies the member name of a partitioned data set allocated to ddname EQQJBLIB. The lines in this member are included immediately after the FETCH directive.

**EXIT=(exit name)**
Specifies an exit to be called when the FETCH statement is invoked. This exit supplies lines to be inserted immediately after the FETCH directive in the job. Refer to Customization and Tuning for more details on exits.

**PHASE=SETUP|SUBMIT**
Specifies whether the FETCH should take effect during the setup or submit phase of the operation.

**COMP=((comparison expression), (comparison expression),...)**
Specifies comparison expressions used to decide whether the FETCH directive should be acted on. If the comparison expression is true, the FETCH directive is honored. For details on defining comparison expressions, see “The COMP keyword on BEGIN and FETCH directives” on page 485.

Usage Notes
The FETCH directive is used to include lines from other partitioned data sets or as supplied by an exit. Lines included by a FETCH directive cannot contain another FETCH directive. BEGIN and END directives with action INCLUDE or EXCLUDE cannot be included in lines inserted by a FETCH directive.

Tivoli Workload Scheduler for z/OS variables can be used to represent the values of any keywords, but not the keywords themselves. A FETCH directive cannot lie between a BEGIN/END directive pair that specifies ACTION=INCLUDE or ACTION=EXCLUDE.

Example

```
//*%OPC SCAN
//*%OPC FETCH PHASE=SUBMIT,
//*%OPC MEMBER=JCL1,
//*%OPC COMP=(&DAY..EQ.1)
```
FETCH directive

FETCH directives can modify the structure of JCL from one run to another. For this reason, to apply again the same directives in a Restart and Cleanup path, ensure that they do not alter the structure of the JCL. For details, refer to “JCL changes considerations” on page 343.
The COMP keyword on BEGIN and FETCH directives

Purpose
A comparison expression lets you specify conditions when BEGIN and FETCH directives will be honored.

Syntax

```
COMP=( (expression1 .EQ. expression2) .NE. .GE. .GT. .LE. .LT. )
```

Parameters

expression1
This specifies a string made up of &-variables and alphanumeric literals. Any global search characters it contains are treated as literals. The value of expression1, arrived at by resolving any variables specified, will be tested against the values given by expression2.

Operators.
These values are operators that specify which comparison should be made between expression1 and any expression2 values.

.EQ. Expression1 must equal one of the expression2 values for the expression to be true.

.NE. All expression2 values must not equal the expression1 value for the expression to be true.

.GT. Expression1 must be greater than the expression2 value for the expression to be true.

.GE. Expression1 must be greater than or equal to the expression2 value for the expression to be true.

.LT. Expression1 must be less than the expression2 value for the expression to be true.

.LE. Expression1 must be less than or equal to the expression2 value for the expression to be true.

expression2
This parameter can be made up of &-variables, literals, or, if .EQ. or .NE. operators are specified, one of the two global search characters, % and *.

The length of the resolved value cannot exceed 44 characters. The % global search character represents any single alphanumeric character. The * global search character represents any alphanumeric string, including a null string.

If GT, GE, LT, or LE is specified:
- Multiple values of expression2 are not supported.
- Global search characters are not supported.
- If the strings on both sides of the operators are of different lengths, the comparison is made using the shorter string.
The COMP keyword on BEGIN and FETCH directives

Note: The % symbol does not signify a % Tivoli Workload Scheduler for z/OS variable within a COMP keyword. The % and ?-variables are not valid within a COMP statement.

Usage Notes
The COMP expression cannot exceed 256 characters unresolved, and cannot be more than 1024 characters after substitution; expression2 can be any &-variable, either user-defined or defined by Tivoli Workload Scheduler for z/OS. Neither expression1 nor expression2 can have embedded blanks.

Consider the following examples:

Example 1

/**<OPC FETCH PHASE=SETUP,
/**<OPC MEMBER=MYJCL,
/**<OPC COMP=(&APPL..EQ.(APPL1,APPL2,APPL3))

If &APPL. is equal to APPL1 or APPL2 or APPL3, the expression is true, and the FETCH directive will be honored.

Note the two periods following &APPL.. The first signifies the end of the variable APPL; the second signifies the start of the comparison operator EQ.

Example 2

/**<OPC FETCH PHASE=SETUP,
/**<OPC MEMBER=MYJCL,
/**<OPC COMP=(&DAY..NE.(1,3,5))

In example 2, if &DAY is not equal to 1 or 3 or 5, the expression is true, and the FETCH directive will be honored. If DAY had been equal to any one of the comparison values, the expression would have been false.

For the COMP keyword to be true, all the comparison expressions that it consists of must be true. This is shown in the following example:

Example 3

/**<OPC BEGIN ACTION=INCLUDE,
/**<OPC COMP=((&APPL..EQ.(APPL1,APPL2,APPL3)),
/**<OPC ((&DAY..NE.((1,3,5)))

For the COMP statement in example 3 to be true, the expressions (&APPL..EQ.(APPL1,APPL2,APPL3)) and (&DAY..NE.((1,3,5))) must both be true.

The expression2 values that you specify can be made up of &-variables, alphanumeric literals, and the * and % global search characters. National characters, left and right parentheses; ( and ), and blanks are not allowed; if they are specified, the results are unpredictable. The * global search character represents a character string of any length; the % global search character represents exactly 1 character. If variables and global search characters are combined, the variables are resolved before any comparisons are made using the global search characters.

Example 4
The COMP keyword on BEGIN and FETCH directives

///%OPC BEGIN ACTION=INCLUDE,
///%OPC     COMP=(&MYVAR..EQ.(TSO199,TSO2%%.,&VALUE1.*))
///%OPC END ACTION=INCLUDE

In example 4, the variable &MYVAR must have one of the following values for the comparison expression to be true:
- TSO199
- TSO2 followed by any 2 alphanumeric characters except blanks
- The value of variable &VALUE1 followed by an alphanumeric string of any length, including length 0.

Example 5
///%OPC BEGIN ACTION=EXCLUDE,PHASE=SUBMIT,
///%OPC     COMP=(&CYYMMDD..GE.000101)
///%OPC END ACTION=EXCLUDE

Note that COMP statements can give unexpected results with some of the date formats of the supplied variables. When date variables are substituted, they are compared as numerals, not as dates.

In example 5, &CYYMMDD is the current date and 000101 represents 1 January 2000. If the value of expression1 is greater than 000101, the comparison expression is true—even though a date in the 20th century is earlier than 1 January 2000. You will also have problems with comparing dates specified in formats like DDMMYY or MMDDYY.
Restrictions on the use of variables

Line-length errors

Normally, job lines are a maximum of 71 characters. In jobs containing in-stream data, the in-stream data lines can be up to 80 characters. If the value of the variable being substituted is longer than the length of the variable name, truncation of the line might occur if the newly created line is longer than the maximum length allowed. For example, if you define a variable &data set, which has a length of 8 characters including the &, but give it a value 'MY.data set.NAME', with a length of 15 characters, you must ensure that there is enough room at the end of the line to allow for the substitution. If Tivoli Workload Scheduler for z/OS calculates that a line-length error will occur, substitution ends and the operation status is set to E, ended-in-error, with the error code OJCV. Tivoli Workload Scheduler for z/OS will, however, truncate comments on lines if necessary without giving an error.

Tivoli Workload Scheduler for z/OS substitutes variables in a series of passes. A truncation error occurs only if, after the final pass of a phase, the line produced is longer than permitted. The final pass is the pass that resolves the last remaining variable. During preceding passes, Tivoli Workload Scheduler for z/OS permits the line to extend beyond the normal limits of a job.

Note: It is recommended that substitution on in-stream data lines occur only for job library members that use the ISPF profile option STD numbers off. When STD numbers are present in columns 73–80, variable substitution on in-stream data lines can produce unpredictable results. Do not use the string 0PCSM56 in positions 73-79, because Tivoli Workload Scheduler for z/OS uses this to position displayed lines at error messages.

Strings you cannot use variables to represent

Tivoli Workload Scheduler for z/OS variables can represent the values of most keywords, but you cannot use variables to represent certain keywords in both standard z/OS JCL and in Tivoli Workload Scheduler for z/OS directives.

- In the case of z/OS JCL, the restricted keywords are:
  - JOB in the JOB statement
  - EXEC in the EXEC statement
  - INCLUDE in the INCLUDE statement
  - DD in the DD statement
  - PROC in the PROC statement
  - PEND in the PEND statement

  If these fields are replaced by variables, the results are unpredictable.

- In the case of the Tivoli Workload Scheduler for z/OS directives, the restricted keywords are:
  - The string /*%OPC
  - The following command names:
    BEGIN
    END
    FETCH
    NOP
    SCAN
Strings you cannot use variables to represent

SEARCH
TABLE
- The keyword to the left of each equals sign:
  ACTION
  COMP
  EXIT
  MEMBER
  NAME
  PHASE
- Certain values to the right of the equals sign:
  APPL
  EXCLUDE
  GLOBAL
  INCLUDE
  NOAPPL
  NOGLOBAL
  NOSCAN
  SETUP
  SUBMIT

The use of variables that contain SCAN terminating keywords should be avoided.

You can substitute the following directive variables:
- Table name in the TABLE directive
- Member name in the FETCH directive
- Exit name in the FETCH directive
- All variables in the comparison expressions on the FETCH, INCLUDE, and EXCLUDE directives

Other directive variables, such as the stepname in the ERRSTEP keyword of the RECOVER statement, cannot be substituted.

Avoiding loops in variable substitution

Tivoli Workload Scheduler for z/OS does not let you define a variable-substitution loop. If you give variable &A the value &B and then give variable &B the value &A, a variable loop is formed. If Tivoli Workload Scheduler for z/OS detects that a variable has recurred in a variable for variable type substitution, the substitution ends in error. There are three possible loops:
- Substitution of variables by variables (described here)
- Dependency of variable A on variable B, which depends on variable A
- A combination of both of these cases

Order of variable substitution

If you specify a compound variable %VAR1%VAR2...DATA, where VAR1 is a promptable variable and VAR2 is a (not promptable) setup variable, the substitution fails, because Tivoli Workload Scheduler for z/OS resolves the compound variable right-to-left.

When you make one variable dependent on another, the independent variable must occur first (you can put it in a comment line), and must be substitutable in the same phase or in an earlier phase. This restriction does not apply to non-centralized scripts.
Using a default calendar name

Using a default calendar name

Tivoli Workload Scheduler for z/OS sometimes needs a calendar when it substitutes variables. The subsystem looks for a calendar in this sequence:

1. An explicitly specified calendar, connecting the calendar name to the application occurrence
2. When you have manual job setup, and only for variables resolved at the setup phase, the calendar specified in the OPTIONS panel
3. The calendar called DEFAULT

Variable substitution for jobs running on fault-tolerant workstations

This section describes the automatic variable substitution provided for job definitions stored in the EQQSCLIB data set. It enables you to specify variables when you set a statement keyword in the job definition. Variable substitution occurs when:

- The Daily Planning process creates the Symphony file.
- The job is added to the plan using the Modify Current Plan dialog.

Activating variable substitution

You activate variable substitution by specifying VARSUB in the first statement of your job definition. Refer to Customization and Tuning for the full syntax. A variable consists of up to 8 alphanumeric characters.

Any non-alphanumeric character, except blanks, can be used as a symbol to indicate that the characters that follow represent a variable. You can define two kinds of symbols using the PREFIX or BACKPREF keywords in the VARSUB statement: it allows you to define simple and compound variables.

Supplied Variables and User-defined Variables tables are supported. Refer to "Supplied variables" on page 449 and "User-defined variables and variable tables" on page 454 for details. For information about associating variable tables with applications, see "Associating variable tables with applications" on page 442.

The following example shows how to use a simple variable. When scheduling the operation, the scheduler automatically sets the simple VUSER variable to the value that is specified in the E2EVARTAB variable table that is defined in the Application Description file.

VARSUB
  TABLES(E2EVARTAB)
  PREFIX(^
  JOBREC
    JOBCMD('dir')
    JOBUSR(&VUSER )

In the following example of a compound variable, two substitution steps are performed automatically. The %OWSID string is processed first. OWSID is an operation-related supplied variable that is automatically set to the workstation ID for the current operation: assuming that FTW1 is the workstation ID, the first substitution step produces &FTW1SDIR\my.cmd string. If you define the variable &FTW1SDIR in the E2EVARTAB application description table and set it to the c:\win32app\maestro value, the final result is that c:\win32app\maestro\my.cmd will be scheduled.
Activating variable substitution

VARSUB
  TABLES(E2EVARTAB)
  PREFIX('&')
  BACKPREF('¥')
  JOBREC
  JOBSCR('¥OWSID.SDIR..my.cmd')
  JOBUSR('tws')

Restrictions
You can have variables that depend on other variables, according to the data specified in the dependency value list as described in "Making one variable dependent on another" on page 457. Dependent and independent variables must be defined inside the same table, and a default value must be assigned to the independent variable.

Variable substitution for job types with advanced options
You can apply variable substitution also to job types with advanced options (jobs that run tasks within the areas of File Transfer, Web Services, Database, Java, and more). If you use the JOBREC command to create or modify these jobs, apply the variable substitution rules documented in the preceding sections. If you use the Dynamic Workload Console, the rules for doing this diverge from standard. This section documents how to implement variable substitution for this type of jobs when you edit them on the Dynamic Workload Console, and the differences and limitations with respect to the standard process.

Adding variables in the job definitions on the Dynamic Workload Console
In the job definition lists of the Dynamic Workload Console, the following job types are categorized as job types with advanced options:
  • Executable
  • Java
  • WebServices
  • FileTransfer
  • J2EE
  • XaJob
  • Database
  • JCL
  • MSSQL Job
  • Any custom jobs for which you develop both the underlying code and the definition and management panels.

When you edit any of these jobs, you can insert variables in the definition fields using the syntax:

$\{variable\_name\}$

where the value for the variable is in one of the tables specified by the VARTABLES keyword described in the next section.

You can use compound variables, that is, more variables joined together as follows:

$\{VAR1\(VAR2\)\…\(VARn\)\}$

where the variables are resolved in several passes starting from the rightmost first and proceeding to the left.
Adding variables in the job definitions on the Dynamic Workload Console

Limitations
Variable substitution for job types with advanced options defined from the Dynamic Workload Console is very straightforward but does not allow most of the options available for normal jobs or for this type of jobs when defined with JOBREC. It is also useful to reiterate that:

- You cannot use variable substitution on passwords. A warning is displayed by the Dynamic Workload Console when you try to save a job containing variables in the password within the user credentials section.
- Variable dependency is not supported, due to the difficulty in these jobs to enforce the rule that the independent variable must come first and to the impossibility to use comment lines to "place" the independent variable before the dependent one.

Available user exits
The SEQQSAMP sample library that was created during installation contains the EQQJVXIT exit, which is a sample exit coded to apply variable substitution. When you specify a job definition variable, you can specify the name of this exit that is called when substitution of the variable is required. You can use the exit to supply the value of a variable. Look in EQQJVXIT and its coded documentation for details.

Configuration requirements
To be able to implement variable substitution on these jobs, specify the following keywords in the HTTPOPTS initialization statement:

**VARSUB (NO | YES)**
Specifies whether variable substitution is enabled for jobs with advanced options. YES means that variable scanning will be run on this type of jobs. NO means that variable scanning will not take place.

**VARTABLES (GLOBAL | APPL | table1, table2, ...)**
Identifies the variable tables that must be searched and the search order:

- **GLOBAL**
  Points to the table defined in the GTABLE keyword of the OPCOPTS controller.
- **APPL**
  Points to the application variable table.
  
  *table1, table2, ...*
  Point to the tables named here. Their names can be no longer than 16 characters and the number of tables in this list is limited to 16.

**VARFAIL (YES | NO)**
Specifies whether the scheduler is to issue an error message when a variable substitution error occurs. If you specify NO, the variable string is left unchanged without any translation.

See the details on the HTTPOPTS initialization statement on the Customization and Tuning guide.
This chapter describes the possible integration and exploitation of the WorkLoad Manager (WLM) component of z/OS. Today, schedulers process different types of work with different completion and resource requirements. Every scheduler aims at making the best use of its resources, at maintaining the highest possible throughput, and at achieving the best possible system response. WLM makes this possible.

With WLM, you define performance goals and assign a business importance to each goal. You define the goals for work in business terms and the system decides the amount of resources, such as CPU and storage, it should provide to meet these goals. WLM constantly monitors the system and adapts processing to meet the goals.

You can also tell WLM which are the resources a job needs to run and WLM will run it on a system where these resources are available.

The workloads, their requirements, and their performance goals are defined to WLM through the service definition. Workload management provides an online panel-based application for setting up and adjusting the service definition. You specify the service definition through this ISPF administrative application. There is one service definition for the entire Sysplex. It contains all the information WLM needs to process, that is:

**Service Policy**
A named set of performance goals (eight characters long) that workload management uses as a guideline to match resources to work. When activated, a service policy is merged with the service definition and applies to all the Sysplex.

**Service Class**
A named group of work (eight characters long), within a workload, with similar performance characteristics:
- Performance goals
- Resource requirements
- Business importance

**Resource Group**
A named amount of processor capacity (eight characters long) across the Sysplex. It can be associated to a service class in the service class definition.

**Classification Rules**
Define how to assign an incoming work to service classes and report classes.

**Application Environment**
A named group of application functions (32 characters long) that execute in server address space and can be requested by a client.

**Scheduling Environment**
A named list of resource names (16 characters long) along with their required states. You can specify up to 999 unique scheduling environments in a service definition, according to the following convention:
- Alphanumeric and the special characters @, $, #, and _ are allowed.
- Underscore (_) must be imbedded
The scheduling environment can be defined in the job card of a JCL (SCHENV=SE_name) so that, at submit time, WLM has a list of the systems where the job can run (a system where the scheduling environment is active), and pick one according to the active service policy.

When no system with an active scheduling environment is available, the job is not started and is placed on the WLM hold queue: it will be started as soon as the scheduling environment becomes active on a system within the Sysplex. If the scheduling environment does not exist at all, the job fails with JCL error

**Workload**
A collection of service classes. It is used for reporting purposes.

**Report Classes**
A group of work utilized for reporting.

Tivoli Workload Scheduler for z/OS is able to submit large quantities of work into z/OS systems. The work is submitted only after a number of predetermined conditions are met, but the scheduler does not know the state of the system (in terms of load and capacity, but also in terms of resource availability) when the work is submitted.

Some of the main objectives of Tivoli Workload Scheduler for z/OS are:
- Running your workload as efficiently as possible, in order to complete all work according to your time requirements.
- Monitoring the workload constantly to detect any possible problems and fix them immediately.

To match these objectives, Tivoli Workload Scheduler for z/OS exploits the services of WLM by providing integration with:

**Service Class**
After the jobs are defined as critical to WLM (at the AD and CP operation levels), Tivoli Workload Scheduler for z/OS uses customizable criteria (either as a controller initial parameter or within the operation definition) to check if these jobs are running late: if they are, the scheduler moves them to the predefined WLM high performance service class.

**Scheduling Environment**
Integration is accomplished by:
- Associating a scheduling environment to the operations (via Tivoli Workload Scheduler for z/OS ISPF panels or programming interfaces). At submission time, the JCLs are tailored by adding the SCHENV=SE_name statement in the JOB cards. A pre-existing SCHENV keyword in the JOB card is taken into account depending on the OPCOPTS SECHECK parameter.
- Monitoring the scheduling environment as follows:
  1. If the associated scheduling environment is not available, the job is not submitted and is assigned the waiting for scheduling environment extended status.
  2. As the scheduling environment becomes available, an exit activated by one of the trackers forwards an event to the scheduler which then submits the job again.
Integrating with the service class

Integrating with the service class

To minimize delays in the execution of the workload, Tivoli Workload Scheduler for z/OS:

1. Detects that a critical job is running late.
2. Calls a documented WLM interface to move the job to a higher performance service class.
3. Conveys additional system resources so that the job can complete in a shorter time.

The delay in the batch workload is therefore reduced.

Environment

To benefit from the WLM function, your system must be a SYSPLEX (at least MONOPLEX) environment running in WLM goal mode and you must set up a higher-performance service class for occasional use (refer to the WLM documentation).

In the OPCOPTS initialization statements, you have to include the WLM statement with the appropriate parameters (refer to Customization and Tuning to enable Tivoli Workload Scheduler for z/OS to send promotion requests to WLM.

When a job should be set as critical

You should always set a job as critical if its delay could imply a delay in the plan. WLM knows the availability of the overall resources against the system workloads when the current plan is executed and the real duration of a job is determined by this factor. Therefore, it is better also that you set as critical those jobs that would be on a critical path if their estimated durations were only minimally longer than at present. To set a job as critical, you have to enter Y in the CRITICAL field on the JOB, WTO, AND PRINT OPTIONS panel (accessible from either the APPLICATION DESCRIPTION panel or the MODIFYING THE CURRENT PLAN panel).

Selecting WLM assistance policies

When you have decided that a job should be considered as critical, Tivoli Workload Scheduler for z/OS allows you to choose between different assistance policies. To activate a particular policy, you have to specify the code that identifies it in the POLICY field on the JOB, WTO, AND PRINT OPTIONS panel (accessible from either the APPLICATION DESCRIPTION or MODIFYING THE CURRENT PLAN panel). If you do not specify a policy for a particular critical job, Tivoli Workload Scheduler for z/OS applies either the common policy specified in the WLM statement of OPCOPTS or the default common policy (if one has not been specified). Valid assistance policy codes are:

L Long Duration. When a critical job runs over the expected duration (as specified by the user), it is promoted to a higher-performance class by the scheduler. This policy is activated by specifying the letter L in the Assist Policy field on the APPLICATION DESCRIPTION panel.

Note: The resources are used also when not necessary, but for a short interval.
Selecting WLM assistance policies

D Deadline. When a critical job runs over its real deadline (calculated as Latest Start Time + Duration), it is promoted to a higher-performance class by the scheduler.

Note: This policy is always recommended when the S(mart) policy is not used. The resources are used only when strictly necessary, but it might be too late.

S Latest Start Time. When a critical job starts after its latest start time, it is promoted to a higher-performance class by the scheduler.

Note: The resources are used also when not strictly necessary, but it can prevent delayed termination of jobs.

C Conditional. The assistance policy is a smart compromise between the Deadline and the Latest Start Time policies as follows:
1. Tivoli Workload Scheduler for z/OS checks if the conditions of the Deadline policy are met and promotes the job.
2. If the conditions of the Deadline policy are not met, the promotion is immediate for jobs started with a significant delay with respect to the remaining time to deadline. The time to deadline can have different weights according to the Threshold value that you specify in the WLM statement of OPCOPTS. The default value is 20%. If you specify a value of 0%, the Latest Start Time policy is automatically applied in this second step. If you specify a value of 100%, the Latest Start Time policy is applied only when the time from the Latest Start Time is greater than that remaining to the deadline.

If a different policy is not specified in the WLM statement, this is the policy that Tivoli Workload Scheduler for z/OS uses as a common policy.

Note: This policy is always recommended, but it requires additional tuning efforts for correct setting of the Threshold value.

The advantages and disadvantages of the assistance policies are:

Table 32. Advantages and disadvantages of assistance policies

<table>
<thead>
<tr>
<th>Assistance Policy (Intervention Option)</th>
<th>Advantages</th>
<th>Disadvantages</th>
</tr>
</thead>
</table>
| Duration                               | • Gentle smoothing  
• Low impact                        | • Possibility of intervention when not actually necessary  
• Limited gain                      | |
| Deadline                               | • Gentle smoothing  
• Low impact  
• 100% certainty that intervention is required | Limited gain |
| Latest Start Time                      | • High gain  
• High probability that intervention is required | High risk of overcompensating |
| Conditional                            | • Best action taken for every situation  
• No decision necessary by you         | Tuning efforts to set the Threshold value |
The Workstation Analyzer checks for late operations every two minutes. When it discovers a late operation that is critical, there might be additional two minutes delay before the WLM action is taken (for example, to promote the job to the service class).

**Integrating with the scheduling environment**

The objective is to facilitate the association of WLM scheduling environments with Tivoli Workload Scheduler for z/OS jobs.

The main benefits are:

- **JCLs no longer have to be manually tailored to associate a scheduling environment with a job or to modify an existing association. In fact:**
  - The association can be made within the definition of the operation (through the ISPF panel or the PIF).
  - Global changes for groups of jobs can be made with Mass Update and Batch Loader support.
- **Jobs can be monitored for problems related to the WLM scheduling environment.**
  - The new Waiting for Scheduling Environment extended status for operations enables a user to track scheduling environment problems via ISPF.
  - The possibility to filter operations by their scheduling environment name enables a user to identify all operations blocked by a specific scheduling environment.
- **The retry mechanism is based on the detection of scheduling environment availability. As soon as the associated scheduling environment becomes available, an event is sent to the controller to retry the operations waiting for it.**
- **The integration is supported across the entire Sysplex.**
  - The availability of the scheduling environments is checked for at the tracker level, where WLM macros are used with sysplex-wide scope.
  - The possibility to define a scheduling environment at the job level preserves the administrator from the need to define scheduling environments by sysplex. For example, scheduling environment SE1 can be defined both in Sysplex A and Sysplex B, but they are two separate objects with different status.

**Associating a scheduling environment with a job**

To associate a Scheduling Environment with an operation at the application description and current plan levels, you can use, via ISPF, PI, BCIT or OCL:

- At submit time, the EQQUX001 controller user exit. EQQUX001 receives in input the scheduling environment value stored in the CP operation definition and has the option to change it. The returned value is then stored in the CP.
- At plan creation time, the EQQDPX01 DP batch user exit. EQQDPX01 is called for every new operation (with the exception of fault-tolerant end-to-end scheduling operations); that is, for each operation extracted from LTP. It receives in input the scheduling environment value associated to the operation and has the option to change it. The returned value is then stored in the CP.

Note that the concept of scheduling environments does not apply to started tasks (the SCHENV= keyword cannot be inserted into a started task).
Submitting prevention and automatic tailoring

If a scheduling environment is associated to an operation, when the time to submit the operation arrives:

- The scheduling environment is monitored for availability at tracker level. The tracker does not submit the operation as long as the scheduling environment is not available.
- The JCL is tailored by inserting the SCHENV=CP defined Scheduling Environment name parameter in the JCL JOB card if the availability check is positive.

Scheduling environments are monitored and defined in the JCLs according to the value of the OPCOPTS SECHECK parameter specified in the tracker that will be submitting the operation. You have the option to:

- Not use scheduling environments (no check nor tailoring).
- Manage scheduling environments only if they are defined at operation level in the current plan.
- Manage scheduling environments also if the related operations do not have the association defined in the current plan but have the JCL containing the SCHENV= keyword.

For details, see Customization and Tuning.

Pre-existing definition in the JCL JOB card

By default, when no scheduling environment name is defined for an operation in the current plan, any SCHENV=SE_name contained in the JCL JOB card is supported. If the operation has a scheduling environment name defined in the current plan, the SCHENV=SE_name contained in the JCL JOB card is overridden. In this way, no extra processing is requested for the tracker submitting the job and you can activate this function at your will.

Checking for scheduling environment availability

The tracker submitting the job uses the WLM Query Service to check if the scheduling environment is available:

- If the scheduling environment is available on any system in the sysplex, the job is submitted and WLM has the task to identify the system where the job will run.

Whenever the new OPCOPTS JESPLEX parameter is specified, the associated list of system names is used as an additional filter in the WLM query to check for scheduling environment availability at JESplex level. This is required when there are more than one JESplex within the same sysplex.

**Note:** If you use virtual workstations with the WLM SCHENV interface, all the virtual destinations must be in the same sysplex and JESplex.

Based on WLM logic, to obtain full workload balancing, you should configure no more than one JESplex in the sysplex. However, deviations from this rule are often found throughout the installed basis. For a description of the new parameter and a troubleshooting for configuration problems, refer to “Multi-sysplex configuration with JESplex matching sysplex” on page 500.

- If the scheduling environment is not available on any system in the sysplex (or at the tracker JESplex level if the OPCOPTS JESPLEX parameter is specified) the operation is assigned the Ready status and the Waiting for scheduling environment extended status.

This process is not followed if the operation comes from the Restart and Cleanup path. In this case, the operation is assigned the SERC (SE not available for
Checking for scheduling environment availability

R&C operation) error status. Operations that are ready but waiting for the scheduling environment to be available are considered not eligible for submission by the controller.

- If the scheduling environment is not defined in the sysplex, the operation status is set to Error with the SEUN (SE undefined) error code.

Restrictions for using job card comments on the right

When the JOB card is tailored to insert or replace the SCHENV keyword, any comments on the right might be ignored and could show truncated in the submitted JCL.

The retry mechanism for operation waiting for SE

To monitor the availability of the scheduling environments, the trackers activate an ENF listener mechanism for event code 57 (the event that reports the status change of a scheduling environment) and for event code 41 (the event that reports on WLM policy activation) according to the value stored in the OPCOPTS SECHECK parameter.

When a scheduling environment becomes again available on a specific image, the tracker generates a new event to report this to the controller. Consequently, the controller resets the extended status of the operation so that it is again eligible for submission.

When a Policy Activation occurs, all the operations waiting for Scheduling Environment belonging to the Sysplex where the Policy change occurred, are resubmitted.

Monitoring operations

All lists currently displaying the operation status, extended status, and error codes, support the new extended status and error codes. By the same token, all commands currently supported for ready operation remain available.

The Scheduling Environment name has been added as filtering criteria in operations list. If the Scheduling Environment name is not defined in the Current Plan at operation level, the Controller stores it in the Current Plan when notified about Scheduling Environment unavailability. If the job submission fails with SEUN or SERC error code, the Scheduling Environment name is not overridden.

Considerations for using the ROUTE JCL statement

A JCL might contain a ROUTE statement that causes the execution of the job on a different image than the Submit task image. JES processes the ROUTE statement after the job submission: at that point, the Submit task already performed the check on scheduling environment availability; therefore the check is effective or not depending on the configuration.

User actions to activate scheduling environment integration

The following is a list of required steps to activate scheduling environment integration.

Note: If you do not want to activate scheduling environment integration, no specific actions are requested. By default, the initialization parameters are set for no integration.
User actions to activate scheduling environment integration

1. To activate the function, for each tracker (and for the controller if it uses the submit function) set the SECHECK parameter to a value other than NO. Depending on your requirements, set to ALL or to OPERONLY:
   - **ALL**: If you do not plan to define the scheduling environment in the Tivoli Workload Scheduler for z/OS database
   - **OPERONLY**: All the other cases.

2. If you use a multi-sysplex configuration, define the new OPCOPTS SYSPLEXID parameter for each tracker. This must be a unique number throughout the entire configuration.

3. If you use multiple JESplex within a sysplex, define the new OPCOPTS JESPLEX parameter for each tracker. This must be a list of system names belonging to the tracker JESplex or use scheduling environment best practice definitions.

The following is a list of optional steps to activate scheduling environment integration:

1. Define a scheduling environment at operation level.
2. Implement the EQQUX001 controller exit routine to set the scheduling environment name if appropriate.
3. Implement the EQQDPX01 exit routine to set the scheduling environment name if appropriate (for details, see "Performance considerations" on page 507).
4. Set the SUPPRESSENF parameter to YES, where applicable, to improve performance (for details, see "Performance considerations" on page 507).

Supported configuration

The following sections describe the more complex scenarios where you must take special care when setting up for WLM integration, that is:

- Multi-Sysplex configurations with matching JESplex and Sysplex
- Multi-Sysplex configurations with more JES complexes within a Sysplex

**Note:** If you use a virtual workstation for an operation that has a scheduling environment defined, then all the destinations in that virtual workstation must belong to the same Sysplex.

Multi-sysplex configuration with JESplex matching sysplex

Invoking the WLM query service at the tracker rather than at the controller level allows for support of multiple Sysplex handled by one controller (macro range is Sysplex-wide). Figure 196 on page 501 shows this type of configuration.
In this configuration, it is important that the controller is told from which Sysplex the tracker event communicating the availability of the scheduling environment comes from, so that the operations waiting for that scheduling environment can be...
submitted again. To do so, use the **SYSPLEXID** (*nn*) parameter, where *nn* can be an integer from 1 to 9999. The default is 1. This parameter applies both to trackers and controllers.

The following scenario, based on [Figure 196 on page 501](#) helps explain how SYSPLEXID is used.

A scheduling environment named DB2x is defined both in Sysplex 1 and in Sysplex 2 and initially it is unavailable in both Sysplexes:

1. You submit a job with SCHENV=DB2x on tracker T3 of system C, Sysplex 1.
2. Tracker T3 finds that the DB2x scheduling environment is not available and sends an IJ4 event, Submit failed, to the controller, with a record of the putting scheduling environment name and of the sysplex ID.
3. The controller places the operation in status READY, WAITING FOR SE and stores both the scheduling environment name (DB2x) and the sysplex ID (1).
4. DB2x becomes available on Sysplex 2. Tracker T5 of Sysplex 2 warns the controller by sending an event.
5. The controller checks the operation in status READY, WAITING FOR SE against the scheduling environment name (DB2x) and the sysplex ID (2) and finds they do not match.
6. DB2x becomes available on tracker T4 of system D, Sysplex 1. Tracker T4 warns the controller by sending an event.
7. The controller checks the operation in status READY, WAITING FOR SE against the scheduling environment name (DB2x) and the sysplex ID (1) and finds that they do match. The operation is resubmitted.

---

**Multi-sysplex configuration with JESplex matching sysplex**

[Figure 197 on page 503](#) shows a configuration where there are multiple JESplex within a single Sysplex.
According to IBM recommendations, the best configuration is when the JESplex (JES2 MAS or JES3 Complex) and the Sysplex match. Nevertheless, it is possible to define more JES complexes (JES2 or JES3) within a Sysplex. In this situation, a job
Multi-sysplex configuration with JESplex not matching sysplex

submitted in a JES complex cannot be routed by WLM to another JES complex, regardless of the output of the IWMSEQRY service.

**Note:** If you use virtual workstations with the WLM SCHENV interface, all the virtual destinations must be in the same sysplex and JESplex.

In the configuration shown in [Figure 198 on page 505](#) suppose there is a scheduling environment named DB2x. This scheduling environment is available on system D only. When the IWMSEQRY service is invoked from system A, the returned output indicates that DB2x is available in the Sysplex, overlooking the existence of the MAS 1 boundary. This is a problem: if a job requires DB2x, and it is submitted on system A, it will hang indefinitely in the JES input queue.
From a WLM point of view, this works as designed. To support this type of configuration, you must use the initial parameter statement OPCOPTS JESPLEX.
that informs trackers of the identity of the multiple JESplex. The **JESPLEX (List of system names)** parameter provides the system names within the JESplex where the tracker belongs.

In the example shown in Figure 198 on page 505, if TA, TB, TC, and TD are the trackers installed on MVS images A, B, C, and D respectively, the **OPCOPTS JESPLEX** parameter for each of the trackers must be the following:

<table>
<thead>
<tr>
<th>Tracker name</th>
<th>JESPLEX parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>TA</td>
<td>JESPLEX (A, B)</td>
</tr>
<tr>
<td>TB</td>
<td>JESPLEX (A, B)</td>
</tr>
<tr>
<td>TC</td>
<td>JESPLEX (C, D)</td>
</tr>
<tr>
<td>TD</td>
<td>JESPLEX (C, D)</td>
</tr>
</tbody>
</table>

When the job is submitted on system A, tracker TA first checks the response of the WLM query if DB2x is available with its own JESplex list (A and B). For this reason, tracker TA understands that DB2x is not available in its own JESplex and therefore does not submit the job.

However, this is a static definition, therefore if you change configuration you have to stop the tracker, update the parameter, and restart the tracker. This problem can be avoided by supporting multiple JESplex in the same Sysplex, without involving the trackers but only following a best practice approach to define and use scheduling environments.

For example, consider each MAS as a resource. If you have MAS 1 and MAS 2, define:

**MAS1** 
Required in **ON** state on MAS 1, in **OFF** state on MAS 2

**MAS2** 
Required in **OFF** state on MAS 1, in **ON** state on MAS 2

This corresponds to the following:

<table>
<thead>
<tr>
<th>Resource State</th>
<th>....System A</th>
<th>System B</th>
<th>System C</th>
<th>System D</th>
</tr>
</thead>
<tbody>
<tr>
<td>MAS 1</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>MAS 2</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

Thus, proceed as follows:
1. Associate MAS 1 with all the scheduling environments for jobs to be submitted on MAS 1
2. Associate MAS 2 with all the scheduling environments for jobs to be submitted on MAS 2

The following scenario, based on the configuration of Figure 197 on page 503 shows how to define a scheduling environment without using the **OPCOPTS JESPLEX** parameter. To define the scheduling environment, apply the following rules:

- Assign a letter to each JESplex in the configuration. For example, in Figure 197 on page 503, **SYSPLEX 1 JES MAS 1** can be identified with X, **SYSPLEX 1 JES MAS 2** can be identified with Y, and **JES3 GLOBAL** with Z.
- Use the same name for the same resource throughout all MVS images. For example, DB2ACT should indicate, when set to **ON**, that the DB2 application is active on a specific image.
Multi-sysplex configuration with JESplex not matching sysplex

- Define for each JESplex the MAS\(_x\) resource, where \(x\) is the letter associated to the JESplex. For example, resources MASX, MASY, and MASZ are to be defined as follows:

<table>
<thead>
<tr>
<th>Resource</th>
<th>SYSPLEX 1 System A</th>
<th>SYSPLEX 1 System B</th>
<th>SYSPLEX 1 System C</th>
<th>SYSPLEX 1 System D</th>
<th>SYSPLEX 2 System A</th>
<th>SYSPLEX 2 System B</th>
<th>SYSPLEX 2 System C</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASX</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>MASY</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>MASZ</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

- If a job requires that application DB2 be active, define 3 different scheduling environments so that they can be associated with the job depending on where the job is run (either MASX, MASY, or MASZ). For example:

**XSEDB2, for the job to run on MASX**
- Resource DB2ACT must be ON
- Resource MASX must be ON

**YSEDB2, for the job to run on MASY**
- Resource DB2ACT must be ON
- Resource MASY must be ON

**ZSEDB2, for the job to run on MASZ**
- Resource DB2ACT must be ON
- Resource MASZ must be ON

- Suppose that job XMYJOB is to be run on MASX with DB2 active. You must associate it with scheduling environment XSEDB2. Suppose you submit this job on MASX system A and the resource status is the following:

<table>
<thead>
<tr>
<th>Resource</th>
<th>SYSPLEX 1 System A</th>
<th>SYSPLEX 1 System B</th>
<th>SYSPLEX 1 System C</th>
<th>SYSPLEX 1 System D</th>
<th>SYSPLEX 2 System A</th>
<th>SYSPLEX 2 System B</th>
<th>SYSPLEX 2 System C</th>
</tr>
</thead>
<tbody>
<tr>
<td>MASX</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>MASY</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
</tr>
<tr>
<td>MASZ</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
<tr>
<td>DB2ACT</td>
<td>OFF</td>
<td>OFF</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
<td>ON</td>
</tr>
</tbody>
</table>

When the tracker invokes the IWMSEQRY macro, XSEDB2 shows to be unavailable (because the status of DB2ACT is OFF) without having had to define the OPCOPTS JESPLEX parameter.

Performance considerations

From a performance point of view, when you use the WLM scheduling environment integration, consider carefully the following items:
- Activating the ENF 57 and ENF 41 listener exit
- Implementing daily plan batch EQQDPX01 exit

For details, see “The ENF 57 and ENF 41 listener exits” on page 508 and “The DP BATCH EQQDPX01 exit” on page 508.
The ENF 57 and ENF 41 listener exits

The ENF 41 listener exits
When the WLM scheduling environment integration is active (the value of OPCOPTS SECHECK is not set to NO for all trackers), by default each tracker activates its own ENF 57 and ENF 41 listener exits. Therefore, for each image, a pair of these exits are at work.

The ENF 41 exit (EQQZNF41) sends an event every time a change in the WLM policy occurs, so that the controller is informed that a change in scheduling environment availability or definition have taken place. As a consequence, the controller resubmit all the operations in READY, WAITING FOR SE status that belong to the Sysplex where the WLM policy change occurred.

The ENF 57 exit (EQQZNF57) sends an event every time a scheduling environment becomes available, so that the controller is immediately notified that it can submit a pending job (in READY, WAITING FOR SE status) again. This means that, for example, if you have 32 MVS images, when scheduling environment SEDB2 becomes active, the Tivoli Workload Scheduler for z/OS controller is sent 32 events containing the same information. If you have many scheduling environments that change their status often, the performance of your system might be affected, depending also on the size of the workload on the controller.

To avoid unnecessary processing, you can suppress the activation of exits ENF 57 and ENF 41 on specific trackers, if this is compatible with your configuration. Considering that the tracker is in the same MVS image as the controller (either in the same or in a separate address space), in a Sysplex hosting both the controller and trackers you can use the SUPPRESSENF parameter to reduce the number of exits to the ones running on the controller. Proceed as follows:
1. Activate the exit on the controller: the value of OPCOPTS SECHECK must not be set to NO.
2. Activate the exit on all the trackers that are not located in the same Sysplex where the controller is: on each tracker the value of OPCOPTS SECHECK must not be NO.
3. Suppress the exit on all the trackers located in the same Sysplex where the controller is: on each tracker the value of OPCOPTS SECHECK must not be set to NO and SUPPRESSENF must be set to YES.

The DP BATCH EQQDPX01 exit
Exit EQQDPX01 can be used to set or change a scheduling environment name when creating the daily plan batch, according to user specifications (you can apply your own rules to set the values). Depending on how you implement it, this exit might affect the performance of the batch creation process, because it is called for each new operation (non-FTA) in the plan. For this reason, use this exit only when really necessary and try to avoid time consuming processing, like for example, unnecessary I/O (for instance, open the file at the first call and close it at the last call).
Chapter 24. Loop detection and analysis

Batch daily planning programs (extend, replan, and trial) prevent you from creating current plans that would cause a dependency loop condition. This is achieved by checking each single network of the plan, that is each set of operations that are linked to each other by some dependencies.

The scheduler checks for both normal and conditional dependencies. A step dependency determines a check at operation level.

If a loop condition is detected, batch daily planning programs stop creating the plan and start an analysis phase that produces:

- A message with summary information about the loop in the EQQMLOG and EQQLOOP data sets
- A list of messages in the EQQLOOP data set about:
  - The operations involved in the loop and their dependencies
  - The dependencies that the batch daily planning programs suggest to remove, one at a time, to solve the loop
  - The application descriptions sorted by the last update date and time in the application definition

Detecting a loop condition

A current plan is made of one or more networks. A network is made of a number of operations that are linked to one another by some dependencies. Every network must contain at least one of the following operations:

**First Operation (FOP)**
Operation with no predecessors. It is considered an entry point to the network.

**Last Operation (LOP)**
Operation with no successors. It is considered an exit point from the network.

To detect a loop condition, batch daily planning programs perform two kinds of check on the network.

The first check performed by batch daily planning programs is to ensure that the network has at least one first and one last operation. If either or both of them are missing, a loop condition is detected and reported as NO ENTRY AND/OR EXIT POINT loop in the EQQMLOG and EQQLOOP data sets. For details about this type of loop, see "NO ENTRY AND/OR EXIT POINT" loop on page 510.

The second check performed by batch daily planning programs is to ensure that all the operations in the network can be assigned an earliest start time (EAS). When processing the network, batch daily planning programs create a current plan where all the operations are assigned a planned start time, to approximately indicate when each operation can run. If, at the end of this process, one or more operations are left with no EAS assigned, a loop condition is detected. This type of loop is reported as SOME NODES COULD NOT BE CHECKED in the EQQMLOG and EQQLOOP data sets. For details, see "SOME NODES COULD NOT BE CHECKED" loop on page 510.
"NO ENTRY AND/OR EXIT POINT" loop

The first check performed by the batch daily planning programs is to ensure that a network has at least a first operation (FOP) and a last operation (LOP).

Suppose you have a plan with the following operations and dependencies:
- Application A, operation 001, predecessor of B, operation 002
- Application B, operation 002, predecessor of C, operation 003
- Application C, operation 003, predecessor of A, operation 001
- Application D, operation 004, predecessor of E, operation 005
- Application E, operation 005, LOP
- Application F, operation 006, predecessor of E, operation 005

Your plan would contain two networks, as shown in Figure 199.

![Figure 199. Example of a NO ENTRY AND/OR EXIT POINT loop condition](image)

In this example, network 2 has two first operations (FOP) and one last operation (LOP), which is allowed. Network 1, instead, has neither FOP nor LOP, which means that a loop condition will surely occur. When processing the network, batch daily planning programs detect the loop and issue the message EQQ3150E in the EQQMLOG and EQQLOOP data sets, specifying NO ENTRY AND/OR EXIT POINT as loop type. In the system log, the message EQQ3163E indicates that a loop condition has occurred.

"SOME NODES COULD NOT BE CHECKED" loop

A more complex mechanism used to detect a possible loop condition is to assign an earliest start time (EAS) to each operation in the network. An operation can be assigned an earliest start time only after an EAS has been assigned to all its predecessor operations. (First operations are assigned only an Input Arrival date and time.)

Suppose you add to your network 1 the following operations, as shown in Figure 200 on page 511:
- Application A, operation 007, predecessor of A, operation 001
- Application C, operation 008, successor of C, operation 003
In this example, network 1 has one FOP (A-007) and one LOP (C-008), which satisfy the first check operated by the batch daily plan programs. Nevertheless, a loop condition is detected when the batch daily plan programs try to assign an earliest start time to the operations in the network, because the only operation that can be assigned a time is A-007. In this case, message EQQ3150E is issued in the EQQMLOG and EQQLOOP data sets, specifying SOME NODES COULD NOT BE CHECKED as loop type and indicating the total number of unchecked operations, including the operations involved in the loop and their direct or indirect successors. In the system log, the message EQQ3163E indicates a loop condition has occurred. The loop analysis phase that starts immediately after a loop condition is detected, discovers that the loop involves only three operations (A-001, B-002, C-003) and batch daily planning programs list them as loop operations (see messages EQQ3151I, EQQ3152I, EQQ3153I, EQQ3154I, EQQ3155I in the EQQLOOP data set).

**Starting the loop analysis**

When a loop is detected, the current plan is not created or extended and a loop analysis process is started to help you resolve the loop. The loop analysis process is performed iteratively on a matrix representing the network, to progressively reduce the operations involved in the network until no operations are left and the loop is completely resolved. The loop analysis process is done on the matrix, without operating any real action in the network. Any bad dependencies removed and actions taken are reported in the EQQLOOP data set as a suggestion for you to solve the loop, according to your requirements.

After creating a matrix of the network, the loop analysis process performs iteratively a loop reduction activity to:

1. Remove any FOPs and LOPs in the network and leave only the operations involved in the loop. The last FOP removed, if any, is considered as the loop entry and its successor is marked as the closest operation to the loop entry. In the EQQLOOP data set, message EQQ3151I describes the loop reduction process performed and, at the first iteration, messages EQQ3152I, EQQ3153I, EQQ3154I, and EQQ3155I list all the operations involved in the loop. For details about these messages, see Messages and Codes.
Starting the loop analysis

2. Identify the dependency to be removed to solve the loop, according to the following criteria in the following order:

   A) IA Time Check
   A predecessor operation whose occurrence has an Input Arrival date (or time) later than the occurrence of the successor operation identifies the dependency to be removed.

   B) Closest To Loop Entry
   If no IA Time Check condition occurs, the dependency between the closest operation to the loop entry, if any, and one of its predecessors is identified as the one to be removed. Only when the network has at least one FOP, the reduction process identifies the closest operation to the loop entry because the loop entry is the last FOP removed. Otherwise, it is not set and the closest to loop entry condition does not occur.

   C) Minimal Net Distortion
   If none of the above conditions occurs, the minimal disturbing dependency is identified as the one to be removed. The disturbance is evaluated based on the throughput affecting the loop operations, such as the number of dependencies for an operation or the presence of external dependencies. The minimal disturbing dependency is identified as the one whose removal would have the least significant impact on the network.

For each dependency to remove, one message EQQ3156I is issued in EQQLOOP explaining the reason of the removal (IA Time Check, Closest To Loop Entry, or Minimal Net Distortion). Examine the EQQLOOP data set to learn the suggestions given to resolve the loop and examine your network to decide the actions to actually take. The information provided by EQQLOOP is a possible path to resolve the loop but you must decide which dependencies you want to remove.

The loop analysis process (reduction, operation analysis, dependency removal) is performed until all the loops in the matrix are solved, that is when no operations are left in the matrix. When all the loops are solved, messages EQQ3157I and EQQ3158I in EQQLOOP detail all the applications involved, listing them by the date and time of the last update made, in descending order. This information helps you identify the changes made in the application definitions that might have caused the loop to occur.

Example of a loop detection and analysis

As an example, consider the network shown in Figure 201 on page 513.
Example of a loop detection and analysis

In this network, which has both FOPs and LOPs, it is assumed that all the operations have the same IA date and time. The example shows a multiple dependency loop made up of four operations. The loop detection process detects a loop of type SOME NODES COULD NOT BE CHECKED, because five nodes could not be assigned an earliest start time (EAS) and therefore are identified as not checked. The loop analysis is performed to identify which dependencies can be the reasons of the loop and suggest how to solve it.

The reduction process, at its first iteration, identifies and lists all the loop operations, along with their dependencies. After an analysis of the loop operations, you are suggested to remove a dependency from the network (reason: CLOSEST TO LOOP ENTRY).

The reduction process starts for the second iteration. After the reduction, three operations are left in the network, meaning that there is another loop involving them. After an analysis of the loop operations, you are suggested to remove a dependency from the network (reason: CLOSEST TO LOOP ENTRY).

The reduction process starts the third iteration. At this stage, the reduction process leaves no operations in the network, meaning that the loop is resolved.

All the applications involved in the loops are listed ordered by the last update date and time, in descending order, in the EQQLOOP data set.
After the loop detection process is completed, the following messages are issued in the EQQMLOG data set:

```
EQQ3150E LOOP FOUND IN AN APPLICATION NETWORK:
EQQ3150I - LOOP TYPE = SOME NODES COULD NOT BE CHECKED
EQQ3150I - NETWORK ID = 000000000002
EQQ3150I - TOTAL OPERATIONS = 000000000009
EQQ3150I - TOTAL DEPENDENCIES = 000000000020
EQQ3150I - NO. OF FOPs = 000000000003
EQQ3150I - NO. OF LOPs = 000000000001
EQQ3150I - NO. OF UNCHECKED NODES = 000000000005
EQQ3150I - FIRST OCCURENCE = LOOPA 051027 1023
```

During the loop analysis process, the following messages are issued in the EQQLOOP data set:

```
EQQ3150E LOOP FOUND IN AN APPLICATION NETWORK:
EQQ3150I - LOOP TYPE = SOME NODES COULD NOT BE CHECKED
EQQ3150I - NETWORK ID = 000000000002
EQQ3150I - TOTAL OPERATIONS = 000000000009
EQQ3150I - TOTAL DEPENDENCIES = 000000000020
EQQ3150I - NO. OF FOPs = 000000000003
EQQ3150I - NO. OF LOPs = 000000000001
EQQ3150I - NO. OF UNCHECKED NODES = 000000000005
EQQ3150I - FIRST OCCURENCE = LOOPA 051027 1023
EQQ3151I LOOP REDUCTION ITERATION 00001 REDUCED LOOP TO 000000004 OPERATIONS
EQQ3152I ADID IDATE IAMT WSD OPNO JOBNM
EQQ3153I LOOP OPERATION: (LOOPC ) 051027 1023 CPU1 006 JOBX
EQQ3154I PREDECESSOR OF: (LOOPD ) 051027 1023 007
EQQ3155I SUCCESSOR OF: (LOOPA ) 051027 1023 002
EQQ3153I LOOP OPERATION: (LOOPD ) 051027 1023 CPU1 008 JOBY
EQQ3154I PREDECESSOR OF: (LOOPC ) 051027 1023 006
EQQ3155I SUCCESSOR OF: (LOOPA ) 051027 1023 007
EQQ3153I LOOP OPERATION: (LOOPD ) 051027 1023 CPU1 007 JOBZ
EQQ3154I PREDECESSOR OF: (LOOPC ) 051027 1023 008
EQQ3155I SUCCESSOR OF: (LOOPA ) 051027 1023 002
EQQ3153I LOOP OPERATION: (LOOPD ) 051027 1023 CPU1 002 JOBZ
EQQ3154I PREDECESSOR OF: (LOOPC ) 051027 1023 007
EQQ3155I SUCCESSOR OF: (LOOPA ) 051027 1023 008
EQQ3156I REMOVED DEPENDENCY: (LOOPD CPU1 0008 JOBY 051027 1023)
EQQ3156I PREDECESSOR OF: (LOOPA CPU1 0002 JOBZ 051027 1023)
EQQ3156I REASON FOR REMOVAL: CLOSEST TO LOOP ENTRY
EQQ3155I LOOP REDUCTION ITERATION 00002 REDUCED LOOP TO 000000003 OPERATIONS
EQQ3156I REMOVED DEPENDENCY: (LOOPC CPU1 0006 JOBX 051027 1023)
EQQ3156I PREDECESSOR OF: (LOOPD CPU1 0007 JOBZ 051027 1023)
EQQ3156I REASON FOR REMOVAL: CLOSEST TO LOOP ENTRY
EQQ3155I LOOP REDUCTION ITERATION 00003 REDUCED LOOP TO 000000000 OPERATIONS
EQQ3157I ADID VALIDTO LASTUPD USER
EQQ3158I LOOP ADID: LOOPD 991231 051104 0107 USER1
EQQ3158I LOOP ADID: LOOPA 991231 051027 0644 USER2
EQQ3158I LOOP ADID: LOOPC 991231 051027 0530 USER3
```

When a batch daily planning ends in error or takes too long, search for message EQQ3150E in the EQQMLOG and EQQLOOP data sets to check whether a loop condition occurred. If a loop condition occurred, detailed information is stored in the EQQLOOP data set.
Example of a loop detection and analysis

According to the resolution actions suggested, check if the removed dependencies are the actual dependencies you can remove based on your application definitions. Check messages EQQ3157I and EQQ3158I to see if some application definitions were recently changed, because they might have caused the loop.

Suggestions to speed up the loop resolution

When you make some changes to the operation definitions, to identify possible loops, run daily planning trials. In this way, there will be smaller networks to analyze and fewer messages to check, if a loop condition is detected.

When a loop is detected, ensure that you promptly modify the application definitions to prevent the network from generating the same loop condition again.

The messages generated by the loop analysis process in the EQQLOOP data set are not intended as the only possible way to resolve the loop. They are only a suggestion path to solve the loop; you can adopt it or not, according to your application definitions requirements. For example, consider a network with three operations involved in a loop, all of them having the same Input Arrival (IA) date and time. Suppose the following dependencies link the operations:

- Operation A is predecessor of operation B
- Operation B is predecessor of operation C
- Operation C is predecessor of operation A

The batch daily planning detects a loop condition (A, B, C) and suggests a path to solve it by removing one of the dependencies. The removed dependency is not necessarily the bad dependency, because the IA date and time is the same for all the operations. In this case, no IA TIME CHECK condition occurs. If the network has one or more FOPs, the matrix reduction sets the closest operation to the loop entry and the removed dependency follows this criteria. If the network has no FOPs, the CLOSEST TO LOOP ENTRY condition does not occur; the MINIMAL NET DISTORTION will determine the dependency to be removed.

When applicable, avoid using the same IA date and time for the operations in the network. Setting the IA date and time according to the dependencies makes it easier for the loop analysis process to identify bad dependencies.
Suggestions to speed up the loop resolution
Part 2. Controlling and monitoring

Chapter 25. Monitoring the workload

Using the Ready List panel 519
Selecting a ready list layout 519
Creating your own ready list layout 520
Ready list layout user exit 522
Invoking the user exit from the panels 522
Defining and setting up the user exit 522
Communicating with the user interface routine 523
Using the ready list 525
Setting the status of an operation 526
Letting the scheduler assign the next status 526
Setting the status explicitly 526
Resetting an operation to its previous state 527
Interrupting an operation 527
Reporting an operation as ended-in-error 527
Viewing operator instructions 527
Preparing jobs at a setup workstation 527
Preparing jobs without unresolved promptable variables 528
Preparing jobs with unresolved promptable variables 529
Other ways of editing a job 530
Delaying an operation, and releasing it 531
Removing an operation from the current plan and restoring it 532
Running an operation immediately with EXECUTE 533
Diagnosing delays 533
Resetting bind information for a shadow job 534
The QCP panel 535
Querying application occurrences 535
Querying operation information 537
Checking the status of a workstation 539
Checking the status of the current plan 540
CRITICAL JOBS 541
A business scenario 543
Roles 543
Setting up the environment 543
Running the scenario 544

Chapter 26. Updating the current plan 547

Using fast paths 549
Accessing the Modifying Current Plan panel 549
Specifying selection criteria 549
Running work on request 549
Adding occurrences to the current plan 550
Selecting occurrences 550
Adding occurrences 551
Specifying an error code 553
Including dependencies defined in the database 553
Changing and adding dependencies 555
Changing and adding conditional dependencies 558
Grouping occurrences 559
Adding an application group to the current plan 559
Excluding some applications 560
Specifying dependency resolution 560
Restarting an occurrence from the beginning 562
Restarting an occurrence from a specific operation 563
Completing an application occurrence 566
Deleting an application occurrence 566
Modifying an application occurrence 567
Changing external dependencies to an occurrence 567
Changing dependencies to an operation 568
Changing the details of an operation 568
Modifying extended information 569
Adding and deleting operations 569
Listing and browsing operations 570
Creating a list of operations 570
Browsing operation information 572
Modifying operations 577
Restarting operations in the history database 578
Updading the history database 578
Processing history operations 579
Selecting a history operation 580
Informing the scheduler of unplanned changes in resources 581
Keeping plans up-to-date 582
Fault-tolerant workstations and replanning 582
Changing workstation availability 584
Active and inactive computer workstations 584
Redirecting work to alternate workstations 586
Changing the global status for a virtual workstation 588
Changing availability for a virtual destination 589
Browsing system information 591
Killing operations on standard agents 591

Chapter 27. Handling operations that end in error 593

Displaying the ended-in-error list for action 594
Selecting an ended-in-error list layout 594
Creating your own ended-in-error list layout 594
Getting rerun or recovery instructions 596
Complting an ended-in-error operation 596
Modifying a job that has failed 597
Restarting ended-in-error operations 597
Restarting ended-in-error operations managing cleanup action 597
Taking action at the occurrence level 599
Handling operations with the OSEQ error code 599
Restarting an operation from a certain step 600
Using cleanup options 601
Specifying automatic restart for operations that fail 602

Chapter 28. Monitoring special resources 605

Understanding special resources 605
Example using data sets 607
Chapter 25. Monitoring the workload

All operations in the current plan are associated with a workstation. You can use Tivoli Workload Scheduler for z/OS panels to display the operations. For an overall view, you use the QUERY CURRENT PLAN panel (described later in this chapter in "The QCP panel" on page 535). However, if you want to find out which operations are due to start or are already started at a workstation, use the workstation ready list.

Using the Ready List panel

The ready list contains operations that have no outstanding predecessors, operations defined to the workstation that are waiting for a particular time or resource, operations that have already started, and operations that have ended in error. The status codes of operations in the ready list can be A, I, R, *, S, or E. The ready list displayed can also include one operation in C status, the last operation that you manually set to complete. This operation is maintained on the ready list to give you an opportunity to reset the operation if you change your mind.

To display the ready list for a workstation, select option 4 from the main menu, which will then display the COMMUNICATING WITH WORKSTATIONS menu (Figure 202). The ready list functions are available by selecting option 1 on this panel.

You can customize the layout of the workstation ready list to suit your needs, either by selecting one of the supplied layouts or by building your own layout.

Selecting a ready list layout

When you select the ready list for a workstation for the first time, you must specify the layout to use for the list. Approximately 90 different fields can be displayed in a ready list (see Appendix F, "Fields displayed in ready and error lists," on page 749), but not all of these can be displayed at once. Information on the fields that are to be displayed, and the order in which they should be displayed, is kept in a named ready list layout. There might be many different layouts in your installation. Each supplied default layout is designed for a specific workstation type.
Selecting a ready list layout

If you have not specified a layout for the workstation that you have selected, Tivoli Workload Scheduler for z/OS displays a list of layouts, so you can then select a layout from this list. The scheduler stores the layout you choose for each workstation in your ISPF profile. When you request the ready list for a workstation, Tivoli Workload Scheduler for z/OS retrieves the name of the layout ID you prefer from your ISPF profile. If you want to change the layout of the ready list, enter the name of another layout in the corresponding field in the ready list. Tivoli Workload Scheduler for z/OS displays the ready list again using the new layout you specified. You can get a list of all the ready list layouts by typing an asterisk (*) in the LAYOUT ID field on the SPECIFYING READY LIST CRITERIA panel and then pressing Enter.

Creating your own ready list layout

Ready list layouts are kept in two ISPF tables: one for your own use, and one that contains your installation-defined layouts. Your own layout overrides the installation layout.

To change a ready list layout, select option 9, the DEFINE RL option on the COMMUNICATING WITH WORK STATION panel. The scheduler then displays a list of both your own and any installation-defined layouts (Figure 204 on page 521). From this list, you can create new layouts or select a layout to delete, copy, modify, or browse. Any modified installation-defined layout is stored as a private copy of the layout in your ISPF profile. You can delete your own layout, but you cannot delete an installation layout.

To make a set of private layouts available for your colleagues:

1. Create a complete set of layouts. If you want to include a supplied (or old) layout, edit and save it so that it becomes part of your private library.
2. Save (by renaming) the old EQQRLDEF member in the common table library.
Creating your own ready list layout

3. Copy the EQQRLOUT member from your ISPF profile library to the common table library, renaming it to EQQRLDEF.

<table>
<thead>
<tr>
<th>Row</th>
<th>Layout</th>
<th>Description</th>
<th>Owner</th>
<th>Last update</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>C1</td>
<td>Layout 1 for CPU work station</td>
<td>LEIFTO2</td>
<td>96/09/15 08.21</td>
</tr>
<tr>
<td>2</td>
<td>C2</td>
<td>Layout 2 for CPU - times</td>
<td>LEIFTO2</td>
<td>96/09/13 13.12</td>
</tr>
<tr>
<td>3</td>
<td>C3</td>
<td>Layout 3 for CPU - options</td>
<td>LEIFTO2</td>
<td>96/09/13 13.19</td>
</tr>
<tr>
<td>4</td>
<td>WTO</td>
<td>Ready list layout for WTO</td>
<td>XRAYNER</td>
<td>97/02/03 12.28</td>
</tr>
<tr>
<td>5</td>
<td>LAYOUT1</td>
<td>My layout</td>
<td>XRAYNER</td>
<td>97/04/20 13.18</td>
</tr>
</tbody>
</table>

Figure 204. EQQRLL - Ready list layouts

When you modify or create a layout, Tivoli Workload Scheduler for z/OS presents a list of the available ready-list fields (see Appendix F, “Fields displayed in ready and error lists,” on page 749 for a complete list). From this list, shown in Figure 205 on page 522, you select the fields to be contained in the ready list layout, and the column order. You can also specify whether the field should be highlighted when Tivoli Workload Scheduler for z/OS displays it. All the items previously selected for display are placed at the top of the list in column order (1).

Note that all fields are not applicable for all kinds of workstations or operations. For example, the actual duration field, ACT DUR, will not be applicable for a general automatic workstation as its operations disappear from the Ready List when complete.

On the CREATING A READY LIST LAYOUT panel, you can also specify a text string used by the ISPF select service to invoke a user exit.
Ready list layout user exit

The user exit is passed the status of an operation in the ready list output. It can set the next logical status plus the error code and duration of the selected operation.

Invoking the user exit from the panels

The user exit is invoked when you enter the row command R (reset) or N (set next status) in a row of the READY LIST panel, EQQRLRLM. It is invoked for only those workstations that have been defined as follows:

- **Type**: General
- **Reporting attribute**: Manual start and completion
- **Job setup**: N

The actual status of the operation must be one of the following:
- Waiting for arrival
- Ready
- Interrupted
- Ready non-reporting

Although defined, the user exit is not invoked for other cases.

Defining and setting up the user exit

You can specify a user exit routine in the USER EXIT field of the CREATING A READY LIST LAYOUT (EQQRLYCL) panel. This specification can be in the form of one of the following command strings, depending on the type of user exit routine:

- `CMD(clistname)`
- `PANEL(panelname)`
- `PGM(modulename)`

Store CLISTs, panels, and programs that comprise the user exit routine in a selected data set that is concatenated in the SYSPROC, ISPLLIB, ISPMLIB, and STEPLIB statements of the procedure or CLIST used for Tivoli Workload Scheduler.
Defining and setting up the user exit

for z/OS setup. (For more information about setting up the controller ISPF environment, refer to Planning and Installation.)

Communicating with the user interface routine

Communication between Tivoli Workload Scheduler for z/OS panel and user exit routine is through the ISPF variable defined in shared pool. The panel passes the following variables to the user exit routine:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>UROSTAT</td>
<td>Current operation status</td>
</tr>
<tr>
<td>URNSTAT</td>
<td>New operation status</td>
</tr>
<tr>
<td>UROPERR</td>
<td>New operation error code</td>
</tr>
<tr>
<td>UROPDUR</td>
<td>New operation duration</td>
</tr>
<tr>
<td>N3P_COM</td>
<td>ISPF variables related to the operation</td>
</tr>
</tbody>
</table>

The UROSTAT variable contains the status value of the current operation. New operation parameters are passed initialized to blank.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPGROUP</td>
<td>Authority group</td>
</tr>
<tr>
<td>OPADI</td>
<td>Application ID</td>
</tr>
<tr>
<td>OPIAD</td>
<td>Application input arrival date after MCP</td>
</tr>
<tr>
<td>OPIAT</td>
<td>Application input arrival time after MCP</td>
</tr>
<tr>
<td>OPTXT</td>
<td>Textual description</td>
</tr>
<tr>
<td>OPJBN</td>
<td>OS job name or blank</td>
</tr>
<tr>
<td>OPWSN</td>
<td>Workstation name</td>
</tr>
<tr>
<td>OPNUM</td>
<td>Operation number</td>
</tr>
<tr>
<td>OPJCL</td>
<td>JOB-, SYSOUT-class, or blank</td>
</tr>
<tr>
<td>OPFRM</td>
<td>From number or blank</td>
</tr>
<tr>
<td>OPPSD</td>
<td>Planned start date</td>
</tr>
<tr>
<td>OPPST</td>
<td>Planned start time</td>
</tr>
<tr>
<td>OPPED</td>
<td>Planned end date</td>
</tr>
<tr>
<td>OPPET</td>
<td>Planned end time</td>
</tr>
<tr>
<td>OPOID</td>
<td>Operation input arrival date</td>
</tr>
<tr>
<td>OPOIT</td>
<td>Operation input arrival time</td>
</tr>
<tr>
<td>OPODD</td>
<td>Operation deadline date</td>
</tr>
<tr>
<td>OPODT</td>
<td>Operation deadline time</td>
</tr>
<tr>
<td>OPLOD</td>
<td>Latest out for operation date</td>
</tr>
<tr>
<td>OPLOT</td>
<td>Latest out for operation time</td>
</tr>
<tr>
<td>OPASD</td>
<td>Actual start date</td>
</tr>
<tr>
<td>OPAST</td>
<td>Actual start time</td>
</tr>
<tr>
<td>OPAAD</td>
<td>Actual arrival date</td>
</tr>
<tr>
<td>OPAAT</td>
<td>Actual arrival time</td>
</tr>
<tr>
<td>OPISD</td>
<td>Intermediate start date if interrupted</td>
</tr>
<tr>
<td>OPIST</td>
<td>Intermediate start time if interrupted</td>
</tr>
<tr>
<td>OPAED</td>
<td>Actual end date</td>
</tr>
</tbody>
</table>
### Communicating with the user interface routine

<table>
<thead>
<tr>
<th>N3P_COM Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPAET</td>
<td>Actual end time</td>
</tr>
<tr>
<td>OPEDU</td>
<td>Estimated duration, hh.mm</td>
</tr>
<tr>
<td>OPADU</td>
<td>Actual duration, hhhh.mm</td>
</tr>
<tr>
<td>OP£PS</td>
<td>Parallel servers required</td>
</tr>
<tr>
<td>OP£R1</td>
<td>WS resources required (r1)</td>
</tr>
<tr>
<td>OP£R2</td>
<td>WS resources required (r2)</td>
</tr>
<tr>
<td>OPCST</td>
<td>Current status</td>
</tr>
<tr>
<td>OPERR</td>
<td>Error code</td>
</tr>
<tr>
<td>OPAEC</td>
<td>Automatic error completion (Y or N)</td>
</tr>
<tr>
<td>OPPRI</td>
<td>Priority</td>
</tr>
<tr>
<td>OPXST</td>
<td>Extended status</td>
</tr>
<tr>
<td>OP£SU</td>
<td>Number of successors</td>
</tr>
<tr>
<td>OP£PR</td>
<td>Number of predecessors</td>
</tr>
<tr>
<td>OP£SR</td>
<td>Number of special resources</td>
</tr>
<tr>
<td>OPAJR</td>
<td>Automatic hold/release (Y or N)</td>
</tr>
<tr>
<td>OPASUOP</td>
<td>Automatic submit (Y or N)</td>
</tr>
<tr>
<td>OPTJT</td>
<td>Time job (Y or N)</td>
</tr>
<tr>
<td>OPRESTA</td>
<td>Restartable operation</td>
</tr>
<tr>
<td>OPRERUT</td>
<td>Reroutable operation</td>
</tr>
<tr>
<td>OPWRER</td>
<td>Operation was rerouted</td>
</tr>
<tr>
<td>OPDAWS</td>
<td>Operation alternate WS</td>
</tr>
<tr>
<td>OPDWTO</td>
<td>Operation deadline WTO</td>
</tr>
<tr>
<td>OPMXLVL</td>
<td>Maximum nesting level</td>
</tr>
<tr>
<td>OPUPDA</td>
<td>Operation userdata field</td>
</tr>
<tr>
<td>OPMHLD</td>
<td>Operation manually held field</td>
</tr>
<tr>
<td>OPNOP</td>
<td>Operation nop field</td>
</tr>
<tr>
<td>OPEXEC</td>
<td>Operation execute field</td>
</tr>
<tr>
<td>OPMCPUP</td>
<td>Last MCP update</td>
</tr>
<tr>
<td>OPJES</td>
<td>Job ID</td>
</tr>
<tr>
<td>OPCMJST</td>
<td>Job log status</td>
</tr>
<tr>
<td>OPEXDEST</td>
<td>Execution destination</td>
</tr>
<tr>
<td>OPEDUS</td>
<td>Estimated duration hh:mm:ss</td>
</tr>
<tr>
<td>OPADUS</td>
<td>Actual duration hh:mm:ss</td>
</tr>
<tr>
<td>OPMON</td>
<td>External monitor (Y or N)</td>
</tr>
</tbody>
</table>
Using the ready list

The following are some of the tasks you can perform using the ready list:

- “Setting the status of an operation” on page 526.
- “Resetting an operation to its previous state” on page 527.
- “Interrupting an operation” on page 527.
- “Reporting an operation as ended-in-error” on page 527.
- “Viewing operator instructions” on page 527.
- “Preparing jobs at a setup workstation” on page 527.
- “Delaying an operation, and releasing it” on page 531.
- “Removing an operation from the current plan and restoring it” on page 532.
- “Running an operation immediately with EXECUTE” on page 533.
- “Diagnosing delays” on page 533.
- “Rerunning operations in the history database” on page 578.
- “Resetting bind information for a shadow job” on page 534.

Select option 1 on the COMMUNICATING WITH WORKSTATIONS panel (Figure 202 on page 519) to see the ready list. Figure 206 shows a typical ready list for a computer workstation.

Figure 206. EQQRLRLM - Ready list

Enter the SORT command to display the list-field headings and their meanings. The ready list by default sorts the operations by job name within status. However, you can change the sort fields presented on the ready list. See Appendix E, “Status, error, and reason codes,” on page 743 for a complete list of status codes and extended status codes. If you sort operations by the actual start times, the jobs that have been running the longest will appear at the top of your display.

You cannot override sort criteria that you have specified on a filter panel with the SORT command.
Setting the status of an operation

Setting the status of an operation
When you work with a ready list, you are normally changing the status of an operation in the cases where the workstation does not report the status change automatically. To do this, you can let Tivoli Workload Scheduler for z/OS assign the next logical status, or you can set the status explicitly.

Letting the scheduler assign the next status
Letting Tivoli Workload Scheduler for z/OS assign the next status is the easier way: enter the row command $N$ beside the operation whose status is to be changed. The scheduler then changes the status to the next logical status for that type of workstation:

- For automatic reporting workstations, the order is: A, R, S, C.
- For manual start and completion workstations, the order is: A, R, S, C.
- For completion only workstations, the order is: A, R, C.
- For non-reporting workstations, the order is: A, C.

Sometimes you see operations on a ready list in *(asterisk) status. This means the operation is ready, but one or more predecessors are defined on a non-reporting workstation.

Note the difference between A and R status. Only operations with no predecessors have an A status. Operations that have all predecessors completed progress to R status directly from W status.

Setting the status explicitly
To set the status explicitly, enter the row command $N-x$, where $x$ is the status you want set for the operation. You might want to do this to immediately set the status of an operation to complete, or to report an operation as interrupted or as ended-in-error. Sometimes Tivoli Workload Scheduler for z/OS requests more information when you set the status; for example, it might request the duration of the operation, or the error code. When Tivoli Workload Scheduler for z/OS requests more information, it displays a panel where you should enter the information.

Notes:
1. Setting the operation status to S (started) for operations on computer workstations does not cause the job or started task that is associated with the operation to be submitted. First establish why the operation is not started. When you have done this, you can remove the cause of the delay so that Tivoli Workload Scheduler for z/OS can run the work normally. See “Diagnosing delays” on page 533.
2. Setting a WTO operation defined to a general workstation to started does not cause a WTO message to be issued.
3. Setting the status of a started operation to E (error), or to any other status, does not cancel the job.
4. Setting the status of an operation to C (completed) allows successor operations to start (other dependencies permitting), even though the job might still be executing.
5. You cannot set the status to S or E for fault-tolerant workstations.

See Chapter 26, “Updating the current plan,” on page 547.
Resetting an operation to its previous state

**Resetting an operation to its previous state**
The scheduler retains on the ready list the most recent operation that you set to the
previous logical state by using the R row command or the OPSTAT TSO command.
You can reset the operation to its previous state only if the status of successor
operations has not changed.

**Interrupting an operation**
The status of an active operation can be set to I (interrupted) automatically if it is
active on a workstation that has the splittable attribute. A computer workstation
never has this attribute. You might want to interrupt job setup, for example, if you
are half-way through editing a job when you have to break for a meeting.

When you interrupt the operation, the resources that the operation uses are freed.
While the operation is in interrupted status, its duration is not incremented. There
is no limit to the number of times that an operation can be started and interrupted.

If the status of an operation is not S (started), use the MCP panel to set the status
to interrupted. For details, see the MODIFYING OPERATIONS IN THE CURRENT
PLAN panel shown in Figure 251 on page 577.

**Reporting an operation as ended-in-error**
The scheduler automatically reports jobs and started tasks that fail with E
(ended-in-error) status. You can manually report operations as ended-in-error on
any type of workstation. In these circumstances, you must decide what the error
status signifies. The advantage of reporting the operation as ended-in-error is that
a problem is highlighted. The scheduler puts the operation in the ended-in-error
list, which you should review regularly so that error-recovery action can be taken.
Refer to Chapter 18, “Automatic recovery of jobs and started tasks,” on page 347
for information on automatic recovery in Tivoli Workload Scheduler for z/OS.

To manually report an operation as ended-in-error, enter the N-E row command
beside the operation. The scheduler then prompts you for an error code. It is good
practice to reserve some error codes for specific error situations. Note that this is
not valid for fault-tolerant workstations.

Setting the status of a started operation to E (error), or to any other status, does
not cancel the job.

**Viewing operator instructions**
Some operations might require specific instructions on how they are to be handled.
These instructions are known as *operator instructions*. You can tell whether
instructions exist for an operation by looking at the code in the Oi ready-list field:

- **N** There are no operator instructions.
- **Y** There are operator instructions.
- **+** Some operator instructions have been changed recently. The default is 30
days, but the definition of “recent” depends on the setting of the
NEWOILIMIT keyword of the JTOPTS parameter at your installation. See
your administrator for Tivoli Workload Scheduler for z/OS.

To browse operator instructions, enter the O command beside the operation. The
scheduler then displays the instructions using the ISPF/PDF browse function.

**Preparing jobs at a setup workstation**
A setup workstation is a general workstation used for preparing jobs. In Tivoli
Workload Scheduler for z/OS, the operation of preparing jobs is immediately
Preparing jobs at a setup workstation

followed by the operation that runs the job on the computer workstation. If it is not waiting for other conditions to be met, the job can be started as soon as job setup is complete.

When you set the next logical status for an operation at a job setup workstation (with the N row command), Tivoli Workload Scheduler for z/OS reports the status of the setup operation as S (started). The action that Tivoli Workload Scheduler for z/OS takes depends on whether it finds promptable variables in the job that have not been resolved.

Note: When you edit the job using the READY LIST or MCP panels, you edit the latest job from the JS file, which is where Tivoli Workload Scheduler for z/OS places modified jobs. The original job is always left unaltered in the partitioned data set allocated to the ddbname EQQJBLIB (JBLIB). To force Tivoli Workload Scheduler for z/OS to read a fresh copy of the job from JBLIB, delete all the lines and end the edit.

Preparing jobs without unresolved promptable variables

Type the N row command beside the job to edit it and start the setup operation. The scheduler automatically resolves any non-promptable job variables that are defined to be resolved at setup time and then invokes the ISPF editor.

You can edit the job as required in the EDITING JCL FOR AN OPERATION panel.

Figure 207.

Preparing jobs at a setup workstation

The job in Figure 207 has several errors:

Figure 207. EQQRJCLE - Editing JCL for an operation

The job in Figure 207 has several errors:
Preparing jobs without unresolved promptable variables

1. The error message refers to the line following . CDATE is a dynamic-format variable whose format must be specified in a SETFORM directive before being used.

2. When variables are not found, other variables, such as those on the job card, are not substituted either, even though Tivoli Workload Scheduler for z/OS has found them. You do not see any substituted variables unless Tivoli Workload Scheduler for z/OS has found them all.

3. The variable on is wrong. You cannot have variables in Tivoli Workload Scheduler for z/OS RECOVER directive. The scheduler has not pointed out this error because it does not scan recovery statements unless the job fails.

When you have finished editing the job, exit by entering one of the following commands:

- END, which causes Tivoli Workload Scheduler for z/OS to save the modified or not modified job in the JCL repository, set the status of the setup operation to C (complete), and start the successor operation (the job itself) unless there are other dependencies.
- CANCEL, which causes Tivoli Workload Scheduler for z/OS to exit without saving the job in the JCL repository. The status of the setup operation is still R (ready).
- TSAVE, which saves your edited job in the JCL repository for later editing and changes the setup operation to status I (interrupted). Use this command if you want to keep your changes and continue editing later.

Preparing jobs with unresolved promptable variables

The scheduler:

1. Scans a job for variables that are not specified as substitute-at-submit variables.

2. Displays all promptable variables on the LIST OF JCL PREPARATION VARIABLES TO BE SET panel so that you can enter values.

3. Ends when you enter one of these commands:
   - CANCEL. The scheduler skips variable substitution and does not save the job.
   - END, which causes Tivoli Workload Scheduler for z/OS to:
     a. Validate all promptable variable values
     b. Substitute promptable variables in the job
     c. Substitute all non-promptable variables

Figure 208. EQQLVAL - List of JCL preparation variables to be set
Preparing jobs with unresolved promptable variables

d. Store the job in the JCL repository.

If errors are found, a message is displayed on the variable list panel.

If you specify that you want to edit the job on the LIST OF JCL PREPARATION VARIABLES TO BE SET panel, Tivoli Workload Scheduler for z/OS invokes the ISPF edit function when you enter the END command and displays the EDITING JCL FOR AN OPERATION panel.

After you have modified the job, the END command saves the job in the JCL repository, and you exit from ISPF edit. If you enter the CANCEL command, you do not cancel the variable values set because they are already stored in the JCL repository.

If the setup operation is for several processor operations (all with the same job name), Tivoli Workload Scheduler for z/OS displays a list of the operations that you can choose from.

Note: The standard ISPF edit SAVE command has no effect when you are editing a job within Tivoli Workload Scheduler for z/OS.

Other ways of editing a job

There are several ways to do job setup using the panels:

- Select the JOB SETUP option from the COMMUNICATING WITH WORK STATIONS panel. You see a selection list that presents a list of operations eligible for setup. Enter J next to one of the operations. This puts you into ISPF edit.

---

```
EQQRJCLE --------------- EDITING JCL FOR AN OPERATION -------------------------

Command ====>

Edit JCL below and press END to complete, CANCEL to reject and reset, or TSAVE to save changes and interrupt the operation.

Application : SCRIPT7 An AIX/6000 transfer script
Operation    : CPU7 015
Jobname      : SCRIPT7 JCL last updated by: XRAYNER

# Create a data file
echo 'greetings ' > $jclfile
echo 'from AIX/6000' >> $jclfile

# Create a file to send
echo 'open SYSTEM' > $ftpfile
echo 'site file=jes' >> $ftpfile
echo 'site lrecl=80' >> $ftpfile
echo "put $jclfile" >> $ftpfile

# Invoke FTP to send the file
ftp < $ftpfile
rm $jclfile

# Invoke command and save return code
$command
src=$?
if [ "$src" -eq 0 ]
  then status=C
else status=E
fi
```

Figure 209. EQQRJCLE - Editing JCL for an operation

After you have modified the job, the END command saves the job in the JCL repository, and you exit from ISPF edit. If you enter the CANCEL command, you do not cancel the variable values set because they are already stored in the JCL repository.

If the setup operation is for several processor operations (all with the same job name), Tivoli Workload Scheduler for z/OS displays a list of the operations that you can choose from.

Note: The standard ISPF edit SAVE command has no effect when you are editing a job within Tivoli Workload Scheduler for z/OS.
**Other ways of editing a job**

edit on the job for that operation. Change the job and press PF3. The changed job is stored in the JCL repository, and the operation is automatically set to C status (complete).

**Note:** You can change any job from the JOB SETUP option, even if the operation does not have a job setup operation associated with it. The changed job is stored in the JCL repository as usual. Because there is no setup operation associated with this activity that can be marked as complete, this editing procedure has no effect on the schedule in Tivoli Workload Scheduler for z/OS.

- Use the MCP panel to edit the job for any operation in the current plan. See “Modifying operations” on page 577.
- Set up the job for an individual occurrence in the LONG TERM PLAN panel. The edited job for the future occurrence is stored in the JCL repository.

**Delaying an operation, and releasing it**

Sometimes you must delay the start of an operation because of a situation beyond your control. For example, the application programmer is manually editing some production files to incorporate an urgent program fix. In such situations, when the operations concerned are already in the current plan and waiting only for a certain time or for predecessors to be complete, you must do something to stop the operation from being started when the scheduling criteria are satisfied. You can:

- Manually HOLD the operation by using the MCP panel, or the ready list if the operation predecessors are already complete.
- Modify the job to include a deliberate error; for example, a comma at the end of the job card for a z/OS job. The job is submitted when all the scheduling criteria are met but does not actually execute until the syntax error is corrected.
- Modify the occurrence to include an extra operation on a general workstation, which becomes a predecessor for the operation you need to delay.

The manual HOLD command, MH, can be issued for an operation on a computer workstation with automatic reporting or on any workstation with no reporting, if the current status of the operation is A, R, *, W, C, or E. The scheduler does not start any operation that has been manually placed in HOLD by a panel user, even though the status of the operation will change when the operation start criteria make the operation eligible to be started. If the operation that was manually placed in HOLD is a time-dependent, suppress-if-late operation, its submission is delayed until the suppress time is reached. When the suppress time has expired, the action set by the SUPPRESSACTION keyword is taken, even if the operation is still manually held. All operations that have been manually placed on HOLD are identified by the extended status code H.

When you no longer want the operation held, you can issue the RELEASE command, MR, and the operation extended status code changes to reflect the current situation. If all start criteria for this operation are met, the operation can start immediately.

If you need to HOLD or RELEASE an operation that is not on the ready list, you can use the MCP panel. See "Modifying operations” on page 577 for more details. These commands can also be entered from the ended-in-error list.

You cannot invoke the next logical status row command against an operation that has been manually held, but you can set a specific status. This does not alter the HOLD indication.
Delaying an operation, and releasing it

The MH command gives the operation the HOLD property. To remove this property from the operation, use the MR command. Neither the MH command nor the MR command changes the status of an operation directly. A HOLD operation is not automatically scheduled for processing.

Removing an operation from the current plan and restoring it

If you need to remove an operation (NOP) that is already in the current plan, enter the NP command beside the row in the ready list. NP can be issued for any operation on the ready list that has status A, R, *, W, or (for computer workstations with automatic reporting only) C.

The scheduler processes status changes for NOP operations until they reach a status of A, R, or * (ready). The scheduler ignores time dependency, use of special resources, and other constraints. When a NOP operation reaches the ready status A, R, or *, Tivoli Workload Scheduler for z/OS immediately sets the operation to status C; the operation is not submitted, and successor operations are eligible to start. NOP operations are identified by the N extended status code.

Attention: Make sure that successor operations in a dependency chain are removed (NOP) before their predecessor operation. This prevents the successor operations from starting when you remove (NOP) their predecessor.

If you want to restore the operation, use the UN command. The UN command does not affect the status of the operation. It merely removes its NOP characteristics. The operation can be treated as any normal operation on the ready list.

The NP command gives to the operation the property NOP. This property can be removed from the operation by the UN command only. Neither the NP command nor the UN command changes the status of an operation directly. A NOP operation is automatically changed to completed status when it reaches ready status, regardless of whether other constraints are met.

Note: The EXECUTE command starts an operation even if it is NOP. See “Running an operation immediately with EXECUTE” on page 533.

Use the MCP panel for an operation that is not on the ready list. See “Modifying operations” on page 577 for more details.

The NP and UN commands can be issued also on operations defined on fault-tolerant workstations, but the result of the commands is different depending on whether the operation has the centralized script option or not. When the centralized script option is used, the NP and UN commands have the same effect they have on operations defined on computer automatic workstations. When the centralized option is not used, the commands result in setting the CANCEL PENDING option in the Symphony file for the related job. Then the job is cancelled locally when the usual conditions are satisfied and consequently the related operation is completed in the current plan. It is also possible for the operations that are handled in the distributed environment to respond differently than usual on time dependencies: if the keyword NOPTIMEDEPENDEENCY(YES) is specified in the TOPOLOGY options when the Symphony is created through the daily planning jobs, the jobs are cancelled only after the time dependencies are locally satisfied. In order to successfully execute the NP and UN commands on an operation that is defined on a fault-tolerant workstation and that does not use the centralized script option, the workstation should be linked.
Running an operation immediately with EXECUTE

Running an operation immediately with EXECUTE

The EXECUTE command overrides normal scheduling rules except dependencies. You can use the EXECUTE command when:

- An operation is waiting for a resource that is not actually required.
- You want only one job to be submitted through Tivoli Workload Scheduler for z/OS.
- Automatic job submission is not active for all operations.
- A planned shutdown of a workstation is in progress and an operation is not submitted by Tivoli Workload Scheduler for z/OS because the job cannot finish in time.

The EXECUTE command, EX, can be issued for an operation on a computer workstation with automatic reporting if the status of the operation is A, *, or R. The EXECUTE command causes Tivoli Workload Scheduler for z/OS to start the operation without regard to normal scheduling criteria. An operation that you EXECUTE will be started even when:

- Job submission is not active.
- Job options for the operation do not specify automatic submit.
- Time dependency for the operation is not satisfied.
- Required resources are not available.
- The operation has H (manual hold) extended status. It remains in held status.
- The operation has N (NOP) extended status. It remains in NOP status.

In order to successfully execute the EX command on an operation that is defined on a fault-tolerant workstation and that does not use the centralized script option, the workstation should be linked. Moreover, for these operations the EXECUTE command works only if the job is in a consistent status (i.e. not in execution or cancelled) at the time it is received by the appropriate fault-tolerant agent. Otherwise, it is discarded.

**Note:** If the workstation the operation is defined to is not active and there is no active alternate workstation connected, or the operation is not re-routable, the EXECUTE command is rejected and the operation status is not changed.

You can also request EXECUTE from the MCP panel. See “Modifying operations” on page 577 for more details.

**Diagnosing delays**

Sometimes you need to run a specific operation in the plan immediately or you need to know why a certain job is not started. Here are some reasons why a computer workstation operation might not be started:

- The workstation is not open.
- Predecessors are not complete.
- No parallel server is available.
- The workstation is offline or has failed and no rerouting is in effect.
- The workstation is active but not connected.
- Not enough workstation resources are available.
- Not all the required special resources are available.
- The operation is waiting for a specific time of day.
- The operation has been manually held.
- The automatic-job-submission option is set to NO for the operation.
- There was an error during job submission.
- The operation is waiting for manual cleanup.
- Restart and cleanup is in progress.
The scheduling environment is not available

Many of these reasons are indicated by a unique extended status code. For example, operations that are waiting until a particular time of day will have T extended status code. If you are not familiar with the code or if there is no code, request additional information about the operation by entering row command I. See Appendix E, “Status, error, and reason codes,” on page 743 for a full list of error codes.

The SELECTNG APPLICATION OCCURRENCE AND OPERATION INFORMATION panel in Figure 210 shows an operation that is ready but cannot be submitted because job submission is deactivated (/SF5800001/SF590000). The ALL DEPS option can be particularly helpful. You can use this function to find out what outstanding predecessors remain before an operation will start and to see the impact of its being late or failing.

![EQQSOPSP SELECTING APPLICATION OCCURRENCE AND OPERATION INFORMATION](image)

The ALL DEPS option can be particularly helpful. You can use this function to find out what outstanding predecessors remain before an operation will start and to see the impact of its being late or failing.

### Reseting bind information for a shadow job

When you define a cross dependency between a job running in your environment and a job running on a different Tivoli Workload Scheduler engine, the bind process associates the shadow job to the remote job instance. In the READY LIST panel, with the BND command you can reset the information collected about the remote job and thus make the shadow job eligible for a new bind process.

The BND command can be issued only for shadow jobs in Ready status.
The QCP panel

The QUERY CURRENT PLAN (QCP) panel provides answers to your production status queries. You can request detailed or summary information on individual applications, operations, or workstations, and summary information concerning all the operations. The QCP panel looks at the current plan, which is continuously updated as the operations are processed. You can use the QCP panel to:

- Determine why an operation has not been started.
- Provide status information.
- Display a list of operations that have ended-in-error.
- Decide if intervention is required to speed up the processing of specific applications. You can display the applications that are most critical and those that have missed, or are close to missing, the defined deadline.
- Check information before making modifications to the current plan.
- Display a list of all dependencies for an operation. This function is of particular benefit to quickly identify which outstanding predecessors are not completed. The scheduler displays up to 999 levels of dependencies.
- Determine the impact of an operation that has ended in error.

You can reach the QCP panel from anywhere in Tivoli Workload Scheduler for z/OS panels by entering =6 at the command prompt. This takes you to the QCP menu, displayed in Figure 211. However, you can invoke QCP functions from many places in Tivoli Workload Scheduler for z/OS. For example, if you enter row command I from the ready list, the QCP panel SELECTING APPLICATION OCCURRENCE AND OPERATION INFORMATION is displayed. This can save time because you do not need to leave an area of the panel to get information.

```
EQQSTOPP -------------- CURRENT PLAN AND STATUS INQUIRY ------------------------
Option ===>
Select one of the following:
1 APPLICATIONS - Query application occurrences
2 MOST CRITICAL - Query most critical uncompleted application occurrences
3 OPERATIONS - Query operations (jobs)
4 ENDED IN ERROR - Query operations ended in error
5 WORK STATIONS - Query work station activities
6 GENERAL - Query general information about current plan
7 CRITICAL JOBS - Query active critical jobs and their critical paths
```

Figure 211. EQQSTOPP - Current plan and status inquiry

The options available from this menu are described in the following sections.

**Querying application occurrences**

When you select option 1 (APPLICATIONS), Tivoli Workload Scheduler for z/OS displays a filtering panel where you can specify the selection criteria that determines the applications displayed. For example, you can list only those applications that are not complete, or applications added by the ETT function. If the selection fields are left blank, all application occurrences in the plan in status W, S, C, E, or U will be displayed. Deleted applications are displayed only when you specifically request applications in D status.
Querying application occurrences

If you request additional information by using the S row command, you see the panel in Figure 212:

![EQQSAOSP - Selecting application occurrence information](image)

From this panel you can request detailed information about the occurrence, the operations defined in the occurrence, or the external dependencies established with the occurrence.

From the CURRENT PLAN AND STATUS INQUIRY panel, select option 2 MOST CRITICAL to display a list of uncompleted occurrences sorted by the latest start time.

The latest start time is the latest time that the operation can start in order to meet the deadline. This calculation considers the operation deadline, estimated duration, resource requirements, and successor processing. When Tivoli Workload Scheduler for z/OS creates the current plan, it calculates the latest start time for all the operations in a chain of dependencies, starting with the last one.

The occurrences that missed or will miss the defined deadline time are at the top of the list, as shown in Figure 213 on page 537.
Querying application occurrences

When you are viewing a list of operations, you can request detailed information about a particular operation by typing S in the row command column next to the operation. This takes you to the SELECTING APPLICATION OCCURRENCE AND OPERATION INFORMATION panel. You can reach the panel from the MCP, QCP, or workstation communication panels, from anywhere that a list of operations can be displayed. The panel is shown in Figure 214 on page 538.
The 9 ALL DEPS option shows the dependencies of an operation and helps you determine the impact of late or failed processing.

When selecting a dependency type, you can also customize nesting level and detail for the output list. You can display up to 999 levels of dependencies. The nesting detail level can be:

**ALL**  Show the higher nesting level.

**NONE**  Show the first nesting level found when exploring the dependency network.

You can display up to 999 levels of dependencies. By customizing nesting level and detail, you can balance proper response time and usability.

After you specify your selection, the panel shown in Figure 215 on page 539 is displayed.

---

**Figure 214. EQQSOISP - Selecting application occurrence and operation information**

The 9 ALL DEPS option shows the dependencies of an operation and helps you determine the impact of late or failed processing.

When selecting a dependency type, you can also customize nesting level and detail for the output list. You can display up to 999 levels of dependencies. The nesting detail level can be:

**ALL**  Show the higher nesting level.

**NONE**  Show the first nesting level found when exploring the dependency network.

You can display up to 999 levels of dependencies. By customizing nesting level and detail, you can balance proper response time and usability.

After you specify your selection, the panel shown in Figure 215 on page 539 is displayed.
Querying operation information

In the list displayed in Figure 215, the immediate predecessor is defined at level 1 (1 in the LEV column) and the subsequent predecessors at level 2.

The list shown in EQQSPG1L panel is ordered by:

**First sort key: LEV**
The nesting level of the dependent operation

**Second sort key: TY**
The dependency type

**Third sort key: APPLICATION ID**
The application name

**Fourth sort key: OPERATION NO**
The operation number

Checking the status of a workstation

Select option 5 WORK STATIONS to display a list of the workstations that includes the current status and reporting attribute. When you request additional summary information with the row command S, the statuses of operations at the workstation are displayed.
Checking the status of a workstation

The estimated duration is the total planned duration (HH.MM) of the operations in that status. For status W, note that the estimated duration is the time that the waiting operations are expected to run once they are started, it does not include the waiting time.

From the BROWSING WORKSTATION ACTIVITY panel, you can browse the system information for the destination of a computer workstation by entering the row command I. However, a non-local workstation can be browsed only if it communicates through XCF, NCF, TCP/IP, or the submit/release data set.

Checking the status of the current plan

Option 6 GENERAL provides details about the current plan, such as the following:
- Creation date and time
- End date and time
- When the last current plan backup was taken
- When the first event was written to the job-tracking-event log since the latest backup of the current plan
- If a new current plan is being produced
- If a new current plan has been produced and is being brought into production
- The run number, processing status, and availability for the Symphony file
- The ddname of the following resources:
  - Current plan
  - Job-tracking-event log
  - JCL repository

The panel is shown in Figure 217 on page 541.
CRITICAL JOBS

Select option 7 CRITICAL JOBS to display the list of jobs in the current plan that are defined as targets of critical paths.

The Risk Level (RL) field can have one of the following values:

**H**  High risk. According to the current estimated end time, the job will not complete by the defined deadline. For example, the estimated end time is beyond the deadline.

**P**  Potential risk. One or more of the critical job predecessors are late, long running, or ended in error.
None. No problem was detected in the network of the critical job predecessors.

The Recalculation (R) field shows the cause of the recalculation; it can have one of the following values:

- **C**: Job on critical path, completed or removed for the hot list
- **D**: Dynamic updates to the plan
- **L**: Late job
- **P**: Daily planning

Enter the right scroll command to scroll the list right:

**Figure 219. EQQSCJO1 - Browsing active critical jobs (right part)**

Enter the $S$ row command to display the list of jobs belonging to the critical path for the target you selected. The BROWSING CRITICAL PATH panel is displayed:

**Figure 220. EQQSCPL1 - Browsing critical path (left part)**

You can scroll the list left and right by entering the left and right scroll commands, respectively.
For a critical job with high or potential risk, you can display a list of the critical job predecessors that are late, long running, or ended with an error, by entering the critical hot list command on the BROWSING CRITICAL JOBS panel.

A business scenario

This scenario shows how an operator can monitor the jobs that are critical for the customer’s business and that must complete by their deadline.

The operator uses the product dialog to meet a Service Level Agreement (SLA) that requires a DB2 database up and running each day by 3 PM, after the database backup.

He needs to be informed whether critical jobs risk missing their deadline, to take the appropriate actions if needed. While the plan is running, the operator expects that the scheduler dynamically controls the network of submitted jobs, detecting when a critical job predecessor is late, long running or ended with an error.

Roles

In this scenario, the scheduling administrator and operator are involved:

Scheduling administrator
  When planning the operations, he defines:
  • Input arrival, duration, and deadline times
  • Critical jobs

Operator
  Controls the submitted workload by using CRITICAL JOBS and CRITICAL HOT lists.

Setting up the environment

Plan and schedule your operations:

1. Mark your critical jobs in the z/OS database. Set DBSTART and DBPRINT as critical jobs, using a job network with the following structure:
Setting up the environment

Running the scenario

After you updated your current plan, you can monitor your critical workload by using the QCP dialog:

1. Select the CRITICAL JOB option. The following list is displayed, showing that the Risk Level (RL) for the DBSTART job is Potential risk:

2. Select the DBSTART job and enter the critical hot list (H) row command. The displayed panel shows that the L column is set to Y in the Flags field, meaning...
Running the scenario

that DBMAINT is a late job:

3. Check the status of the CPU2 workstation. It is offline.
4. Activate the workstation. The DBMAINT job starts to run.
5. Return to the BROWSING ACTIVE CRITICAL JOBS panel. The refreshed list shows that the Risk Level (RL) for the DBSTART job is now No risk.
Running the scenario
Chapter 26. Updating the current plan

This chapter describes the MODIFYING CURRENT PLAN (MCP) panel, which is the most important tool for operators controlling the daily processing of Tivoli Workload Scheduler for z/OS.

The scheduler schedules work according to the current plan, which is created from information in Tivoli Workload Scheduler for z/OS databases. However, unplanned situations that require one-time changes to schedules constantly arise.

Situations that might require changes in plans are the following:
- Hardware failures
- Urgent hardware maintenance
- Last-minute changes to business schedules
- Business system reruns
- Unplanned jobs that must run immediately
- A change in work priorities
- Jobs that fail
- Data that arrives late
- Operations that run longer than planned

If you are new to Tivoli Workload Scheduler for z/OS, read this chapter as a guide to the MODIFYING CURRENT PLAN panel. If you are already familiar with Tivoli Workload Scheduler for z/OS, use Table 36 on page 548 to find the information you need quickly.

The history function lets you rerun completed operations that are no longer in the current plan. “Rerunning operations in the history database” on page 578 describes how to use the history function.

Two options in the MODIFYING CURRENT PLAN panel are large enough to need separate chapters:
- Chapter 27, “Handling operations that end in error,” on page 593 describes the error list, which you use to handle failed operations.
- Chapter 28, “Monitoring special resources,” on page 605 describes the Special Resource Monitor, which you use to change the status of resources and the allocation of resources.

You can manage most situations using the MODIFYING CURRENT PLAN panel, but you can also perform some actions from the READY LIST, which is described in Chapter 25, “Monitoring the workload,” on page 519.

Note: You can edit jobs with the modify current plan panel (for example, using option 6), but you must perform job setup operations from the READY LIST.
### Table 36. Using the MODIFYING CURRENT PLAN panel

<table>
<thead>
<tr>
<th>If you need to ...</th>
<th>Use fast path</th>
<th>See page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Diagnose delays</td>
<td>4.1 or 6.3</td>
<td>533 or 537</td>
</tr>
<tr>
<td>View the job log</td>
<td>5.2 or 5.4</td>
<td>563 or 597</td>
</tr>
<tr>
<td>Run work on request</td>
<td></td>
<td>549</td>
</tr>
<tr>
<td>Restart an application occurrence from the beginning</td>
<td>5.2</td>
<td>562</td>
</tr>
<tr>
<td>Rerun an occurrence from a specific operation</td>
<td>5.2</td>
<td>563</td>
</tr>
<tr>
<td>Delete an application occurrence</td>
<td>5.2</td>
<td>566</td>
</tr>
<tr>
<td>Change external dependencies to an occurrence</td>
<td>5.2</td>
<td>567</td>
</tr>
<tr>
<td>Rerun an operation in the history database</td>
<td>4.1, 5.3, 5.4 or 7</td>
<td>578</td>
</tr>
<tr>
<td>Change dependencies to an operation</td>
<td>5.3 or 5.2</td>
<td>568 or 555</td>
</tr>
<tr>
<td>Change the details of an operation</td>
<td>5.3 or 5.2</td>
<td>568 or 577</td>
</tr>
<tr>
<td>Set the status of an operation</td>
<td>4.1 or 5.2</td>
<td>526 or 568</td>
</tr>
<tr>
<td>Reset an operation to its previous state</td>
<td>4.1</td>
<td>527</td>
</tr>
<tr>
<td>Interrupt an operation</td>
<td>4.1</td>
<td>527</td>
</tr>
<tr>
<td>Report an operation as ended-in-error</td>
<td>4.1</td>
<td>527</td>
</tr>
<tr>
<td>Prepare jobs at a setup workstation</td>
<td>4.1</td>
<td>527</td>
</tr>
<tr>
<td>Delay an operation, and release it</td>
<td>4.1 or 5.3</td>
<td>531 or 577</td>
</tr>
<tr>
<td>Add and delete operations</td>
<td>5.2 or 5.3</td>
<td>569 or 577</td>
</tr>
<tr>
<td>Remove and restore operations with NOP and UNNOP</td>
<td>4.1 or 5.3</td>
<td>532 or 577</td>
</tr>
<tr>
<td>Handle operations that have ended in error</td>
<td>5.4</td>
<td>593</td>
</tr>
<tr>
<td>Complete an ended-in-error operation</td>
<td>5.4</td>
<td>596</td>
</tr>
<tr>
<td>Modify a job that has failed</td>
<td>5.4</td>
<td>597</td>
</tr>
<tr>
<td>Restart an ended-in-error operation</td>
<td>5.4</td>
<td>597</td>
</tr>
<tr>
<td>Get rerun or recovery instructions (view operator instructions)</td>
<td>5.4 or 4.1</td>
<td>596 or 527</td>
</tr>
<tr>
<td>Specify automatic restart for operations that fail</td>
<td>5.4</td>
<td>602</td>
</tr>
<tr>
<td>Recover when automatic restart of an operation fails</td>
<td>5.4</td>
<td>602</td>
</tr>
<tr>
<td>Control and monitor cleanup and restart</td>
<td>5.4</td>
<td>601</td>
</tr>
<tr>
<td>Inform Tivoli Workload Scheduler for z/OS of unplanned changes in resources</td>
<td>5.5</td>
<td>581</td>
</tr>
<tr>
<td>Change workstation availability</td>
<td>5.5</td>
<td>584</td>
</tr>
<tr>
<td>Redirect work to an alternate workstation</td>
<td>5.5</td>
<td>586</td>
</tr>
<tr>
<td>Keep plans up-to-date</td>
<td>3.1 or 3.2</td>
<td>582</td>
</tr>
<tr>
<td>Maintain special resources</td>
<td>5.7</td>
<td>605</td>
</tr>
<tr>
<td>Complete an application occurrence</td>
<td>5.2</td>
<td>566</td>
</tr>
<tr>
<td>Handle operations with the OSEQ error code</td>
<td>5.2</td>
<td>599</td>
</tr>
<tr>
<td>Select an ended-in-error list layout</td>
<td>5.4</td>
<td>594</td>
</tr>
<tr>
<td>Create your own ended-in-error list layout</td>
<td>5.9</td>
<td>594</td>
</tr>
<tr>
<td>Run an operation immediately with EXECUTE</td>
<td>4.1 or 5.3</td>
<td>533 or 577</td>
</tr>
</tbody>
</table>
Using fast paths

Table 36 on page 548 shows the fast path to the correct panel in the MODIFYING CURRENT PLAN panel. For example, if the fast path is 5.1, enter =5.1 on the command line to reach the panel, which is the same as selecting option 5 (MODIFYING CURRENT PLAN panel) in the main menu and selecting option 1 (ADD) in the MODIFYING CURRENT PLAN panel. You can avoid the filter panel, if there is one, by adding .0; for example, 5.3.0 instead of 5.3. If you do this, Tivoli Workload Scheduler for z/OS uses the previous filter criteria.

Accessing the Modifying Current Plan panel

To enter the panel, select the MCP option from the main menu, and the panel in Figure 225 is displayed.

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 ADD</td>
<td>Add a new occurrence to the current plan</td>
</tr>
<tr>
<td>2 LIST</td>
<td>List existing occurrences for further processing</td>
</tr>
<tr>
<td>3 OPERATIONS</td>
<td>List existing operations for further processing</td>
</tr>
<tr>
<td>4 ERROR HANDLING</td>
<td>Handle operations in error</td>
</tr>
<tr>
<td>5 WORK STATIONS</td>
<td>Change status and open interval of work stations</td>
</tr>
<tr>
<td>6 JOB SETUP</td>
<td>Prepare JCL for jobs in the current plan</td>
</tr>
<tr>
<td>7 SPECRES</td>
<td>Special resource monitor</td>
</tr>
<tr>
<td>9 DEFINE EL</td>
<td>Define alternative error list layouts</td>
</tr>
</tbody>
</table>

Figure 225. EQQMTOPP - Modifying the current plan

Specifying selection criteria

When you select an option from the main menu, you might see a panel where you can specify selection criteria to reduce the number of entries in the list. The selection criteria is saved between sessions, so you can bypass the panel (for example, by specifying 2.0 instead of 2 for the LIST panel) if you do not need to change the criteria.

Running work on request

You can add work to the plan on request using the MODIFYING CURRENT PLAN panel. Before you can add an application occurrence to the plan, however, a description of the application must exist in the application description database.

If your installation frequently adds work to the plan that does not have an application description defined for it, consider using this method:

1. Create dummy model applications that match the work most commonly added. The simplest model application consists of one computer workstation operation. The job names in these model applications should be dummy names indicating that they are only models.
Running work on request

Note: Do not specify run cycles for these applications; if you do, Tivoli Workload Scheduler for z/OS automatically adds occurrences for these applications to the long-term plan and current plan.

2. When you must add work to the plan, select the model application that best corresponds to the work you want to add using the method described in “Adding occurrences to the current plan.” Change the operation details to match those of the work that you want to add. For example, you can change the job name, workstation name, and input arrival times for the operations. You can also add or delete operations, external dependencies, and internal dependencies.

Adding occurrences to the current plan

To add an application occurrence to the current plan, you can do any of the following:

• Select option 1 from the MODIFYING CURRENT PLAN panel. (Figure 225 on page 549).

• Enter =5.1 from the command prompt in any other part of Tivoli Workload Scheduler for z/OS panels.

• Use the CREATE command from the LIST panel (option =5.2).

Note: If you are using the advanced panels, you can add an application occurrence by selecting Action > Add from the OPERATIONS IN THE CURRENT PLAN panel. See Figure 248 on page 575 for more information.

Selecting occurrences

You specify selection criteria on the ADDING APPLICATIONS TO THE CURRENT PLAN panel, as shown in Figure 226.

The following fields can have a significant effect on the added occurrences:

• ERROR CODE, for details see “Specifying an error code” on page 553.

• RESOLVE REQUIRED and AUTOMATIC DEP, for details see “Including dependencies defined in the database” on page 553.
GROUP DEFINITION, for details see “Grouping occurrences” on page 559.

Values specified in this panel are carried forward to the ADDING AN APPLICATION TO THE CURRENT PLAN panel for each application that you select. INPUT ARRIVAL DATE always contains the current date. The following fields contain the values from your last MCP add:

- APPLICATION ID
- AUTOMATIC DEP
- RESOLVE REQUIRED

You might find it faster to specify only the application ID, group definition, and dependency indicators. If you do not specify input arrival or deadline times, Tivoli Workload Scheduler for z/OS extracts them from the run cycles defined to the application in the application description database (if any run cycles exist). These fields are filled in when you get to the ADDING AN APPLICATION TO THE CURRENT PLAN panel (see “Adding occurrences”).

When you need to add multiple occurrences, or if you do not remember the name of the application, leave the APPLICATION ID field blank, or specify a generic application ID such as PAY* to generate a list of applications.

Select the single occurrences that you want to add from this list with the row command A. To add part or all of the application group to which an application belongs, select option G. For a detailed description about how to add application groups to the current plan, see “Adding an application group to the current plan” on page 559.

Adding occurrences

When you select an individual application you want to add to the current plan, Tivoli Workload Scheduler for z/OS displays the following panel:
Adding occurrences

Note: If you are using the advanced panels, you can also add occurrences by selecting Add in the Action menu of the OPERATIONS IN THE CURRENT PLAN panel. See Figure 248 on page 575 for more information.

Any values specified on the ADDING APPLICATIONS TO THE CURRENT PLAN panel (see Figure 226 on page 550) are carried forward to this panel. The priority is extracted from the application description. The panel indicates the number of operations in the occurrence and the number of external predecessors.

The scheduler does not accept any command, except CANCEL, until the input arrival and deadline dates and times are specified. If you want the input arrival time to be the current time, you can either specify the necessary time or press Enter. The scheduler uses the current time as default for the input arrival time. If you press Enter again, Tivoli Workload Scheduler for z/OS uses the current date as default for the deadline date.

You can use the ISPF command delimiter to set up a chain command to do this quickly. For example, if you type ;; and press Enter, the panel is re-displayed with the cursor at the deadline time field. All other date and time fields are set by Tivoli Workload Scheduler for z/OS.

Note: If the added application has a run cycle defined, the input arrival and deadline time is taken from the first run cycle description. If you try to add an application with the same date and time as another occurrence of the same application already in the plan (even if deleted or completed), Tivoli Workload Scheduler for z/OS rejects it. If you mean to do this, change the time by one minute until the added occurrence is unique. But be careful if you want Tivoli Workload Scheduler for z/OS to resolve external dependencies: external dependencies can depend on the input arrival time of the added occurrence. For example, if application B depends on application A, what happens if you add an extra occurrence of A and B? When you add A, you must give its occurrence a different input arrival time to the regular occurrence. When you add B, you give its occurrence the same or later input arrival time, and it should become dependent on the
added occurrence of A, because that is the closest occurrence with an equal or earlier input arrival time. See “Including dependencies defined in the database” for a description of how dependencies are resolved when there are several candidates. But you can have problems if the operations in A and B have explicit input arrival times: these are not affected by the occurrence input arrival time that you specify on the ADDING AN APPLICATION TO THE CURRENT PLAN panel, and you should alter these dependencies manually. See “Changing external dependencies to an occurrence” on page 567 for details.

You can use the DEP and OPER commands to alter dependencies and operation details for this occurrence. When you have specified all the occurrence information, add it to the current plan by issuing the END command.

**Specifying an error code**

ERROR CODE is an optional field containing a user-defined value up to 4 characters. This value is included in the daily planning reports for the occurrence when it completes. You can use this to identify particular categories of added processing.

For example, you could have an installation standard that all report reprints are added to the current plan with an error code of REPT. Similarly, occurrences added as part of a business system rerun could have an error code RER.

The specification of an error code does not affect the submission or tracking of operations, but it can be a useful tool for measuring the unplanned workload on your system.

You can use the error code to identify the name or the department of the user who requested the processing. This can be helpful when you are converting regularly added occurrences to an automated method by using ETT or PIF.

**Including dependencies defined in the database**

When you add application occurrences to the current plan, decide if the occurrence or group should be added with the dependencies defined in the application database. With the AUTOMATIC DEP field, you can:

- Add both predecessor and successor dependencies.
- Add only predecessor dependencies.
- Add only successor dependencies.
- Ignore any dependencies specified in the application description.

Because conditional dependencies are considered external dependencies, if set AUTOMATIC DEP to N they are not added even if they refer to internal operations. If an application definition contains conditions and you set AUTOMATIC DEP to N, a confirmation panel (EQQMCADC) is displayed to avoid losing conditions from the plan.

When you request that dependencies be considered for individual occurrences, you can also specify whether or not the predecessors must be resolved. If you specify RESOLVE REQUIRED=YES, the scheduler issues a message if any predecessors defined in the application cannot be resolved and the scheduler does not add the application occurrence to the current plan until you enter DEP and delete the unresolved dependency.
**Including dependencies defined in the database**

If predecessors are added to an application in the AD database, this does not take effect in the current plan until that application is scheduled by daily planning or manually added to the current plan.

You can also delete external dependencies or specify additional dependencies when you are adding an occurrence. For detailed information, see "Changing and adding dependencies" on page 555.

Successor dependencies are added automatically only if the dependency is defined in the applications database at the time the successor occurrence was added to the current plan. When an occurrence is added to the plan, either manually using the MODIFYING CURRENT PLAN panel or automatically scheduled by daily planning, Tivoli Workload Scheduler for z/OS creates a potential predecessor record for predecessors that are not in the current plan. This can happen if the predecessor (job A) is run on demand. When its successor (job B) is added to the plan (at daily planning), Tivoli Workload Scheduler for z/OS notices that its predecessor (A) is missing. When you add the on-demand application (A), and specify that Tivoli Workload Scheduler for z/OS is to resolve successor dependencies, Tivoli Workload Scheduler for z/OS looks for predecessors that match, and the added occurrence A becomes a predecessor to B, unless B has started or has already completed.

If there are several occurrences of B in the plan, Tivoli Workload Scheduler for z/OS must choose which occurrence of B to make the successor. The best successor is the first not-started operation that has the correct application name, the correct operation number, and an occurrence input arrival later than or equal to that of the occurrence being added.

To handle the resolution of conditional successors, the scheduler might automatically generate conditions, to be consistent with the database definitions and prevent a condition from referring to different occurrences of the same applications. You can change only the description of the conditions that are automatically generated.

For each missing predecessor, Tivoli Workload Scheduler for z/OS can keep track of up to 1000 successors. If this limit is reached, no more successors are added and a warning message is issued.

If you add an occurrence to the current plan from the MODIFYING CURRENT PLAN panel, and later run a daily planning EXTEND or REPLAN job, daily planning never makes the manually-added occurrence a predecessor or successor of an occurrence that it adds from the long-term plan, even if the dependency is specified in the database. This is because daily planning does not search the current plan to resolve dependencies, but it takes the dependencies directly from the long-term plan. For example:

1. You add an occurrence of PAYDAILY using the MODIFYING CURRENT PLAN panel. PAYDAILY is specified in the database as a predecessor of PAYBACKP.
2. You extend the current plan, which adds an occurrence of PAYBACKP.
3. Even though the application description for PAYBACKP states that it has an external predecessor PAYDAILY, PAYBACKP will not have this dependency, because occurrence dependencies are not added when the current plan is extended unless they were present in the long-term plan.
Including dependencies defined in the database

When you specify resolution of predecessor dependencies, Tivoli Workload Scheduler for z/OS looks for occurrences that are predecessors to the added operations, but also considering the following rules:

- The predecessor operation can be completed (or started).
- If the added operation has an explicit input arrival time, Tivoli Workload Scheduler for z/OS takes the predecessor occurrence that has both the following characteristics:
  - The input arrival time is earlier and closest to, or the same as, the input arrival time of the added operation.
  - Contains a candidate predecessor operation.
- If the added operation has no explicit input arrival time, Tivoli Workload Scheduler for z/OS takes the predecessor occurrence that has both the following characteristics:
  - The input arrival time is earlier and closest to, or the same as, the input arrival time of the added occurrence, as specified on the ADDING AN APPLICATION TO THE CURRENT PLAN panel in Figure 228 on page 552.
  - Contains a candidate predecessor operation.

Note: When resolving a dependency, Tivoli Workload Scheduler for z/OS uses the input arrival time of the predecessor occurrence, not the input arrival time of the operation specified in the dependency.

Changing and adding dependencies

When you add an occurrence, you can make the operations dependent on operations in the plan, even though these dependencies are not defined in the database.

On the ADDING AN APPLICATION TO THE CURRENT PLAN panel, shown in Figure 228 on page 552, enter the OPER command, and the panel in Figure 229 displays, in which you select the operation.

```
Eqummopl --------- MODIFYING OPERATIONS IN THE CURRENT PLAN - ROW 1 TO 1 OF 1
Command ===> Scroll ===> PAGE

Enter the GRAPH command above to view operations graphically or change data in the rows, and/or enter any of the following row commands:
   I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
   J - Edit JCL, O - Browse operator instructions, S - Modify operation details
   L - Browse joblog

Application : PAYQUERY AD-HOC payroll query
Owner : SAMPLE payroll application
Input arrival : 03/07/28 14.11
Status : Being added

Row Operation Jobname PS Duration Opt Dep Res Stat
  cmd ws no. text HH.MM.SS S T S/P S R1 R2 N Cu
'' CPU1 050 run as required PAYQUERY 1 00.05.00 Y N Y 0 0 A

Figure 229. EQQMMOPL - Modifying Operations in the Current Plan panel
```

Note: If you are using the advanced panels, you can also open the panel by selecting Modify in the Occurrence menu of the OPERATION IN THE CURRENT PLAN panel (see 576 for more information).
Changing and adding dependencies

Select the operation. The panel in Figure 230 appears.

| EQQMODP -------- MODIFYING AN OPERATION IN THE CURRENT PLAN ---------- |
| Option ===>

Select one of the following:

1 DEPENDENCIES - Delete and add (internal and external)
2 SPEC RESOURCES - Special resources
3 AUTOMATIC OPTIONS - Job, WTO, and print options
4 TIME - Time specifications
5 JCL - Edit JCL for MVS job
6 GENERAL - General information
7 CLEANUP OPTIONS - Cleanup Options
8 EXTENDED INFO - Extended Information
9 AUTOMATION INFO - Automation info
10 USER FIELDS - User fields operation info
11 REMOTE JOB INFO - Remote job information

Application : PAYQUERY AD-HOC payroll query
Operation : CPU1 050 run as required
Jobname : ROXINO
Input arrival : 03/07/28 14.11
Duration (HH.MM.SS) : 0005.00
External predecessors : No Conditional predecessors : No
External successors : No Conditional successors : No
Special resources : Yes Number of conditions : 0

Figure 230. EQQMODP - Modifying an Operation in the Current Plan panel

Note: If you are using the advanced panels, you can display the MODIFYING OPERATION IN THE CURRENT PLAN by either:

- Entering M in the Row cmd column next to the selected operation in the OPERATIONS IN THE CURRENT PLAN panel (see Figure 243 on page 572).
- Selecting Modify in the TABLE ROW COMMANDS panel after selecting an operation in the OPERATIONS IN THE CURRENT PLAN panel (see Figure 244 on page 573).
- Selecting Modify in the Operation menu of the OPERATION IN THE CURRENT PLAN panel (see Figure 249 on page 576).

In panel EQQMODP, select option 1 (DEPENDENCIES). The panel shown in Figure 231 on page 557 is displayed.
Changing and adding dependencies

To create non-conditional dependencies, enter the CREATE command. The panel shown in Figure 232 is displayed.

You can specify the details of the predecessor or successor in the CREATING A DEPENDENCY IN THE CURRENT PLAN panel, but it is easier to use a generic search character in the ID (P* in Figure 232) and press Enter.

The DEFINING DEPENDENCIES IN THE CURRENT PLAN, shown in Figure 233 on page 558 is displayed.
### Changing and adding dependencies

A wildcard search such as the one in Figure 232 on page 557 is useful when you have several operations with the same name. Use the following row commands to specify the dependencies:

- **P** Makes the operation in the row a predecessor of the operation that you are adding.
- **S** Makes the operation in the row a successor of the operation that you are adding.
- **blank** Removes the dependency between the operation in the row and the one that you are adding.

**Note:** Changes that result in a conflicting status (such as specifying that an operation is a predecessor of a completed operation) are rejected.

#### Changing and adding conditional dependencies

To handle conditional dependencies, enter the **COND** command from the panel shown in Figure 231 on page 557. The panel shown in Figure 234 on page 559 is displayed.

---

**Table: EQQMDLL - Defining dependencies in the current plan**

<table>
<thead>
<tr>
<th>Application id</th>
<th>Input arrival</th>
<th>Jobname</th>
<th>Operation</th>
<th>date</th>
<th>time</th>
<th>ws</th>
<th>no.</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYBACKP</td>
<td>09/06/15</td>
<td>12.00</td>
<td>PAYBACKP CPU1 015 Daily payroll backup</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>09/06/15</td>
<td>12.00</td>
<td>PAYBACKP WTO1 030 PAYX OPEN DATASET</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>09/06/17</td>
<td>12.00</td>
<td>PAYBACKP CPU1 015 Daily payroll backup</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>09/06/17</td>
<td>12.00</td>
<td>PAYBACKP WTO1 030 PAYX OPEN DATASET</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>09/06/20</td>
<td>12.00</td>
<td>PAYBACKP CPU1 015 Daily payroll backup</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>09/06/20</td>
<td>12.00</td>
<td>PAYBACKP WTO1 030 PAYX OPEN DATASET</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>09/06/21</td>
<td>12.00</td>
<td>PAYBACKP CPU1 015 Daily payroll backup</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>09/06/21</td>
<td>12.00</td>
<td>PAYBACKP WTO1 030 PAYX OPEN DATASET</td>
<td>W</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>09/06/17</td>
<td>12.00</td>
<td>PAYDAILY WTO1 005 PAYX CLOSE DATASET</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>09/06/17</td>
<td>12.00</td>
<td>PAYDAILY SETP 010 Job setup for paydaily</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>09/06/17</td>
<td>12.00</td>
<td>PAYDAILY CPU1 020 Runs pay04 and pay06</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>09/06/20</td>
<td>12.00</td>
<td>PAYDAILY WTO1 005 PAYX CLOSE DATASET</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>09/06/20</td>
<td>12.00</td>
<td>PAYDAILY SETP 010 Job setup for paydaily</td>
<td>C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Changing and adding dependencies

By entering the available row commands, you can:

- Change a condition, depending on its status. In fact you can:
  - Modify type, operator and operands (status or return code values) of a selected condition dependency, until the status is undefined both for the selected dependency and the condition itself.
  - Manually reset the status of a step-level dependency that is undefined because of missing step-end information. Row commands are supported to reset the status to true or false.
  - Delete or add condition dependencies, until the condition status is undefined.

- Delete undefined conditions and create new ones, until the overall condition status for the selected operation is undefined. Typically, you can delete or create conditions for operations in ready or waiting status.

Grouping occurrences

When you add occurrences to the current plan, you can specify that the occurrence should become a member of an existing occurrence group or that a new occurrence group should be created. Related occurrences can be handled as a single entity when they are defined in an occurrence group. An occurrence group, and therefore all member occurrences, can be completed or deleted from the current plan using a single panel request.

The daily planning process automatically creates occurrence groups in the current plan if such a group exists in the long-term plan for the corresponding period.

When an occurrence is defined as a group member, it cannot be individually completed or deleted. Group members must be removed from the group if individual complete or delete is required.

Adding an application group to the current plan

If you enter the G row command in the SELECTING APPLICATIONS TO ADD TO THE CP panel beside an application that belongs to a group, you can add all or part of that group to the current plan.
Excluding some applications

When Tivoli Workload Scheduler for z/OS displays the ADDING AN OCCURRENCE GROUP TO THE CP panel, you see a list of all applications that are members of the group you have selected. From the list, you can exclude specific applications defined in the group. In this panel, you enter the input arrival and deadline time that will be used for all occurrences in the group. Note that only the applications that are valid for the input arrival date and time that you specify are contained in the list. If an application that is normally part of the group is not valid for the specified input arrival date, it is not listed as part of the group.

Specifying dependency resolution

You can specify whether dependencies should be resolved and if all occurrences in the group should be added using the same priority or JCL variable table. When you request automatic dependency resolution for an application group, Tivoli Workload Scheduler for z/OS will first attempt to establish any dependencies to other applications within the group that is being added.

The applications in an application group are handled as a single entity, but when the group is added to the current plan, the occurrences are generated one at a time. If Tivoli Workload Scheduler for z/OS is abnormally terminated during such a transaction, the occurrences already added to the current plan will not be backed out, and the other occurrences will not be automatically added when Tivoli Workload Scheduler for z/OS is restarted. The order that Tivoli Workload Scheduler for z/OS creates the occurrences depends both on the application name and the structure of the dependencies between the applications. The order does not affect how the dependencies are created.

When an occurrence group is added, Tivoli Workload Scheduler for z/OS resolves dependencies only within the group if you specify option G on the AUTOMATIC DEP field in Figure 235. For all other options, dependencies will be resolved as normal for each added occurrence, with the exception that dependencies are always resolved within the group if possible.

Before adding any occurrence, Tivoli Workload Scheduler for z/OS checks for dependency loops in the occurrence group. If it detects a loop, one of the
dependencies forming the loop will not be resolved within the group. However, if there is another occurrence, external to the group, that matches the dependency criteria, the dependency will be resolved to this external occurrence if option Y or P is specified in the AUTOMATIC DEP field. A warning message is issued.

If an application defines a dependency that cannot be satisfied within the group, Tivoli Workload Scheduler for z/OS will attempt to establish the dependency to another occurrence, outside the group, in the current plan. You can see the dependencies established for occurrences created by adding the application group by entering the GRAPH command on the MODIFYING OCCURRENCES ADDED TO THE CURRENT PLAN panel (Figure 236).

When you have entered the required data and deleted any rows that you do not want to include in the group, enter END or press PF3. The applications required in the group are added to the plan. Initially, these occurrences are held by Tivoli Workload Scheduler for z/OS to let you modify the operation data or modify dependencies for any occurrences in the group. If you do not want to proceed with the group add, enter CANCEL on the ADDING AN OCCURRENCE GROUP TO THE CP panel (Figure 235 on page 560).

When the occurrences are added to the plan, Tivoli Workload Scheduler for z/OS displays this panel, which lists only those occurrences added as a result of adding the group:

```

<table>
<thead>
<tr>
<th>Application</th>
<th>Input arrival</th>
<th>S</th>
<th>P</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYM1</td>
<td>03/03/14 12.00</td>
<td>W</td>
<td>5</td>
<td>Y</td>
</tr>
<tr>
<td>PAYM2</td>
<td>03/03/14 12.00</td>
<td>W</td>
<td>5</td>
<td>Y</td>
</tr>
</tbody>
</table>
```

Figure 236. EQQMAMOL - Modifying occurrences added to the current plan

**Note:** If you are using the advanced panels, you can perform these occurrence tasks from either the Table Row Commands table (see Figure 244 on page 573) or from the Occurrence menu of the OPERATION IN THE CURRENT PLAN panel (see 576).

From the MODIFYING OCCURRENCES ADDED TO THE CURRENT PLAN panel, you have access to the complete range of MCP services. At this point, the occurrences are still in a hold state; if you enter CANCEL from this panel, the occurrences remain held. The occurrences are already added to the plan, but you can mark them deleted with the DELETE command. If you do this, you cannot add these occurrences later with exactly the same input arrival date and time (because the occurrences, even though marked deleted, are still in the plan). If you need to add them again later, change the arrival time slightly.

After making any modifications, enter END or press PF3 to release the occurrences. If your conversation with Tivoli Workload Scheduler for z/OS subsystem is cancelled for any reason while the occurrences are still in a hold state, they remain held.
Specifying dependency resolution

Note: While you are using the panel in Figure 236 on page 561, nobody else can access the listed occurrences. Release the lock as soon as you can.

Restarting an occurrence from the beginning

The LIST option of the modify current plan panel takes you to the MODIFYING OCCURRENCES IN THE CURRENT PLAN panel:

To restart an application occurrence from the beginning, enter W in the row command field for the occurrence. After you confirm the request, Tivoli Workload Scheduler for z/OS resets the status of all operations in the occurrence to W (waiting). All operations (whose predecessors have completed) are then automatically resubmitted if all other criteria for submission are fulfilled. In this process, all external and conditional successors that have a status of ready are reset to W (waiting) status. If they are conditional successors, the scheduler might reset the related conditions to undefined, meaning that:

- Single dependencies in existing conditions are reset to undefined.
- Each condition is reset to undefined depending on the specified rule.

If an external or conditional successor has started, however, the request to restart is rejected, and an error message is displayed. If this happens, consider deleting the external successor dependency and then reset the occurrence to waiting status. See “Changing dependencies to an operation” on page 568 for details on how to delete dependencies.

Note: If you are using the advanced panels, you can rerun an occurrence from either the Table Row Commands panel (see Figure 244 on page 573) or from the Occurrence menu of the OPERATION IN THE CURRENT PLAN panel (see 576).

If you want to reset the status of successors as well, you need RERUN instead of SET TO WAITING. See “Rerunning an occurrence in the current plan from a specific operation” on page 563.

Notes:
1. Row commands are usually executed immediately. Thus, if you cancel the rerun against which you have issued a NOP command, the operation remains removed (no-oped), because the NOP command has already been processed.
2. If at some point an operation was set to HOLD or NOP, the HOLD or NOP status will stay in effect when the occurrence is set to waiting. To see which operations have been set to HOLD or NOP, request a list of the operations using option 5.3.

3. You might need to HOLD or NOP operations that will be reset by the command.

4. When an operation is rerun, Tivoli Workload Scheduler for z/OS uses the job that was last submitted for the operation.

5. Same restrictions apply as for changing the operation status to ready by selecting option 6 (GENERAL) from the MODIFYING AN OPERATION IN THE CURRENT PLAN panel. In particular, a step or job restart request might imply a request to change to ready the status of an operation with conditional successors already started, completed, suppressed by condition, or ended in error. In this case the scheduler issues message EQQM208E. Only when rerunning an occurrence you might indirectly obtain this kind of change.

**Rerunning an occurrence in the current plan from a specific operation**

If an occurrence is still in the current plan, you can rerun it from a particular operation if all external dependencies for that occurrence are in READY status (R, *, or A). To do so, select the occurrence with the row command R on the MODIFYING OCCURRENCES IN THE CURRENT PLAN panel [Figure 237](#). The RERUNNING AN OCCURRENCE IN THE CURRENT PLAN panel [Figure 238](#) is displayed, where you can specify which operation in the occurrence you want to restart from.

**Note:** If an operation ends in error, use the error list to rerun it.

<table>
<thead>
<tr>
<th>Application</th>
<th>APPI application 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input arrival</td>
<td>03/04/29 20.17</td>
</tr>
<tr>
<td>Calendar name</td>
<td>DEFAULT</td>
</tr>
<tr>
<td>DEADLINE DATE</td>
<td>03/05/01 Date in format YY/MM/DD</td>
</tr>
<tr>
<td>DEADLINE TIME</td>
<td>22.00 Time in format HH.MM</td>
</tr>
<tr>
<td>PRIORITY</td>
<td>7 1-9 where 1=low, 8=high and 9=urgent</td>
</tr>
<tr>
<td>ERROR CODE</td>
<td>____ For reporting purposes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Row Operation</th>
<th>Jobname</th>
<th>Status</th>
<th>Error Man</th>
<th>Nop</th>
<th>Cln</th>
<th>Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd ws no. text</td>
<td>code Hld T S Stat</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1 050 job1 in appl</td>
<td>JOB1 E JCL N N A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 238. EQQMROCL - Rerunning an occurrence in the current plan**

**Note:** If you are using the advanced panels, you can rerun an occurrence and enter the commands in Figure 238 by either:

- Using the TABLE ROW COMMANDS panel (see Figure 244 on page 573)
Rerunning an occurrence in the current plan from a specific operation

When rerunning an occurrence, you can choose to restart the step or the job. To do this, you must select which step will be the restart point by using the S command on the RERUNNING AN OCCURRENCE IN THE CURRENT PLAN panel. The operation status of the restart point is then set to R (ready). From that step forward in the occurrence, the statuses of the operations are automatically modified to waiting. In cases where an operation changes from C (complete) status to either R (ready) or W (waiting) status, you must decide if cleanup actions on the data sets should be performed. When you request restart and cleanup actions and job log information is available, you can choose to restart the operation from a particular step. The scheduler automatically tailors and verifies the job for restart. Use the OPERATION RESTART AND CLEANUP panel (Figure 239 on page 564) to initiate restart and cleanup actions.

For additional details about restart and cleanup actions at the job or step-level, see Chapter 27, “Handling operations that end in error,” on page 593.

If there are conditional or external dependencies defined on the operation, they are included on the LIST DEPENDENCY STATUS CHANGE panel (Figure 240 on page 565) that is displayed automatically. From the RERUNNING AN OCCURRENCE IN THE CURRENT PLAN panel, you can modify the jobs of the dependent operations before Tivoli Workload Scheduler for z/OS resets them. The modified job is saved in the JCL repository when you exit and confirm the occurrence rerun. As in rerunning occurrences, the same procedures apply to dependencies.

You can use RERUN to reset dependent occurrences that have already completed. This can save you the trouble of adding the occurrences again if you need to rerun all or part of a business system and occurrences have not been planned out of the current plan. You select the point from which you must start the rerun and all successor dependencies are automatically set to waiting status.

**Note:** The real status of the operations is still unchanged. Whether any further row commands will be allowed depends on the real status, rather than on the status currently displayed. The effective rerun is executed only after
Rerunning an occurrence in the current plan from a specific operation

confirmation on the LIST DEPENDENCY STATUS CHANGE panel.

<table>
<thead>
<tr>
<th>Application id</th>
<th>Input arrival</th>
<th>Status</th>
<th>Calendar name</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYBACKP</td>
<td>09/04/29 12.00</td>
<td>W</td>
<td>MONTHLY</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>09/04/29 12.00</td>
<td>E</td>
<td>MONTHLY</td>
</tr>
</tbody>
</table>

Figure 240. EQQMOSTL - List dependency status change

The list includes both non-conditional and conditional dependencies.

Depending on whether you rerun an occurrence before or after a daily planning process, it might result in different outputs. In fact, daily planning removes from the current plan:

- Normal dependencies on completed predecessors. This is the default behavior, however, you can configure the behavior so that dependencies are retained, even after they are resolved. Refer to the **KEEPCOMPDEPS** parameter in the BATCHOPT initialization statement documented in *Customization and Tuning*.

- Evaluated conditions, both true and false. Single condition dependencies are removed provided that the corresponding condition is not undefined.

Once you selected an operation as restart point, the scheduler automatically:

- Modifies conditional predecessors to completed and conditional successors to waiting, as for standard dependencies.

- Sets to True - Forced by rerun the conditions on the conditional predecessors. It applies even if those conditions are on operations that are not involved in the rerun processing;

- Sets to Undefined - Forced by rerun the conditions on the conditional successors. Such conditions might include dependencies with different status values, for example a combination of all the possible status values: undefined because not evaluated yet, true, false, or undefined because of missing step-end information.

**Notes:**

1. Row commands are usually executed immediately. Thus, if you cancel the rerun against which you have issued a NOP command, the operation remains removed (no-oped), because the NOP command has already been processed.

2. If at some point an operation was set to HOLD or NOP, the HOLD or NOP status will stay in effect when the occurrence is set to rerun. To see which operations have been set to HOLD or NOP, request a list of the operations using option 5.3.
Rerunning an occurrence in the current plan from a specific operation

3. Rerunning an occurrence is a powerful request that might reset the status of the job network including the restart point. In particular, it is the only manual request that can reset the suppressed by condition status. Therefore consider using HOLD or NOP command for any impacted operation that is to be prevented from starting.

4. When an operation is rerun, Tivoli Workload Scheduler for z/OS uses the job that was last submitted for the operation.

5. For a fault-tolerant workstation, you cannot issue the S, RC, NP, or UN row commands in the LIST DEPENDENCY STATUS CHANGE panel.

6. Same restrictions apply as for changing the operation status to ready by selecting option 6 (GENERAL) from the MODIFYING AN OPERATION IN THE CURRENT PLAN panel. In particular, a step or job restart request might imply a request to change to ready the status of an operation with conditional successors already started, completed, suppressed by condition, or ended in error. In this case the scheduler issues message EQQM208E. Only when rerunning an occurrence you might indirectly obtain this kind of change.

Completing an application occurrence

When you want to complete the entire application occurrence, enter C in the row command field for the occurrence on the MODIFYING OCCURRENCES IN THE PLAN panel. The scheduler presents a confirmation panel that includes a list of the dependent operations to the occurrence for which you have requested completion. The normal dependencies listed are also considered complete if you enter Y. From this panel, you can request that Tivoli Workload Scheduler for z/OS HOLD or NOP a dependent operation. After your confirmation, Tivoli Workload Scheduler for z/OS sets all operations to C (complete) status, except for the suppressed by condition operations, that keep their status.

Notes:

1. If you are using the advanced panels, you can set a selected operation to complete from the:
   - TABLE ROW COMMANDS panel (see Figure 244 on page 573).
   - Action menu in the OPERATION IN THE CURRENT PLAN panel (see Figure 248 on page 575).
   - Occurrence menu in the OPERATION IN THE CURRENT PLAN panel (see 576).

2. If you set the status of an application occurrence to complete, Tivoli Workload Scheduler for z/OS can submit its successor operations.

3. The selected occurrence can contain an in-error operation, with conditional successors in final status, because of already evaluated conditions. The scheduler does not evaluate again these conditions after the occurrence status change, in fact no operation reruns, but sets them to the Frozen by complete extended status.

Deleting an application occurrence

You can delete an application occurrence from the current plan by entering D in the row command field for the occurrence in the MODIFYING OCCURRENCES IN THE CURRENT PLAN panel. The scheduler then displays a confirmation panel showing all external dependencies for the occurrence. After your confirmation, all operations in the occurrence are deleted. Additionally, you can delete occurrences on a fault-tolerant workstation in the same way.
Deleting an application occurrence

Note: If you are using the advanced panels, you delete an application occurrence from the TABLE ROW COMMANDS panel. See Figure 244 on page 573 for more information.

Except for conditional dependencies, for each operation in the deleted occurrence, Tivoli Workload Scheduler for z/OS automatically connects all its external predecessors to its external successors. By doing this, Tivoli Workload Scheduler for z/OS maintains the consistency of the network of operations. For conditional dependencies, network split cannot be prevented because the scheduler cannot redefine the rule criteria.

To prevent unwanted dependent operations from starting after the deletion, first change the external dependencies, as described under "Changing external dependencies to an occurrence." Alternatively, you can HOLD or NOP a dependent operation from the list displayed.

Note: The scheduler retains the deleted occurrences for reporting purposes and to ensure that the long-term plan is updated if the occurrence exists in the long-term plan. So you cannot add another occurrence of the same application to the current plan with the same input arrival time as the deleted occurrence. You can display deleted occurrences using the QUERY CURRENT PLAN panel.

Modifying an application occurrence

Requests to modify application occurrences in the plan might include requests to change:

• Input arrival and deadline times
• Priority
• Operation information
• JCL variable table used by the occurrence operations
• External and internal dependencies

To change an occurrence, enter M in the row command field for the occurrence in the MODIFYING OCCURRENCES IN THE CURRENT PLAN panel. You can change occurrence times and priority on the resulting MODIFYING AN OCCURRENCE IN THE CURRENT PLAN panel.

You can also change operations (see "Changing the details of an operation" on page 568 for detailed information) and display the list of external dependencies for verification and deletion. See "Changing external dependencies to an occurrence" for detailed information.

Note: If you are using the advanced panels, you modify an application occurrence from the Occurrence menu in the OPERATION IN THE CURRENT PLAN panel. See 576 for more information.

Changing external dependencies to an occurrence

You can display all external dependencies to an occurrence by entering the DEP primary command on the MODIFYING AN OCCURRENCE IN THE CURRENT PLAN panel. This takes you to the RESOLVING EXTERNAL DEPENDENCIES IN THE CP panel, where you can verify or delete the dependencies. You cannot add external dependencies from this panel. You can add external dependencies only by modifying individual operations.
Changing external dependencies to an occurrence

Note: If you are using the advanced panels, you can change external dependencies of an occurrence from the Operation menu on the OPERATION IN THE CURRENT PLAN panel. See Figure 249 on page 576 for more information.

Changing dependencies to an operation
To change dependencies to an operation, get the list of operations by entering OPER in the command line of the MODIFYING AN OCCURRENCE IN THE CURRENT PLAN panel. In the list of operations, enter $ in the row command field to get details of the operation. Select option 1 (DEPENDENCIES) in the MODIFYING AN OPERATION IN THE CURRENT PLAN, as described in “Changing and adding dependencies” on page 555. You can then delete dependencies and create new ones. You can handle both internal, external, and conditional dependencies.

In instances where you change dependencies for a fault-tolerant workstation, ensure that there is enough time for the dependency event to reach the distributed agent. If the operation starts before the dependency event reaches the agent, the operation status is set to error with the OSEQ error code.

Changing the details of an operation
To change the details of an operation, display the list of operations by entering OPER in the command line of the MODIFYING AN OPERATION IN THE CURRENT PLAN panel. The MODIFYING OPERATIONS IN THE CURRENT PLAN panel is displayed, where you can modify items such as:

- Workstation name
- Job name
- Operation text
- Use of workstation resources
- Submit options
- Time options
- Operation status
- User fields

Note: If you are using the advanced panels, you can change the details of an operation by either:
- Selecting Modify from the TABLE ROW COMMANDS panel (see Figure 244 on page 573).
- Selecting Modify from the Operation menu in the OPERATION IN THE CURRENT PLAN panel (see Figure 249 on page 576).

You can perform several actions by specifying the appropriate value in the command field of an operation, for example:

J To edit the JCL for the job.
L To view the output of completed operations. Tivoli Workload Scheduler for z/OS retrieves the output from the operation. Enter L a second time to browse the output.
O To browse the operator instructions for the operation.
S To display more details about the operation by returning to the MODIFYING AN OPERATION IN THE CURRENT PLAN panel.

Notes:
1. If you modify an operation by mistake, when you are returned to the MODIFYING AN OPERATION IN THE CURRENT PLAN panel use the
Changing the details of an operation

CANCEL command to discard the changes. However, if you modified the JCL using the J command, the changes that you made take effect even if you exit from the MODIFYING AN OPERATION IN THE CURRENT PLAN panel using CANCEL.

2. PF3 is the confirmation of a modify request. By entering PF3, the resulting operation values comprise the changes specified together with the existing values that have not been changed. If a modify request is entered and PF3 is issued without any change, the operation values are set to the existing values (for example, if the submit option was set to Y, the operation is submitted). If a modify request is entered by mistake, when you are returned to the MODIFYING AN OPERATION IN THE CURRENT PLAN panel, use CANCEL to discard the request.

3. Changes that would force a status change of a started or completed dependent operation are rejected.

Modifying extended information

From the MODIFYING AN OPERATION IN THE CURRENT PLAN panel, enter 8 to modify the information describing the operation. You can use the string to filter queries of operations.

Note: If you are using the advanced panels, you can change the details of an operation by either:

- Selecting Modify from the TABLE ROW COMMANDS panel (see Figure 244 on page 573).
- Selecting Modify from the Operation menu in the OPERATION IN THE CURRENT PLAN panel (see Figure 249 on page 576).

In the MODIFYING EXTENDED INFO IN THE CURRENT PLAN panel, type the information in the Operation Extended Name field and (or) in the Operation Scheduling Environment Name field, as shown in this example.

```
EQQMXDP --- MODIFYING EXTENDED INFO IN THE CURRENT PLAN --------
Command ===> Application: PAYDAILY daily payroll jobs
Operation: CPU1 020 Run pay04 and pay06
Job name: PAYDAILY
Enter change data below:
Operation Extended Name: daily payroll data for accounting___________________________
Operation Scheduling Environment Name: DB2ACTIVE___________________________

Figure 241. EQQMXDP - Modifying extended info in the current plan
```

After the scheduling environment name is added or changed in the plan, at job submission time it is used to check if the job can be submitted and, if this is so, to tailor the JCL by adding the SCHENV=SE_name keyword in the JOB card.

For more information, see Chapter 23, “Job scheduling and WLM,” on page 493.

Adding and deleting operations

To add or delete operations, request a list of operations by entering OPER in the command line of the MODIFYING AN OCCURRENCE IN THE CURRENT PLAN panel. In the list, you can enter these row commands:

- Inserts a blank row for a new operation
Adding and deleting operations

| R | Copies existing operations to be added as new |
| D | Deletes operations. |

**Note:** If you are using the advanced panels, you can perform these same tasks from the Occurrence menu on the OPERATION IN THE CURRENT PLAN panel. See 576 for more information.

When you generate a new row with the insert or repeat row command, the operation number is initially set to 000. This shows that the operation is not yet defined, and that you must assign a unique number to the operation. By default, the new operation is set up as a successor to the operation preceding it in the list.

When an operation is repeated with the repeat row command, all information except the operation number is copied to the new operation.

When an operation is deleted, Tivoli Workload Scheduler for z/OS automatically connects all its predecessors with its successors to maintain the consistency of the network of operations. You cannot delete an operation whose deletion splits the operation network of the application occurrence into several operation networks. If you need to delete an operation that would do this, you must first change the dependencies in the application so that the network is not split.

---

### Listing and browsing operations

To save you time and help you work more efficiently, Tivoli Workload Scheduler for z/OS allows you to create a list of all the operations in the current plan. In one panel, this list summarizes the high-level information of all operations such as workstation name, job name, and duration. From this list you can select an operation and, in a single panel, browse all the detailed information for that operation. At a glance, the color of the job name indicates the status of the operation (see “Operation status codes” on page 743).

This section explains how to use these list and browse functions and which tasks you can perform.

**Note:** To perform these list and browse tasks, you must use the advanced panels (see “Panels Style” on page 41). You can change from the basic style of panels (default) to the advanced style panels from the main menu by entering 0.8.

### Creating a list of operations

To create a list of operations, enter 5.3 from the main menu to open the SELECTING OPERATIONS panel. Leave all the criteria fields blank in this selection panel to see the entire list of operations in the current plan, or enter criteria to create a more specific operations list. The OPERATIONS IN THE CURRENT PLAN panel opens, as shown in Figure 242 on page 571.
Creating a list of operations

At a glance you can see all the operations in the current plan, and all the key information. The >> symbol indicates when there is more data available to the right. Use F11 to scroll right. Based on which view you are in (Compact, Full, or Job Detail), different amounts and types of data are available. The example shown in Figure 242 show input arrival time, dependencies, conditional dependences, operation status (S), extended status codes (X), and unexpected return codes (U). The color-coding of the job name, status, and unexpected return codes help you immediately see the operation status. For more information on the color codes, see "Operation status codes" on page 743. The Full and Job Detail views provide additional types of information such as planning details, high-level monitoring, and operation text description.

The OPERATIONS IN THE CURRENT PLAN panel has the following three menus that allows you to quickly and efficiently access other information and perform other tasks:

**Action**
- Allows you to:
  - Add an application to the current plan. This is the same as using the ADD primary command.
  - View the operations in the History database. This is the same as using the HIST primary command.
  - Sort a table (see "Sorting list output" on page 46). This is the same as using the SORT primary command.

**View**
- Allows you to choose the type and amount of operation data displayed. The name of the template the panel is using is displayed next to the name of the view. There is a different template for each type of view of each type of advanced panel. See "Specifying panel views" on page 44 for more information about the types of views.

**Help**
- Provides general help information and defines all the primary commands.
  Alternatively, you can enter HELP in the command line or press PF key 1 for the same information.

In the OPERATIONS IN THE CURRENT PLAN panel, enter B or S in the Row cmd column next to the operation you want to browse. Alternatively, you can enter a forward slash (/) in the Row cmd column of the operation and enter 1 in the TABLE ROW COMMANDS panel (see Figure 244 on page 573).
When you use the advanced panels (see "Panels Style" on page 41), you can browse, in a single panel, all the information for an operation in your current plan. This browse panel removes the need to go to several different panels to obtain data such as operation status, dependencies, operation times, operation options, and resources.

To open the browsing panel, create a list of your operations as described in "Creating a list of operations" on page 570. From the list, select the operation you want to browse by entering either of the following:

- B or S in the Row cmd column next to the operation.
- Forward slash (/) in the Row cmd column of the operation (as shown in Figure 243), and then selecting Browse in the TABLE ROW COMMANDS panel (see Figure 244 on page 573).

Figure 243. EQQMOPRV - Enter forward slash to select an operation
The browse panel, named OPERATION IN THE CURRENT PLAN, opens, as shown in Figure 245 on page 574. Scroll down to see all the operation details. The details are categorized into sections. Each section is prefixed by a code so that you can easily jump to the required section using the find command, without having to scroll through each section. For example, to quickly display the Dependencies section, type find DEP- at the command prompt.

**Figure 244. EQQSRLCP - Table Row Commands panel**

Choose the numeric option of the following commands

<table>
<thead>
<tr>
<th>Line 1 of 33</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Browse details (B/S)</td>
</tr>
<tr>
<td>2. Modify (M)</td>
</tr>
<tr>
<td>3. Browse JCL (BJ)</td>
</tr>
<tr>
<td>4. Edit JCL (J)</td>
</tr>
<tr>
<td>5. Browse joblog (L)</td>
</tr>
<tr>
<td>6. Browse operator instructions (O)</td>
</tr>
<tr>
<td>7. Recovery Info (RI)</td>
</tr>
<tr>
<td>8. Target Critical Job (TCJ)</td>
</tr>
<tr>
<td>9. Hold operation (MH)</td>
</tr>
<tr>
<td>10. Release held operation (MR)</td>
</tr>
<tr>
<td>11. NOP operation (NP)</td>
</tr>
<tr>
<td>12. UN-NOP the operation (UN)</td>
</tr>
<tr>
<td>13. EXECUTE operation (EX)</td>
</tr>
<tr>
<td>14. Kill (K)</td>
</tr>
<tr>
<td>15. Set NEXT logical status (N)</td>
</tr>
<tr>
<td>16. Set specific status (N-x)</td>
</tr>
<tr>
<td>17. Reset status (R)</td>
</tr>
<tr>
<td>18. Set the operation to complete (C)</td>
</tr>
<tr>
<td>19. Simple Job Restart (SJR)</td>
</tr>
<tr>
<td>20. Attempt automatic recovery (ARC)</td>
</tr>
<tr>
<td>21. Restart and Cleanup (RC)</td>
</tr>
<tr>
<td>22. Fast path to run Job Restart (FJR)</td>
</tr>
<tr>
<td>23. Fast path to run Step Restart (FSR)</td>
</tr>
<tr>
<td>24. Fast path to Start Cleanup (FSC)</td>
</tr>
<tr>
<td>25. Modify the occurrence (MOD)</td>
</tr>
<tr>
<td>26. Rerun the occurrence (RER)</td>
</tr>
<tr>
<td>27. Set the occurrence to complete (CMP)</td>
</tr>
<tr>
<td>28. Set to wait all ops. of occ. (WOC)</td>
</tr>
<tr>
<td>29. Delete Operation (D)</td>
</tr>
<tr>
<td>30. Delete Occurrence (DEL)</td>
</tr>
<tr>
<td>31. Complete the occurrence group (CG)</td>
</tr>
<tr>
<td>32. Remove from group (RG)</td>
</tr>
<tr>
<td>33. Delete the occurrence group (DG)</td>
</tr>
<tr>
<td>34. Reset bind information (BND)</td>
</tr>
</tbody>
</table>
Browsing operation information

Figure 245. EQQSOPSD - Operation in the Current Plan panel (showing main information)

Figure 246. EQQSOPSD - Operation in the Current Plan panel (showing dependencies)
In addition to displaying all the information for the operation, the OPERATION IN THE CURRENT PLAN panel has a menu that provides you with a quick and efficient way to access other data or perform other tasks, such as creating operation user fields. The following sections describe the menu options.

### Action

**Action** allows you to perform administration actions on the status of an operation. [Figure 248](#) shows a list of the possible actions.

<table>
<thead>
<tr>
<th>Action</th>
<th>Operation</th>
<th>Occurrence</th>
<th>View</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>MH</td>
<td>Hold</td>
<td>operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MR</td>
<td>Release</td>
<td>held</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NP</td>
<td>NOP</td>
<td>operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>UN</td>
<td>UN-NOP</td>
<td>the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EX</td>
<td>EXECUTE</td>
<td>operation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BND</td>
<td>Reset</td>
<td>bind</td>
<td></td>
<td></td>
</tr>
<tr>
<td>K</td>
<td>Kill</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>Set</td>
<td>NEXT logical</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N-x</td>
<td>Set</td>
<td>specific</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>Reset</td>
<td>status</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>Set</td>
<td>the</td>
<td></td>
<td></td>
</tr>
<tr>
<td>SJR</td>
<td>Simple</td>
<td>Job</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ARC</td>
<td>Attempt</td>
<td>automatic</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FJR</td>
<td>Fast</td>
<td>path</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSC</td>
<td>path</td>
<td>Start</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSR</td>
<td>path</td>
<td>Step</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

[Figure 248. List of administrative tasks available from the Action menu](#)

---

**Figure 247. EQQSOPSD - Operation in the Current Plan panel (showing user fields)**

In addition to displaying all the information for the operation, the OPERATION IN THE CURRENT PLAN panel has a menu that provides you with a quick and efficient way to access other data or perform other tasks, such as creating operation user fields. The following sections describe the menu options.

**Action** allows you to perform administration actions on the status of an operation. [Figure 248](#) shows a list of the possible actions.
**Browsing operation information**

Alternatively, you can enter any of the commands (in parentheses) in the panel command line or in the Row cmd column of the OPERATIONS IN A CURRENT PLAN panel. The Primary commands section of the Help menu explains these commands.

**Operation**

Allows you to perform actions and commands to modify or browse either the properties of an operation or its JCL. Figure 249 shows a list of the possible actions and commands.

<table>
<thead>
<tr>
<th>Action</th>
<th>Operation</th>
<th>Occurrence</th>
<th>View</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Modify (M)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>Edit JCL (E)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td>Browse JCL (JB)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 249. List of operation actions available from the Operation menu*

Alternatively, you can enter any of the commands (in parentheses) in the panel command line or in the Row cmd column of the OPERATIONS IN A CURRENT PLAN panel. The Primary commands section of the Help menu explains these commands.

**Occurrence**

Allows you to perform actions and commands to modify or browse the application that contains the operation. Figure 250 shows a list of the possible actions and commands.

<table>
<thead>
<tr>
<th>Action</th>
<th>Operation</th>
<th>Occurrence</th>
<th>View</th>
<th>Help</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Modify the occurrence (MOD)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Rerun the occurrence (RER)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td>Set the occurrence to complete (CMP)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td>Set to wait all ops. of occ. (WOC)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Complete the occurrence group (CG)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Remove from group (RG)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>Detailed application occurrence information (OCC)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Operations of the application occurrence (OPER)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Immediated external dependencies of the occurrence (OCCDEP)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Figure 250. List of operation actions available from the Occurrence menu*

Alternatively, you can enter any of the commands (in parentheses) in the panel command line or in the Row cmd column of the OPERATIONS IN A CURRENT PLAN panel. The Primary commands section of the Help menu explains these commands.

**View**

Allows you to change the view of the information for the operation. The full view provides more extensive data than the shortened, compact view. The name of the template the panel is using is displayed next to the name of the view. There is a different template for each type of view of each type of advanced panel.
Help

Provides general help and defines all the primary commands. Alternatively, you can enter HELP in the command line or press PF key 1 for the same information.

Modifying operations

The most direct way to modify an operation is to select option 3 (OPERATIONS) from the MODIFYING THE CURRENT PLAN panel (fast path 5.3) to open the Selecting operations panel. After specifying the criteria for selecting the operation, the MODIFYING OPERATIONS IN THE CURRENT PLAN panel opens, as shown in Figure 251.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>J</td>
<td>Edit JCL</td>
</tr>
<tr>
<td>M</td>
<td>Modify</td>
</tr>
<tr>
<td>B</td>
<td>Browse details</td>
</tr>
<tr>
<td>DEL</td>
<td>Delete Occurrence</td>
</tr>
<tr>
<td>MM</td>
<td>Man. HOLD</td>
</tr>
<tr>
<td>MR</td>
<td>Man. RELEASE oper</td>
</tr>
<tr>
<td>Q</td>
<td>Browse operator instructions</td>
</tr>
<tr>
<td>NP</td>
<td>NOP oper</td>
</tr>
<tr>
<td>UN</td>
<td>UN-NOP oper</td>
</tr>
<tr>
<td>EX</td>
<td>EXECUTE operation</td>
</tr>
<tr>
<td>D</td>
<td>Delete Oper</td>
</tr>
<tr>
<td>RG</td>
<td>Remove from group</td>
</tr>
<tr>
<td>L</td>
<td>Browse Joblog</td>
</tr>
<tr>
<td>K</td>
<td>Kill</td>
</tr>
<tr>
<td>RI</td>
<td>Recovery Info</td>
</tr>
<tr>
<td>RC</td>
<td>Restart and CleanUp</td>
</tr>
<tr>
<td>FJR</td>
<td>Fast path JR</td>
</tr>
<tr>
<td>TCJ</td>
<td>Target Critical Job</td>
</tr>
<tr>
<td>FSR</td>
<td>Fast path SR</td>
</tr>
<tr>
<td>BND</td>
<td>Reset bind information</td>
</tr>
<tr>
<td>FSC</td>
<td>Fast Path SC</td>
</tr>
</tbody>
</table>

Figure 251. EQQMOPRL - Modifying operations in the current plan (left part)

This panel lists the operations in the current plan that fulfill the list criteria you specified. Using row commands, you can perform the full range of MCP services on these operations.

Note: If you are using the advanced panels, you can perform these modification tasks from the:

- TABLE ROW COMMANDS panel (see Figure 244 on page 573).
- Action menu in the OPERATION IN THE CURRENT PLAN panel (see Figure 248 on page 575).
- Operation menu in the OPERATION IN THE CURRENT PLAN panel (see Figure 249 on page 576).

To display job options and expected duration time, scroll the list to the right:
Modifying operations

Select an operation that belongs to a critical path and enter the row command to browse information about the target critical job for the selected operation.

To perform actions on operations that are no longer in the current plan, use the HIST primary command.

Note: If you are using the advanced panels, refer to “Browsing operation information” on page 572 for information on how to browse operations.

Rerunning operations in the history database

You might want to rerun an operation that is not part of the current plan, because the daily plan removed it, as not requiring further processing (for details refer to “Removing data from NCP by running daily planning” on page 250). To do this, you use the history function.

The history function stores operation data in the history database, a DB2 database that contains operations from the most recently completed occurrences from a previous current plan. The HIST command lets you select operations from the history database for further processing, provided that the operations are defined on computer workstations. It is not possible to perform Restart and Cleanup (RC) for an operation that is retrieved from the history database. Message EQQM600E is issued if this is attempted.

Updating the history database

If you do not use the history function, the daily plan process creates a current plan that includes new occurrences as well as the non-completed occurrences from the...
old current plan. However, completed occurrences from the old plan do not appear in the new one. To save information about completed occurrences, you must use the history function.

When the history function is active, the daily plan process stores information about completed occurrences in the history database when it extends the current plan. The history update stores information about the most recently completed occurrence of an application unless the RETAINOPER keyword is specified in the BATCHOPT statement. If the RETAINOPER keyword is specified, the value of the keyword determines how many calendar days the operation is kept in the database before being removed or replaced by a newer occurrence of the operation. If RETAINOPER is not specified, the operation is kept in the history database indefinitely or until it is replaced by a more recent occurrence.

Before you can use the history function, a DB2 database must be installed with Tivoli Workload Scheduler for z/OS table definitions and the OPERHISTORY and DB2SYSTEM keywords must have been defined on the BATCHOPT and OPCOPTS statements for your subsystem. This is described in Planning and Installation and Customization and Tuning.

### Processing history operations

You can display a list of the history operations by selecting option 7 (Old Operations) on the main menu, or by entering the HIST command on any of these panels:

- MODIFYING OPERATIONS IN THE CURRENT PLAN (5.3)
- HANDLING OPERATIONS ENDED IN ERROR (5.4)
- READY LIST (4.1).

**Note:** If you are using the advanced panels, you can click **Action > History** from the menu on the OPERATIONS IN THE CURRENT PLAN panel. See Figure 248 on page 575 for more information.

**Note:** After you issue the HIST command, a confirmation panel displays. Enter Y to confirm the command.

After the HIST command is processed, the following panel displays:
Processing history operations

The history command might take a long time to process, depending on:

- The quantity of data that is to be retrieved from the DB2 tables.
- The quantity of archived CP data that is stored in the DB2 database.
- Whether you specified any filtering options. To considerably reduce the command processing time, specify some filtering criteria.

During the processing time, the GENERAL SERVICE task that manages the panel requests locks the CP, preventing other Tivoli Workload Scheduler for z/OS requests from accessing the CP. The maximum amount of data that can be returned from the DB2 database to the TSO dialog is set by the MAXHISTORYROWS parameter of the OPCOPTS initialization statement. To run the HIST command, ensure that you have at least READ access for the HIST fixed resource.

In the OPERATIONS HISTORY LIST panel, you see the row commands that can be issued against the operations. Two row commands, B and J, can be issued before you select an operation.

The B row command lets you browse the operation details. If you issue the command before selecting the operation, the details you see will be from the history database. In particular, occurrence token data is displayed as blank characters in hexadecimal format. If you have selected the operation and changed input arrival time before browsing, the details will show the new input arrival time and updated occurrence token data.

The J row command displays the job for editing. If it exists in the JS data set, the job you edit will be the one used when the occurrence was most recently run. If this copy does not exist, the job will be taken from the job library data set.

**Note:** If any scheduler or system variables are used in the JCL, the job submitted by the RESTART processing might be very different from the one originally submitted.

**Selecting a history operation**

When you select an operation, you add it to the current plan and make it available for processing with additional row commands.

---

**Figure 253. EQQHISTL - Operations history list**

The history command might take a long time to process, depending on:

- The quantity of data that is to be retrieved from the DB2 tables.
- The quantity of archived CP data that is stored in the DB2 database.
- Whether you specified any filtering options. To considerably reduce the command processing time, specify some filtering criteria.

During the processing time, the GENERAL SERVICE task that manages the panel requests locks the CP, preventing other Tivoli Workload Scheduler for z/OS requests from accessing the CP. The maximum amount of data that can be returned from the DB2 database to the TSO dialog is set by the MAXHISTORYROWS parameter of the OPCOPTS initialization statement. To run the HIST command, ensure that you have at least READ access for the HIST fixed resource.

In the OPERATIONS HISTORY LIST panel, you see the row commands that can be issued against the operations. Two row commands, B and J, can be issued before you select an operation.

The B row command lets you browse the operation details. If you issue the command before selecting the operation, the details you see will be from the history database. In particular, occurrence token data is displayed as blank characters in hexadecimal format. If you have selected the operation and changed input arrival time before browsing, the details will show the new input arrival time and updated occurrence token data.

The J row command displays the job for editing. If it exists in the JS data set, the job you edit will be the one used when the occurrence was most recently run. If this copy does not exist, the job will be taken from the job library data set.

**Note:** If any scheduler or system variables are used in the JCL, the job submitted by the RESTART processing might be very different from the one originally submitted.

**Selecting a history operation**

When you select an operation, you add it to the current plan and make it available for processing with additional row commands.
Selecting a history operation

You select a history operation with the S row command. Selecting an operation displays this panel, where you can change the input arrival or deadline time:

```
Application name : PAYDAILY
Operation number : 30
Input arrival:
  DATE ===> 03/07/01 Date in format YY/MM/DD
  TIME ===> 08.00 Time in format HH.MM
Deadline:
  DATE ===> 03/09/01 Date in format YY/MM/DD
  TIME ===> 08.00 Time in format HH.MM
```

Figure 254. EQQHIPUP - Specifying occurrence input arrival

After you have selected an operation and specified data on the Specifying OCCURRENCE INPUT ARRIVAL panel, you return to the previous panel. The input arrival time has been updated to reflect the time you specified, and the operation has been added to the current plan with the status AH (arriving, manually held).

You should note that no dependency information is stored in the history database. Operations that are added do not have any predecessors or successors even though these might be defined in the application description. Use the M command to add dependencies after you have selected an operation.

Select an operation before performing any of these row commands:

- DS The DS command deletes the operation from the current plan.
- L Browse the job log. The command retrieves and displays the job log if it exists for the selected operation.
- M This command is the same as the M command on the Modifying operations in the current plan. It lets you change operation details. If you want to run the history operations that you add to the current plan in any certain order, you must add the dependencies here.

Informing the scheduler of unplanned changes in resources

Sometimes a vital resource suddenly becomes unavailable because:
- A hardware error occurs on a tape or disk drive
- A processor is down
- A link is down
- Important staff are not available
- Resources must suddenly be reallocated.

When a vital resource becomes unavailable, you usually change the workstation definitions to prevent Tivoli Workload Scheduler for z/OS from starting operations in the wrong order or producing inaccurate schedules. If the resource is a special resource, refer to Chapter 28, “Monitoring special resources,” on page 605 for details of the RESOURCE MONITOR panel. But to change workstation parallel servers and the workstation fixed (R1 and R2) resources, use the MODIFYING THE CURRENT PLAN panel and select the WORK STATIONS option to display a
Informing the scheduler of unplanned changes in resources

list of all workstations. Enter the $S$ row command beside the workstation you want to select. You can then change the reporting attributes and the open interval information for the workstation. You can also change the alternate workstation for a workstation in this panel.

Any changes to open intervals that you make here remain in effect until they elapse or until you change them again in the panel. These changes always override the information calculated from the workstation descriptions when you extend the current plan.

Note: If you assign no parallel servers to an interval and specify that Tivoli Workload Scheduler for z/OS should control on servers for the workstation, Tivoli Workload Scheduler for z/OS will not submit any jobs for that workstation during the interval.

Keeping plans up-to-date

When you have changed the current plan, or when the duration of some work has varied significantly from its schedule, the planned times in the current schedule might be inaccurate. To make Tivoli Workload Scheduler for z/OS recalculate these times, taking the current information into account, replan the current plan. You can do this in two ways:

1. Enter the Replan panel from the DAILY PLANNING panel, by selecting the 1 (REPLAN) option on the PRODUCING OFC DAILY PLANS panel. The scheduler will then submit the replan job.
2. Define the replan job as an application in Tivoli Workload Scheduler for z/OS so you can schedule the replan as you would any other application, and Tivoli Workload Scheduler for z/OS will submit it automatically.

The REPLAN job removes all completed occurrences from the current plan panels.

Fault-tolerant workstations and replanning

To replan, you can configure Tivoli Workload Scheduler for z/OS to automatically send the Symphony file to IBM Tivoli Workload Scheduler after you modify the current plan. To send the new Symphony file automatically, set the CPUAUTOLNK parameter of the CPUREC initialization statement to ON.

Otherwise, you can control manually when the Symphony file is delivered to the distributed agent and set the CPUAUTOLNK parameter to OFF. You can improve scheduling performance in these ways:

- Reduce network congestion by controlling when the file is delivered
- Keep jobs from being scheduled on a workstation

For the syntax and description of CPUAUTOLNK, refer to Scheduling End-to-end with Fault Tolerance Capabilities.

When replanning, the distributed agent can receive and run the Symphony file and report job status back to IBM Tivoli Workload Scheduler for z/OS only if the workstation is active and linked. From the MODIFYING WORK STATIONS IN THE CURRENT PLAN panel, you can view and modify the status of the fault-tolerant workstation. Select option 5.5 from the main menu to display the panel:
Fault-tolerant workstations and replanning

In the S column, the possible values for status are the following:

A  Active. The workstation is running and scheduling its own plan.
O  Offline. The workstation is not running and scheduling its own plan.

The Fully Active (FA) column shows if all the destinations of the virtual workstation are active:

Y  All the destinations are active.
N  Some or all the destinations are not active.

In the L column, the possible values are the following:

F  Fully linked. The workstation is fully linked through its primary connection (either the domain manager or the OPCMASTER, if the fault-tolerant agent is a domain manager) and all secondary connections (the full-status fault-tolerant agents that are in the same domain). You can define a fully linked workstation only if ENABLESWITCHFT is set to YES in the TOPOLOGY statement. Fully linked workstations can be only fault-tolerant agents, domain managers, and standard agents.
L  Linked. Communication is established between Tivoli Workload Scheduler for z/OS and the workstation.
U  Unlinked. There is not an established connection to send and exchange events.

Job status includes information such as job started, job run, and job completed. When the new plan is produced, the distributed agents are stopped, and the scheduling on the agents is interrupted. The scheduling is resumed only when the new Symphony file is delivered.

When you send the Symphony file (CPUAUTOLNK=OFF) manually, you must manually link the fault-tolerant workstation. Perform the following steps:

1. In the MODIFYING WORK STATIONS IN THE CURRENT PLAN panel, use the S row command to select the fault-tolerant workstation.
Fault-tolerant workstations and replanning

2. In the MODIFYING A WORK STATION IN THE CURRENT PLAN panel, enter one of the following commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>L</td>
<td>To link the workstation</td>
</tr>
<tr>
<td>U</td>
<td>To unlink the workstation</td>
</tr>
<tr>
<td>S</td>
<td>To start workstation</td>
</tr>
<tr>
<td>P</td>
<td>To stop workstation</td>
</tr>
</tbody>
</table>

   Work station : FMAX COMPUTER AUTOMATIC FTW
   Type          : Computer FT Work Station
   Status        : Active
   Status link   : Linked

   Figure 256. EQQMWSTP - Modifying the status of a fault-tolerant workstation in the CP

3. The MODIFYING WORK STATION STATUS IN THE CURRENT PLAN panel displays the change.

   In some situations the Symphony file in IBM Tivoli Workload Scheduler is not created. For example, there might be a recovery from a disaster situation. In this example, the new current plan is created, but the Symphony file is not. In this situation, you can keep the current plan up-to-date by selecting the 5 (SYMPHONY RENEW) option from the PRODUCING OPC DAILY PLANS panel.

   The following are some other examples about the need of renewing the Symphony file:
   - You change the TCP/IP port number on a workstation.
   - You modify the CPUREC or DOMREC initialization statements.
   - You make changes to the job in the script library EQQSCLIB.
   - End-to-end scheduling with fault tolerance capabilities has configuration errors, for example the specified directories do not exist, the file system is full, or the workstation is not correctly defined.

Changing workstation availability

Before Tivoli Workload Scheduler for z/OS can start an operation, the workstation that it runs on must be available (open and active). If control on servers has been defined for the workstation, servers must be available, too. Select option 5.5 from the main menu to specify the times that a workstation is open for work.

Active and inactive computer workstations

To run work, a computer workstation must be active and open. A workstation is active if the controller can communicate with it. An active computer workstation can be open or closed. A closed computer workstation is not eligible to have work scheduled on it even if the controller can communicate with the workstation.
Workstation status can be changed dynamically (either manually, using Tivoli Workload Scheduler for z/OS panels, or automatically, in response to changes in your systems).

An inactive workstation can have one of the following statuses:

**FAILED**
The operating system has detected a failure on the system that the workstation is defined on. The workstation status is automatically set to ACTIVE when the system is available again, provided that the controller and tracker stopped correctly and that the WFAILURE and WSOFFLINE keywords were correctly defined in the JTOPTS initialization statement. You can manually set a computer workstation to status FAILED, using the panels. The WSTAT command or the EQUSIN subroutine can be used to report FAILED status for a workstation.

**OFFLINE**
Communication is lost between the controller and tracker on the system that the workstation represents. This might be because the tracker is not yet started or because it ended abnormally. The scheduler does not take any reroute or restart actions until the time specified for the OFFDELAY initialization statement elapses. The status is automatically set to ACTIVE status when the tracker is started. The WSTAT command or the EQUSIN subroutine can be used to report OFFLINE status for a workstation.

**UNKNOWN**
The scheduler has detected that the workstation is inactive, but no other diagnostic information is available. You can manually reset the workstation to ACTIVE, using the panels, the WSTAT command, or the EQUSIN subroutine, or Tivoli Workload Scheduler for z/OS can do this automatically. A workstation normally connected via XCF, NCF, or TCP/IP will be reported in UNKNOWN status, if the XCF, NCF, or TCP/IP task is not started for the controller. Workstations that specify a user-defined destination ID are set to UNKNOWN status when they are first added to the current plan.

In addition to the workstation status, text that describes the status value can be displayed by Tivoli Workload Scheduler for z/OS in the current plan panel. The possible status descriptions are:

**Waiting for manual intervention**
The controller has received an automatic online notification for a workstation previously in offline or failed status. The AVAIL action parameter value specified by the WFAILURE or WSOFFLINE initialization statement specifies manual activation. If the workstation should be receiving work from the controller, vary the workstation status to ACTIVE from the MODIFYING THE CURRENT PLAN panel.

**Waiting for connection**
The status of the workstation has been set to ACTIVE, and the controller is waiting for the corresponding tracker to communicate. No operations are started on the workstation until the tracker and the controller synchronize the correct submit position. This condition can identify an error in Tivoli Workload Scheduler for z/OS configuration. Perhaps the tracker events are not being communicated to the controller. Check that the tracker has either EWSEQNO(n) specified in the EWTROPTS or that an event reader has been started by specifying ERDRTASK(n) in the OPCOPTS for the tracker.
Active and inactive computer workstations

Offline actions pending
An offline notification has been received. The time Tivoli Workload Scheduler for z/OS has been instructed to wait before performing offline actions has not yet been reached.

Status was set manually
The status of the workstation has been set by a panel user.

Status set by WSSTAT
The workstation status was set as the result of a workstation availability event generated by the WSSTAT TSO command or by the EQQUSIN subroutine.

Status set by EQQUX009
The workstation status was set to the current value from the return code issued by the operation-initiation exit, EQQUX009.

Redirecting work to alternate workstations
The scheduler supports the redirecting of work from one workstation to another. If a workstation becomes inactive, you can specify, for each of its open intervals, an alternate workstation where work will be redirected. At any time, you can manually redirect the work to the workstation by using the MODIFYING THE CURRENT PLAN panel (option 5.5 from the main menu). This function does not apply to either fault-tolerant or virtual workstations.

When you select a workstation to be modified, the current status of the workstation is presented (see Figure 257). In most cases, the workstation is active and no further explanation is required. If the workstation status has been modified manually or if an exception has occurred, status text that describes the current state is also displayed.

If you make changes on the MODIFYING A WORK STATION IN THE CURRENT PLAN panel, the changes take effect when you press PF3 (End). Press PF12 (Cancel) if you do not want to make the typed modifications.

If you need to manually change the status of a workstation, enter V from the MODIFYING A WORK STATION IN THE CURRENT PLAN panel, and the following panel is displayed.

---

**EQQMWSRP -- MODIFYING A WORK STATION IN THE CURRENT PLAN**

Command ==> 

Enter/change data below:
Enter the O command above to modify open time intervals
Enter the V command above to vary workstation status

<table>
<thead>
<tr>
<th>Workstation</th>
<th>CPU1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Computer Active</td>
</tr>
<tr>
<td>Status</td>
<td>Active</td>
</tr>
</tbody>
</table>

REPORTING ATTR ===> A A automatic, S manual start and complete C complete only or N non reporting

CONTROL ON SERVERS ===> Y Control on parallel servers, Y or N
CONTROL ON R1 ===> N Control on critical resource R1, Y or N
CONTROL ON R2 ===> N Control on critical resource R2, Y or N

Figure 257. EQQMWSRP - Modifying a workstation in the current plan
Redirecting work to alternate workstations

The values you enter on the MODIFYING WORK STATION STATUS IN THE CURRENT PLAN panel determine the effect on the workload. If you modify the workstation status to failed or offline, you must specify what should happen to operations that are already started on the workstation and whether Tivoli Workload Scheduler for z/OS should reroute operations to an alternate workstation.

Notes:
1. Be careful if you specify that operations are to be restarted on an alternate workstation. They might still be running on the original workstation. The scheduler does not know this. You must decide if they can be safely started on the alternate workstation. If you set them to error, you can rerun them after taking any necessary recovery actions; this is a safer option.
2. When you issue END from this panel, the status change takes effect immediately, even if you subsequently issue CANCEL on panel EQQMWSRP.
3. When defining your operations, you can specify that a particular operation should not be redirected under any circumstances.

Figure 259 on page 588 shows how you can adjust the workstation fixed resources and parallel servers for a workstation, and specify an alternate workstation.
Redirecting work to alternate workstations

You cannot use this panel to make the workstation unavailable: use the panel in Figure 258 on page 587 instead. In practice, setting the number of parallel servers to zero means that operations cannot run on the workstation.

If you add or modify an open interval, ensure that the begin time is the same or later than the current time, otherwise the update is not made.

If you alter the interval details, the altered interval is never replaced by daily planning extend and replan functions. Consider this sequence of events:
1. You modify the number the number of parallel servers between 18.00 and 23.59 on 03/03/10 to 25.
2. You change the database to close CPU1 at 20.00 on 03/03/10.
3. You extend the plan at 15.00.

The result is that the workstation stays open past 20.00, because there is a modified interval that lasts until 23.59. The scheduler considers that it is modified even if you change the number back to the original value.

Changing the global status for a virtual workstation

If you need to manually change the global status of a virtual workstation, enter the V row command from the MODIFYING A WORK STATION IN THE CURRENT PLAN panel. The following panel is displayed:

---

**EQQMWSOL --------- MODIFYING OPEN TIME INTERVALS IN THE CP - ROW 1 TO 3 OF 3**

Command ==> Scroll ==> PAGE

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

Work station : CPU1 Main JES processor
Control on servers : Yes
Control on R1 : No
Control on R2 : No

<table>
<thead>
<tr>
<th>Row</th>
<th>From YY/MM/DD HH.MM</th>
<th>To YY/MM/DD HH.MM</th>
<th>Planned ps R1 R2 AltWS</th>
<th>Modified ps R1 R2 AltWS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>03/03/10 00.00</td>
<td>03/03/10 05.59</td>
<td>99 99 99</td>
<td>20 99 99</td>
</tr>
<tr>
<td>2</td>
<td>03/03/10 06.00</td>
<td>03/03/10 17.59</td>
<td>99 99 99</td>
<td>10 99 99 CPU2</td>
</tr>
<tr>
<td>3</td>
<td>03/03/10 18.00</td>
<td>03/03/10 23.59</td>
<td>99 99 99</td>
<td>20 99 99</td>
</tr>
</tbody>
</table>

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

---

**Figure 259. EQQMWSOL - Modifying open time intervals in the CP**

**You cannot use this panel to make the workstation unavailable: use the panel in Figure 258 on page 587 instead.** In practice, setting the number of parallel servers to zero means that operations cannot run on the workstation.

If you add or modify an open interval, ensure that the begin time is the same or later than the current time, otherwise the update is not made.

If you alter the interval details, the altered interval is never replaced by daily planning extend and replan functions. Consider this sequence of events:
1. You modify the number the number of parallel servers between 18.00 and 23.59 on 03/03/10 to 25.
2. You change the database to close CPU1 at 20.00 on 03/03/10.
3. You extend the plan at 15.00.

The result is that the workstation stays open past 20.00, because there is a modified interval that lasts until 23.59. The scheduler considers that it is modified even if you change the number back to the original value.
Changing availability for a virtual destination

If you need to manually change the status of a virtual destination, follow these steps:

1. Enter the S row command from the MODIFYING A WORK STATION IN THE CURRENT PLAN panel. The following panel is displayed:

   **Figure 260. EQQMWS1P - Modifying workstation status in the current plan**

   **Changing availability for a virtual destination**

   If you need to manually change the status of a virtual destination, follow these steps:

   1. Enter the S row command from the MODIFYING A WORK STATION IN THE CURRENT PLAN panel. The following panel is displayed:

   **Figure 260. EQQMWS1P - Modifying workstation status in the current plan**

2. Enter the S row command. The following panel is displayed:

   **Figure 261. EQQMWDDES - Modifying a virtual workstation destination status in the CP**

3. Enter the V command. The following panel is displayed:

   **Figure 262. EQQMWSRV - Modifying a virtual workstation destination status in the CP**

Chapter 26. Updating the current plan  589
Changing availability for a virtual destination

The values you enter on this panel affect the workload. If you change the
destination status to F or O, specify also an action for the operations that are
already started on the workstation, by using the STARTED OPERATIONS field.

A status change on a virtual destination might cause a change in the global
status of the virtual workstation it refers to.

Use this panel to make the virtual destination unavailable. You can set the
global status to:

**Offline**
All the destinations are offline, meaning that communication is lost
between the controller and the tracker on the system that the
destination represents.

**Failed**
All the destinations are in failed status, meaning that the operating
system detected a failure on the system that the destination represents.

**Active**
At least one destination is active.

**Inactive**
Some destinations are offline and others are in failed status.

**Note:** When you enter the END command on this panel, the status change is
effective immediately, even if you enter the CANCEL command on the
EQQMWSRV panel.

To modify open time intervals and adjust the destination fixed resources and
parallel servers, enter the O command on the EQQMWSRV panel. The
following panel is displayed:
To indicate that no operation can run on the selected destination, set the number of parallel servers to 0.

If you alter the interval details, the altered interval is never replaced by daily planning extend and re-plan functions. For example, consider the following sequence of events:

a. You modify the number of parallel servers between 18.00 and 23.59 on 07/06/07 to 25.

b. You change the database to close VDEST2 at 20.00 on 07/06/07.

c. You extend the plan at 15.00.

The result is that the destination stays open after 20.00, because there is a modified interval that lasts until 23.59. The scheduler considers that it is modified even if you change the number back to the original value.

### Browsing system information

From the MODIFYING WORK STATIONS IN THE CURRENT PLAN panel, you can browse the system information for the destination of a computer workstation by entering the I row command. This option does not work for fault-tolerant or virtual workstations. However, a non-local workstation can be browsed only if it communicates through XCF, NCF, TCP/IP, or the submit/release data set. To browse the system information for the destinations of a virtual workstation, enter the S row command. A list of its destinations is shown. From this list, enter the I row command on the specific destination to retrieve system information.

### Killing operations on standard agents

You can use the kill command in the MODIFYING WORK STATIONS IN THE CURRENT PLAN (EQQMOPRL) panel to kill jobs running on standard agents that are directly connected to the end-to-end server (OPCMASTER).

The same command can be used in the Operation Recovery Info panel (EQQREINP, accessible by using the RI row command in panel EQQMOPRL) to kill recovery jobs.

**Note:** If you are using the advanced panels, you can also kill jobs by selecting:

- Kill in the TABLE ROW COMMANDS panel (see Figure 244 on page 573).
Killing operations on standard agents

- Kill in the Action menu in the OPERATIONS IN THE CURRENT PLAN panel (see Figure 248 on page 575).

The kill command can be issued only on STARTED jobs or recovery jobs that are in the EXECUTING status, so that their JOBID is known. The JOBID is required to identify the operation that is to be killed.

You can issue the kill command on the same job as many times as you need to.
Chapter 27. Handling operations that end in error

This chapter explains how to handle failed operations. You use the ended-in-error list, which is part of the MODIFYING THE CURRENT PLAN (MCP) panel, which was introduced in Chapter 26, “Updating the current plan,” on page 547.

When an operation that is under the control of Tivoli Workload Scheduler for z/OS does not complete successfully, it is reported as having ended in error. The scheduler automatically reports failures of jobs or started tasks, and you can manually report failures of manual operations via Tivoli Workload Scheduler for z/OS panels. The scheduler keeps a list of all operations that have ended in error. You can display this list using either the MODIFYING THE CURRENT PLAN panel, which lets you take some action on the operation, or the QCP panel, where you can only browse the operation.

When you are using the HANDLING OPERATIONS ENDED IN ERROR panel (the ended-in-error list), you have access to a wide range of options, which include the following:
- Restarting the operation
- Completing or deleting the operation or the occurrence
- Requesting automatic recovery for the operation
- Complete or delete an occurrence group
- Initiating, discarding, or displaying restart and cleanup actions

**Note:** Because in some cases EQQCLEAN might delete a data set by mistake, it is recommended that you protect critical data sets from deletion by using either the RCLOPTS parameters (DDPROT, DDPRMEM, DSNPROT, DSNPRMEM) or the EQQUXCAT exit.

Figure 265 shows an example of an ended-in-error list, showing the PAYDAILY job having ended with an abend.

```
<table>
<thead>
<tr>
<th>Cmd</th>
<th>Ended time</th>
<th>Application</th>
<th>ws no.</th>
<th>Jobname</th>
<th>Errc</th>
</tr>
</thead>
<tbody>
<tr>
<td>95/05/01 19.30</td>
<td>PAYDAILY</td>
<td>CPU1</td>
<td>20</td>
<td>PAYDAILY</td>
<td>SOC4</td>
</tr>
</tbody>
</table>
```

Figure 265. EQQMEP1L- Handling operations ended in error (left part)

Sometimes you might need to restart an operation even though it has completed successfully. See “Restarting an occurrence from the beginning” on page 562 for information on restarting operations that have not failed.
Displaying the ended-in-error list for action

To display the ended-in-error list for action, select option 4 (ERROR HANDLING) from the MODIFYING THE CURRENT PLAN panel. The SPECIFYING ENDED IN ERROR LIST CRITERIA panel is displayed. Specify selection criteria to display only the list items you want to examine.

You can customize the layout of the ended-in-error list to suit your needs, either by selecting one of the installation-defined layouts or by building your own layout. This is described in the following sections.

After you select an error list layout and display your error list, the list is updated with the latest information when you press Enter or any of the PF keys.

Selecting an ended-in-error list layout

When you select the ended-in-error list, you can also define a layout to use for the list. About 90 fields can be displayed in an ended-in-error list (see Appendix E, “Fields displayed in ready and error lists,” on page 749). Not all of these can be presented at once in the same ended-in-error list. But you can specify two panels for your layout and switch between these panels using the LEFT and RIGHT commands. Information on the fields that are to be displayed, and the order in which they should be displayed, is kept in a named ended-in-error list layout. There might be many different layouts in your installation.

If you have not specified a layout, Tivoli Workload Scheduler for z/OS displays a list of existing layouts. You can then select a layout from this list. If you want to change the layout of the ended-in-error list, enter the name of another layout in the corresponding field in the ended-in-error list, and Tivoli Workload Scheduler for z/OS will display the ended-in-error list using the layout you specified. You can get a list of all the ended-in-error list layouts by entering an asterisk (*) in the LAYOUT ID field on the SPECIFYING ENDED IN ERROR LIST CRITERIA panel:

```
Figure 266. EQQMERRP - Specifying ended in error list criteria
```

Creating your own ended-in-error list layout

Ended-in-error list layouts are kept in two ISPF tables: one for your own use and one that contains the installation-defined layouts. Your own version of a layout
Creating your own ended-in-error list layout

overrides the installation layout for your user ID. To change an ended-in-error list layout, select the DEFINE EL option on the MODIFYING THE CURRENT PLAN panel. The scheduler then displays a list of both your own layouts and any installation-defined layouts. From this list, you can create new layouts or select an existing layout to delete, copy, modify, or browse. Modified installation-defined layouts are stored in your own library. You can delete your own layout versions, but you cannot delete an installation-defined layout.

When you modify or create a layout, Tivoli Workload Scheduler for z/OS presents a list of the available ended-in-error list fields. See Appendix F, “Fields displayed in ready and error lists,” on page 749. From this list, you select and order the fields to be contained in the ended-in-error list layout. You can also specify whether the field should be highlighted when Tivoli Workload Scheduler for z/OS displays it. All the items you select for display are placed as near the top of the selection list as possible.
Creating your own ended-in-error list layout

Table: EQQELYCL - Creating an Error List Layout

<table>
<thead>
<tr>
<th>Column title</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PRE</td>
<td>Number of predecessor operations</td>
</tr>
<tr>
<td>PSE</td>
<td>No. of parallel servers req. by the op.</td>
</tr>
<tr>
<td>UPR</td>
<td>Number of uncompleted predecessor op.</td>
</tr>
<tr>
<td>R1E</td>
<td>No. of 1st WS resources req. by the op.</td>
</tr>
<tr>
<td>R2E</td>
<td>No. of 2nd WS resources req. by the op.</td>
</tr>
<tr>
<td>SRE</td>
<td>Special resources referenced by the op.</td>
</tr>
<tr>
<td>SUE</td>
<td>Number of successor operations</td>
</tr>
<tr>
<td>Arrived</td>
<td>Operation arrival date actual if arrived</td>
</tr>
<tr>
<td>Ard</td>
<td>Operation arrival day, actual if arrived</td>
</tr>
<tr>
<td>time</td>
<td>Operation arrival time actual if arrived</td>
</tr>
</tbody>
</table>

Figure 268. EQQELYCL - Creating an error list layout

Getting rerun or recovery instructions

When an operation fails, the person responsible for rerunning the operation often needs rerun instructions. These instructions can reside either in the job or in the operator instruction database.

To display the instructions for the failing operations, enter the O row command beside those operations on the HANDLING OPERATIONS ENDED IN ERROR panel. To see the job, enter J in the row command field.

Completing an ended-in-error operation

Sometimes you might consider that an operation has ended successfully, even though it is listed as ended-in-error. To set its status to C (complete), enter the C row command beside that operation in the HANDLING OPERATIONS ENDED IN ERROR panel.
Modifying a job that has failed

If the job or started task has failed, you might be able to correct the error by changing the job. You can do this from the panel by entering the J row command beside that operation in the HANDLING OPERATIONS ENDED IN ERROR panel. The scheduler uses the ISPF/PDF editor to display the job for you to edit. In the case of a fault-tolerant workstation, you cannot edit the JCL.

The scheduler keeps a separate copy of the job for each run. The job you edit using the J row command is always the job that was used during the failing run of the job (unless you have edited it since). When you end the edit with the END command, Tivoli Workload Scheduler for z/OS saves the modified job in the JCL repository and re-displays the ended-in-error list.

If the job contained resolved variables and you want to use the same variables again with new values, perhaps from another JCL variable table, delete the job in the ISPF editor and enter the END command. This forces Tivoli Workload Scheduler for z/OS to use a new copy of the job from EQQJBLIB.

JCL variable values are stored in the JCL record and will be used again if referenced in the current copy of the job. This is useful in situations in which you want to set promptable variables in the READY LIST panel, and then refer to them during the submit phase, probably in a COMP statement. For more information about job tailoring, refer to Chapter 22, “Job tailoring,” on page 441.

Note: Editing and saving the job does not restart the operation.

Restarting ended-in-error operations

To restart an operation that ended in error:
1. Correct the cause of the error (by modifying the job, for example)
2. Enter the SJR (Simple Job Restart) row command beside the operation in the HANDLING OPERATIONS ENDED IN ERROR panel. Using SJR the operation status is set to Ready. You can use SJR if you do not need specific restart and cleanup actions, such as cleanup or step restart. In this way, you can avoid installing and customizing the data store.

Restarting ended-in-error operations managing cleanup action

To restart an operation that ended in error performing a cleanup action, enter the RC (restart and cleanup) row command beside that operation in the HANDLING OPERATIONS ENDED IN ERROR panel. The following panel is displayed:
### Restarting ended-in-error operations managing cleanup action

**EQQRCLSE------------- OPERATION RESTART AND CLEANUP --------------------------**

<table>
<thead>
<tr>
<th>Option ====&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application : PAYDAILY 00/12/13 20.36</td>
</tr>
<tr>
<td>Operation : CPUA 1</td>
</tr>
<tr>
<td>Jobname and jobid : PAYDAILY JOB01475</td>
</tr>
<tr>
<td>Expanded JCL : N</td>
</tr>
</tbody>
</table>

**Clean Up Result :**

**Edit JCL ====> N Edit JCL before Restart (SR/JR only)**

Select one of the following:

1. **STEP RESTART** Request a Step Restart
2. **JOB RESTART** Request a Job Restart
3. **START CLEANUP** Request Cleanup
4. **DISPLAY CLEANUP** Display Cleanup result

---

**Figure 269. EQQRCLSE - Operation restart and cleanup**

**Note:** In the case of a fault-tolerant workstation, you cannot restart the operation.

From this panel you choose among the restart and cleanup functions. For step restart, see "Restarting an operation from a certain step" on page 600. Also, refer to Chapter 17, "Restart and cleanup," on page 313 for more information about setting cleanup options.

For job restart or step restart, perform the following steps:

1. Select job restart or step restart for an operation. If cleanup is required, confirm that the cleanup type is other than none.
2. For step restart only, the Step RESTART SELECTION panel is displayed. Enter the GO command, and the MODIFYING THE CLEANUP ACTION panel is displayed.
3. In the MODIFYING THE CLEANUP ACTIONS panel, enter the GO command. If you specified Y (yes) for the Edit JCL option in the OPERATION RESTART AND CLEANUP panel, you can correct the JCL. The CONFIRM RESTART panel, shown in Figure 270 on page 599, is displayed.

The start cleanup options can be requested only for an operation with the cleanup type set to automatic, immediate, or manual. You can also issue a job restart or step restart to perform the cleanup together with the job run instead of separately before the run.
When you restart a job, you can give a reason, which is written to the track log for audit purposes.

When you confirm the restart, Tivoli Workload Scheduler for z/OS:
1. Resets the operation status to ready.
2. Initiates any cleanup actions.
3. Resubmits the operation when all conditions are met.

If you cancel out of restart and cleanup, the statuses are reset to their original values.

**Taking action at the occurrence level**

If the operation cannot be restarted because you need to change the application occurrence, perform the following steps:

1. In the MODIFYING OCCURRENCES IN THE CURRENT PLAN panel (option 5.2), enter the M row command beside the occurrence. The MODIFYING AN OCCURRENCE IN THE PLAN panel is displayed.
2. Change the occurrence, as necessary.
3. In the MODIFYING OCCURRENCES IN THE CURRENT PLAN panel, enter the R row command to rerun the occurrence. The RERUNNING AN OCCURRENCE IN THE CURRENT PLAN panel is displayed.
4. Enter the S row command to set the start point.
5. Issue the RC command, if needed.

Alternatively, you can directly rerun the occurrence without making modifications. Use the R row command in the MODIFYING OCCURRENCES IN THE CURRENT PLAN panel, performing steps 5 to 5.

**Handling operations with the OSEQ error code**

If Tivoli Workload Scheduler for z/OS detects the running of a job with the same name as one in the plan, but out of sequence, it gives the operation in the plan an
Handling operations with the OSEQ error code

OSEQ error code, which prevents its running. If the planned operation should be run, use the SJR row command to reset it to waiting status. If the planned operation should not run, delete it.

Restarting an operation from a certain step

If the job log is available for the operation, you can view the STEP RESTART SELECTION LIST panel, shown in Figure 271.

To restart an operation from a certain step, select the steps that you want to include in the restart:

1. In the Step RESTART SELECTION LIST panel, use the S row command to specify the first step and the E row command for the last step to be executed in the rerun. S defaults to the best step.
2. To have only one step in the rerun, specify S for that step and exclude (X) all the other steps.
3. To include the step if it is inside the restart range, specify I for that step.
4. To set the step completion code with a desired value, use the F row command.

The codes shown in the Rest column (the ability to restart) indicate whether the related steps can be restarted and also can assist you in deciding which step is best. Their meanings are as follows:

Table 37. Codes for the ability to restart a step

<table>
<thead>
<tr>
<th>Ability-to-Restart Code</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Y (yes)</td>
<td>The step can be selected as the start step</td>
</tr>
<tr>
<td>N (no)</td>
<td>The step cannot be selected as the start step</td>
</tr>
<tr>
<td>B (best)</td>
<td>The suggested best step to restart</td>
</tr>
</tbody>
</table>
Restarting an operation from a certain step

If you decide a step-level restart is not needed after viewing the job, enter the CANCEL command and select option 2 (JOB RESTART) to initiate restart and cleanup actions at the job level.

When you have selected the steps, you see the CONFIRM RESTART panel, shown in Figure 270 on page 599.

Using cleanup options

The cleanup options help you recover z/OS jobs and started tasks by deleting or uncataloging data sets that were created in the failing job, and cataloging data sets that were uncataloged. Without cataloging or deleting data sets, a rerun will fail with an error message saying that a data set already exists, is already cataloged, or is not found.

The scheduler restores the catalog status of data sets from the restart step, up to and including the failing step. It cannot restore the data set itself. The scheduler deletes or uncatalogs the data set if it was created and catalogs it if it was uncataloged.

Restart and cleanup works like this:

1. The scheduler retrieves the job log. Refer to "Selecting a history operation" on page 580 for a full description of job log retrieval.
2. It makes a list of actions to restore the data set status. All data sets defined with DISP=NEW are marked as eligible for potential deletion. The operator can refuse the deletion if this action is not in line with the logic of the job to be restarted.
3. Depending on the kind of restart and cleanup requested (job restart, step restart, or standalone cleanup) and the cleanup type set (automatic, immediate, or manual), the cleanup is completed in one of the following ways:
   • At the time the job is submitted by the controller, a step is added to the JCL before the existing steps.
     The name of the added step is EQQCLEAN. It uses as input the list of data sets to be deleted, cataloged, or uncataloged. This list is obtained from the history of previous runs. To get this history, the controller requests the job log from the data store and then merges the history in the local repository of the controller.
   • A single-step job, created by the controller, is run before the job run.

You can modify or browse cleanup information from a number of places in Tivoli Workload Scheduler for z/OS. In most cases, you will probably be using the HANDLING OPERATIONS ENDED IN ERROR panel to browse, modify, execute, or discard cleanup actions. When you select a job restart, the MODIFYING CLEANUP ACTIONS panel, shown in Figure 272 on page 602, is displayed.
Using cleanup options

The scheduler maintains status information to describe the progress of the cleanup action. If you use RC, include the CLN_Ty and CLN_Re status fields in your ended-in-error layout. After the request for cleanup, you also can view the cleanup status for processes not yet completed on the OPERATION RESTART AND CLEANUP panel. The panel displays the status, dynamically updated at each step of the pending process. The statuses can be summarized in the following way:

- Collecting job log data
- JCL tailoring in progress
- Process started

When the cleanup action ends, the Cleanup Result field displays Completed or Ended in Error.

To view the result of a cleanup action, select option 4 (DISPLAY CLEANUP) on the OPERATION AND RESTART CLEANUP panel. The VIEW CLEANUP RESULTS panel is displayed; it provides the following status information:

- Whether the data set cleanup was completed or ended-in-error
- The job name and job ID that performed the cleanup actions
- Whether Tivoli Workload Scheduler for z/OS removed the data set from the catalog, uncataloged the data set, or cataloged the data set

Specifying automatic restart for operations that fail

The automatic recovery function lets you specify, for each job or started task, automatic recovery actions for specific failures. You specify recovery actions by including Tivoli Workload Scheduler for z/OS recovery statements in the job. Automatic recovery parameters that refer to job steps are supported only for jobs that execute on z/OS systems. For detailed information, refer to "Rerunning operations in the history database" on page 578.

If automatic job recovery fails or if a job does not have recovery statements when it fails, you can modify the job and create or change the recovery statements. Use the J row command on the HANDLING OPERATIONS ENDED IN ERROR panel to edit the job. The EDITING JCL FOR AN MVS JOB panel in Figure 273 on page 603 shows some recovery statements (1 to 3). The beginning of the recovery statement 1 has been changed from /*% to //*> showing that action has already been taken on the statement.
Specifying automatic restart for operations that fail

You can add further recovery directives to the job, save the job, and initiate recovery by entering the ARC row command beside the failing operation on the HANDLING OPERATIONS ENDED IN ERROR panel.

You can also use the ARC command to initiate recovery outside the hours specified for automatic recovery.

If a request to start automatic recovery is received while a restart and cleanup action is still in progress, the request is denied.

---

Figure 273. An example of RECOVER statements

You can add further recovery directives to the job, save the job, and initiate recovery by entering the ARC row command beside the failing operation on the HANDLING OPERATIONS ENDED IN ERROR panel.

You can also use the ARC command to initiate recovery outside the hours specified for automatic recovery.

If a request to start automatic recovery is received while a restart and cleanup action is still in progress, the request is denied.
Specifying automatic restart for operations that fail
Chapter 28. Monitoring special resources

This chapter shows you how to monitor and change special resources. You can do this using option 7 (SPECRES) in the MODIFYING THE CURRENT PLAN (MCP) panel. This part of the MCP is called the SPECIAL RESOURCE MONITOR.

Understanding special resources

You can use special resources to represent any type of limited resource, such as tape drives, communication lines, or a database. The administrator defines resources using the SPECIAL RESOURCE DEFINITION panel. The SPECIAL RESOURCE DEFINITION panel updates the resource database, which has these details of each resource:

- **Name**
  - The resource name. It can be up to 44 characters.

- **Availability**
  - Available (Y) or not available (N).

- **Connected workstations**
  - A list of the workstations where operations can allocate the resource.

- **Quantity**
  - It can be from 1 to 999999.

- **Used for**
  - Specifies how Tivoli Workload Scheduler for z/OS is to use the special resource. Allowed values are:
    - **P** Planning
    - **C** Control
    - **B** Both control and planning
    - **N** Neither control nor planning

- **On-error action**
  - Specifies the action to take if the operation that allocates this resource ends in error (and does not have an overriding keep-on-error specification in the operation definition). Possible values are:
    - **F** Free all
    - **FX** Free exclusively-held resources
    - **FS** Free shared resources
    - **K** Keep all

  Tivoli Workload Scheduler for z/OS uses the attribute specified at operation level first. If this is blank, it uses the attribute specified in the resource database. If this is also blank, it uses the ONERROR keyword of the RESOPTS statement.

- **On Complete**
  - Specifies the value to which the global availability is reset after the operation that uses the resource completes. It can be one of the following:
    - **Y** Sets the global availability to Yes.
    - **N** Sets the global availability to No.
Understanding special resources

Set the global availability to blank.

Blank

Uses the system default, according to the following order:
1. The On Complete value set at operation definition level, if not blank.
2. The On Complete value set at special resource definition level, if not blank.
3. The ONCOMPLETE or DYNONCOMPLETE keyword value, respectively set for the not dynamically added resources or the dynamically added resources, in all the other cases.

Max Usage Limit

Specifies after how many allocations of the special resource, its availability is reset according to the value set for Max Usage Type. An internal usage counter is increased each time an operation allocates the resource. When the internal counter reaches the Max Usage Limit value, the global availability is reset to the value specified with Max Usage Type.

The default value is 0, meaning that no usage counter check is done.

Max Usage Type

Specifies the value to which the global availability is reset when the Max Usage Limit is reached. This value is valid only if the Max Usage Limit is different from 0. Possible values are:

Y Sets the global availability to Yes.

N Sets the global availability to No.

R Sets the global availability to blank.

The quantity, availability, and list of workstations can vary with time. The administrator can associate different intervals with the resource.

The administrator also specifies, for each operation, the special resources that it uses: how (shared or exclusive), how many (quantity), and the on-error attribute.

The long-term plan is built without taking the special resources into account, but when you extend the current plan, it schedules operations taking account of all the special resources that are used for planning though the daily planning program does not take manually changed availability, quantity, and deviation into account. This is because they are usually assumed to be temporary changes and the values will be reset to the normal values when, for example, an engineer has repaired a tape unit.

If a special resource is needed in the current plan, Tivoli Workload Scheduler for z/OS copies the details from the RD database and stores them in the current plan extension data set. These details include the information from the resource database, but also have these overriding (global) fields:

Quantity

1 through 999999 or blank. If you specify a quantity, this overrides the scheduled quantity from the database.

Availability

Y or N or blank. If you specify an availability, this overrides the scheduled availability from the database.
Deviation
-999999 through 999999 or blank. You use the deviation to make a temporary alteration to the scheduled quantity.

You can change the quantity and availability of a special resource, and the connected workstations, using the Special Resource Monitor, which is described in this chapter. You might need to make a resource unavailable (to prevent the submission of all jobs needing a database, if some corruption is suspected), alter the quantity by specifying a deviation (if a tape drive is broken), or change the list of connected workstations (to include a workstation that will take over processing from the normal one).

Other ways of changing resource attributes are:

**EQQUSIN subroutine**
Refer to *Customization and Tuning*.

**SRSTAT command**
Refer to "SRSTAT" on page 664.

If the availability of a resource is known to the Resource Object Data Manager (RODM), Tivoli Workload Scheduler for z/OS can, by subscribing to RODM for that resource, be notified automatically of any changes. This is the best alternative where RODM is installed.

Changes to special resources using any of these methods override the scheduled quantity and availability, but you can at any time reset the values to those specified for the current interval.

These are some examples of how you can use the Monitor:
- Browse the special resources in the current plan, and see how they are allocated.
- Deallocate resources from an operation that is running
- Change the resource requirements of an operation that has not yet run
- See what special resource an operation is waiting for, if its extended status is X (for example, status RX), and see what operations are using it

**Example using data sets**

The payroll application has many jobs that use the payroll database and therefore must not run together. You could let z/OS resolve the contention problem, using DISP=OLD in the JCL, but a job that waits in z/OS uses a JES initiator and other resources. This can reduce your batch throughput.

To prevent Tivoli Workload Scheduler for z/OS from scheduling or starting an operation that uses the payroll database when it is already being used, define a resource PAYROLL.DATABASE that represents the payroll database. Give each update job, such as PAYDAILY, exclusive control. Jobs that merely read the database, such as PAYQUERY, can have shared control.

In this case, the resource has a quantity of 1 (there is one database). Specify keep on error, because operators want to correct and resubmit a job without another job taking control of the database in the meantime.

When you extend the current plan, Tivoli Workload Scheduler for z/OS makes sure the jobs are not scheduled to run together or run at a time when the resources are unavailable. This is how Tivoli Workload Scheduler for z/OS makes use of the resource at the planning stage. When Tivoli Workload Scheduler for z/OS is ready...
Example using data sets

to submit each job, it checks that the resource is available. This is how Tivoli Workload Scheduler for z/OS makes use of the resource at the control stage.

Specify the resource like this:

Name
PAYROLL.DATABASE. Make sure that all operations specify this name exactly.

Quantity
1

Used for
B (both planning and control).

On-error action
Keep all.

Workstations
Connect to all workstations that can use the data set.

Availability
The data set is always available, so no intervals are required.

Example using tape drives

Tape drives are usually owned by only one machine, but they can be used by a started task workstation and a computer workstation on the same machine, so you can, for example, allocate a tape pool to workstations CPU1 and STC1.

Operations that use tapes must allocate them exclusively. Do not let your operations keep this resource on error, because you normally want to release a tape drive for other work while you prepare a rerun of a failed job.

Specify the resource like this:

Name
TAPES

Usage
Give all operations exclusive allocation.

Quantity
10 (for example). You need not make all the drives available for automatic allocation.

Used for
B (both planning and control).

On-error action
Free all.

Workstations
Connect to all workstations that can use the tapes.

Example using communication lines

Lines are often shared between processors and can even be shared between operations. Lines are never allocated by z/OS, because they are owned by a communication controller, but you might want to ration the number of file transfer
Example using communication lines

jobs, for example. If you have a number of lines from New York to London, with a
total capacity of 256 kbaud, you can define a quantity of 256. If you give a file
transfer operation exclusive use of 20 units, for example, that gives the lines a limit
of 12 file transfers.

You can protect online and voice systems that use the same lines by giving them a
shared allocation of, for example, 50 units each. Because they share units, they do
not compete with each other; you can have an unlimited number of operations
each sharing 50 units, but this limits the quantity available for file transfers to 206.

Specify the resource like this:

Name
   LINES.TO.LONDON

Quantity
   256

Used for
   B (both planning and control)

On-error action
   Free all

Workstations
   Connect to all workstations that can use the lines

Intervals
   Reduce the number available at peak hours, which will keep more transfer jobs
   out and improve the performance of online systems.

How does the scheduler use special resources?
The scheduler keeps a record of the state of each resource and its allocation. The
scheduler does not know that PAYROLL.DATABASE is a database and it is
unaware that TAPES resources are tape drives. Only you know this and you are
responsible for making sure that Tivoli Workload Scheduler for z/OS knows the
true availability of the objects that the resources represent.

VSAM will not tell Tivoli Workload Scheduler for z/OS when the payroll database
is opened, and z/OS is not going to tell Tivoli Workload Scheduler for z/OS when
you vary a tape drive offline. This is your responsibility.

The best way to tell Tivoli Workload Scheduler for z/OS about changes in special
resources is through the RODM interface. System components such as AOC/MVS
inform RODM about changes to their resources. You can subscribe to RODM
updates by setting the RODMTASK keyword on the OPCOPTS initialization
statement, and using the RODMOPTS initialization statement. Refer to
Customization and Tuning for details of initialization statements.

If you do not have RODM installed, and for resources that RODM does not know
about, you can automatically notify Tivoli Workload Scheduler for z/OS about
changes in resources by intercepting messages (NetView can do this) and issuing
Tivoli Workload Scheduler for z/OS SRSTAT command, or calling the EQQUSIN
subroutine. If you cannot automatically notify Tivoli Workload Scheduler for z/OS
of a change in resource status, use the Special Resource Monitor.
Using the special resource monitor

Operations hold and release special resources automatically according to their descriptions in the current plan and the resources are available and connected to workstations as scheduled in the current plan. So why do you need to monitor them?

Resources represent something, such as tape drives, and the current plan is built assuming that a certain number of tape drives will be available. If one breaks down and you do not use RODM or otherwise automatically notify Tivoli Workload Scheduler for z/OS, Tivoli Workload Scheduler for z/OS no longer has a true picture of the real world—Tivoli Workload Scheduler for z/OS continues to allocate the broken tape drive to a job that needs one. The job will wait, because z/OS knows that the tape drive is offline. To avoid Tivoli Workload Scheduler for z/OS starting a job that will only wait and use operating system resources, use the SPECIAL RESOURCE MONITOR panel to reduce the number of tape drives by one. You specify a deviation of -1 (minus one) to show that the quantity is reduced. When the engineer hands it back and you vary it online again, reset the deviation.

If a job is waiting with extended status X, it is waiting for a resource. Use the SPECIAL RESOURCE MONITOR panel to check its availability, and see what other operations are using the resource.

Understanding availability intervals

Before you use the Special Resource Monitor, be sure that you understand how availability intervals work. Table 38 shows how planned availability is affected by unplanned events such as input from the Special Resource Monitor, the SRSTAT command, and the EQQUSIN subroutine. Notice that manually altered attributes are honored across an interval boundary and batch planning EXTEND and REPLAN jobs; to make Tivoli Workload Scheduler for z/OS revert to the scheduled values after a manual alteration, you must reset the attribute, as at 11.20 in Table 38.

Table 38. How attributes are preserved across intervals

<table>
<thead>
<tr>
<th>Start of interval / time of event</th>
<th>Planned values</th>
<th>Actual values</th>
<th>Deviation</th>
<th>Number available</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Planned quantity</td>
<td>Planned availability</td>
<td>Actual quantity</td>
<td>Actual availability</td>
</tr>
<tr>
<td>08.00</td>
<td>8</td>
<td>N</td>
<td>Interval specifies quantity 8, not available</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>N</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>08.40</td>
<td>You set the availability to Y with the EQQUSIN subroutine</td>
<td>8</td>
<td>Y</td>
<td>0</td>
</tr>
<tr>
<td>09.00</td>
<td>8</td>
<td>N</td>
<td>A new interval specifies the resource unavailable</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>Y</td>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>09.40</td>
<td>You set a deviation of -1 with the SRSTAT command</td>
<td>8</td>
<td>Y</td>
<td>-1</td>
</tr>
<tr>
<td>09.41</td>
<td>You set a deviation of -1 with the SRSTAT command</td>
<td>8</td>
<td>Y</td>
<td>-2</td>
</tr>
<tr>
<td>09.42</td>
<td>You set the deviation to -1 with the Special Resource Monitor</td>
<td>8</td>
<td>Y</td>
<td>-1</td>
</tr>
<tr>
<td>10.00</td>
<td>9</td>
<td>Y</td>
<td>Extend CP, and interval specifies 9 available</td>
<td></td>
</tr>
</tbody>
</table>
### Understanding availability intervals

Table 38. How attributes are preserved across intervals (continued)

<table>
<thead>
<tr>
<th>Planned values</th>
<th>Actual values</th>
</tr>
</thead>
<tbody>
<tr>
<td>9 Y -1 8</td>
<td></td>
</tr>
<tr>
<td>10.20</td>
<td>You set the quantity to 6 with the SRSTAT command</td>
</tr>
<tr>
<td>6 Y -1 5</td>
<td></td>
</tr>
<tr>
<td>11.00</td>
<td>Interval specifies 8 available</td>
</tr>
<tr>
<td>6 Y -1 5</td>
<td></td>
</tr>
<tr>
<td>11.20</td>
<td>You reset the quantity with the SRSTAT command</td>
</tr>
<tr>
<td>8 Y -1 7</td>
<td></td>
</tr>
</tbody>
</table>

The number available (the last column) is the actual number available for allocation, taking into account the actual quantity, the deviation, and the actual availability.

The three events starting at 09.40 show the difference between altering the deviation with SRSTAT (or a subroutine) and with the Special Resource Monitor. SRSTAT adds the specified deviation to the current deviation, but the panel replaces the current deviation with the value you specify.

If you change values other than the overriding (global) quantity, availability, and deviation, or the values for an interval, you lose the changes at the next daily planning run, but the job issues a warning message about any manually changed values that will be lost. For example, if you change the default quantity (the quantity used where intervals are not specified) in the current plan, this is replaced at the next daily planning run with the value from the database.

### Accessing the special resource monitor

Follow these steps to use the Special Resource Monitor to remove a resource from an operation:

1. From the MODIFYING THE CURRENT PLAN main menu, select option 7 (SPECRES). You see the SPECIFYING RESOURCE MONITOR LIST CRITERIA panel, shown in Figure 274.

2. Use the panel to limit the resources displayed. The SPECIAL RESOURCE and SPECRES GROUP ID fields can have Tivoli Workload Scheduler for z/OS filter
Accessing the special resource monitor

characters such as * (asterisk) to specify a range of resources. Specify the values shown, for example, to show all available resources whose names begin with PAY.

3. Press Enter to list the resources. You see the SPECIAL RESOURCE MONITOR panel, shown in Figure 275.

<table>
<thead>
<tr>
<th>Command</th>
<th>SPECIAL RESOURCE MONITOR</th>
<th>ROW 1 TO 1 OF 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQQMLSL</td>
<td>Special Resource Monitor</td>
<td>B - Browse, M - Modify, I - In use list, W - Waiting queue</td>
</tr>
<tr>
<td>Resource</td>
<td>RDM Adjust Used Used W</td>
<td></td>
</tr>
<tr>
<td>PAYROLL.DATABASE</td>
<td>Y NNN 1 0 0 N</td>
<td></td>
</tr>
</tbody>
</table>

Figure 275. EQQMLSL - Special resource monitor

Each resource is described with these fields:

A Availability. Y or N. When a global availability has been set (for example, with the SRSTAT command), this is shown, rather than the value specified for the interval.

RDM RODM can update three resource fields:

A Availability
D Deviation
Q Quantity

The panel shows a separate status for each of the RDM fields:

N Not monitored. The scheduler does not subscribe to RDM for this field.

I Inactive. RODM is inactive, or communication has been lost. The scheduler tries to reconnect every 5 minutes.

P Pending. A status request has been sent to RODM, but no reply has yet been received.

A Active. RODM monitoring is active and has updated the field.

Adjust Qty

The current quantity, taking any deviation into account. When a global quantity has been set (for example, with the SRSTAT command), this is used, rather than the value specified for the interval or the default value.

Used Shared

Quantity allocated as shared.

Used Excl

Quantity allocated as exclusive.

W Indicates that there are operations waiting to allocate the resource (Y), or no waiting operations (N).

The available row commands are:

B Browse the resource.
M Modify the resource.
I Display the operations that have allocated this resource.
W Display the operations that are waiting to allocate this resource.
4. Enter the I row command to see the operations that are using the resource. You see the SPECIAL RESOURCE MONITOR - IN USE LIST panel, shown in Figure 276.

<table>
<thead>
<tr>
<th>Command</th>
<th>Scroll</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enter any of the row commands below:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S - Select details, D - Delete from list</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Special resource: PAYROLL.DATABASE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Text: serializes access to the Paymore database</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Row Actual Start</td>
<td>Operation Jobname</td>
<td>Est</td>
</tr>
<tr>
<td>cmd Date Time ws no. Dur</td>
<td></td>
<td></td>
</tr>
<tr>
<td>'' 95/06/09 12.00 CPU1 020 PAYDAILY 0060.00 E 1 X</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 276. EQQQMIML - Special resource monitor - in use list

The operations at the top of the list have been using the resource for the longest time and are more likely to release their allocation soon. The duration helps you estimate when the operations will release their allocation of the resource.

The list has these columns:
- Actual start date and time
- Operation ws (workstation)
- Operation no. (operation number)
- Jobname
- Est Dur (the estimated operation duration)
- S (status of the operation)
- Qty (number allocated to this operation)
- Type (type of allocation: shared (S) or exclusive (X)).

5. Enter the D row command beside the operation to release the resources from the operation.

**Note:** This is a logical release. In Figure 276, for example, the D command causes PAYDAILY to free Tivoli Workload Scheduler for z/OS resource PAYROLL.DATABASE, but if its status is S, it can still be writing to the real payroll database.

6. You see the CONFIRMING DELETION OF AN OPERATION FROM QUEUE OR LIST panel. Enter Y to deallocate the resource from the operation.

### Looking at the operations waiting for a resource

To see the operations that are waiting for a resource, enter the W row command beside the resource in the SPECIAL RESOURCE MONITOR panel. The following panel is displayed:
The operations at the top of the list have earlier latest-out times and are therefore more likely to get the resource when a sufficient quantity becomes available. The list has these columns:

- Latest out date and time
- Operation ws (workstation)
- Operation no. (operation number)
- Jobname and priority
- Qty (the amount of the resource that the operation needs)
- Type (type of allocation—shared (S) or exclusive (X))
- Reason Wait (the reason that this operation must wait). It can have these codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>FEWINF</td>
<td>According to the values of the LOOKAHEAD initialization parameter and the planned job duration, there will not be enough quantity to satisfy the waiting operation.</td>
</tr>
<tr>
<td>INVRES</td>
<td>The resource is not valid.</td>
</tr>
<tr>
<td>NOWSC</td>
<td>No workstation is connected.</td>
</tr>
<tr>
<td>OTHRES</td>
<td>The operation needs this resource, but is waiting for another special resource.</td>
</tr>
<tr>
<td>RODMP</td>
<td>The scheduler is waiting for a status update from RODM.</td>
</tr>
<tr>
<td>TOOFEW</td>
<td>There is not enough quantity to satisfy the operation that is waiting.</td>
</tr>
<tr>
<td>UNAVL</td>
<td>The resource is not available.</td>
</tr>
<tr>
<td>UNAVLF</td>
<td>According to the values of the LOOKAHEAD initialization parameter and the planned job duration, the resource will not be available for the required time.</td>
</tr>
</tbody>
</table>

An operation is taking all the resource.

Enter the D row command beside a row to remove the dependency of the operation on the resource so that it can start (but it might be waiting for other resources, too). Do this with care, because the operation might not have access to the resource that it needs to run successfully. In the PAYQUERY example, if you remove the dependency of PAYQUERY on the payroll database resource, the PAYQUERY job will start, but it will wait for the database if it is still allocated to
Looking at the operations waiting for a resource

PAYDAILY. You see the CONFIRMING DELETION OF AN OPERATION FROM QUEUE OR LIST panel. Enter Y to remove the resource dependency from the operation.

Modifying a special resource

From the SPECIAL RESOURCE MONITOR panel, you can browse or modify special resource details using the B or M row commands. To modify a resource, enter the M row command beside a resource. You see the MODIFYING A SPECIAL RESOURCE panel, shown in Figure 278.

This is similar to the SPECIAL RESOURCE DEFINITION panel, except that here you are not updating the RESOURCE DESCRIPTION database. When you change resource details with the MCP Resource Monitor, you are updating the current plan resource information, which is stored in the current plan extension (CX) data set. That is why this panel has extra fields that are not present in the SPECIAL RESOURCE DEFINITION panel:
- DEVIATION (1)
- AVAILABLE (2)
- QUANTITY (3)
- Active LIFESPAN (4)
- The reason that caused the global availability to be reset, Event Change in this example (5)

If no change has been made to the current plan, DEVIATION is zero or blank (no deviation) and AVAILABLE and QUANTITY are blank (they are specified in the current interval data, if any, or take the default value). If you have changed these...
Modifying a special resource

fields, either with this panel, with the SRSTAT command, or with the EQUSIN subroutine, the deviation is added to the quantity to give the actual quantity available and the changed availability overrides the scheduled availability until it is reset (set to blank).

Note: The resource details are retained in the current plan as long as any of the fields 1 to 3 are set (QUANTITY or AVAILABILITY are non-blank, or DEVIATION is not zero or blank). If you set TAPES unavailable, for example, using field 2, TAPES remains unavailable indefinitely, past daily planning EXTENDs and REPLANs, for weeks or even years. You must set them to blank (or, in the case of deviation, to zero) manually for the database values to take effect. When you change other fields on the panel, such as the default quantity, the value will be replaced by the value from the database the next time a daily planning EXTEND or REPLAN is run. Daily planning issues a warning message when it replaces a manually changed value with a value from the database.

The Active LIFESPAN line of the MODIFYING A SPECIAL RESOURCE panel in Figure 278 on page 615 indicates whether there is a pending action about the global availability of the resource. It has the following format:

Active LIFESPAN: Action = ACTION_TYPE  Expiration Date= EXPIRATION_DATE

ACTION_TYPE
The action taken to set the global availability when the expiration time is reached. It can be one of the following values:

Y     Sets the global availability to Yes.
N     Sets the global availability to No.
R     Sets the global availability to blank.
Blank  No action taken.

EXPIRATION_DATE
The date and time when the global availability of the resource will be changed.

The last line of the MODIFYING A SPECIAL RESOURCE panel is Last Updated by, showing how the resource came to be in the current plan and the reason that caused the global availability to be reset. The Last Updated by line has the following format:

Last updated by: USERID at DATE on TIME  GlobalAvailChangeReason

USERID
Who or how the resource was added. The resource details might have been added during batch daily planning because a planned operation references the resource. If there have been no manual alterations to the resource, this shows who last updated the database record.

If the resource details have been changed with the SRSTAT command, this shows who issued the command.

If it is not possible to identify the user who changed the resource (for example, because the resource was added to the plan by a batch daily planning program), this shows the process that changed the occurrence:

*DYNAADD*  Batch daily planning dynamically added the resource to the plan, because an operation referenced it, but the resource is not in the database.
**Modifying a special resource**

*DSPADD* Batch daily planning dynamically copied the resource to the plan, because an operation referenced it (the resource is in the database).

*OPC* An internal process, generated for example by the On Complete action, changed the resource in the plan.

**DATE and TIME**
The date and time when the resource was last updated.

**GLOBALAVAILCHANGEREASON**
The reason that caused the global availability of the resource to be changed.
Possible values are:

**Event Change**
A special resource event occurred. Events are generated when you issue an SRSTAT command or you invoke a program such as, for example, EQQUSIN or when data set triggering is active.

**Max Usage Limit Change**
The maximum usage limit was reached.

**LIFESPAN Change**
A LIFESPAN active condition expired. LIFESPAN active conditions are generated when you issue an SRSTAT command with the LIFESPAN parameter or when data set triggering table definitions use the LIFESPAN parameter.

**Blank**
The global availability was either not changed or changed through a panel.

To change the intervals, select option 1 (Intervals). The following panel is displayed:

```
**EQQODIML ------- MODIFYING INTERVALS FOR A SPECIAL RESOURCE**
Command ==> Scroll ==> PAGE

Enter any of the row commands below:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete, or,
S - Work stations

Special resource : TAPES
Text : tape drives on CPU1 and STC1

Row Date From To Qty A
cmd Time Time
'' 95/06/10 00.00 07.59 8 Y
'' 95/06/10 08.00 22.00 6 Y
'' 95/06/10 22.01 23.59 8 Y
'' 95/06/11 00.00 07.59 8 Y

*******************************************************************************

Figure 279. EQQODIML - Modifying intervals for a special resource
```

When you change interval data in the current plan, this is not replaced by values from the database when you next run daily planning; a manually altered interval remains in the plan.

To change the workstations where operations can allocate the resource, enter the S row command beside the interval. You see the MODIFYING CONNECTED WORKSTATIONS FOR A SPECIAL RESOURCE panel, shown in Figure 280 on page 618.
Modifying a special resource

Figure 280. EQQDWML - Modifying connected workstations for a special resource

Change the connected workstations as required and press PF3 (End).
Chapter 29. Using Tivoli Business Systems Manager

This chapter describes how to use Tivoli Workload Scheduler for z/OS with Tivoli Business Systems Manager, Version 1.5 with APAR 0W50948 or later. Monitoring batch processes within Tivoli Business Systems Manager provides quick determination of problems that threaten batch schedules from completing correctly and on time. You can immediately map and understand the impact of any outage in the system, thus facilitating a fast response for problem handling and resolution.

Tivoli Business Systems Manager enables you to monitor and manage events related to resources, applications, and subsystems with the objective of providing continuous availability for the enterprise. The product receives events and inputs from z/OS systems and Tivoli managed distributed systems. It then calculates propagation rules and applies them according to the business rules of the enterprise so that you can determine immediately the overall impact of an outage.

Using Tivoli Business Systems Manager with IBM Tivoli Workload Scheduler for z/OS, you can manage strategic applications from a unique business systems perspective. The scheduler identifies jobs to be actively monitored in Tivoli Business Systems Manager. The scheduler then sends jobs events to Tivoli Business Systems Manager for job status changes to started, completed, and alert conditions. The scheduler also sends events to signal job additions in the plan.

This chapter contains the following sections:
- Enabling monitoring by Tivoli Business Systems Manager
- How monitoring works
- Discovery of the scheduler

Enabling monitoring by Tivoli Business Systems Manager

The following steps are necessary to enable Tivoli Workload Scheduler for z/OS to be monitored by Tivoli Business Systems Manager. If you used Tivoli Business Systems Manager with previous versions of Tivoli Workload Scheduler for z/OS, it is no longer necessary to use the Tivoli Business Systems Manager user exit 7 and to activate the WTO options in the ALERT initialization statement.

Scheduler start options

For Tivoli Business Systems Manager to monitor Tivoli Workload Scheduler for z/OS, you must configure Tivoli Workload Scheduler for z/OS start options. Specify the EXTMON parameter on the OPCOPTS statement. This statement defines the runtime options to Tivoli Workload Scheduler for z/OS and is used by the tracker, controller, or the standby controller.

Refer to Customization and Tuning for OPCOPTS parameters and syntax information.

At initialization time, if the EXTMON start option is set to YES, Tivoli Workload Scheduler for z/OS loads the Tivoli Business Systems Manager module AOPEDI. The module must be present in a library visible to the controller and tracker. If the
Scheduler start options

Load fails, the value of the EXTMON start option is automatically set to NO, and the message EQQZ232 appears in Tivoli Workload Scheduler for z/OS message log.

Identifying jobs for monitoring

In order to monitor a job using Tivoli Business Systems Manager, set the EXTERNAL MONITOR job option of an operation to YES. This option is modifiable in the application description database and in the current plan. You can use any of the following to set or browse the option:

- ISPF panels
- The Scheduler programming interface
- Dynamic Workload Console

By default, jobs are not monitored by Tivoli Business Systems Manager.

Monitoring can also be configured so that jobs are automatically selected. This is done by establishing a monitoring policy, using the MONPOL initialization statement. For further details on this initialization statement refer to Customization and Tuning.

Setting monitors from the ISPF panels

You can browse and set the EXTERNAL MONITOR job option of an operation from the JOB, WTO, AND PRINT OPTIONS panel (see Figure 281 on page 620 for an example).

<table>
<thead>
<tr>
<th>Command ===&gt;</th>
<th>Enter/Change data below:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Application</td>
<td>VALE1</td>
</tr>
<tr>
<td>Operation</td>
<td>CPU1 001</td>
</tr>
<tr>
<td>JOB CLASS ===&gt;</td>
<td>Job class of corresponding job</td>
</tr>
<tr>
<td>ERROR TRACKING ===&gt;</td>
<td>Y Y means error automatically tracked</td>
</tr>
<tr>
<td>HIGHEST RETURNCODE ===&gt;</td>
<td>Highest return code not in error</td>
</tr>
<tr>
<td>EXTERNAL MONITOR ===&gt;</td>
<td>Y Job monitored by external product (Y/N)</td>
</tr>
<tr>
<td>CENTRALIZED SCRIPT ===&gt;</td>
<td>N Centralized script Y/N (for FTW only)</td>
</tr>
<tr>
<td>CRITICAL ===&gt;</td>
<td>P Critical job: N P W</td>
</tr>
<tr>
<td>POLICY ===&gt; L CLASS ===&gt; WLMCLASS</td>
<td>WLM policy and service class</td>
</tr>
<tr>
<td>Job release options:</td>
<td>Answer Y or N for options below:</td>
</tr>
<tr>
<td>SUBMIT ===&gt;</td>
<td>Y Automatically submitted</td>
</tr>
<tr>
<td>HOLD/RELEASE ===&gt;</td>
<td>Y Automatically held and released</td>
</tr>
<tr>
<td>TIME DEPENDENT ===&gt;</td>
<td>N Run the job at specified time</td>
</tr>
<tr>
<td>SUPPRESS IF LATE ===&gt;</td>
<td>N Suppress the job if not on time</td>
</tr>
<tr>
<td>DEADLINE WTO ===&gt;</td>
<td>Y Deadline WTO, Y or N</td>
</tr>
<tr>
<td>WS fail options:</td>
<td></td>
</tr>
<tr>
<td>RESTARTABLE ===&gt;</td>
<td>Operation is restartable</td>
</tr>
<tr>
<td>REROUTABLE ===&gt;</td>
<td>Operation is eligible for reroute</td>
</tr>
<tr>
<td>Print options:</td>
<td></td>
</tr>
<tr>
<td>FORM NUMBER ===&gt;</td>
<td>_____ SYSOUT CLASS ===&gt; _</td>
</tr>
</tbody>
</table>

Figure 281. EQQAMJBP - Job, WTO, and print options

In the current plan, you can browse if a specific operation has an external monitor from the BROWSING OPERATIONS panel. The BROWSING DETAILED OPERATION INFORMATION panel in Figure 282 on page 621 shows an operation that has an external monitor.
Setting monitors from the ISPF panels

Refer to Programming Interfaces for additional information about using the programming interface to set up monitors for Tivoli Business Systems Manager. In addition to using the programming interface, monitoring of operations can be enabled from the batch loader and the BCIT. Refer to Programming Interfaces for information on how to use the batch loader.

How monitoring works

To monitor Tivoli Workload Scheduler for z/OS, Tivoli Business Systems Manager scans the daily planning report in Tivoli Workload Scheduler for z/OS and discovers all batch objects in the scheduler plan. Events to signal job status changes and alert conditions are sent to Tivoli Business Systems Manager only for jobs that have the external monitor job option set to YES.

Tivoli Business Systems Manager receives the following events from Tivoli Workload Scheduler for z/OS for monitored jobs:
How monitoring works

- Addition of an operation
- Operation in late
- Operation waiting for resource
- Start of an operation
- Long duration for an operation
- End of an operation

The scheduler also sends to Tivoli Business Systems Manager an event for the following alert conditions:
- Alert durations (monitored jobs only)
- Alerts for late operations (monitored jobs only)
- Alerts for operations that ended in error (all jobs)
- Alerts for the time out of a special resource (monitored jobs only)
- Alerts for Tivoli Workload Scheduler for z/OS subtasks or subsystems that ended in error
- Alerts for Tivoli Workload Scheduler for z/OS subtask queues that exceed the threshold value

Tivoli Business Systems Manager is notified for all job error situations, such as job abended. When a non-monitored job abends, the EXTERNAL MONITOR job option is automatically set to YES, and Tivoli Business Systems Manager begins to monitor it.

Discovery of the scheduler

The Tivoli Business Systems Manager administrator must accomplish several tasks during the discovery session. These tasks are divided into the following categories:
- Collecting the current plan
- Organizing the applications in Tivoli Business Systems Manager
- Loading Tivoli Workload Scheduler for z/OS files into the Tivoli Business Systems Manager database

Tivoli Business Systems Manager discovers the scheduled jobs by reading the daily operating plan. To customize Tivoli Business Systems Manager for discovery of Tivoli Workload Scheduler for z/OS, refer to the Tivoli Business Systems Manager documentation regarding installation.
Chapter 30. Using IBM Tivoli Monitoring

This chapter describes how to use Tivoli Workload Scheduler for z/OS with IBM Tivoli Monitoring Version 6.1 with Fix Pack 2 installed. It describes how this integration works and the IBM Tivoli Monitoring components involved.

Tivoli Workload Scheduler for z/OS can send events and alerts to the IBM Tivoli Monitoring (ITM) product through the services provided by a component called Tivoli Universal Agent. Tivoli Workload Scheduler for z/OS establishes an IP connection with the Socket Data Provider of the Tivoli Universal Agent which acts like a listener task. Every time a change of status of the monitored operations occurs, or an alert is issued, the related event is sent to the Tivoli Universal Agent.

The Tivoli Universal Agent interprets the data sent by Tivoli Workload Scheduler for z/OS using the information contained in a metafile which is provided with Tivoli Workload Scheduler for z/OS and previously loaded on the Tivoli Universal Agent by an installation script called `tepconfig`. This script is explained in “Tivoli Enterprise Portal Configuration options” on page 625. The Tivoli Universal Agent stores the event information in tables that are consolidated by the Tivoli Enterprise Monitoring Server. The collected data is then retrieved and displayed by a component called Tivoli Enterprise Portal.

The Tivoli Enterprise Portal is a user interface for viewing and monitoring your enterprise network. The Tivoli Enterprise Portal client connects to the Tivoli Enterprise Portal server that enables retrieval, manipulation, and analysis of data collected by all existing IBM Tivoli Monitoring agents in your enterprise.

Using the Tivoli Enterprise Portal you can display and manage events related to Tivoli Workload Scheduler for z/OS operations such as operation start and end time. In the same way, you can configure automatic selection of monitored operations and subtasks providing a single point of management and control for all your enterprise resources.

Because the Tivoli Enterprise Portal component provides the end-user interface for viewing and managing Tivoli Workload Scheduler for z/OS monitored objects, in this publication we refer to the integration with IBM Tivoli Monitoring by referring to Tivoli Enterprise Portal. For detailed information on IBM Tivoli Monitoring components refer to the IBM Tivoli Monitoring documentation.
Using IBM Tivoli Monitoring

This chapter contains the following sections:

- “Enabling monitoring by IBM Tivoli Monitoring” on page 624
- “How monitoring works” on page 626
- “Identifying jobs for monitoring” on page 628

Enabling monitoring by IBM Tivoli Monitoring

This section describes the steps you follow to enable Tivoli Workload Scheduler for z/OS to be monitored by Tivoli Enterprise Portal.

The components that must be installed on one of the supported operating systems to allow integration with Tivoli Workload Scheduler for z/OS are the following:

- Tivoli Enterprise Monitoring Server (TEMS)
- Tivoli Enterprise Portal Server (TEPS)
- Tivoli Enterprise Portal (TEP)
- Tivoli Universal Agent

Note: For information on the supported operating systems for each of the above components refer to IBM Tivoli Monitoring documentation.

Controller Configuration options

For Tivoli Enterprise Portal to work with Tivoli Workload Scheduler for z/OS, you must include the data set used by the monitoring task in your JCL procedures and configure the controller start options.

EEQMONDS is the DD Name of the data set used by the monitoring task to store events for IBM Tivoli Monitoring. This is an optional data set and so you must include this data set in your JCL procedures if you want to be able to monitor events with the Tivoli Enterprise Portal. Refer to the information in Tivoli Workload Scheduler for z/OS Planning and Installation about creating JCL procedures and including optional data sets in the procedure.
Controller Configuration options

To enable monitoring by Tivoli Enterprise Portal, use the MONOPTS initialization statement. This statement defines the runtime options to Tivoli Workload Scheduler for z/OS and is used by the controller or the standby controller.

At initialization time, if the MONOPTS option is set, a specific controller subtask called “monitoring task” is started. This subtask is dedicated to sending events to the Tivoli Enterprise Portal.

Optionally, you can use the OPCOPTS initialization statement to set the codepage to use in the communication between Tivoli Workload Scheduler for z/OS and Tivoli Enterprise Portal, and the ALERTS and MONALERT statement to define for what conditions an alert should be sent to the Tivoli Enterprise Portal monitoring agent.

You can also configure monitoring so that operations are automatically selected to be monitored. Define a monitoring policy using the MONPOL initialization statement.

If you use a TCP/IP stack name different from TCPIP default value, code the TCPOPTS statement with the TCPIPJOBNAME parameter set to the TCP/IP stack name.

For a detailed description of the parameters and syntax information for the ALERTS, MONOPTS, MONPOL, OPCOPTS, and TCPOPTS initialization statements, refer to Customization and Tuning.

Tivoli Enterprise Portal Configuration options

The Tivoli Enterprise Portal integration package is contained in a CD delivered with the product. It includes the following items:

- A sample Metafile, TWSz86SampleMeta.mdl, needed by the Tivoli universal Agent to access the monitoring data
- Customization scripts for the following environments:
  - UNIX: tepconfig.sh
  - Windows: tepconfig.bat

For a description of the product CD, refer to Memo to Users.

To enable monitoring by Tivoli Enterprise Portal, perform the following steps:

1. Download the configuration package (in tar format) to the target Tivoli Enterprise Portal system. Download the metafile and customization scripts to the system where the Tivoli Universal Agent is installed.

   Note: Ensure you save the metafile and customization scripts to a directory where the name of the directory does not contain spaces. If the script is run from a directory name containing spaces, the installation fails.

2. Launch the metafile configuration script on the system where the Tivoli Universal Agent is installed, from the path where the metafile resides:

   Use:
   
   `tepconfig -d Universal_Agent_install_dir`
   `-h hostname[portnumber] [-t time_to_live_for_monitored_data]`
   
   on UNIX:
   
   `./tepconfig.sh -d Universal_Agent_install_dir`
   `-h hostname[portnumber] [-t [time_to_live_for_monitored_data]`
   
   on Windows:
Tivoli Enterprise Portal Configuration options

```
tepecfg.bat -d Universal_Agent_install_dir -h hostname[portnumber] [-t [time_to_live_for_monitored_data]]
```
Where:

- **-d** The fully qualified path to the Tivoli Universal Agent installation directory. The script does not accept a path name containing blanks.

- **-h** The host name of the z/OS system(s) where the Tivoli Workload Scheduler for z/OS controller is installed, optionally followed by the port number used by the Controller to send data to the Tivoli Universal Agent. If specified, the port number must match with the value specified for the LOCPORT parameter in the MONOPTS statement. The host name is used to set the `<location>` variable in the SOURCE control statement of the data definition metafile.

- **-t** The amount of time in seconds which the monitored data is kept by Tivoli Enterprise Portal. This parameter sets the TTL value in the NAME control statement in the data definition metafile. The value that you specify for this parameter also determines when the metafile application will go offline in Tivoli Enterprise Portal after a disconnection between the Socket Client Program and the Socket Data Provider has occurred. This parameter is optional. If not set, the default value is 86400. For more information on the NAME Control statement refer to the *IBM Tivoli Universal Agent User’s Guide*.

**Example:**

The following example shows that monitoring is configured for a Tivoli Workload Scheduler for z/OS controller residing on a system with host name zos18c8, and a port number of 2500. The monitored data is kept by Tivoli Enterprise Portal for 3600 seconds. The host name parameter is fully qualified.
```
tepecfg.bat -d C:\IBM\ITM -h zos18c8.ibm.com[2500] -t 3600
```

Log in to the TEP client to start monitoring.

**How monitoring works**

If the Tivoli Enterprise Portal Monitoring is activated, monitoring is performed only on those operations that have the EXTERNAL MONITOR option set to YES.

A Tivoli Workload Scheduler for z/OS event is sent to the Tivoli Enterprise Portal when the following situations occur:

- Addition of an operation
- Late operation
- Operation waiting for resource
- Start of an operation
- Long duration of an operation
- End of an operation
- Any subsequent status changes to operations that ended in error

All application occurrences containing monitored operations are also monitored. The Tivoli Enterprise Portal receives events from Tivoli Workload Scheduler for z/OS when the following situations occur:

- Addition of an application occurrence
- Start of an application occurrence
- End of an application occurrence

---

626 IBM Tivoli Workload Scheduler for z/OS: Managing the Workload
How monitoring works

- Delete of an application occurrence

**Discovery of monitored objects**
You obtain the initial status of the monitored objects by running a bulk discovery using the BULKDISC TSO command, or using a batch job. The bulk discovery command can be used each time an updated status of all monitored objects is required.

The bulk discovery can also be configured to start at every current plan creation, extension, or replan by using the MONOPTS initialization statement. If you set the BULKDISC parameter to YES in the MONOPTS statement, then a bulk discovery is performed automatically for example, each time a status change occurs.

For further information on how to configure the bulk discovery and on the BULKDISC command, refer to *Customization and Tuning* and the command “BULKDISC” on page 651.

**Identifying and displaying alerts**
The scheduler also sends alerts to Tivoli Enterprise Portal based on the MONALERT and MONOPER parameters of the ALERTS statement. The types of alerts that are sent depend on the monitoring policy provided using the MONPOL statement. For example, to send alerts that can be viewed in the Tivoli Enterprise Portal pertaining to specific operation statuses (CRITICAL, ERROR, LATE, DURATION, MANUAL), use the MONPOL statement to specify the types of operations to be monitored. The alerts can be viewed from the Jobs view in the Tivoli Enterprise Portal.

The ALERTS statement instead, specifies the conditions under which an alert should be generated and an alert can be sent to the Tivoli Enterprise Portal as a consequence of the generated alert. This type of alert can be viewed from the Universal Agent view in the Tivoli Enterprise Portal.

For a detailed description of the ALERTS and MONPOL statements and their parameters, refer to *Customization and Tuning*.

Alerts can be sent on the following conditions:

- For monitored jobs or for all jobs (in parentheses are the alert names that are displayed by the Tivoli Enterprise Portal interface):
  - Alerts for operations that ended in error (JobInError)
  - Alerts for late operations (LateJob)
  - Alerts for long durations (LongDurationJob)
  - Alerts for long time in the input queue (LongDurationJob)
  - Alerts for jobs promoted by WLM (WLMPromotedJob)
  - Alerts for the time-out of a special resource (ResourceTimeout). The timeout is specified in the RESOPTS CONTENTIONTIME parameter.

- For Tivoli Workload Scheduler for z/OS subtasks (in parentheses are the alert names that are displayed by the Tivoli Enterprise Portal interface):
  - Alerts for Tivoli Workload Scheduler for z/OS subtasks or subsystems that ended in error (TaskInError)
  - Alerts for Tivoli Workload Scheduler for z/OS subtask queues that exceed the threshold value (QueueThresholdExceeded)
Identifying and displaying alerts

- Alerts for Tivoli Workload Scheduler for z/OS controllers that lose monitoring events
- Tivoli Enterprise Portal specific alerts:
  - Some events collected by the monitoring task are lost. A bulk discovery could be necessary to restore data consistency in the Tivoli Enterprise Portal interface (QueueLostMonitorEvents).
  - An error occurred during a bulk discovery. A new bulk discovery could be necessary to restore data consistency in the Tivoli Enterprise Portal interface (ErrorDuringBulkDiscovery).

Note: If you recycle the Tivoli Universal Agent (TUA) events are lost. If the Tivoli Universal Agent is stopped for any reason, events are lost unless you have saved them using the UA history function. The Tivoli Universal Agent can also be stopped as a result of a TCP/IP connection failure. In these cases, you should issue a bulk discovery to refresh the status of all monitored objects.

Identifying jobs for monitoring

To monitor an operation using Tivoli Enterprise Portal, set the EXTERNAL MONITOR job option of an operation to YES. You can modify this option in the application description database and in the current plan. You can use one of the following interfaces to set or browse the option:

- ISPF panels
- The Scheduler programming interface
- Dynamic Workload Console

By default, jobs are not monitored.

You can also configure job monitoring so that jobs are automatically flagged as "monitored". You do this by establishing a monitoring policy. Use the MONPOL statement to automatically select jobs to be monitored. For further information, refer to Customization and Tuning.

Setting monitors from the ISPF panels

You can browse and set the EXTERNAL MONITOR job option of an operation from the JOB, WTO, AND PRINT OPTIONS panel (see Figure 284 on page 629 for an example).
In the current plan, you can browse if a specific operation has an external monitor from the BROWSING OPERATIONS panel. The BROWSING DETAILED OPERATION INFORMATION panel in Figure 285 shows an operation that has an external monitor.

In addition, you can use the mass update utility, in the application description, to update multiple operations with the EXTERNAL MONITOR job option. For more information on mass update, refer to Managing the Workload.
Setting monitors from the programming interface

Setting monitors from the programming interface
In addition to enabling monitoring by Tivoli Enterprise Portal from the panels, you can also write special scripts with the programming interface to specify monitors for Tivoli Enterprise Portal.

With the programming interface, you can specify that one operation of an application or occurrence be monitored. You can also specify that all the operations of an occurrence be monitored. Moreover, you can use the MONITOR option as a filter for the following:
- Operations in the current plan
- Applications or occurrences

Refer to Programming Interfaces for additional information about using the programming interface to set up monitors for Tivoli Enterprise Portal. In addition to using the programming interface, monitoring of operations can be enabled from the batch loader and the BCIT.

Setting monitors from the graphical user interfaces
Using the Dynamic Workload Console, you can specify that one operation of an application or occurrence be monitored. You can also specify that all the operations of an occurrence be monitored. Moreover, you can use the MONITOR option as a filter for the following:
- Operations in the current plan
- Applications or occurrences

For additional information, refer to the Dynamic Workload Console documentation.
Chapter 31. Reporting with Tivoli Workload Scheduler for z/OS

This chapter describes how the product supports the Dynamic Workload Console reporting feature, collecting historical workload data and archiving it in dedicated tables in a DB2 database. The console process retrieves the archived data and returns it to you by using the Business Intelligent Report Tool (BIRT) Report Viewer. The same reports can also be generated using a command-line interface.

Using reporting with Tivoli Workload Scheduler for z/OS, you archive historical data about jobs and workstation workloads in the DB2 database. The historical data consist of old current plan data collected at daily planning time.

For example, you can extend or replan your current plan more than once a day but decide to run the archiving process only once, selecting the new ARCHIVE option from the daily planning dialog. This triggers the submission of a batch job that archives the historical data in the DB2 database, by Java Data Base Connectivity (JDBC) services.

To use the new capabilities, some customization actions are required at the following sides:

Host system
- Specify the new optional keywords to the initialization statements of the batch component and the server.
- Run the new sample jobs to create feature-specific files.

Distributed system with the DB2 database installed
Create the dedicated tables.

The following sections provide detailed information about the reporting feature. They include setting up the environment, setting up the database applicable to running reports from both the Dynamic Workload Console and the command-line interface. They also explain how you customize the security support to protect access to your historical run data using a RACF fixed resource and sub-resource and an encryption utility for the password used when archiving data to DB2.

This chapter contains the following sections:
- "Archiving historical data" on page 632
- "Setting up the environment" on page 635
- "Setting up the database" on page 637
- "Encrypting the password in DBOPT statement" on page 639
- "Setting up RACF authorization" on page 640
- "Running batch reports from the command line interface" on page 640
- "Keeping historical data in the database" on page 645
Archiving historical data

The Dynamic Workload Console is a web-based user interface for Tivoli Workload Scheduler. By connecting Dynamic Workload Console with a scheduler, you can view and control scheduling activities, and display historical run data and generate reports. Following is a list of the reports that are provided:

**Job run statistics**
It shows, based on the selection criteria, information about one or more jobs and runtime statistics for each job. You can use it, for example, to detect success or error rates, minimum, maximum, and average duration, late and long duration statistics.

**Job run history**
It shows historical data about jobs that already run. This report is usually requested for a specific time frame. You can use it, for example, to detect jobs that ended in error or late jobs, missed deadline, long duration.

**Workstation workload runtimes**
It displays the time periods while the jobs run. This report provides information about the actual jobs that run on a specific workstation, along with their actual start times and duration. You can also list the job instances that caused job workload on a workstation, in a specific interval of time. You can use it to make necessary capacity planning adjustments, such as workload modeling and workstation tuning.

**Workstation workload summary**
It collects data about the workload on the workstations. It shows the number of jobs that actually run on a specific workstation in a specific interval of time. You can use it to make necessary capacity planning adjustments, such as workload modeling and workstation tuning.

**Custom SQL**
It enables you to create reports by running your own SQL queries. The reports will display a table with the column name as specified in the SELECT part of the SQL statement.

Depending on the interface from where you run the report or the operating system of the engine, the following output formats are available:

<table>
<thead>
<tr>
<th>Name of the report</th>
<th>Output formats supported by the Dynamic Workload Console</th>
<th>Output formats supported by batch reports</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job Run History Report</td>
<td>HTML, CSV</td>
<td>HTML, CSV, PDF</td>
</tr>
<tr>
<td></td>
<td>Only table format</td>
<td>Only table format</td>
</tr>
<tr>
<td>Job Run Statistics Report</td>
<td>HTML, CSV</td>
<td>HTML, CSV, PDF</td>
</tr>
<tr>
<td></td>
<td>Table and chart formats</td>
<td>Table and chart formats</td>
</tr>
<tr>
<td>Workstation Workload Summary Report</td>
<td>HTML, CSV</td>
<td>HTML, CSV, PDF</td>
</tr>
<tr>
<td></td>
<td>Table and chart formats</td>
<td>Table and chart formats</td>
</tr>
<tr>
<td>Workstation Workload Runtimes Report</td>
<td>HTML, CSV</td>
<td>HTML, CSV, PDF</td>
</tr>
<tr>
<td></td>
<td>Table and chart formats</td>
<td>Table and chart formats</td>
</tr>
<tr>
<td>Custom SQL Report</td>
<td>HTML, CSV</td>
<td>HTML, CSV, PDF</td>
</tr>
<tr>
<td></td>
<td>Only table format</td>
<td>Only table format</td>
</tr>
</tbody>
</table>
For more information about how to create and run reports from the Dynamic Workload Console, see the Reporting section of the Dynamic Workload Console User's Guide.

### Historical reports

The following tables summarize the historical reports in terms of their:

- Functionality
- Selection criteria
- Output content options

#### Table 40. Summary of historical reports

<table>
<thead>
<tr>
<th>Report name</th>
<th>Description</th>
<th>Selection criteria</th>
<th>Output content options</th>
</tr>
</thead>
<tbody>
<tr>
<td>Job run history</td>
<td>Collects historical job execution data during a time interval. Helps you find:</td>
<td>• Job name, job stream name, workstation name, and workstation name (job stream). Each field can be specified using a wildcard.</td>
<td>You can select from the following:</td>
</tr>
<tr>
<td></td>
<td>• Jobs ended in error</td>
<td>• Submission destination</td>
<td>• Actual start time</td>
</tr>
<tr>
<td></td>
<td>• Late jobs</td>
<td>• Execution destination</td>
<td>• Estimated duration</td>
</tr>
<tr>
<td></td>
<td>• Missed deadlines</td>
<td>• Status (Success, Error).</td>
<td>• Actual duration</td>
</tr>
<tr>
<td></td>
<td>• Long duration</td>
<td>• Delay indicators</td>
<td>• Job identifier</td>
</tr>
<tr>
<td></td>
<td>• Rerun indicators for reruns</td>
<td>• Job execution interval</td>
<td>• Started late (delay)</td>
</tr>
<tr>
<td></td>
<td>• Other historical information</td>
<td></td>
<td>• Ended late (delay)</td>
</tr>
<tr>
<td>Job run statistics</td>
<td>Collects job execution statistics. Helps you find:</td>
<td>• Job name. The field can be specified using a wildcard.</td>
<td>• Status</td>
</tr>
<tr>
<td></td>
<td>• Success/error rates</td>
<td>• Percentage of jobs in Success, Error, Started late, Ended late, and Long duration</td>
<td>• Iteration number</td>
</tr>
<tr>
<td></td>
<td>• Minimum and maximum elapsed CPU times</td>
<td>• Total runs and total reruns</td>
<td>• Long duration</td>
</tr>
<tr>
<td></td>
<td>• Average duration</td>
<td></td>
<td>• Error code</td>
</tr>
<tr>
<td></td>
<td>• Late and long duration statistics</td>
<td></td>
<td>• Submission destination</td>
</tr>
<tr>
<td></td>
<td>• Total runs and total reruns</td>
<td></td>
<td>• Execution destination</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Report format:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Table view</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Include table of contents</td>
</tr>
</tbody>
</table>

Archiving historical data
### Table 40. Summary of historical reports (continued)

<table>
<thead>
<tr>
<th>Report name</th>
<th>Description</th>
<th>Selection criteria</th>
<th>Output content options</th>
</tr>
</thead>
</table>
| **Workstation workload summary** | Provides data on the workload in terms of the number of jobs that have run on each workstation. Helps making the necessary capacity planning adjustments (workload modeling, and workstation tuning). | • Workstation names. Fields can be specified using a wildcard.  
  • Submission destination  
  • Execution destination  
  • Date ranges or specific days for workload filtering.  
  • Relative time intervals (allows to reuse the same report task for running each day and getting the report of the production of the day before) | You can select from the following:  
  • Workstation information granularity arranged by:  
    – Hour  
    – Day  
  • Information aggregation options:  
    – Workstation  
    – Run date  
    – Submission destination  
    – Execution destination  
  • Report format:  
    – Charts view (bar, line)  
    – Table view  
    – Include table of contents by date or by workstation  

| **Workstation workload runtimes** | Provides data on the job runs (time and duration) on the workstations. Helps making the necessary capacity planning adjustments (workload modeling, and workstation tuning). | • Job and workstation names. Each field can be specified using a wildcard.  
  • Workload execution period  
  • Daily time intervals | You can select from the following:  
  • Job information:  
    – Workstation name  
    – Actual duration  
    – Status  
    – Iteration number  
    – Submission destination  
    – Execution destination  
    – Job number  
  • Information grouped by:  
    – Workstation  
    – Run date  
    – Submission destination  
    – Execution destination  
  • Report format:  
    – Charts view  
    – Table view  
    – Include table of contents  

| **Custom SQL**                  | A wizard helps you define your custom SQL query (only on the database views which you are authorized to access). The resulting report has a table with the column name as specified in the SELECT part of the SQL statement. | You can either write an SQL query statement to be run, or import a file containing the SQL statement that you want to use. | You can select from the following:  
  • Report format:  
    – Table view  
    – Include SQL statement |
To use the archiving capabilities of the host system and to be able to archive and run historical reports, set up the environment on the controller by performing the following steps:

1. Run EQQJOBS to create or modify the following skeletons:
   - **EQQDBARS**  Historical data archiver.
   - **EQQDPEXS**  Daily plan extend skeleton.
   - **EQQDPRCS**  Daily plan replan skeleton.
   
   and samples:
   - **EQQPCS08**  To set up the working directory.
   - **EQQPCS09**  To allocate VSAM and GDG files.
   - **EQQDBOPT**  To set DBOPT parameters.
   - **EQQDBENC**  To encrypt the password specified in the DBOPT statement.
   - **EQQSERP**  To set server parameters.
   - **EQQCONP**  To set batch options, if controller and tracker run in the same address space.
   - **EQQCONOP**  To set batch options, if controller and tracker run in separate address spaces.

2. Run EQQPCS09 to allocate a VSAM data set, used to archive data, and a Generation Data Group (GDG) root with the following characteristics:
   - **NOEMPTY**  Specifies that only the oldest GDG generation is to be deleted and uncataloged when the generation number reaches the maximum limit.
   - **SCRATCH**  Specifies that all the control information, related to a GDG generation, is to be deleted when the system deletes that generation.
   - **LIMIT(255)**  Specify how many versions will be kept in the generation data group.

3. Run EQQPCS08 to customize the working directory.

4. Customize the parameter library as follows. For details about the BATCHOPT, DBOPT, SERVOPTS, and AUTHDEF statements, refer to Customization and Tuning.
   - In the BATCHOPT initialization statement, set the JRUNHISTORY parameter to YES.
   - In the parameter library, add a member that contains only the DBOPT statement, specifying the parameters to connect to the database and manage the historical data archiving process.
   - In the SERVOPTS statement, set the DBOPTPRM parameter to the name of the member containing the DBOPT statement.
Setting up the environment

- Optionally, in the AUTHDEF statement set SUBRESOURCES (RP:REPTYPE) to include the RP:REPTYPE subresource. For details, refer to section “Setting up RACF authorization” on page 640.
- Encrypt the password to access the database. For details, refer to section “Encrypting the password in DBOPT statement” on page 639.

5. Set up the relational database. For details, refer to section “Setting up the database” on page 637.

6. Set up the RACF definitions. For details, refer to section “Setting up RACF authorization” on page 640.

7. Make sure you installed the REXX Compiler and you defined the TSO Compiler Programming Table, IRXCMPTM.

For details about steps 1-3, refer to Planning and Installation.

To populate the relational database with historical data, perform the following steps:

1. Run a daily planning EXTEND or REPLAN. During this step, the old current plan is copied into a Generation Data Grouping (GDG) entry data set by using a REPRO command. The GDG data set is identified in the daily planning EXTEND or REPLAN batch job by the EQQOCPBK ddname.

2. Periodically, archive the old current plans. Select option 3 (DAILY PLANNING) from the main menu. The following panel is displayed:

   EQQDPLNP ----------------- PRODUCING TWSz DAILY PLANS --------------------------
   Option ===> 

   Select one of the following:

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

3. Select option 6 (ARCHIVE) to display the following panel:
4. Press Enter to generate a JCL that, for each GDG entry related to the root you specified, will start a program that:
   • Converts the sequential GDG dataset to the predefined VSAM using REPRO.
   • Reads the records from the VSAM and archives the historical data in the database.
   • Deletes the processed GDG, if the process is successful.

   **Note:** You obtain a JCL tailored according to the EQQDBARS skeleton JCL, generated by the EQQJOBS dialog. Before submitting the JCL, specify at least 128 MB as region value.

## Setting up the database

To set up the database, use the product CD that contains the EQQD2* files:

- **EQQD2ZIP** To be used with Windows operating system.
- **EQQD2TAR** To be used with UNIX and Linux operating systems.

For a description of the product CD, refer to *Memo to Users*.

The previous files contain the script and SQL files you need to create the database. For a procedure to create the database, refer to “Creating the database” on page 637 section.

Extract the correct file content on the workstation where the DB2 server is installed.

### Creating the database

To create the database where historical run data is stored, you can run the following scripts, depending on the operating system you use:

- `dbsetup.bat` for Windows.
- `dbsetup.sh` UNIX and Linux.
Creating the database

In the same directory, the following files are also located:

- create_database.sql
- create_tables.sql
- create_constraints.sql
- create_indexes.sql

To set up the database, perform the following steps:

1. Save the script and SQL files locally on your workstation. The DB2 user needs read and write privileges on the directory where you save these files.

2. Depending on the operating system you are using, do the following:
   - If you use Windows, open a DB2 command window and access the directory where you saved the files.
   - If you use UNIX or Linux, log on as DB2 administrator and access the directory where you saved the files.

3. Before you activate the reporting feature for the first time, run `dbsetup` as follows:
   ```
   dbsetup dbName dbTableSpace dbDataDir port db2Admin db2AdminPwd
   ```
   where the parameters are positional and indicate the following:
   - `dbName`  
     The name of the database to be created.
   - `dbTableSpace`  
     The name of the user table space.
   - `dbDataDir`  
     The name of the directory where DB2 is to store the data.
   - `port`  
     The port number.
   - `db2Admin`  
     The DB2 administrator user ID.
   - `db2AdminPwd`  
     The DB2 administrator password.

   Example:
   ```
   dbsetup dbweb tsdbweb twsdbweb_data 50000 db2inst1 mypwd
   ```

4. The `dbsetup` command triggers the SQL definitions to create the database, the following DB2 tables, and the constraints, indexes, and views related to those tables:

   - **MDL.JOS_JOB_STATISTICS**
     To store statistics run data.
   - **MDL.JHR_JOB_HISTORY_RUNS**
     To store job run data.

5. To grant a user access to the views created in the previous step, use the `dbgrant` script. From a DB2 command prompt, enter the following command:
   ```
   dbgrant dbUser dbName db2Admin db2AdminPwd
   ```
   where the parameters are positional and indicate the following:
   - `dbUser`  
     The name of the user to be granted.
Creating the database

$dbName$
The name of the database to be used for the reporting feature.

$db2Admin$
The DB2 administrator user ID.

$db2AdminPwd$
The DB2 administrator password.

Example:
dbgrant dbweb webuser goofy goofypwd

Migrating the database to the latest version
To migrate an existing database to the latest schema version, you can run the following scripts, depending on the operating system you use:

• dbmigrate.bat for Windows.
• dbmigrate.sh UNIX and Linux.

In the same directory, there are other files used by the dbmigrate script. To migrate the database, perform the following steps:

• Save the script and SQL files locally on your workstation. Ensure that the DB2 user has read and write privileges on the directory where you save these files.
• Depending on the operating system you are using, do the following:
  – If you use Windows, open a DB2 command window and access the directory where you saved the files.
  – If you use UNIX or Linux, log on as DB2 administrator and access the directory where you saved the files.

• Run dbmigrate as follows:
dbmigrate $dbname $db2Admin $db2AdminPwd

where the parameters are positional and indicate the following:

$dbname$
The name of the database to be created.

$db2Admin$
The DB2 administrator user ID.

$db2AdminPwd$
The DB2 administrator password

For example:
dbmigrate DBWEB DB2INST1 MYPWD

• The dbmigrate command triggers the SQL definitions to modify the database tables, constraints, indexes, and views related to those tables as needed to align the database to the latest definitions.

Encrypting the password in DBOPT statement
To make the password set in the DBPSW parameter of the DBOPT statement not readable you can use either one of these three encrypting methods:

• Write the password in plaintext in the DBPSW parameter. The first time you run the archiving process, the password is automatically encrypted.
• Write the password in plaintext in the DBPSW parameter, and then encrypt it by submitting the EQQBENC JCL, which in turn runs a stand-alone application that checks the validity of the statement and encrypts the password.
Encrypting the password in DBOPT statement

- If you do not want to write the password in plaintext, you can set it already encrypted by running the shell command `RunChangeEncryptPwd` as follows:

  ```
  sh RunChangeEncryptPwd -c jarpath -d dataset -m membername -p pwd [-j javapath]
  ```

  where the parameters indicate the following:

  - **jarpath** The complete path where the jar files are stored.
  - **dataset** The data set name of the parameter library.
  - **membername** The member where the DBOPT statement is located.
  - **pwd** The password you are setting.
  - **javapath** The complete path where IBM SDK for z/OS is stored.

Setting up RACF authorization

Before Tivoli Workload Scheduler for z/OS executes any your request, a security verification check is passed to the System Authorization Facility (SAF) to ensure that you are authorized to access all the resources required to perform the request. You can use fixed resources and subresources to protect Tivoli Workload Scheduler for z/OS functions and data. When a console user requires a report, fixed resources (RP) are checked. Subresources (RP.REPTYPE) are checked only if they are defined in the AUTHDEF statement, for this reason you are required to add subresource RP.RETYPE in the AUTHDEF statement.

Allowed report types are RUNSTATS, RUNHIST, WWR, WWS, and SQL. Define these reports to RACF with the following command:

```
RDEF ibmopc RPT.reptype UACC(read)
```

Then authorize the reports to RACF with the following command:

```
PE RPT.reptype ID(userid) ACCESS(read) CLASS(ibmopc)
```

For detailed information about how to protect resources through RACF, refer to the `Customization and Tuning` manual.

Running batch reports from the command line interface

This section describes how you can run from the command line the reports listed in "Historical reports" on page 633.

The ability to run such reports from a command line interface allows you to schedule your reports so that you can trigger them on a timely basis.

A sample business scenario

To avoid unexpected slowing down in the workload processing, the analyst of a big company needs weekly reports collecting historical information about the processed workload to determine and analyze any workload peaks that might occur.

To satisfy this request, the Administrator creates `Workload Workstation Summary Reports` (WWS) and `Workload Workstation Runtimes Reports` (WWR).

To accomplish his task, he runs the following steps:
A sample business scenario

1. He customizes the property files related to the Workload Workstation Summary and Workload Workstation Runtimes reports, specifying the format and content of the report output.

2. He schedules jobs to obtain WWS and WWR reports:
   - The first job generates a WWS report to be saved locally
   - The second job runs a WWR report overnight on expected workload peaks time frames. The report output is sent using an e-mail to the analyst. The information collected are used to optimize the workload balance on the systems.

3. He adds the two jobs to a job stream scheduled to run weekly and generates the plan.

Setting up for command line reporting

Before running batch reports you must run a few setup steps:

1. The software needed to run batch reports is contained in a package included in the Tivoli Workload Scheduler installation image. If you plan to run batch reports from within a scheduled job, extract the package file on one of the platforms listed in [http://www.ibm.com/support/docview.wss?rs=672&uid=swg27020800](http://www.ibm.com/support/docview.wss?rs=672&uid=swg27020800).

   After you unzip you will have the following file structure:

```
| config |
| jars |
| jre |
| notification |
| ReportEngine |
| reports |
| common_logging.properties |
| logging.properties |
| reportcli.cmd |
| reportclish |
```

Because the native UNIX tar utility does not support long file names, if you are extracting the files on AIX, Solaris, or HP-UX systems, ensure that the latest GNU version of tar (gtar) is installed to extract the files successfully.

Notes:

a. Make sure you run the following commands in the directory where you extracted the files:

   **On UNIX**
   ```
   chmod -R +x *
   chown -R username *
   ```

   **On Windows**
   Make sure Tivoli Workload Scheduler is installed.
   ```
   setown -u username *
   ```

   where `username` is the user that will run the reports.

b. If you plan to schedule jobs that run batch reports, the system where you extract the package must be accessible as network file system from a Tivoli Workload Scheduler for z/OS defined in the local scheduling environment.

2. Configure the template file `.\config\common.properties` specifying the information to:
   a. Connect to the database where the historical data are stored.
Setting up for command line reporting

b. Set the date and time format, including the time zone. The file
   .\config\timezone.txt contains a list of time zones supported by Tivoli
   Workload Scheduler and the information on how to set them. The time zone
   names are case sensitive.

c. Make available the report output on the URL specified in ContextRootUrl
   field. This is an example of configuration settings:

   ##############################################################################
   # HTTP Server information
   ##############################################################################

   # Specify the context root where the report will be available
   # To leverage this possibility it needs to specify in the report output dir
   # the directory that is referred by your HTTP Server with this context root

   ContextRootUrl=http://myserver/reportoutput

   In this case make sure that the output_report_dir
   specified when running the batch reports command points to the same directory specified in the
   ContextRootUrl field.

d. Send the report output using a mail. This is an example of configuration
   settings:

   ##############################################################################
   # Email Server configuration
   ##############################################################################

   PARAM_SendReportByEmail=true
   #SMTP server
   mail.smtp.host=myhost.mydomain.com
   #IMAP provider
   mail.imap.socketFactory.fallback=false
   mail.imap.port=993
   mail.imap.socketFactory.port=993
   #POP3 provider
   mail.pop3.socketFactory.fallback=false
   mail.pop3.port=995
   mail.pop3.socketFactory.port=995

   ##############################################################################
   # Email properties
   ##############################################################################

   PARAM_EmailFrom=user1@your_company.com
   PARAM_EmailTo=user2@your_company.com,user3@your_company.com
   PARAM_EmailCC=user4@your_company.com
   PARAM_EmailBCC=user5@your_company.com
   PARAM_EmailSubject=Test send report by email
   PARAM_EmailBody=This is the report attached

   An explanation of all the customizable fields is contained in the template file.

   Note: If you plan to run Workstation Workload Runtime reports ensure that the
   file system where database is installed has enough free space. If a shortage
   of disk space occurs an SQL exception like the following is triggered:

   DB2 SQL error: SQLCODE: -968, SQLSTATE: 57011

Running batch reports

The \reports\templates directory contains a sample template file for each type of
report.

Before running any of these reports make sure you customize the corresponding
template file.
In that file, named *report_name.properties*, you can specify:

- The information to display in the report header.
- How to filter the information to display the expected result.
- The format and content of the report output.

For more information about the specific settings see the explanation provided in the template file beside each field.

If you are using DBCS characters to specify the parameters in the template .properties files, ensure you save the file in UTF-8 encoding.

After you set up the environment as it is described in "Setting up for command line reporting" on page 641, and you configured report template file, use the following syntax to run the report:

```
reportcli -p report_name.property
  [-o output_report_dir]
  [-r report_output_name]
  [-k key=value ]
  [-k key=value ]
  .......
```

where

- **-p** *report_name.property*
  Specifies the path name to the report template file.

- **-o** *output_report_dir*
  Specifies the output directory for the report output.

- **-r** *report_output_name*
  Specifies the name of the report output.

- **-k key=value**
  Specifies the value of a settings. This value override the corresponding value, if defined, in the common.properties file or in the report_name.properties file.

### Examples

1. In this example the reportcli.cmd is run with the default parameter and produces the report *jrh1*:

   ```
   reportcli.cmd -p D:\ReportCLI\TWSReportCli\reports\templates\jrh.properties -r jrh1
   ```

2. In this example the reportcli.cmd is run using the -k parameter to override the values set for **PARAM_DateFormat** in the .\config\common.properties file and produces the report *jrh2*:

   ```
   reportcli.cmd -p D:\ReportCLI\TWSReportCli\reports\templates\jrh.properties -r jrh2 -k PARAM_DateFormat=short
   ```

3. In this example the reportcli.cmd is run using the -k parameter to override the format specified for the report output in the PROPERTIES file and produces the report *wwr3*:

   ```
   ./reportcli.sh -p /TWSReportCli/REPCLI/reports/templates/wwr.properties -r wwr3 -k REPORT_OUTPUT_FORMAT=html -k OutputView=charts
   ```

4. Do the following if you want to run a Custom SQL report and make available the output of the report at the following URL as http://myserver/reportoutput/report1.html:
Exmaples

a. Configure the `ContextRootUrl` parameter in the common.properties files as follows:

```
# HTTP Server information

#Specify the context root where the report will be available
#To leverage this possibility it needs to specify in the report output dir
#the directory that is referred by your HTTP Server with this context root

ContextRootUrl=http://myserver/reportoutput
```

b. When you run a command specify as `output_report_dir` a directory that points to the same HTTP directory specified in the `ContextRootUrl`. For example, if you mapped locally the `http://myserver/` as `R:` driver, you can run the following command:

```
reportclibat
   -p REPORT_CLI_DIR\reports\TWS\historical\templates\sql.properties
   -r report1
   -o R:\reportoutput
```

c. As a confirmation for the successful run of the report, the following message is displayed:

```
```

This URL shows where the report output is available.

Note: If the report is run through a Tivoli Workload Scheduler job, the output of the command is displayed in the job output.

Logs and traces for batch reports

The file `./common_logging.properties` contains the parameters you can use to configure tracing and logging.

The file contains the following settings:

```
logFileName=reportcli.log
traceFileName=trace.log
trace=off
birt_trace=off
```

where

- `logFileName` specifies the name of the file containing generic information, warning about potential problems, and information about errors. This file is stored under `./log`.

- `traceFileName` specifies the name of the file containing traces. If you set `trace=on` the trace file is stored under `./log`.

- `trace` specifies whether to enable or not traces. Enable the traces by setting `trace=on` if you want to investigate further about an error.

- `birt_trace` specifies whether to enable or not traces to diagnose errors in BIRT engine. If you set `birt_trace=on` a file containing the trace and named `ReportEngine_aaaa_mm_dd_hh_mm_ss.log` is stored in the `/ReportEngine/logs` folder.
Keeping historical data in the database

The data stored in the database is kept according to the clean up policy you specified with the CLEANUPPOLICY parameter of the DBOPT initialization statement. With this parameter you can specify the number of days you want to keep the data in the tables. The default value is 10.

For details about the DBOPT statement, refer to Customization and Tuning.
Keeping historical data in the database
Appendix A. TSO commands

This appendix describes the TSO commands and how you can use them in batch mode, using the event-generating batch program, EQQEVPGM. See "EQQEVPGM - Issue commands in batch" on page 691 for an example of the JCL that you need for EQQEVPGM. Examples of these commands are provided for both online and batch invocation.

If you want to invoke these as TSO commands from a system where a controller is not running, they must first be defined as authorized TSO commands on that system. Your system programmer can do this by adding the commands to the list defined by the NAMES keyword of the AUTHCMD statement in member IKJTSOxx of SYS1.PARMLIB. Refer to the Planning and Installation for more information.

Note: When EQQEVPGM is used to issue TSO commands, statement data must be in columns 1 through 72. Information in columns 73 through 80 is ignored. You can abbreviate keywords to their shortest unambiguous form. For example, you can shorten the AVAIL keyword to an ‘A’.

If you plan to issue the TSO commands many times per day from a long-running non-TSO address space (for example, NetView) it is recommended that you use a Tivoli Workload Scheduler for z/OS subroutine instead. If you plan to run PIF applications many times per day from a long-running non-TSO address space (for example, NetView), it is recommended that you do not specify the EQQYPARM ddname. When you either do one of the following:

- Issue the commands from TSO or as input to the EQQEVPGM program
- Run a PIF application by specifying the EQQYPARM ddname

a TSO environment must be established each time and some of the resources remain allocated until the task ends, which might lead to a storage shortage if the commands are issued many times.

For detailed information about the subroutines, see Customization and Tuning.

Important considerations for using the TSO commands

Consider these points when using the TSO commands:

- The parameters that you pass to the TSO commands are checked only for the correct format; that is, numeric fields must contain only numbers within a valid range, date fields must contain valid dates, and so on. The parameters are not checked for their validity for a particular Tivoli Workload Scheduler for z/OS address space. For example, the workstation name that you specify for OPSTAT is not verified against the actual workstations that exist in a particular current plan. In addition, a single event record can be generated to be used in two or more Tivoli Workload Scheduler for z/OS address spaces. A particular parameter (for example, the application description ID) might be valid for one address space and not another.

As long as the minimum parameter requirements are met and the parameters are in the correct format, the TSO commands will execute successfully and generate an event record.
Important considerations for using the TSO commands

When the event is processed by the controller, it is checked for validity. If any errors are found, an error message is written to the controller message log (EQQMLOG).

- You can use the TSO commands even if Tivoli Workload Scheduler for z/OS (in particular, the event writer subtask) is not active. Event records are still generated and placed in the event writer queue. When the event writer starts, the event records are removed from the queue and written to the event data set.

For a description of the TSO commands and their usage, refer to the following sections. The commands are listed alphabetically.

BACKUP

Purpose

The BACKUP command is used to initiate a backup of the current plan (CP) or the JCL repository (JS) on request. You can request the backup process by issuing the BACKUP command from your TSO session or from within a batch job. You can schedule current plan or JS backups by defining the job as an operation in the current plan.

The JS and current plan files are managed by the controller. A request generated by the BACKUP command is communicated to the controller as an event record processed by the event writer task of a tracker. The BACKUP command can be issued from any z/OS system that runs a tracker from Tivoli Workload Scheduler for z/OS Release 2 or a later release. If you request a BACKUP on multiple systems at the same time, you will cause multiple file backups to occur when the events reach the controller.

A current plan backup is performed automatically under these circumstances:
- During normal shutdown of the controller.
- At the beginning and end of the daily planning process.
- When the number of new job-tracking records is greater than the value specified by the JTOPTS keyword BACKUP. But if the JTOPTS BACKUP keyword specifies NO, these regular backups of the current plan are not taken.

You can also request an immediate current plan backup at any time by issuing the BACKUP command for the current plan resource. You might do this to:
- Request a backup at a predefined time for disaster recovery purposes.
- Ensure regular backups are taken when Tivoli Workload Scheduler for z/OS activity is low.
- Take backups only at set times, in which case the JTOPTS BACKUP keyword is probably set to NO.

When the JTOPTS BACKUP keyword specifies a numeric value, a counter increases by one every time a new job-tracking record is written. The counter is reset to zero after every current plan backup.

Backup of the JS file is performed automatically at regular intervals based on the value defined in the MAXJSFILE keyword of the JTOPTS initialization statement. You can also issue the BACKUP command for the JS resource at any time to schedule an immediate backup of the JS file. If the value of the MAXJSFILE keyword is specified as NO, regular backups of the JS file are not taken. You might do this to ensure that backups are only taken at a time when the disruption is
minimized. During a JS file backup, the current plan resource is locked to prevent panel users and other Tivoli Workload Scheduler for z/OS tasks from updating the JCL for operations in the current plan.

**Usage Notes**

You can invoke the BACKUP command as a TSO command or by using a batch job that executes program EQQEVPGM. If you invoke BACKUP as a TSO command, you must allocate the EQQMLIB data set to the address space of the TSO user, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command execution.

Use of the BACKUP command can be restricted with the fixed resource code BKP. The authority of the requestor is verified by the subsystem name identified in the command if an AUTHDEF statement is defined for that subsystem. When SUBSYS(MSTR) is specified, all subsystems defined on the z/OS system to which the command is directed will attempt to verify the authority of the requestor before an event is generated. It is possible to be rejected by one subsystem and accepted by another.

You must be defined with update authority to the BKP resource to use the BACKUP command. Resource codes are documented in *Customization and Tuning*. Check with your administrator to confirm that you have this authority before you use BACKUP.

The subsystem to which you direct the command does not have to be active when the command is issued. An event will be generated and queued in CSA along with other job-tracking events. If the subsystem is not active when the command is issued, the authority of the requestor is verified using the class name specified in the AUTHDEF statement when the subsystem was last started. If the subsystem has not been started since a z/OS IPL, no authority verification can be performed.

**Syntax**

```
BACKUP RESDS (CP | JS) [TRACE(trace level)] SUBSYS (MSTR | subsystem name | OPC)
```

**Parameters**

**RESDS (CP | JS)**

The RESDS keyword specifies which data set the backup will be performed on. If you specify CP as the keyword value, a current plan backup will be performed. If you specify JS as the value for this keyword, the JCL repository data set will be copied to the alternate JS file.

The RESDS keyword must be specified.

**SUBSYS (MSTR | subsystem name | OPC)**

The name of the tracker subsystem that the BACKUP command is directed to.
The name can be up to four characters long. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase. If the SUBSYS keyword is omitted the on-request backup command will be directed to a subsystem called OPCA.

If you specify **MSTR**, the BACKUP command is directed to all tracker subsystems on the z/OS system that the BACKUP command was directed to.

**Attention:** This will cause multiple backups to occur if more than one tracker subsystem is active on the system.

**Subsystem name** is the name of the subsystem that the BACKUP command will be directed to. If the tracker and controller run in separate address spaces in your Tivoli Workload Scheduler for z/OS configuration, specify the name of the tracker subsystem in this parameter.

**TRACE** *(level 0)*
Event tracing indicator. When a positive number is specified, a trace entry is created for each event generated by the BACKUP command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records.

**Examples**
These two examples demonstrate how you can use the BACKUP command in TSO, or in a batch job (using the batch program EQQEVPGM).

<table>
<thead>
<tr>
<th>BACKUP</th>
<th>Example 1 - TSO command</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE</td>
<td></td>
</tr>
<tr>
<td>BACKUP RESDS(CP) SUBSYS(OPCA)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Example 2 - Batch job</th>
</tr>
</thead>
<tbody>
<tr>
<td>//CPBACKUP JOB (ACCOUNT),'Backup CP',CLASS=A</td>
</tr>
<tr>
<td>//STEP1 EXEC PGM=EQQEVPGM</td>
</tr>
<tr>
<td>//STEPLIB DD DSN=OPC.LOAD MODULE.LIBRARY,DISP=SHR</td>
</tr>
<tr>
<td>//EQQMLOG DD SYSOUT=A</td>
</tr>
<tr>
<td>//SYSIN DD *</td>
</tr>
<tr>
<td>BACKUP RESDS(CP) SUBSYS(OPCA)</td>
</tr>
</tbody>
</table>

In both of these examples, the current plan data set will be copied to the old current plan data set, on a subsystem called OPCA.
Purpose

The BULKDISC command is used to initiate a bulk discovery request. You can request a bulk discovery by issuing the BULKDISC command from your TSO session or from within a batch job. You can run a complete job discovery manually or each time a planning activity is performed (create, extend or replan), by setting the MONOPTS runtime option for the controller.

A request generated by the BULKDISC command is communicated to the controller as an event record processed by the event writer task of a tracker. The BULKDISC command can be issued from any z/OS system that runs a controller from Tivoli Workload Scheduler for z/OS Release 3, or later. The command initiates a bulk discovery. While the bulk discovery is in progress, the current plan backup resource is locked to prevent other Tivoli Workload Scheduler for z/OS tasks from updating the same file concurrently.

Usage

You can invoke the BULKDISC command as a TSO command or by using a batch job that runs program EQQEVPGM. If you invoke BULKDISC as a TSO command, you must allocate the EQQMLIB data set to the address space of the TSO user, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command completion.

Use of the BULKDISC command can be restricted with the fixed resource code BUL. The authority of the requestor is verified by the subsystem name identified in the command if an AUTHDEF statement is defined for that subsystem. When SUBSYS(MSTR) is specified, all subsystems defined on the z/OS system to which the command is directed attempt to verify the authority of the requestor before an event is generated. It is possible to be rejected by one subsystem and accepted by another.

You must be defined with update authority to the BUL resource to use the BULKDISC command. Resource codes are described in Customization and Tuning. Check with your administrator to confirm that you have this authority before you use BULKDISC.

If the subsystem to which you direct the command is not active when the command is issued, an event is generated and queued in CSA together with other job-tracking events. In this case, the authority of the requestor is verified using the class name specified in the AUTHDEF statement when the subsystem was last started. If the subsystem has not been started since a z/OS IPL, no authority verification can be performed. For the bulk discovery to run successfully, the monitoring task must be active when the BULKDISC command is issued.

Format

```
BULKDISC

TRACE(trace level) SUBSYS(subsystem name)

OPCA
```
Parameters

**CRITJ**
If you work with IBM Tivoli Monitoring and specified the CRITICAL value in the MONPOL initialization statement, use this parameter to obtain all the available information about the critical jobs or the jobs in a critical path.

**SUBSYS (MSTR | subsystem name | OPCA)**
The name of the controller that the BULKDISC command is directed to. The name can be up to four characters long. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase. If the SUBSYS keyword is omitted, the on-request BULKDISC command is directed to a subsystem called OPCA.

If you specify MSTR, the BULKDISC command is directed to all controller subsystems on the z/OS system that the BULKDISC command was directed to.

*Note*: Specifying MSTR causes multiple bulk discoveries to occur, if more than one controller subsystem is active on the system.

Subsystem name is the name of the subsystem that the BULKDISC command is directed to. If the tracker and controller run in separate address spaces in your Tivoli Workload Scheduler for z/OS configuration, specify the name of the tracker subsystem in this parameter.

**TRACE (level | 0)**
Event tracing indicator. When a positive number is specified, a trace entry is created for each event generated by the BULKDISC command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 does not generate trace records.

Examples
These two examples demonstrate how you can use the BULKDISC command in TSO, or in a batch job (using the batch program EQQEVPGM).

```
BULKDISC

Example 1 - TSO command
ALLOC F(EQQLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE
BULKDISC SUBSYS(OPCA)

Example 2 - Batch job
//BULKJOB JOB (ACCOUNT), 'Bulkdj', CLASS=A
//STEP1 EXEC PGM=EQQEVPGM
//STEPLIB DD DSN=OPC.LOAD.MODULE.LIBRARY, DISP=SHR
//EQQLIB DD DSN=OPC.MESSAGE.LIBRARY, DISP=SHR
//EQQMLOG DD SYSOUT=A
//SYSIN DD *
BULKDISC SUBSYS(OPCA)
/*
```
**JSUACT**

**Purpose**

The JSUACT command is used to activate or inactivate the job submission function.

**Usage**

You can invoke the JSUACT command as a TSO command or by using a batch job that executes program EQQEVPGM. If you invoke JSUACT as a TSO command, you must allocate the EQQMLIB data set to the TSO user address apace, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command execution.

In any case, appropriate messages are issued in the controller log to indicate whether or not job submission has been activated.

Use of the JSUACT command can be restricted using the fixed resource JSUB. The authority of the requestor is verified by the subsystem name identified in the command if an AUTHDEF statement is defined for that subsystem. When SUBSYS(MSTR) is specified, all subsystems defined on the z/OS system to which the command is directed will attempt to verify the authorization. It is possible to be rejected by one subsystem and accepted by another.

You must be defined with update authority to the JSUB resource to be able to use the JSUACT command. Resource codes are described in *Customization and Tuning*. Check with your scheduler administrator to confirm that you have this authority before trying to use JSUACT.

The subsystem to which you direct the command does not have to be active when the command is issued. An event will be generated and queued in CSA along with other job-tracking events. If a subsystem is not active when the command is issued, the authorization of the requestor is verified by using the class name specified in the AUTHDEF statement when the subsystem was started. If the subsystem has not been started since a z/OS IPL, no authorization verification can be performed.

**Format**

```
JSUACT ACT(YES | NO) SUBSYS(MSTR subsystem name)
 TRACE(trace level) TYPE(F | B)
```

**Parameters**

**ACT (Y | N)**

If you want to activate the job submission function specify Y, otherwise N.
JSUACT

**SUBSYS (MSTR | subsystem name | OPCA)**

The name of the tracker the JSUACT is directed to.

This parameter can be four characters in length. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase.

If you specify MSTR, the JSUACT command is directed to all scheduler subsystems on the z/OS system where the JSUACT command was issued.

**TRACE (level | 0)**

Event tracing indicator. When a nonzero positive number is specified, a trace entry is created for each event generated by the JSUACT command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records.

**TYPE (B | F | H)**

Indicates whether the job submission must be deactivated for the host (H) or for the distributed environment (F). To deactivate job submission in both environments, specify B.

**Examples**

These two examples demonstrate how you can use the JSUACT command in TSO, or in a batch job (using the batch program EQQEVPGM).

```
//JSUACT JOB (ACCOUNT),'Deactivate',CLASS=A
//STEP1 EXEC PGM=EQQEVPGM
//STEPLIB DD DSN=OPC.LOAD.MODULE.LIBRARY,DISP=SHR
//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY,DISP=SHR
//EQQMLOG DD SYSOUT=A
//SYSIN DD *
JSUACT ACT(N) SUBSYS(OPCB)
/*
```
Purpose

Use the OPINFO command to set the user data of an operation in the current plan. The user data can contain any data you require. In many installations, the field is used to record the problem record number for operations that have ended in error. OPINFO can be incorporated into your Information/Management interface to enable immediate feedback of the problem report to Tivoli Workload Scheduler for z/OS.

Usage

You can invoke OPINFO as a TSO command or by using a batch job which executes program EQQEVPGM. If you invoke OPINFO as a TSO command, allocate the EQQMLIB data set to the address space of the TSO user, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command execution.

You use the OPINFOSCOPE keyword of the JTOPTS to specify the scope of the command. If OPINFOSCOPE is set to (or defaults to) IP, Tivoli Workload Scheduler for z/OS considers only in-progress operations (where the current status is R, A, *, S, I, or E), and will not action OPINFO events for operations in any other status. If OPINFOSCOPE is set to ALL, Tivoli Workload Scheduler for z/OS also considers operations with W and C status.

With the OPINFO command, you specify the WSNAME, ADID, IA, OPNUM, or JOBNAME parameters to identify the operation whose user data is to be updated. If the OPINFOSCOPE keyword is IP, which is the default, WSNAME is a required parameter. If OPINFOSCOPE is ALL, the ADID or JOBNAME parameter is required. Also, for printer workstations, you can specify the CLASS and FORM parameters to identify the operation.

If you do not provide enough information to uniquely identify the operation, and Tivoli Workload Scheduler for z/OS finds more than one operation that matches your parameters, Tivoli Workload Scheduler for z/OS chooses the best operation. If OPINFOSCOPE is IP, Tivoli Workload Scheduler for z/OS uses this list, taking each item until it gets a single operation:
1. Priority 9
2. Earliest latest start time
3. Highest priority, if lower than 9
4. Earliest operation input arrival time, or occurrence input arrival time if the operation does not have input arrival specifically defined
5. Longest in Ready status

That is, if Tivoli Workload Scheduler for z/OS determines that there is more than one in-progress operation in the current plan, the operation with priority 9 is updated. If more than one operation specifies priority 9, or there are no priority 9 operations, the operation with the earliest latest start time is updated. The latest start is blank if the operation is added using the MCP panel. If the latest start time is equal, the operation with the highest priority is updated, and so on.
If OPINFOSCOPE is ALL, Tivoli Workload Scheduler for z/OS uses the same list as for OPINFOSCOPE(IP) to find a single operation. If no in-progress operation that matches your parameters is found, Tivoli Workload Scheduler for z/OS also searches operations with status C and W in the current plan. The operation with the earliest latest start time is selected.

Use of the OPINFO command can be restricted with the fixed resource code CP. The authority of the requestor is verified by the subsystem name identified in the command if an AUTHDEF statement is defined for that subsystem. When SUBSYS(MSTR) is specified, all tracker subsystems defined on the z/OS system where the OPINFO command is issued will attempt to verify the authority of the requestor before data will be passed. You might be rejected by one subsystem and accepted by another.

You need update authority to resource code CP to use this command. Resource codes are documented in *Customization and Tuning*.

The subsystem to which you direct the command does not have to be active when the command is issued. An event will be generated and queued in CSA along with other job-tracking events. If the subsystem is not active when the command is issued, the authority of the requestor is verified using the class name specified in the AUTHDEF statement when the subsystem was last started. If the subsystem has not been started since a z/OS IPL, no authority verification can be performed.

**Format**

```plaintext
<table>
<thead>
<tr>
<th>Format</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPINFO</td>
</tr>
<tr>
<td>ADID(ADID)</td>
</tr>
<tr>
<td>CLASS(CLASS)</td>
</tr>
<tr>
<td>IA(IA)</td>
</tr>
<tr>
<td>JNAME(JOBNAME)</td>
</tr>
<tr>
<td>OPCODE(OPNUM)</td>
</tr>
<tr>
<td>OPCODE(SUBSYS)</td>
</tr>
<tr>
<td>OPCODE(TRACE)</td>
</tr>
<tr>
<td>OPCODE(USERDATA)</td>
</tr>
<tr>
<td>OPCODE(WNAME)</td>
</tr>
</tbody>
</table>
```

**Parameters**

**ADID (application description identifier)**

The application identifier of the operation whose user data is to be updated. If the OPINFOSCOPE keyword of the JTOPTS statement is ALL, ADID is a required parameter.

**CLASS (printer sysout class)**

For a printer workstation, specifies the printer SYSOUT class of the operation whose user data is to be updated.
FORM *(printer form name)*
For a printer workstation, specifies the printer FORM name of the operation whose user data is to be updated.

IA *(yymmddhhmm)*
The input arrival date and time of the occurrence containing the operation whose user data is to be updated. You must specify this in the format *yymmddhhmm*.

**Note:** Tivoli Workload Scheduler for z/OS interprets the *yy* part as follows:

<table>
<thead>
<tr>
<th>YY</th>
<th>Year</th>
<th>72 - 99</th>
<th>1972 - 1999</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>00-71</td>
<td>2000 - 2071</td>
<td></td>
</tr>
</tbody>
</table>

JOBNAME *(job name)*
The job name associated with the operation whose user data is to be updated. If the OPINFOSCOPE keyword of the JTOPTS statement is ALL, JOBNAME is a required parameter.

OPNUM *(operation number)*
The operation number of the operation whose user data is to be updated.

SUBSYS *(MSTR | subsystem name | OPCA)*
Name of the tracker subsystem that the OPINFO command is directed to. The name can be up to 4 characters long. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase.

If you specify MSTR, the OPINFO command is directed to all tracker subsystems on the z/OS system where the OPINFO command is issued.

**Note:** If the trackers and controller run on different subsystems in your configuration, specify the name of the tracker subsystem in this parameter.

TRACE *(level | 0)*
Event tracing indicator. When a positive number is specified, a trace entry is created for each event generated by the OPINFO command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records.

USERDATA *(any data)*
You can use this 16-character parameter to pass information about an operation to the current plan in the *operation user data*. The USERDATA field cannot contain any blanks.

WSNAME *(workstation name)*
The name of the workstation for the operation whose user data is to be updated. If the OPINFOSCOPE keyword of the JTOPTS statement is IP, WSNAME is a required parameter.

**Examples**
These two examples demonstrate how you can use the OPINFO command in TSO, or in a batch job (using the batch program EQQEVPGM).
Example 1 - TSO command

ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE

OPINFO W(BDEC) J(DNCD3000) A(ACLMSDLY) U(USEREDATAHERE)

Example 2 - Batch job

//OPINFOUS JOB (ACCOUNT),'Set completed',CLASS=A
//STEP1 EXEC PGM=EQQEVPGM
//STEPLIB DD DSN=OPC.LOAD.MODULE.LIBRARY,DISP=SHR
//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY,DISP=SHR
//EQQMLOG DD SYSOUT=A
//SYSIN DD *
OPINFO W(BDEC) J(DNCD3000) A(ACLMSDLY) U(PROBREC01234567)
/*/

In both of these examples, an operation for application ACLMSDLY at workstation BDEC will have the data fed back to it.
OPSTAT

Purpose

The OPSTAT command lets you set the status of an operation at any workstation, except workstations that have the nonreporting attribute. Events generated by OPSTAT are matched against operations on the ready list. Events received for operations in waiting (W), suppressed by condition (X), or complete (C) status are ignored. Jobs and started tasks that are running are always allowed to finish; for rules governing the changing of operation status, see the READY LIST panel description in “Using the ready list” on page 525.

You need update authority to resource code RL to use this command. Resource codes are documented in Customization and Tuning.

The OPSTAT command gives you a portable method of using automatic event reporting (AER) facility. AER can help you coordinate many tasks that are not normally seen by Tivoli Workload Scheduler for z/OS. For example, you can use AER to trigger the start of an operation when a particular step in a job is complete, or as acknowledgment that a file has been received across the network.

The OPSTAT command also gives you the facility to automatically report the status of work executing in operating environments that do not support a tracker.

Usage

You can invoke OPSTAT as a TSO command or by using a batch job that executes program EQQEVPGM. If you invoke OPSTAT as a TSO command, allocate the EQQMLIB data set to the address space of the TSO user, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command execution.

OPSTAT is an alternative to the EQQUSIN subroutine for implementing automatic event reporting for general workstations.

With the OPSTAT command, you must specify the WSNAM E parameter to identify the workstation at which the operation is changing status. Tivoli Workload Scheduler for z/OS then changes the status of the operation at that workstation to the status you have specified.

If there is more than one operation at the workstation, you can optionally specify the ADID, IA, OPNUM, or JOBNAME parameters to identify the particular operation whose status is to be changed. Also, for printer workstations, you can specify the CLASS and FORM parameters to identify an operation.

If you do not provide enough information to uniquely identify the operation and Tivoli Workload Scheduler for z/OS finds more than one operation which matches the criteria you specified, Tivoli Workload Scheduler for z/OS determines the most appropriate operation to update. Tivoli Workload Scheduler for z/OS chooses the most appropriate operation by investigating its characteristics in this order:

1. The operation has priority 9.
2. Earliest latest start time.
3. Priority 8-1.
4. Input arrival time specified for the operation, or the occurrence input arrival if
the operation does not have input arrival specifically defined.

Therefore, if you define only the WSNAMESPACE parameter and Tivoli Workload
Scheduler for z/OS determines that there is more than one operation in the current
plan for that workstation in status R, A, *, S, I, or E, the operation with priority 9 is
updated. If more than one operation specifies priority 9, or there are no priority 9
operations, the operation with the earliest latest-start time is updated. If you add
the operation using the MCP panel, the latest start time is blank. If latest start is
equal, the operation with the highest priority is updated. If priority is equal, the
operation which specifies the earliest input arrival time is updated. If input arrival
is also equal, the update is performed on a first-in-first-out basis.

Use of the OPSTAT command can be restricted with fixed resource code RL and
the subresource RL.WSNAMESPACE. The authority of the requestor is verified by the
subsystem name identified in the command if an AUTHDEF statement is defined
for that subsystem. When SUBSYS(MSTR) is specified, all tracker subsystems
defined on the z/OS system where the OPSTAT command is issued will attempt to
verify the authority of the requestor before an event is generated. You might be
rejected by one subsystem and accepted by another.

The subsystem to which you direct the command does not have to be active when
the command is issued. An event will be generated and queued in CSA along with
other job-tracking events. If the subsystem is not active when the command is
issued, the authority of the requestor is verified using the class name specified in
the AUTHDEF statement when the subsystem was last started. If the subsystem
has not been started since a z/OS IPL, no authority verification can be performed.

Format

```
OPSTAT
    ADID( application description ID )

    CLASS( printer sysout class )
    DURATION( hhm )

    ERRORCODE( operation error code )
    EVDATE( yymmmdd )

    EVTME( hhm )
    FORM( printer form name )

    IA( yymmmddhhmm )
    JOBNAME( job name )

    NUMJOB( job number )
    OPCODE( operation number )

    TOKEN( token value )
    STATUS( )

    SUBSYS( MSTR subsystem name )
    TRACE( trace level )
```
OPSTAT

Parameters

ADID (application description identifier)
The application identifier of the operation whose status you want to change.

CLASS (printer sysout class)
For a printer workstation, specifies the printer SYSOUT class of the operation whose status you want to change.

DURATION (hhmm)
If you are specifying STATUS(C) to set the operation status to complete, you can optionally specify a duration for the completed operation. You specify the duration in hours and minutes, in the format hhmm.

ERRORCODE (operation error code)
If you are specifying STATUS(E) to set the operation status to ended-in-error, you can optionally specify an error code for the operation. The error code can be any 4 characters.

EVDATE (yymmda)
The date of this operation status event. You must specify the date in the format yymmda.

See the comments under the following parameter, EVTIME.

EVTIME (hhmm)
The time of this operation status event. You must specify the time in the format hhmm.

You can use the EVDATE and EVTIME parameters if you want to indicate that the operation changed status at a time other than the current time. If you do not supply these parameters, the operation is considered to have changed status at the time Tivoli Workload Scheduler for z/OS processes the OPSTAT command.

FORM (printer form name)
For a printer workstation, specifies the printer FORM name of the operation whose status you want to change.

IA (yymmdhhmm)
The input arrival date and time of the occurrence containing the operation whose status you want to change. You must specify this in the format yymmdhhmm.

Note: Tivoli Workload Scheduler for z/OS interprets the yy part as follows:

<table>
<thead>
<tr>
<th>YY</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>72</td>
<td>1972 - 1999</td>
</tr>
<tr>
<td>00</td>
<td>2000 - 2071</td>
</tr>
</tbody>
</table>

JOBNAME (job name)
The job name associated with the operation whose status you want to change.

NUMJOB (job number)
Use this optional parameter to specify a job number for an operation on a user-defined computer automatic workstation. Specify a number from 0 to 999999. Tivoli Workload Scheduler for z/OS builds a job number in the format USRnnnnn or Unnnnnnn, depending on the value and padding the number with zeros on the left.
OPSTAT

OPNUM (operation number)
The operation number of the operation whose status you want to change.

TOKEN (token value)
The token assigned for the operation whose status you want to change. A token is automatically assigned for operations started on workstations that specify a user-defined destination ID. The token can be used to uniquely identify the operation.

When TOKEN is used in conjunction with the ADID, IA, JOBNAME, or OPNUM parameters, all values must match the target operation. For example, if the token identifies the operation but OPNUM is also specified and does not match, the event will be rejected and message EQQE091E written to the controller message log.

Specify the token as a hexadecimal value in the format TOKEN(X'00ABCDEF').

STATUS (C)
Use the default value to set the new operation status as successfully completed at the workstation.

Changes to operation status using OPSTAT follow the same rules as status changes in the Ready List. In particular:

- You cannot change an operation status from W to C. This is because predecessor jobs might not be completed.
- You cannot change an operation status from X (suppressed by condition) or to X.
- To change the status of an operation to the previous logical status, specify STATUS(X). As from the previous point, do not use it to change the operation status to suppressed by condition.

For more information, refer to “Using the ready list” on page 525.

SUBSYS (MSTR | subsystem name | OPCA)
Name of the tracker subsystem that the OPSTAT command is directed to. The name can be up to 4 characters long. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase.

If you specify MSTR, the OPSTAT command is directed to all tracker subsystems on the z/OS system where the OPSTAT command is issued.

Note: If the trackers and controller run on different subsystems in your configuration, specify the name of the tracker subsystem in this parameter.

TRACE (level | 0)
Event tracing indicator. When a positive number is specified, a trace entry is created for each event generated by the OPSTAT command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records.

WSNAME (workstation name)
You must supply the name of the workstation that you are reporting the status of an operation for.

Examples
These examples demonstrate how you can use the OPSTAT command in TSO, or in a batch job (using the batch program EQQEVPGM).
Example 1 - TSO command
ALLOC F(EQMLIB) DA('OPC.MESSAGE.Library') SHR REUSE
OPSTAT W(BDEC) ST(C) J(DNCD3000) A(ACLMSDL)Y

Example 2 - Batch job
//OPSTATUS JOB (ACCOUNT), 'Set completed', CLASS=A
//STEP1 EXEC PGM=EQQEVPGM
//STEPLIB DD DSN=OPC.LOAD.MODULE.LIBRARY, DISP=SHR
//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY, DISP=SHR
//EQQMLOG DD SYSOUT=A
//SYSIN DD *
OPSTAT W(BDEC) ST(C) J(DNCD3000) A(ACLMSDL)

In both of these examples, an operation for application ACLMSDL at workstation
BDEC is reported as completed.
Purpose

The SRSTAT command lets you change the overriding (global) availability, quantity, and deviation of a special resource. You can do this to prevent operations from allocating a particular resource, or to request the ETT function to add an application occurrence to the current plan.

Usage

You can invoke SRSTAT as a TSO command or by using a batch job which executes program EQQEVPDM. If you invoke SRSTAT as a TSO command, you must allocate the EQQMLIB data set to the address space of the TSO user, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command execution.

Use of the SRSTAT command can be restricted with fixed resource code SR and subresource SR.SRNAME. The authority of the requestor is verified by the subsystem name identified in the command if an AUTHDEF statement is defined for that subsystem. When SUBSYS(MSTR) is specified, all tracker subsystems defined on the z/OS system where the SRSTAT command is issued will attempt to verify the authority of the requestor before an event is generated. It is possible to be rejected by one subsystem and accepted by another.

The subsystem to which you direct the command does not have to be active when the command is issued. An event will be generated and queued in CSA along with other job-tracking events. If the subsystem is not active when the command is issued, the authority of the requestor is verified using the class name specified in the AUTHDEF statement when the subsystem was last started. If the subsystem has not been started since a z/OS IPL, no authority verification can be performed.

See [Chapter 5, “Creating special resources,” on page 81](#) for information about special resources, how to connect them to workstations, and how to specify intervals.

Format

```
SRSTAT 'resource name'
```

```
KEEP
AVAIL(KEEP
RESET
NO
YES
CREATE(YES
NO
)
)

DEVIATION(KEEP
amount
RESET
)

LIFESPAN(interval, new availability value)
```
Parameters

'resource name'

The name of the resource whose availability or quantity you want to change. This parameter must be contained within single quotes and can be up to 44 characters in length. All lowercase characters are converted to uppercase.

You must supply this parameter.

**AVAIL (RESET | NO | YES | KEEP)**

YES indicates that the availability status of the resource should be set to YES. Operations requiring the resource can start, as long as there are no other conditions preventing them from starting. For example, if two operations both require a resource for exclusive (X) use, Tivoli Workload Scheduler for z/OS lets only one of them start.

NO indicates that the availability status of the resource should be set to NO. Any operation that requires the resource will be prevented from starting, regardless of whether it was specified for shared (S) or exclusive (X) use.

RESET sets the overriding availability to blank so that the interval or default value is used.

KEEP, the default, does not change the availability status.

When you set the availability with SRSTAT (or other interfaces such as the EQQUSIN subroutine or the MCP panel), the specified availability lasts over interval boundaries, even though the next interval can specify a different availability, and persists after a daily planning job. Specify RESET to restore the planned availability.

**CREATE (NO | YES)**

NO indicates that the resource is not to be added to the current plan of the receiving Tivoli Workload Scheduler for z/OS subsystem, if it does not exist in the database. If the resource exists in the database, CREATE(NO) does not have any effect. You can specify CREATE(NO) if the resource is being used only as a means to generate an event for ETT—the event is generated even if the resource does not exist.

If YES is specified or defaulted, and the DYNAMICADD keyword of the RESOPTS initialization statement is set to YES or EVENT, Tivoli Workload Scheduler for z/OS adds the resource to the current plan of the receiving Tivoli Workload Scheduler for z/OS subsystem, if the resource is not in the database. It uses the following default values:

**Text**

Blank.

**Specres group ID**

Blank.
Hiperbatch
No.

Used for
Control.

On error
Blank. If an error occurs, Tivoli Workload Scheduler for z/OS uses the value specified in the operation details or, if that is also blank, the value of the ONERROR keyword of RESOPTS.

On Complete
Blank.

When the operation completes, the On Complete value is considered according to the following order:
1. The On Complete value set at operation definition level, if not blank.
2. The On Complete value set at special resource definition level, if not blank.
3. The ONCOMPLETE or DYNONCOMPLETE keyword value, respectively set for the not dynamically added resources or the dynamically added resources, in all the other cases.

Max Usage Limit
0.

Max Usage Type
Reset.

Usage Counter
0.

Overriding availability, quantity, and deviation
The value specified by SRSTAT, or blank.

Default quantity
1. The default quantity is automatically increased if contention occurs.

Default availability
Yes.

Intervals
No intervals are created.

Workstations
* (all workstations can allocate the resource).

DEVIATION (RESET | amount | KEEP)
To make a temporary change to the quantity, you can specify a deviation, which is an amount to be added to (positive number) or subtracted from (negative number) the current quantity. A specified amount can be from −999 999 to +999 999. The default, KEEP, does not alter the deviation.

Note: The effect of deviation is cumulative. If you issue two SRSTAT commands with DEVIATION(-1), for example, this subtracts two from the deviation.

When the deviation is not zero, the value lasts over interval boundaries and persists after a daily planning job. Specify RESET or zero (0) to set the deviation to zero.

LIFESPAN (interval, new availability value)
The interval of time, in minutes, after which the global availability of the
special resource is changed according to the value specified as new availability value. When you specify this parameter, a pending LIFESPAN action (the one specified as new availability value) comes into effect for the resource. The controller runs that action as soon as the LIFESPAN interval of time expires.

Only one pending LIFESPAN action can exist for one resource. This means that issuing an SRSTAT command with LIFESPAN replaces any existing pending LIFESPAN action.

To cancel a pending LIFESPAN action, issue an SRSTAT command with an interval of 0. When the interval of time is 0, no LIFESPAN action is run.

Interval can be an integer from 0 to 99999. New availability value can be one of the following (you must always specify one, because no default is provided):

YES  The global availability is changed to Yes
NO   The global availability is changed to No
RESET The global availability is changed to blank

QUANTITY (RESET | amount | KEEP)
To change the overriding (global) quantity, specify the amount, from 1 to 999 999.

RESET sets the overriding quantity to blank so that the interval or default value is used. KEEP does not alter the quantity.

When you set the quantity with SRSTAT (or other interfaces such as the EQQUSIN subroutine or the MCP panel), the specified quantity lasts over interval boundaries, even though the next interval can specify a different quantity, and persists after a daily planning job. Specify RESET to restore the planned quantity.

SUBSYS (subsystem name | MSTR | OPCA)
The name of the tracker subsystem that the SRSTAT command is directed to. This parameter can be up to 4 characters long. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase.

If you specify MSTR, the SRSTAT command is directed to all tracker subsystems on the z/OS system where the SRSTAT command is issued.

Note: If the trackers and controller in your configuration run on different subsystems, specify the name of the tracker subsystem in this parameter.

TRACE (level | 0)
Event tracing indicator. When a positive number is specified, a trace entry is created for each event generated by the SRSTAT command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records.

Examples
These examples demonstrate how you can use SRSTAT in TSO, or in a batch job (using the batch program EQQEVPGM).
Example 1 - TSO command
ALLOC F(EQQLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE
SRSTAT 'IMS.DATA.BASE' SUBSYS(OPCB) AVAIL(YES)

Example 2 - Batch job
//CHSTATUS JOB (ACCOUNT), 'Change DB status', CLASS=A
//STEP1 EXEC PGM=EQQEVPGM
//STEPLIB DD DSN=OPC.LOAD.MODULE.LIBRARY, DISP=SHR
//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY, DISP=SHR
//EQQMLOG DD SYSOUT=A
//SYSIN DD *
SRSTAT 'IMS.DATA.BASE' SUBSYS(OPCB) AVAIL(YES)
/*

Example 3 - Reduce tape pool
ALLOC F(EQQLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE
SRSTAT 'TAPES' SUBSYS(OPCB) DEV(RESET) Q(6)
SRSTAT 'TAPES' SUBSYS(OPCB) DEV(-1)
SRSTAT 'TAPES' SUBSYS(OPCB) DEV(-1)
SRSTAT 'TAPES' SUBSYS(OPCB) DEV(0)

In examples 1 and 2, the availability status of the resource IMS.DATA.BASE is changed to YES. In example 3, the number of tapes is set to 6, 5, 4, and then back to 6 (a deviation of 0 is a special value and means the same as reset).
WSSTAT

Purpose
The WSSTAT command lets you change the status of a workstation in the current plan. The status information is communicated to the controller to indicate a workstation as active, offline, or failed. When you use the WSSTAT command you can optionally define restart and routing options for the workload defined on the workstation when you are reporting a status of offline or failed.

Usage
You can invoke WSSTAT as a TSO command or by using a batch job which executes program EQQEVPGM. If you invoke WSSTAT as a TSO command, you must allocate the EQQMLIB data set to the address space of the TSO user, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command execution.

Use of the WSSTAT command can be restricted with fixed resource code RL and subresource RL.WSSTAT. The authority of the requestor is verified by the subsystem name identified in the command if an AUTHDEF statement is defined for that subsystem. When SUBSYS(MSTR) is specified, all tracker subsystems defined on the z/OS system where the WSSTAT command is issued will attempt to verify the authority of the requestor before an event is generated. It is possible to be rejected by one subsystem and accepted by another.

The subsystem to which you direct the command does not have to be active when the command is issued. An event will be generated and queued in CSA along with other job-tracking events. If the subsystem is not active when the command is issued, the authority of the requestor is verified using the class name specified in the AUTHDEF statement when the subsystem was last started. If the subsystem has not been started since a z/OS IPL, no authority verification can be performed.

Note: If the status of a workstation has been set to offline manually, using the panels, then you are not allowed to reset it to active using the WSSTAT command.

Format

```
WSSTAT
  ALTWS(altworkstation name)
  REROUTE(R)
  STARTOPR(R)
  CMD(L)
  MANAGES(domain name)
  STATUS(O)
```
WSSTAT

Parameters

ALTWS (alternate workstation name)
When the workstation status is set to offline or failed, you can specify the alternate workstation where reroutable operations should be started.

If this parameter is omitted the value defined for the current workstation open interval will be used. If the REROUTE parameter specifies L, or if the default specifies no rerouting, the value of ALTWS is ignored.

The parameter is optional.

CMD (L | P | S | U)
Use this parameter to change the status of the fault-tolerant workstation to one of the following statuses:

L   To link the workstation
P   To stop workstation
S   To start workstation
U   To unlink the workstation

You must specify either this or one of the alternative parameters.

MANAGES (domain name)
Used in conjunction with the WSNAME parameter, it instructs the workstation specified by WSNAME to become the new domain manager of the domain specified by MANAGES. It cannot be specified in conjunction with STATUS or CMD. Example:

WSSTAT SUBSYS(TWST) WSNAME(U001) MANAGES(UK-DM)

This command instructs the U001 workstation to become the new domain manager of the UK-DM domain. The command is sent through the TWST tracker subsystem.

The name of the domain, consisting of up to 16 characters, must start with a letter. It can contain alphanumeric characters, dashes and underscores. No other characters must be used because they can generate unpredictable situations.

REROUTE (R | L)
This parameter is optional. When the workstation status is set to offline or failed, you can specify R for operations to be rerouted to the alternate workstation, or L for no rerouting (to leave the operations at the inactive workstation).

If this parameter is omitted, the value defined in either the WSOFFLINE or the WSFAILURE keyword on the JTOPTS initialization statement will be used as default.

STARTOPR (R | E | L)
This parameter is optional. When the workstation status is set to offline or failed, you can specify what Tivoli Workload Scheduler for z/OS should do with operations that are currently in started status at the workstation, where:
**WSSTAT**

| **R** | Restart operations automatically on the alternate workstation. |
| **E** | Set all started operations to ended-in-error. |
| **L** | Leave the operations in started status. |

If this parameter is omitted, the value defined in either the WSOFFLINE or the WSFAILURE keyword on the JTOPTS initialization statement is used as default.

**Notes:**
1. If you select STARTOPR(E), a started job continues to run. Tivoli Workload Scheduler for z/OS never cancels jobs that have started.
2. For remote engine workstations, this parameter supports only the value L. If you specify any other value, it is forced to L.

**STATUS (A | O | F)**
The status you want to report for the workstation
- **A** Active
- **O** Offline
- **F** Failed

You must specify either this or one of the alternative parameters.

**SUBSYS (subsystem name | MSTR | OPCA)**
The name of the tracker subsystem that the WSSTAT command is directed to. This parameter can be up to 4 characters long. The first character must be alphabetic; the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase.

If you specify **MSTR**, the WSSTAT command is directed to all tracker subsystems on the z/OS system where the WSSTAT command is issued.

**Note:** If the tracker trackers and controller in your configuration run on different subsystems, specify the name of the tracker subsystem in this parameter.

**TRACE (level | 0)**
Event tracing indicator. When a nonzero positive number is specified, a trace entry is created for each event generated by the WSSTAT command. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 will not generate trace records.

**WSNAME (workstation name)**
The name of the workstation to be updated. This parameter is required.

The WSSTAT parameters are checked for validity and consistency. The validity checks are carried out in the tracker where the command is executed. The validity check processes parameter names, length and type of parameter values.

**Note:** If you try to set the status of a fault-tolerant workstation to Failed (an invalid status for this type of workstation), you do not receive an immediate message. Instead, you receive message EQQE112E in the controller message log, and the workstation status is not changed.

If the input is valid, a *workstation status event* is generated and communicated to the controller. The processing of the event includes a consistency check of the values specified in the parameters. The following consistency checks are made:
- The workstation name is checked for existence.
- The alternate workstation is checked for existence.
If the value given in the STATUS parameter equals current status of the workstation, the command will be ignored.

Examples

The following two examples demonstrate how you can use WSSTAT in TSO, or in a batch job (using the batch program EQQEVPGM).

| WSSTAT
| Example 1 - TSO command |
| ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE |
| WSSTAT SUBSYS(OPCB) WSNAME(AS4H) STATUS(O) START(R) |

| Example 2 - Batch job |
| //CHSTATUS JOB (ACCOUNT), 'Change WS status', CLASS=A |
| //STEP1 EXEC PGM = EQQEVPGM |
| //STEPLIB DD DSN = OPC.LOAD.MODULE.LIBRARY, DISP = SHR |
| //EQQMLIB DD DSN = OPC.MESSAGE.LIBRARY, DISP = SHR |
| //EQQMLOG DD SYSOUT = A |
| //SYSIN DD * |
| WSSTAT SUBSYS(OPCB) WSNAME(AS4H) STATUS(A) |
| /* |

In the first example the status of workstation AS4H is set to offline. Started operations will be restarted on the alternate workstation.

In the second example the status of workstation AS4H will be set to active status.
Appendix B. Batch programs

This appendix describes the JCL and parameters that the batch programs need, although you do not need this if you use the panels to generate and submit them.

For a description of how to create JCL skeletons for the batch jobs, refer to the section about EQQJOBS in the Planning and Installation.

Most batch programs are invoked by a program named EQQBATCH, and are specified by a parameter in the JCL EXEC statement. This program must be authorized and, like all other batch load modules in Tivoli Workload Scheduler for z/OS, must reside in an authorized library. These batch programs are not invoked with EQQBATCH: EQQYLTOP, EQQPURGE, and EQQEVPGM.

After running batch programs, the generated reports can optionally be sent to other users by email.

DD statements referenced by batch jobs

The batch jobs use three categories of DD statements:

- DD statements that are required by EQQBATCH
- DD statements that refer to Tivoli Workload Scheduler for z/OS databases
- DD statements that refer to work data sets

For DCB and SPACE requirements of data sets that are referred to by these ddnames, see the JCL in each batch program example.

The corresponding JCL examples define which batch jobs use which DD statements.

DD statements used by EQQBATCH

The basic batch program, EQQBATCH, handles common functions necessary to initialize the batch jobs. These DD statements are required for all batch jobs:

- **EQQDMSG**: Output; contains diagnostic messages.
- **EQQDUMP**: Output; contains diagnostic trace information.
- **EQQMLIB**: Input; messages, translatable terms, and headers.
- **EQQMLOG**: Output; messages.
- **EQQPARM**: Input; initialization parameters.

This DD statement in the JCL references the member in the parameter library that contains the BATCHOPT statement for the Tivoli Workload Scheduler for z/OS subsystem. The BATCHOPT statement defines runtime options for the batch jobs. These options can include:

- Date format for reports
- Report headers
- Page size for reports
- Subsystem name

For a more detailed description of the BATCHOPT statement, refer to Customization and Tuning.

© Copyright IBM Corp. 1991, 2011

673
DD statements used by EQQBATCH

<table>
<thead>
<tr>
<th>DD Statement</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>STEPLIB</td>
<td>Input; step library. (This statement is not required if the Tivoli Workload Scheduler for z/OS modules are part of the z/OS link library.)</td>
</tr>
<tr>
<td>SYSIN</td>
<td>Input; inline input parameters to the batch programs.</td>
</tr>
<tr>
<td>SYSMDUMP</td>
<td>Output; the dump data set</td>
</tr>
</tbody>
</table>

DD statements describing the data

Tivoli Workload Scheduler for z/OS data is kept in VSAM as well as non-VSAM data sets:

- **EQQADDS** | VSAM; application descriptions and JCL variable tables |
- **EQQAD2DS** | VSAM; application descriptions (used by the batch loader) |
- **EQQCKPT** | Sequential; checkpoint data set |
- **EQQCP1DS** | VSAM; primary-current-plan data set |
- **EQQCP2DS** | VSAM; alternate-current-plan data set |
- **EQQCXDS** | VSAM; current plan extension, containing special resource information |
- **EQQJTx** | Sequential; job-tracking-log data set (maximum 99) |
- **EQQJTARC** | Sequential; job-tracking-archive data set |
- **EQQLTBKP** | VSAM; long-term plan backup |
- **EQQLTDS** | VSAM; long-term plan data set |
- **EQQNCPDS** | VSAM; new-current-plan data set |
- **EQQNCXDS** | VSAM; new current plan extension, containing special resource information |
- **EQQOIDS** | VSAM; operator instruction database |
- **EQQRDDS** | VSAM; resource definition database |
- **EQQSCLIB** | Partitioned; script library |
- **EQQSCPDS** | VSAM; current plan copy to produce the Symphony file |
- **EQWWSDS** | VSAM; workstation description and calendar database |

These data sets are common to most batch programs and to the Tivoli Workload Scheduler for z/OS subsystem. Ensure that your batch jobs use the same data sets as the Tivoli Workload Scheduler for z/OS subsystem (except when you creating a separate database using the batch loader program, as described in Table 18 on page 193).

The batch jobs also allocate work data sets.

Security and batch programs

Most batch programs do not use subresource security checking to update a database. If you have authority to update the application description database, for example, batch programs allow you to update any application. An exception to this rule is when the batch program uses the program interface to make the updates (as the EQQOIBLK does when you comment out the EQQOIDS ddname).
Batch programs and JCL examples

The batch programs are described, in alphabetical order, in the following pages. Each program is described with the format of the required SYSIN and a JCL example for the batch job.

Each job must contain:

```
//SYSIN DD *
data
/*
```

where `data` meets the conditions under the heading “SYSIN requirements” in the following descriptions. If the job does not require SYSIN, the DD statement for SYSIN is still required; it should be immediately followed by the `/*` statement.

Tivoli Workload Scheduler for z/OS interprets the year part (YY) in the SYSIN data streams as follows:

<table>
<thead>
<tr>
<th>YY</th>
<th>Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>72 - 99</td>
<td>1972 - 1999</td>
</tr>
<tr>
<td>00 - 71</td>
<td>2000 - 2071</td>
</tr>
</tbody>
</table>

In the following example, a batch program for creating a new long-term plan with the SYSIN DD statement used in the example will produce a long-term plan for the period 1998/12/01 to 2000/04/30:

```
//SYSIN DD *
981201000430
/*
```

All batch programs that use the long-term plan must have a start day later than 1978/01/01 and an end date before 2066/12/31, because of the way Tivoli Workload Scheduler for z/OS handles calendars internally.

Table 41 summarizes the batch program functions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQQADCOP</td>
<td>Print applications: calculate and print application run days.</td>
</tr>
<tr>
<td>EQQADDEP</td>
<td>Print applications: cross-reference external dependencies.</td>
</tr>
<tr>
<td>EQQADMUP</td>
<td>Perform mass update of application descriptions.</td>
</tr>
<tr>
<td>EQQADPRT</td>
<td>Print applications: detailed.</td>
</tr>
<tr>
<td>EQQADXRF</td>
<td>Print applications: cross-reference job names.</td>
</tr>
<tr>
<td>EQQAUDIT</td>
<td>Print formatted audit reports</td>
</tr>
<tr>
<td>EQQAXR00</td>
<td>Print applications: cross-reference different items.</td>
</tr>
<tr>
<td>EQQCLPRC</td>
<td>Print calendars.</td>
</tr>
<tr>
<td>EQQCLPRP</td>
<td>Print periods.</td>
</tr>
<tr>
<td>EQQDNTOP</td>
<td>Create or extend the current plan.</td>
</tr>
<tr>
<td>EQQDOTOP</td>
<td>Print statistics for current planning period.</td>
</tr>
<tr>
<td>EQQDRTOP</td>
<td>Replan current planning period.</td>
</tr>
<tr>
<td>EQQDSTOP</td>
<td>Renew the Symphony file.</td>
</tr>
<tr>
<td>EQQDTTOP</td>
<td>Produce a trial plan.</td>
</tr>
</tbody>
</table>

Table 41. Batch programs
Table 41. Batch programs (continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQQEVPGM</td>
<td>Change status of resources or operations.</td>
</tr>
<tr>
<td>EQQJVPR</td>
<td>Print JCL variables.</td>
</tr>
<tr>
<td>EQLTCRE</td>
<td>Create a new long-term plan.</td>
</tr>
<tr>
<td>EQLTMOA</td>
<td>Modify the long-term plan for all applications or extend the long-term plan.</td>
</tr>
<tr>
<td>EQLTMOO</td>
<td>Modify the long-term plan for one application.</td>
</tr>
<tr>
<td>EQLTPTR</td>
<td>Print the long-term plan.</td>
</tr>
<tr>
<td>EQLTTRY</td>
<td>Make a trial long-term plan.</td>
</tr>
<tr>
<td>EQQOIBAT</td>
<td>Print operator instructions.</td>
</tr>
<tr>
<td>EQQOIBLK</td>
<td>Operator instructions: perform mass update.</td>
</tr>
<tr>
<td>EQQPURGE</td>
<td>Purge DLF (data lookaside facility) object.</td>
</tr>
<tr>
<td>EQQSLTOP</td>
<td>File parse program, Process parse statement.</td>
</tr>
<tr>
<td>EQQWSPRT</td>
<td>Print all workstation descriptions.</td>
</tr>
<tr>
<td>EQQYLTOP</td>
<td>Batch loader.</td>
</tr>
</tbody>
</table>

**EQQADCOP - Print applications: calculate and print application run days**

The EQQADCOP batch program calculates and prints application run days.

**SYSIN requirements**

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The application ID is specified, starting in column 1.

**JCL example**

```plaintext
//*********************************************************************
//* CALCULATE AND PRINT APPLICATION RUNDATES
//*********************************************************************
//ADCOP EXEC PGM=EQQBATCH,PARM='EQQADCOP',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APPLIB
//EQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//LTREPORT DD SYSOUT=*,
//   DCB=(RECFM=FBA,LRECL=121,BKSIZE=6050)
//EQMLOG DD SYSOUT=*,
//SYMOUT DD SYSOUT=*
//SYSMLOG DD SYMLOG=*,
//EQQDUMP DD SYSLOG=*
//EQMDUMP DD SYMDUMP=*
//LTACIN DD DCB=(RECFM=FB,LRECL=672,BKSIZE=6048),
//   SPACE=(CYL,(3,1)),UNIT=3390
//EQQADD DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//EQQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR
//SYSIN DD *
//PAYDAILY
//** = APPLICATIONID
//** APPLICATION ID = SELECTED APPLICATION
```
**EQQADDEP - Print applications: cross-reference external dependencies**

The EQQADDEP batch program provides a printout of a cross-reference of external dependencies on the application description database. Note that EQQADDEP does not support applications with "valid from" dates later than the date on which the batch job runs.

**SYSIN requirements**

None.

**JCL example**

```plaintext
EXEC PGM=EQQBATCH,PARM='EQQADDEP',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APPLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//ADREPORT DD SYSOUT=*
//DCB=(RECFM=FBA,LRECL=121,BKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//EQQADDEP DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQAD2DS DD DSN=EID.EIDAR3.AD,DISP=SHR
//SYSIN DD *
```

**EQQADMUP - Perform mass update of application Descriptions**

The EQQADMUP batch program performs a mass update of application descriptions in the AD database.

**SYSIN requirements**

**Attention:** The SYSIN parameter string is in a binary format. Do not try to create it manually, because any errors could corrupt your application description database. Always use the mass update dialog to generate the parameter.

**JCL example**

```plaintext
EXEC PGM=EQQBATCH,PARM='EQQADMUP',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APPLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//MUREPORT DD SYSOUT=*
//DCB=(RECFM=FBA,LRECL=121,BKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//ADDPIN DD DCB=(LRECL=065,BKSIZE=3250,RECFM=FB),
//SPACE=(CYL,(3,1)),DISP=(,DELETE),UNIT=3390
//ADDPW01 DD DCB=(LRECL=065,BKSIZE=3250,RECFM=FB),
//SPACE=(CYL,(1,3)),UNIT=3390
//ADDPw02 DD DCB=(LRECL=065,BKSIZE=3250,RECFM=FB),
//SPACE=(CYL,(1,3)),UNIT=3390
//ADDPW03 DD DCB=(LRECL=065,BKSIZE=3250,RECFM=FB),
//SPACE=(CYL,(1,3)),UNIT=3390
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQAD2DS DD DSN=EID.EIDAR3.AD,DISP=SHR
//SYSIN DD *
```
EQQADMUP - Perform mass update of application Descriptions

//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMP
//EQQDUMP DD SYSOUT=* 
//EQQDSMG DD SYSOUT=* 
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR 
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR 
//*****************************************************************************
// CAUTION
// RENUMBERING THIS MEMBER AS WELL AS UPDATING THE SYSIN CARDS
// MAY DAMAGE THE INPUT PARAMETERS TO THE AD MASS UPDATE BATCHJOB.
//*****************************************************************************
//SYSIN DD *
XRAYNER U OG ( SAMPLE
CPU1 Y Y
OG - SAMPLE CPU1
D
/*

EQQADPRT - Print applications: detailed

The EQQADPRT batch program provides a printout of application information from the application database.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The definition of the input parameters is:

COLUMNS 1 2 3
123456789012345678901234567890
ABCDEFGHIJKLMNOP
LINE 1 (REQUIRED)
XXXXXXXXXXXXXXXX LINE 2-n (OPTIONAL)

where:

A = 1 APPLICATIONS, 2 DEPENDENCIES, 3 OPERATIONS, 4 APPL vs OWNER
B = 1 ALL APPLICATIONS, 2 SELECTED ITEM, 3 FROM FILE
CDEFGH = (YMMDD)
IJ = Number of lines following
XXXXXXXXXXXXXX = Application ID (number specified by IJ)

Notes

If you specify a data set name on the select panel, the ADUSERDS DD is set to the data set containing application IDs. These report options are available:

- Detailed printout of applications
- Internal dependencies for applications
- Operation using particular workstations
- Applications with particular owners

The input type restricts the printout to one of the following:

- Applications that are valid only on the date specified
- The application IDs that are provided
- The application IDs in the specified data set

JCL example

/*******************************************************************************
// APPLICATION PRINT PROGRAM
*******************************************************************************
//ADPRT EXEC PGM=EQQBATCH,PARM='EQQADPRT',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQ Parm DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//ADREPORT DD SYSOUT=*,
// DCB=(RECFM=FBA,RECL=121,BLKSIZE=6050)
//ADREPORT DD DB=(RECFM=FBA,RECL=121,BLKSIZE=6050),
// SPACE=(CYL,(3,1)),DISP=(NEW,DELETE),UNIT=3390
//EQQMLOG DD SYSOUT=*

*******************************************************************************
EQQADXR - Print applications: cross-reference job names

The EQQADXR batch program provides a printout of cross-references of job names on the application description database.

SYSIN requirements

None.

JCL example

/usr=t/r=000000001

EQQAUDIT - Print formatted audit reports

The EQQAUDIT batch program provides a printout of formatted audit reports.
EQQAUDIT - Print formatted audit reports

Notes:
1. When you run the EQQAUDIT batch program interactively only, all the parameters are converted to upper-case. Because the extended name field input is case-sensitive, query results will not be found if you include a lower- or mixed-case extended name in your search criteria. To avoid this problem, run the sample in batch.
2. Run the sample in batch if you include the following characters in your search criteria:
   / + & * -

JCL example

//******************************************************************
//* Print formatted audit reports
//******************************************************************
/** This program will take input from either the JTARC/JTn-files (for reporting in real time) or from the //EQQTROUT-file of the daily plan extend/replan programs (for after-the-fact reporting).
/** If you don't allocate //LIVEMLOG dummy, it will also read that file and, looking at the time-stamps, merge those messages with the formatted report-lines from the tracklog-files.
/** If not excluded by the user-defined filters, each tracklog-event will generate one formatted report-line. The program will also:
/** - Print all JCL-lines if AMOUNT(DATA) specified for FILE(JS)
/**   (If PASSWORD= is encountered, the password will be blanked out and not appear in the listing)
/** - Print all variable values of a job if AMOUNT(DATA) was specified for FILE(VAR)
/** - Print all origin dates of a period if AMOUNT(DATA) was specified for FILE(PER)
/** - Print all specific dates of a calendar if AMOUNT(DATA) was specified for FILE(CAL)
/** You have to specify AMOUNT(DATA) for FILE(CAL) to have calendar identifier listed
/** You have to specify AMOUNT(DATA) for FILE(LTP) to have for example deleted occurrences identified in a meaningful way
/** An MCP-request will be broken down into sub-transactions and each sub-transaction listed in connection to the originating request
/** - All WS intervals will be printed if availability of a WS is changed
/** - All contained group occurrences will be listed for a 'group' MCP request
/** If you specify a search-string to the program it will select events with the specified string in it somewhere. That MAY mean that you will not see the string in the report-line itself as the line may have been too 'busy' to also have room for your string value, whatever it may be.
/** MLOG messages have only MM/DD time-stamps. Therefore You should avoid running the program with input files spanning the year boundary.
//******************************************************************
EQQAUDIT - Print formatted audit reports

The EQQAUDIT program provides cross-references between different items on the application database.
**SYSIN requirements**
The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The definition of the input parameters is:

```
COLUMNS 1 2 3
123456789012345678901234567890
AABBCCDDEE LINE 1 (REQUIRED)
```

where **AA, BB, CC, DD, EE** are the codes for the sort order (maximum of 5):

- 01 = APPLICATION ID
- 02 = APPLICATION TEXT
- 03 = OP ID
- 04 = OP TEXT
- 05 = PERIOD NAME
- 06 = RUN CYCLE DESCRIPTION
- 07 = JOB NAME
- 08 = JOB CLASS
- 09 = PRIORITY
- 10 = SYSOUT CLASS
- 11 = FORM NUMBER
- 12 = SPECIAL RESOURCES
- 13 = WS RESOURCE R1
- 14 = WS RESOURCE R2
- 15 = CALENDAR ID

**JCL example**
```
//Application cross-reference of selected fields
//***************************************************************************
//ADREPORT EXEC PGM=EQQBATCH,PARM='EQQAXR00',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APPLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//ADREPORT DD SYSOUT=*,
//DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//ADWKIN DD DCB=(LRECL=220,BLKSIZE=4400,RECFM=FB), 
//SPACE=(CYL,(3,1)),UNIT=3390
//ADWKOUT DD DCB=(LRECL=220,BLKSIZE=4400,RECFM=FB), 
//SPACE=(CYL,(3,1)),UNIT=3390
//ADWKWK01 DD SPACE=(CYL,(3,1)),UNIT=3390
//ADWKWK02 DD SPACE=(CYL,(3,1)),UNIT=3390
//ADWKWK03 DD SPACE=(CYL,(3,1)),UNIT=3390
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//SYSIN DD *
//0701031215 /*
```

**EQQCLPRC - Print calendars**
The EQQCLPRC batch program provides a printout of calendar information from the calendar database. All calendar information from the date of submission to the end date that is specified will be printed.

**SYSIN requirements**
The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The definition of the input parameters is:

```
COLUMNS 1 2 3
123456789012345678901234567890
YYMMDD LINE 1 (REQUIRED)
```

where **YYMMDD** = End date for the report (YY = YEAR, MM = MONTH, DD = DAY)

**JCL example**
```
//*******************************************************************
//* PRINT ALL CALENDARS
//*******************************************************************
```
EQQCLPRP - Print periods
The EQQCLPRP batch program provides a printout of period information from the
calendar database.

SYSIN requirements
The SYSIN data is normally supplied from the select panel; it is the input
parameter to the batch program. The definition of the input parameters is:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYMMDD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CCCC CCCC CCCC CCCC CCCC CCCC CCCC</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LINE 1</td>
<td>REQUIRED</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

* YYMMDD = End date for the report (YY = YEAR, MM= MONTH, DD= DAY)
* CCCC CCCC CCCC CCCC CCCC CCCC CCCC = Calendar for the calculation of the origin date

JCL example

/* = END DATE OF PERIODS PRINT (YYMMDD)

EQQDNTOP - Create or extend the current plan
The EQQDNTOP batch program creates or extends the current planning period.

Note: Job tracking uses two data sets to log each event that has been processed. If
you need to restart Tivoli Workload Scheduler for z/OS from a checkpoint,
Tivoli Workload Scheduler for z/OS uses the older of the two current plan
data sets. Events logged on the current log data set are then reprocessed.
When this reprocessing finishes, a current plan backup copy is created so
that the two current plan data sets are again equal. Job tracking continues to
process new events as they occur. You do not need to re-create the current
plan.

SYSIN requirements
The SYSIN data is normally supplied from the select panel; it is the input
parameter to the batch program. The layout of the SYSIN DD statement for the
first step is:

```
COLUMNS 1 2 3
123456789012345678901234567890
YYMDDHHHMM hmmm abcdefgh (format 1)
YYMDDHHHMM yymmddhhmm abcdefgh (format 2)
```

where:
- `YYMDDHHHMM` = Plan start date and time (both formats)
- `hmmm` = Hours and minutes to be extended (format 1)
  (for example, 02400 is 24 hours)
- `W` = Work day flag. Set to W or blank. (format 1)
  (for example, 04800W means extend for 48 days,
  excluding any free days)
- `yymmddhhmm` = Plan end date and time (format 2)
- `abcdefgh` = Report flags. Set to 1 or 0. (both formats)
  - a = WORKSTATION SUMMARY
  - b = DAILY OPERATING PLANS
  - c = WORK STATION PLANS
  - d = WORK STATION INPUT ARRIVAL LISTS
  - e = NONREPORTING WORK STATIONS
  - f = PREVIOUS PERIOD RESULTS
  - g = PLANNED RESOURCE UTILIZATION
  - h = ACTUAL RESOURCE UTILIZATION

Examples:
- `02400 11111111` To extend by 24 hours and have all reports.
- `951201120010000000` To extend to 1 December 1995 at 12.00 and
  have the workstation summary report only.

JCL example
```
//*********************************************************************
//* DAILY PLANNING - PLAN NEXT PERIOD
//*********************************************************************
//DNEQDNTOP EXEC PGM=EQQBATCH,PARM='EQQDNTOP',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//EQQDIN DD DSN=&&A,DISP=(NEW,PASS),
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580),
//EQQTROUT DD DUMMY,DCB=BLKSIZE=100
//* ----- ----
//* CHANGE WHEN TRACKLOG IS TO BE SAVED
//* DISP=MOD MUST BE SPECIFIED
//* DCB PARAMETERS MUST BE DEFINED IN DD STATEMENT:
//* LRECL=32000 RECFM=VBS
//*********************************************************************
//EQQMLOG DD SYSOUT=* 
//EQQDUMP DD SYSOUT=* 
//EQQDMSG DD SYSOUT=* 
//SYSDUMPDD DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB 
//EQQWDSDD DD DISP=SHR,DSN=EID.EIDA3.WS 
//EQQADDS DD DISP=SHR,DSN=EID.EIDA3.AD 
//EQRDSDS DD DISP=SHR,DSN=EID.EIDA3.RD 
//EQLTDS DD DISP=SHR,DSN=EID.EIDA3.LT 
//EQLTBP DD DISP=SHR,DSN=EID.EIDA3.LB 
//EQQCP1DS DD DISP=SHR,DSN=EID.EIDA3.CP1 
//EQQCP2DS DD DISP=SHR,DSN=EID.EIDA3.CP2 
```
EQQDNTOP - Create or extend the current plan

//EQQSCPD S DD DISP=SHR,DSN=EID.EIDAR3.SCP
//EQQSCLIB DD DISP=SHR,DSN=EID.EIDA.SCRPTLIB
//EQQNCPS D DD DISP=SHR,DSN=EID.EIDAR3.NCP
//EQNXCXDS DD DISP=SHR,DSN=EID.EIDAR3.NCX
//EQQJARC DD DISP=SHR,DSN=EID.EIDA.JTARC
//EQQCKPT DD DISP=SHR,DSN=EID.EIDA.CKPT//SYSIN DD *
   02400  111111

/** = PLSTRTTYPENDTIME REPORTS
** PLSTRTTIME(YYMMDDHHMM) = PLAN START DATE,TIME IF REFRESH
** PLENDTIME (YYMMDDHHMM) = PLAN ENDTIME
** (HHMM ) = PLAN EXTENSION IN HOURS AND MINUTES
** COUNTERING ALL DAYS
** (HHMMW ) = PLAN EXTENSION IN HOURS AND MINUTES
** COUNTERING ONLY WORK DAYS
** REPORTS(ABCDEFGH) = REQUESTED REPORTS (1/0)
** A = WORKSTATION SUMMARY
** B = DAILY OPERATING PLANS
** C = WORKSTATION PLANS
** D = WORKSTATION INPUT ARRIVAL LISTS
** E = NONREPORTING WORKSTATIONS
** F = PREVIOUS PERIOD RESULTS
** G = PLANNED RESOURCE UTILIZATION
** H = ACTUAL RESOURCE UTILIZATION
*/

//*********************************************************************
// STEP2 DAILY PLAN - SORT OF REPORT RECORDS
//*********************************************************************
//* EXEC PGM=SORT,REGION=4096K,TIME=1440,COND=(8,LT)
//SYSPRINT DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SORTIN DD DSN=&A,DISP=(OLD,DELETE)
//SORTOUT DD DISP=(NEW,PASS),
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580), 
// SPACE=(CYL,(3,1)),UNIT=3390
//SORTWK01 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580), 
// SPACE=(CYL,(3,1)),UNIT=3390
//SORTWK02 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580), 
// SPACE=(CYL,(3,1)),UNIT=3390
//SORTWK03 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580), 
// SPACE=(CYL,(3,1)),UNIT=3390
//SYSIN DD *
  
  SORT FIELDS=(1,130,CH,A)

//*********************************************************************
// STEP3 DAILY PLAN - PLAN NEXT PRINT THE REPORTS
//*********************************************************************
//* EXEC PGM=EQQBATCH,PARM='EQQDPRPT',COND=(8,LT),
// REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APLLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//SYSPRINT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*,
//EQQDUMP DD SYSOUT=*,
//EQQDMSG DD SYSOUT=*
EQQDNTOP - Create or extend the current plan

//SYSOUT DD SYSOUT=*  
//EQQPOUT DD DSN=*.SORT.SORTOUT,DISP=(OLD,DELETE)  
//SYSIN DD *  
//*

EQQDOTOP - Print statistics for current planning period

The EQQDOTOP batch program prints current plan statistics.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The layout of the SYSIN DD statement for the first step is:

COLUMNS 1 2 3
123456789012345678901234567890
YYMMDHHHMyymmdhhmm

where:
YYMMDHHHM = Plan start date and time
yymmdhhmm = Plan end date and time

JCL example

//*********************************************************************  
//* DAILY PLANNING - PRINT CURRENT PERIOD RESULTS  
//*********************************************************************  
//DOTOP EXEC PGM=EQQBATCH,PARM='EQQDOTOP',REGION=4096K  
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APPLIB  
//EQQMLIB DD DISP=SHR,DSN=OPCTEST.ESA131.SEQMMSG0  
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)  
//SYSPRINT DD SYSOUT=*  
//EQQMLOG DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//EQQDIN DD DSN=&&A,DISP=(NEW,PASS),  
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580),  
// SPACE=(CYL,(3,1)),UNIT=3390  
//EQQCP1DS DD DSN=EID.EIDAR3.CP1,DISP=SHR  
//EQQCP2DS DD DSN=EID.EIDAR3.CP2,DISP=SHR  
//EQQCKPT DD DSN=EID.EIDA.CKPT,DISP=SHR  
//SYSIN DD *  
95061012009506111600  
//* = PLSTRTIMEPLENDTIME  
//* PLSTRTIME(YYMMDHHHM) = PLAN START DATE,TIME IF REFRESH  
//* PLENDTIME (YMMMDHHMM) = PLAN END DATE,TIME/*  
//*********************************************************************  
//* STEP2 DAILY PLAN - SORT OF REPORT RECORDS  
//*********************************************************************  
//SORT EXEC PGM=SORT,REGION=4096K,TIME=1440,COND=(8,LT)  
//SYSPRINT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SORTIN DD DSN=&&A,DISP=(OLD,DELETE)  
//SORTOUT DD DISP=(NEW,PASS),  
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580),  
// SPACE=(CYL,(3,1)),UNIT=3390  
//SORTWK01 DD DISP=(NEW,DELETE),  
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580),  
// SPACE=(CYL,(3,1)),UNIT=3390  
//SORTWK02 DD DISP=(NEW,DELETE),  
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580),  
// SPACE=(CYL,(3,1)),UNIT=3390  
//SORTWK03 DD DISP=(NEW,DELETE),  
// DCB=(RECFM=FB,LRECL=558,BLKSIZE=5580),
EQQDRTOP - Replan current planning period

The EQQDRTOP batch program provides a new plan for the current planning period.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The layout of the SYSIN DD statement for the first step is:

```
COLUMNS 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
abcdefgh
```  

where:

- `abcdefgh` = Report flags. Set to 1 or 0.
  - `a` = WORKSTATION SUMMARY
  - `b` = DAILY OPERATING PLANS
  - `c` = WORK STATION PLANS
  - `d` = WORK STATION INPUT ARRIVAL LISTS
  - `e` = NONREPORTING WORK STATIONS
  - `f` = PREVIOUS PERIOD RESULTS
  - `g` = PLANNED RESOURCE UTILIZATION
  - `h` = ACTUAL RESOURCE UTILIZATION

JCL example

```
//*********************************************************************
//* DAILY PLANNING - REPLAN CURRENT PERIOD
//*********************************************************************
//** DRTOP EXEC PGM=EQQBATCH,PARM='EQQDRTOP',REGION=4096K
//** STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//** EQQMLIB DD DISP=SHR,DSN=OPCTEST.ESA131.SEQMSG0
//** EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//** SYSPRT DD SYSOUT=*, DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//** EQQLOG DD SYSOUT=*
//** SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//** EQQDUMP DD SYSOUT=*
//** EQQMSG DD SYSOUT=*
//** SYSPRT DD SYSOUT=*
//** EQQPRINT DD DSN=*.SORT.SORTOUT,DISP=(OLD,DELETE)
//** SYSIN DD *
//*********************************************************************
```
EQQDRTOP - Replan current planning period

The EQQDSTOP batch program renews the symphony file. The file is created from the active current plan.
EQQDSTOP - Renew the symphony file

SYSIN requirements

None.

JCL example

//*********************************************************************
//* DAILY PLANNING - REFNEW SYMPHONY FROM CP
//*********************************************************************
//DSTOP EXEC PGM=EQQBATCH,PARM='EQQDSTOP',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APLLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//*********************************************************************
//SYSIN DD *

EQQDTTOP - Produce a trial plan

The EQQDTTOP batch program provides a trial plan for the current planning period. A long-term plan must already exist before you can make a trial plan.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The layout of the SYSIN DD statement for the first step is:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789012345678901234567890</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YYMMDDHHMMhhmmW Tabcdefgh (format 1)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YYMMDDHHMMymmdhhmmTabcddefgh (format 2)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

- **YYMMDDHHMM** = Plan start date and time (both formats)
- **hhmm** = Hours and minutes to be extended (format 1)
  - (for example, 02400 is 24 hours)
- **W** = Work day flag. Set to W or blank. (format 1)
  - (for example, 04800W means extend for 48 days, excluding any free days)
- **ymmdhhmm** = Plan end date and time (format 2)
- **T** = Type of trial plan (C=REPLAN,N=NEXT,F=FUTURE)
- **abcdefg** = Report flags. Set to 1 or 0. (both formats)
  - **a** = WORKSTATION SUMMARY
  - **b** = DAILY OPERATING PLANS
  - **c** = WORKSTATION PLANS
  - **d** = NOT USED. ALWAYS 0.
  - **e** = NONREPORTING WORKSTATIONS
  - **f** = NOT USED. ALWAYS 0.
  - **g** = PLANNED RESOURCE UTILIZATION
  - **h** = NOT USED. ALWAYS 0.

JCL example

//*********************************************************************
//* DAILY PLANNING - PLAN A TRIAL PERIOD
//*********************************************************************
//EQQDTTOP EXEC PGM=EQQBATCH,PARM='EQQDTTOP',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APLLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
**EQQDTTOP - Produce a trial plan**

```plaintext
//EQQPARM DD DISP=SHR, DSN=EID.EIDA.PARM(EIDA)
//EQQDIN DD DSN=&A, DISP=(NEW, PASS),
   // DCB=(RECFM=FB, LRECL=558, BLKSIZE=5580),
   // SPACE=(CYL,(3,1)), UNIT=3390
//EQQTROUT DD DUMMY, DCB=BLKSIZE=100
/**
 /* CHANGE WHEN TRACKLOG IS TO BE SAVED
 /* DISP=MOD MUST BE SPECIFIED
 /* DCB PARAMETERS MUST BE DEFINED IN DD STATEMENT:
 /* LRECL=32000 RECFM=VBS
//*****************************************************************************
//EQQMLOG DD SYSOUT=*  
//EQQDMSG DD SYSOUT=*  
//EQQDUMP DD SYSOUT=*  
//SYSMDUMP DD DISP=MOD, DSN=EID.EIDA.SYSDUMPB 
//EQQWSDS DD DISP=SHR, DSN=EID.EIDAR3.WS 
//EQQADDS DD DISP=SHR, DSN=EID.EIDAR3.AD 
//EQQLTDS DD DISP=SHR, DSN=EID.EIDAR3.LT 
//EQQCP1DS DD DISP=SHR, DSN=EID.EIDAR3.CP1 
//EQQCP2DS DD DISP=SHR, DSN=EID.EIDAR3.CP2 
//EQQNCPDS DD DISP=SHR, DSN=EID.EIDAR3.NCP 
//EQQTARC DD DISP=SHR, DSN=EID.EIDA.JTARC 
//EQQCKPT DD DISP=SHR, DSN=EID.EIDA.CKPT 
/**
//SYSIN DD *
  950505083902400 F11101010 
  /* = PLSTRRTIME PLENDTIME REPORTS 
  /** PLSTRRTIME(YYMMDDHHMM) = PLAN START DATE, TIME IF REFRESH 
  /** PLENDTIME (YYMMDDHHMM) = PLAN END TIME 
  /** (HHHMM) = PLAN EXTENSION IN HOURS AND MINUTES 
  /** COUNTING ALL DAYS 
  /** (HHHMMW) = PLAN EXTENSION IN HOURS AND MINUTES 
  /** COUNTING ONLY WORK DAYS 
  /** TYPE OF TRIAL PLAN (T) = REQUESTED TYPE (C=REPLAN, N=NEXT, F=FUTURE) 
  /** REPORTS(ABCDEFGH) = REQUESTED REPORTS (1/0) 
  /** A = WORKSTATION SUMMARY 
  /** B = DAILY OPERATING PLANS 
  /** C = WORKSTATION PLANS 
  /** D = NOT USED ALWAYS 0 
  /** E = NONREPORTING WORKSTATIONS 
  /** F = NOT USED ALWAYS 0 
  /** G = PLANNED RESOURCE UTILIZATION 
  /** H = NOT USED ALWAYS 0 
*****************************************************************************
//STEP2 DAILY PLAN - SORT OF REPORT RECORDS 
*****************************************************************************
/**
//SORT EXEC PGM=SORT, REGION=4096K, TIME=1440, COND=(8, LT)
//SYSPRINT DD SYSOUT=*  
//SYSOUT DD SYSOUT=*  
//SORTIN DD DSN=&A, DISP=(OLD, DELETE)  
//SORTOUT DD DISP=(NEW, PASS),
   // DCB=(RECFM=FB, LRECL=558, BLKSIZE=5580),
   // SPACE=(CYL,(3,1)), UNIT=3390 
//SORTWK01 DD DISP=(NEW, DELETE),
   // DCB=(RECFM=FB, LRECL=558, BLKSIZE=5580),
   // SPACE=(CYL,(3,1)), UNIT=3390 
//SORTWK02 DD DISP=(NEW, DELETE),
   // DCB=(RECFM=FB, LRECL=558, BLKSIZE=5580),
   // SPACE=(CYL,(3,1)), UNIT=3390 
//SORTWK03 DD DISP=(NEW, DELETE),
   // DCB=(RECFM=FB, LRECL=558, BLKSIZE=5580),
   // SPACE=(CYL,(3,1)), UNIT=3390 
//SYSIN DD *
   SORT FIELDS=(1,130,CH,A)
```
EQQDTTOP - Produce a trial plan

/*
 ********************************************
//** STEP3 DAILY PLAN - PLAN NEXT PRINT THE REPORTS
 ********************************************
/*
//DPREPORT EXEC PGM=EQQBATCH,PARM='EQQDPRPT',COND=(8,LT),
 // REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//SYSPRINT DD SYSOUT=*,
 //DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSMDUMP DD SYSOUT=*
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//EQQPOUT DD DSN=*.SORT.SORTOUT,DISP=(OLD,DELETE)
//SYSSIN DD *
//*/

EQQEVPGM - Issue commands in batch
Use EQQEVPGM to issue the TSO commands for Tivoli Workload Scheduler for z/OS in batch.

SYSIN requirements
The SYSIN data can consist of one or more BACKUP, OPINFO, OPSTAT, SRSTAT, or WSSTAT commands. These commands have the same syntax as the corresponding TSO commands, which are described, with examples, in the Appendix A, "TSO commands," on page 647.

JCL example
//STEP1 EXEC PGM=EQQEVPGM
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQMLOG DD SYSOUT=*
//SYSSIN DD *
SRSTAT 'TAPES' SUBSYS(EIDA)
 AVAIL(YES)
 DEVIATION(-1)

EQQJVPRT - Print JCL variables
The EQQJVPRT batch program prints a single JCL variable table or all JCL variable tables (when no SYSIN data is specified).

SYSIN requirements
The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The parameter is the name of the variable table, starting in column 1.

JCL example
//'**************
//'** PRINT JCL VARIABLE TABLES
//'**************
//@JVPRT EXEC PGM=EQQBATCH,PARM='EQQJVPRT',REGION=4096K
//@STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//@EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//@EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//@JVRPT DD SYSOUT=*,
//@ DCB=(RECFM=FBA,LRECL=121,BLKSZIE=6050)
//@EQQMLOG DD SYSOUT=*

EQQLTCRE - Create a new long-term plan

The EQQLTCRE batch program creates a completely new long-term plan. The input parameters to the batch program give the start date and end date to the long-term plan that is created.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The layout of the SYSIN DD statement is:

COLUMNS 1 2 3
123456789012345678901234567890
YYMMDD yymmdd LINE 1 (REQUIRED)

where:

YYMMDD = STARTDATE (YY = YEAR, MM = MONTH, DD = DAY)
yymmdd = ENDDATE (yy = YEAR, mm = MONTH, dd = DAY)

JCL example

//*********************************************************************
//* LONG TERM PLANNING - CREATE THE LONG TERM PLAN
//*********************************************************************
//LTCREATE EXEC PGM=EQQBATCH,PARM='EQQLTCRE',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCTEST.ESA131.SEQQMSG0
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//LTREPORT DD SYSOUT=*
//   DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=* 
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//LTPRIN DD DCB=(RECFM=FB,LRECL=66,BLKSZIE=4620),
//   SPACE=(4620,(300,300)),UNIT=SYSDA
//LTPROUT DD DCB=(RECFM=FB,LRECL=66,BLKSZIE=4620),
//   SPACE=(4620,(300,300)),UNIT=SYSDA
//LTOCIN DD DCB=(RECFM=FB,LRECL=735,BLKSZIE=4410),
//   SPACE=(4410,(2400,2400)),UNIT=3390
//LTOCOUT DD DCB=(RECFM=FB,LRECL=735,BLKSZIE=4410),
//   SPACE=(4410,(2400,2400)),UNIT=3390
//LTPRWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOCWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOCWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOCWK03 DD SPACE=(CYL,(1,5)),UNIT=3390

EQQJVPRT - Print JCL variables
EQQLTMOA - Modify the long-term plan for all applications or extend the long-term plan

The EQQLTMOA batch program can modify the long-term plan for all applications or extend the long-term plan. (If the start date of the long-term plan does not move forward each time you extend it, you might have an uncompleted occurrence at the start of the long-term plan.)

SYSIN requirements

SYSIN data is required only for extending the long-term plan. The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. Specify either an end date or the number of days to be extended.

**JCL example**

```plaintext
//************************************************************
// ** LONG TERM PLANNING - EXTEND THE LONG TERM PLAN
//************************************************************
/#LTEXTEND EXEC PGM=EQQBATCH,PARM='EQQLTMOA',REGION=4096K
//!STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//!EQQMLIB DD DISP=SHR,DSN=OPCTEST.ESA131.SEQQMSG0
//!EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//!LTREPORT DD SYSOUT=*,
//!   DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//!EQQMLOG DD SYSOUT=*
//!SYSPRINT DD SYSOUT=*
//!SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//!EQQDUMP DD SYSOUT=*
//!EQQDMSG DD SYSOUT=*
//!LTOLIN DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
//!   SPACE=(CYL,(1,5)),UNIT=3390
//!LTOLOUT DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
//!   SPACE=(CYL,(1,5)),UNIT=3390
//!LTPRIN DD DCB=(RECFM=FB,LRECL=66,BLKSIZE=4620),
//!   SPACE=(4620,(300,300)),UNIT=3390
//!LTPROUT DD DCB=(RECFM=FB,LRECL=66,BLKSIZE=4620),
//!   SPACE=(4620,(300,300)),UNIT=3390
//!LTOCIN DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
//!   SPACE=(4410,(300,300)),UNIT=3390
//!LTOCOUT DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
//!   SPACE=(4410,(300,300)),UNIT=3390
//!LTOLWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//!LTOLWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//!LTOLWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
```
**EQQLTMOA - Modify the long-term plan for all applications or extend the long-term plan**

//LTOCWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOCWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOCWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//EQQADD5 DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//EQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR,
  //AMP=('BUFNI=10, TUFN=10')
//EQLTBP DD DSN=EID.EIDAR3.LB,DISP=SHR
//EQLODS DD DSN=EID.EIDAR3.LD,DISP=SHR,
  //AMP=('BUFNI=10, TUFN=10')
//SYSIN DD *
0002
  // YMMDD  DDD  WHERE
  // YMMDD  = EXTEND DATE OR BLANK
  // DDD  = PLAN EXTENSION IN DAYS
  // DDD  = COUNTING ALL DAYS
  // OR BLANK
/

**EQQLTMOO - Modify the long-term plan for one application**

The EQQLTMOO batch program modifies the long-term plan for one application.

**SYSIN requirements**

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The parameter is the application ID, beginning in column 1.

**JCL example**

[code]
EXEC PGM=EQQBATCH,PARM='EQQLTMOO',REGION=4096K
STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
LTREPORT DD SYSOUT=*,
  DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
EQQMLOG DD SYSOUT=*
SYSPRINT DD SYSOUT=*
SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
EQQDUMP DD SYSOUT=*
EQQDMSG DD SYSOUT=*
LTOCIN DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
  SPACE=(4410,(2400,2400)),UNIT=3390
EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
EQQADD5 DD DSN=EID.EIDAR3.AD,DISP=SHR
EQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR
SYSIN DD *
APP4
/* = APPLICATIONID
/* APPLICATION ID = SELECTED APPLICATION
[/code]

**EQQLTPRT - Print the long-term plan**

The EQQLTPRT batch program prints information on the long-term plan. The input parameters to the batch program support these print functions:

- Print the long-term plan for all applications.
- Print the long-term plan for one application.

**SYSIN requirements**

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The layout of the SYSIN DD statement is:
**JCL example**

```jcl
//*********************************************************************
//* LONG TERM PLANNING - PRINT THE LONG TERM PLAN
//*********************************************************************
//LTPPRINT EXEC PGM=EQQBATCH,PARM='EQLTPRT',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APPLIB
//EQQLIB DD DISP=SHR,DSN=OPCTEST.ESA131.SEQQMGR
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//LTREPORT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQLOG DD SYSOUT=*
//EQQPRINT DD SYSOUT=*
//SYSDUMP DD SYSOUT=*
//SYSCOUNT DD SYSOUT=*
//SYSSDUMP DD DDIE.SYS.SDSMGR
//EQQDUMP DD SYSOUT=*
//EQQMSG DD SYSOUT=*
//LTOOLIN DD DCB=(RECFM=FBS,LRECL=32756,BLKSIZE=13200),
// SPACE=(CYL,(1,5)),UNIT=3390
//LTOOLOUT DD DCB=(RECFM=FBS,LRECL=32756,BLKSIZE=13200),
// SPACE=(CYL,(1,5)),UNIT=3390
//LTPRIN DD DCB=(RECFM=FB,LRECL=66,BLKSIZE=4620),
// SPACE=(4620,(300,300)),UNIT=3390
//LTPROUT DD DCB=(RECFM=FB,LRECL=66,BLKSIZE=4620),
// SPACE=(4620,(300,300)),UNIT=3390
//LTORIN DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
// SPACE=(4410,(2400,2400)),UNIT=3390
//LTOROUT DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
// SPACE=(4410,(2400,2400)),UNIT=3390
//LTOLWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOLWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOLWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTCWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTCWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTCWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//EQQLDS DD DSN=EID.EIDAR3.LD,DISP=SHR,
// AMP=('BUFDI=10,BUFDN=10')
//SYSIN DD *
95080307519511222400DRALL
/* = STARTDATE (YYMMDD)
/* = STARTTIME (HHMM)
/* = ENDDATE (YYMMDD)
/* = ENDTIME (HHMM)
/* = REPORTTYPE (F=FULL, D=DEPENDENCIES)
/* = SORT (R,O,A)
/* = ALL
/*
```
EQQLTTRY - Make a trial long-term plan

EQQLTTRY - Make a trial long-term plan

The EQQLTTRY batch program simulates an long-term plan create, extend, or modify all.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The layout of the SYSIN DD statement is as follows:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>YYMMDD</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>yymmdd</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>dddd</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

- YYMMDD = plan start date, or blank
- yymmdd = plan end date (format 1)
- dddd = extension length, in days (format 2)

You can specify the following for a trial long-term plan:

- **Create**  Specify a start date and an end date to produce a trial create.
- **Extend**   To produce a trial extend, either specify an extension length, or an end date that is later than the end of the current long-term plan but no start date.
- **Modify all** Do not specify a start date, end date, or extension length to produce a trial modify all.

JCL example

```plaintext
//*********************************************************************
//* LONG TERM PLANNING - PLAN A TRIAL PERIOD
//*********************************************************************
//LTPRYS EXEC PGM=EQQBATCH,PARM='EQQLTTRY',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCTEST.ESA131.SEQQMSG0
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//LTREPORT DD SYSOUT=*,
//   DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//LTOLIN DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
//   SPACE=(CYL,(1,5)),UNIT=3390
//LTOLOUT DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
//   SPACE=(CYL,(1,5)),UNIT=3390
//LTPRIN DD DCB=(RECFM=FB,LRECL=66,BLKSIZE=4620),
//   SPACE=(4620,(300,300)),UNIT=SYSDA
//LTPROUT DD DCB=(RECFM=FB,LRECL=66,BLKSIZE=4620),
//   SPACE=(4550,(300,300)),UNIT=SYSDA
//LTOCIN DD DCB=(RECFM=FB, LRECL=735, BLKSIZE=4410),
//   SPACE=(4410,(2400,2400)),UNIT=SYSDA
//LTOCOUT DD DCB=(RECFM=FB, LRECL=735, BLKSIZE=4410),
//   SPACE=(4410,(2400,2400)),UNIT=SYSDA
//LTLWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTLWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTLWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTPRWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOCWK01 DD SPACE=(CYL,(1,5)),UNIT=3390
```
EQQLTTRY - Make a trial long-term plan

//LTOCWK02 DD SPACE=(CYL,(1,5)),UNIT=3390
//LTOCWK03 DD SPACE=(CYL,(1,5)),UNIT=3390
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//EQQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR
//EQQLTBKPD DD DSN=EID.EIDAR3.LB,DISP=SHR
//EQQLDDS DD DSN=EID.EIDAR3.LD,DISP=SHR,
// AMP=('BUFNI=10,BUFND=10')
//SYSIN DD *
0003
//* YMMDDYMMDDDDDDDD WHERE
//* YMMDD = NEW START DATE OR BLANK
//* YMMDD = NEW END DATE OR BLANK
//* DDDD = PLAN EXTENSION IN DAYS
//* COUNTING ALL DAYS
//* OR BLANK
/*

EQQOIBAT - Print operator instructions

The EQQOIBAT batch program prints or removes the operator instructions from
the operator instruction database. These print options can be selected:
• Print operator instructions in AD ID order.
• Purge old temporary operator instructions, up to the date specified.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input
parameter to the batch program. The definition of the input parameters is:

COLUMN 1 2 3
123456789012345678901234567890
5 format 1 - print in application order
7ymmd format 4 - purge instructions
where:
   for format 4, instructions are purged if the valid-to date is
   less than yymmd.

JCL example

******************************************************************************
/* OPERATOR INSTRUCTIONS - BATCH PROGRAM
******************************************************************************
//OIBAT EXEC PGM=EQQBATCH,PARM='EQQOIBAT',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//OIREPORT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*
// SYSPRINT DD SYSOUT=*
// SYSPUT DD SYSOUT=*
// SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//O1WKIN DD DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),
// SPACE=(CYL,(2,4)),UNIT=3390
//O1WKOUT DD DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),
// SPACE=(CYL,(2,4)),UNIT=3390
//O1WKKW01 DD SPACE=(CYL,(1,4)),UNIT=3390
//O1WKKK02 DD SPACE=(CYL,(1,4)),UNIT=3390
//O1WKKK03 DD SPACE=(CYL,(1,4)),UNIT=3390
//EQQOIDS DD DSN=EID.EIDAR3.OI,DISP=SHR
//SYSIN DD *
5
*/
EQQOIBLK - Operator instructions: perform mass update

EQQOIBLK - Operator instructions: perform mass update

The EQQOIBLK batch program updates operator instructions in the operator instruction database. These mass update options are supported:

- Update using sequential input without replacing old operator instructions
- Update using sequential input and replace old operator instructions

If the subsystem is active, comment out the EQQOIDS ddname, and the program uses the program interface to update the instructions. If the subsystem is stopped, include the EQQOIDS ddname, and the program uses VSAM I/O to update the database. The subsystem name is specified by the SUBSYS keyword of the BATCHOPT statement.

SYSIN requirements

The SYSIN data is normally supplied from the select panel; it is the input parameter to the batch program. The definition of the input parameters is:

BY (INSERT NEW, REPLACE OLD, OPERATOR INSTRUCTIONS)
BN (INSERT NEW ONLY)

The EQQOPIN DD will reference the name of the selected sequential file. For a description of the contents of this file, see “Layout of the operator instruction data set” on page 699.

JCL example

//********************************************************************
//* OPERATOR INSTRUCTIONS - BATCH INPUT FROM A SEQ. data set
//* Note:
//* If output is to a stopped opc subsystem, change *QQOIDS to EQQOIDS
//********************************************************************
//OIBLK EXEC PGM=EQQBATCH,PARM='EQQOIBLK',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.STAGE.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCSHIP.ESA131.MSGS
//EQQOPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//OIREPORT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//OIWKIN DD DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),
// SPACE=(CYL,(2,4)),UNIT=3390
//OIWKOUT DD DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),
// SPACE=(CYL,(2,4)),UNIT=3390
//OIWKWK01 DD SPACE=(CYL,(1,4)),UNIT=3390
//OIWKWK02 DD SPACE=(CYL,(1,4)),UNIT=3390
//OIWKWK03 DD SPACE=(CYL,(1,4)),UNIT=3390
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//********************************************************************
//* IF THE CONTROLLER IS STOPPED WHEN RUNNING THIS JOB,
//* CHANGE //*QQOIDS TO //EQQOIDS,
//********************************************************************
//EQQOPIN DD DSN=XRAYNER.OPERATOR.MESSAGES,DISP=SHR
//SYSIN DD *
//BY *
/*
Layout of the operator instruction data set

Specify your operator instructions in the sequential data set defined by the EQQOPIN DD statement. The data set should consist of 80-byte records and be fixed blocked.

Each instruction has two parts:
- Header (a single record)
- Instruction (one or more records)

A header record must precede each operator instruction. Here is the format for the header:

```
OPC KEY=aaaaaaaaaaaaaaaannntttttttttt,VALFROM=eeeeeeeeee
```

The characters OPC must be in columns 1 to 3. The variables have the following meaning:

- `aaaaaaaaaaaaaaa` Application ID. This is required and must be 16 characters (padded with blanks to the right, if necessary).
- `nnn` A valid operation number. This is a required field.
- `ttttttttttt` Specifies when the operator instruction is valid to. This field is required if you specify a valid-from (VALFROM) date and time. Use the YYMMDDHHMM format. If both valid-to and valid-from dates and times are specified, the instruction is temporary. If both are blank, the instruction is considered permanent.
- `eeeeeeeeee` Specifies when the operator instruction is valid from. This field is required if you specify a valid-to date and time. Use the YYMMDDHHMM format. If both valid-from and valid-to dates and times are specified, the instruction is temporary. If both are blank, the instruction is considered permanent.

The keywords KEY and VALFROM can be in any order, but they must be separated by either blanks or a comma.

The operator instruction itself consists of 80-byte records following the instruction header. The first 72 positions contain the instruction text; the last 8 positions are ignored. A maximum of 443 records is allowed for any one instruction.

For example:

```
OPC KEY=APPLICATIONABC 0059308311200,VALFROM=9308011200
If the job ends abnormally, it can be restarted from
the CPU_005 operation without any JCL changes.
```

```
OPC KEY=APPLICATIONXXX 015
If the job ends abnormally, it can be restarted
from the CPU_020 operation without changing the JCL.
```

You should include the name of the operator-instruction sequential-input data set on the SPECIFYING SEQUENTIAL FILE NAME panel. You can display this panel by selecting option 1.5.5.2 from the main menu.
EQQPURGE - Purge data lookaside facility object

The EQQPURGE batch program receives a data set name as input and purges the corresponding data lookaside facility (DLF) object by invoking the macro COFSDONO. Sample JCL is in the EQQPROC member of the samples library. Refer to Customization and Tuning for more details.

SYSIN requirements
None.

JCL example

```shell
//EQQPROC EXEC PGM=EQQPURGE,PARM=
//**********************************************************************
//* THIS PROCEDURE IS STARTED FROM OPC TO INITIATE PURGE
//* PROCESSING OF DLF OBJECTS USED BY OPC JOBS.
//* EQQPROC INVOKES PROGRAM EQQPURGE.
//* PROGRAM EQQPURGE READS JCL FROM data set WITH DDNAME
//* JCLIN AND UPDATES IT WITH THE NAME OF THE DLF OBJECT TO BE
//* PURGED, WHEN THE JCL IS UPDATED EQQPURGE WRITES THE JCL TO
//* JES INTERNAL READER.
//* SAMPLE EQQJCLIN CONTAINS SAMPLE JCL FOR FILE JCLIN
//* JCLIN ORGANIZATION : PS (OR A PDS MEMBER)
//* RECORD LENGTH: 80
//* TO GET THIS PROCEDURE TO WORK, YOU SHOULD DO THE FOLLOWING:
//* 1. CHANGE STEPLIB DSN TO THE Tivoli Workload Scheduler for z/OS LOAD LIBRARY NAME
//* 2. CHANGE JCLIN DSN TO data set NAME CONTAINING JCL
//* 3. EQQPROC MUST BE DEFINED AS A STARTED TASK ACCORDING TO
//* Documentation for the Security Product Used.
//* WHEN THE INPUT TO PROGRAM EQQPURGE IS INVALID, ONE OF THE FOLLOWING
//* WTO'S IS WRITTEN TO THE OPERATOR CONSOLE:
//* EQQPURGE : PARAMETER CARD SPECIFIES AN INVALID NAME
//* EQQPURGE : ERROR OPENING FILE WITH DDNAME JCLIN
//* EQQPURGE : INVALID JCLIN RECORD LENGTH, MUST BE 80
//* EQQPURGE : ERROR OPENING INTERNAL READER
//* STEPLIB DD DISP=SHR,DSN=Tivoli Workload Scheduler for z/OS LOAD LIBRARY NAME
//* JCLIN DD DISP=SHR,DSN=INPUT JCL data set NAME
//* JCLOUT DD SYSOUT=(A,INTRDR)
//* SYSPRINT DD SYSOUT=* 

EQQPURGE returns one of the following codes upon completion:
CC = 00: Retained object found and deleted.
Explanation: The DLF object has been purged from Hiperspace.
Action: EQQPURGE terminates.
User response: None.

CC = 02: The object did not exist in DLF.
Explanation: This is a return code from COFSDONO.
Action: EQQPURGE terminates; no object is purged.
User response: None.

CC = 04: The data set name could not be located in catalog
Explanation: The catalog locate for this data set name was unsuccessful.
Action: EQQPURGE terminates; no object is purged.
User response: None.

Note: data sets that are to be handled by Tivoli Workload Scheduler for z/OS Hipertask support must be cataloged, because the JCL invoking EQQPURGE is distributed to all systems where DLF is
running, according to routing information in the JCL. Expect this return code when EQQPURGE executes on a system where the DLF object does not exist.

**CC = 08:** The data set name passed has an invalid length.  
**Explanation:** The data set name passed to EQQPURGE is either of length 0 or longer than 44 characters.  
**Action:** EQQPURGE terminates without further processing.  
**User response:** Verify that EQQPURGE was invoked with a valid Tivoli Workload Scheduler for z/OS resource name.

**CC = 40:** DLF is not active.  
**Explanation:** EQQPURGE received return code 40 from the COFSDONO macro indicating that DLF is not active.  
**Action:** EQQPURGE terminates without further processing.  
**User response:** Start DLF if it should be active on this system, or update the file in EQQPROC with ddname JCLIN to avoid routing jobs to this system.

**CC = 44:** Unexpected error in DLF.  
**Explanation:** EQQPURGE received return code 44 from the COFSDONO macro indicating that an unexpected error occurred in DLF when processing the purge request.  
**Action:** EQQPURGE terminates without further processing.  
**User response:** Contact your system programmer.

If EQQPURGE receives invalid input, one of the following WTOs is written to the operator console:

**EQQPURGE:** Parameter card specifies an invalid name.  
**Explanation:** The object name passed to EQQPURGE is either of length 0 or longer than 44 characters.  
**Action:** The JCL from the JCLIN data set is neither substituted nor written to the internal reader.  
**User response:** Verify that EQQPROC has been started by Tivoli Workload Scheduler for z/OS.

**EQQPURGE:** Error opening file with ddname JCLIN.  
**Explanation:** EQQPURGE could not open input file JCLIN.  
**Action:** EQQPURGE terminates.  
**User response:** Verify that the JCLIN statement is present in EQQPROC.

**EQQPURGE:** Invalid JCLIN record length; must be 80.  
**Explanation:** The record length of the data set with ddname JCLIN is not 80.  
**Action:** EQQPURGE terminates; no JCL is processed.  
**User response:** Reallocate JCLIN with the correct record length.

**EQQPURGE:** Error opening internal reader.  
**Explanation:** EQQPURGE could not open JES internal reader data set.  
**Action:** EQQPURGE terminates; no JCL is processed.
**EQPURGE - Purge data lookaside facility object**

**User response:**
Verify that the JCLOUT statement is present and correctly specified in EQQPROC.

**EQQSLTOP - File parse program, process parse statement**

The EQQSLTOP batch program provides a syntactic check of the script library members. In the EQQSCLIB ddname, there can be allocated more than one script library. For every member with a problem, the parsing process issues a message explaining the problem found and the message EQQ4033I, indicating the script library where the failing member is defined. In this way, even if different members with the same name are defined in different script libraries allocated in the EQQSCLIB ddname, the members in error are located immediately. If all script libraries defined in the EQQSCLIB ddname are empty, the message EQQ4031W is displayed.

**SYSIN requirements**

You can customize the program as shown hereafter.

The following example shows that all members of script libraries allocated in the EQQSCLIB ddname are processed:

```
SYSIN DD *
PARSE
```

The following example shows that a specific member of the script libraries allocated in the EQQSCLIB ddname are processed:

```
SYSIN DD *
PARSE MEMBER (membername)
```

where membername is the name of the script library member to be processed. It can be long up to 8 characters, and one row can contain only one parse statement. Empty rows are allowed, all the rows are processed until the end of the SYSIN ddname.

**JCL example**

```c
//********************************************************************
//* FILE PARSE PROGRAM
//********************************************************************
//PARSE EXEC PGM=EQQBATCH,PARM='EQQSLTOP',REGION=4M
//STEP1 LIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=OPCID.EIDA.PARM(EIDA)
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQMLOG DD SYSOUT=*
//EQQMSG DD SYSOUT=*
//SYSIN DD *
//PARSE MEMBER (membername)
//PARSE MEMBER (....)
```

**EQQWSPRT - Print all workstation descriptions**

The EQQWSPRT batch program provides a printout of all workstation descriptions from the workstation description database.

**SYSIN requirements**

None.
**EQQWSPRT - Print all workstation descriptions**

**JCL example**

```plaintext
//PRTWORK EXEC PGM=EQQBATCH,PARM='EQQWSPRT',REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APPLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//WSREPORT DD SYSOUT=*,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTEM DD SYSOUT=*
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//SYSIN DD *
```

**EQQYLTOP - Batch loader**

The EQQYLTOP batch program lets you create or update information in the application-description or operator-instruction databases. For a complete description, including SYSIN requirements and JCL examples, see “What is the batch loader?” on page 189.

---

**Sending generated reports by email**

Reports generated by running batch programs can optionally be sent to other users by email. If you want to use this facility, perform the following steps:

1. On your computer, create a copy of the file `SENDREPORT.MAC`, available in the `C:\Send Report Macro` directory of the TEP Int Filewatch Feat Agent CD. Change the name, but keep the extension `.MAC`. You can create more than one copy of this file to suit your needs; each copy must have a different name.

2. Copy this file into the installation path of the Personal Communications tool; for example: `C:\Documents and Settings\Administrator\Application\Data\IBM\Personal Communications`.

3. Customize this file with all the following required parameters, as shown in the following example:

   ```plaintext
   [PCOMM SCRIPT HEADER]
   LANGUAGE=VBSCRIPT
   DESCRIPTION=MAIL
   [PCOMM SCRIPT SOURCE]
   autECLSession.SetConnectionByName(ThisSessionName)

   REM This line calls the macro subroutine
   subSub1_

   sub subSub1_()
     autECLSession.autECLOIA.WaitForAppAvailable
     autECLSession.autECLOPS.SendKeys "[home]"
     autECLSession.autECLOPS.SendKeys "start"
     autECLSession.autECLOPS.SendKeys "[enter]"
     autECLSession.autECLOPS.SendKeys "6"
     autECLSession.autECLOPS.SendKeys "[enter]"
     autECLSession.autECLxfer.ReceiveFile "outputfile", "zOSreport", "ASCII CRLF NOCLEAR"
   ```

---

Appendix B. Batch programs
Sending generated reports by email

```vbscript
autECLSession.autECLPS.SendKeys "swap"
autECLSession.autECLPS.SendKeys "[enter]"

Dim wsh
Set wsh=CreateObject("WScript.Shell")
wsh.Run "mailto:receiveremail" + "?Subject=emailsubject" + "&Body=emailtext"
+ "&attach=outputfile" end sub
```

where:

- **outputfile**
  - Is the report file on your PC, for example, C:\audit.txt.

- **zOSreport**
  - Is the z/OS report you want to send, for example, TWS.AUDIT.LIST.

- **receiveremail**
  - Is the email address you want to send the report to, for example, user@company.com.

- **emailsubject**
  - Is the subject of your email, for example, Automatically%20sent%20by %20zOS.

- **emailtext**
  - Is your email body text, for example, Received%20file%20from%20zOS %20and%20prepare%20the%20email%20with%20attachment.

4. Send the report by using either the macro icon **Start a macro** in the Personal Communications toolbar or, from the **Actions** menu, by clicking **Start Playing Macro/Script**.

**Note:** When customizing the parameters, you cannot insert spaces, but must use the %20 sequence for every single space character, for example, Automatically%20sent%20by%20zOS.
Appendix C. Report examples

This appendix provides examples of the following reports:
- Calendars
- Periods
- Workstation descriptions
- Application descriptions
- JCL variable tables
- Mass update
- Operator instructions
- Long-term plan
- Daily planning

A description of the batch programs and examples of the JCL are provided in the Appendix B, “Batch programs,” on page 673.

Note: On many reports, the year and month are not shown for the planned start, planned end, and latest out dates. Only the day number is shown. It is assumed that the year and month of these dates are the same as the current year and current month. However, in some cases a date in a preceding month or a following month has been calculated. This is indicated after the day number by a < for a preceding month or a > for a following month.

Calendar reports

PRINTOUT OF THE CALENDAR
========================
GENERAL INFORMATION
---------
CALENDAR ID : DEFAULT
CALENDAR DESCRIPTION : Default calendar
WORK DAY END TIME : 01.00
START DATE OF PRINTOUT: 03/02/14
END DATE OF PRINTOUT: 03/12/31
LATEST USER UPDATE : 03/02/13
UPDATED BY : KATARIN
DEFAULT STATUS : MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY SUNDAY
WORK WORK WORK WORK WORK FREE FREE

Figure 286. Calendars - General information
Calendar reports

<table>
<thead>
<tr>
<th>DATE</th>
<th>STATUS</th>
<th>COMMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/04/14</td>
<td>FREE</td>
<td>Good Friday</td>
</tr>
<tr>
<td>03/12/24</td>
<td>FREE</td>
<td>Christmas Eve</td>
</tr>
<tr>
<td>03/12/25</td>
<td>FREE</td>
<td>Christmas Day</td>
</tr>
<tr>
<td>03/12/26</td>
<td>FREE</td>
<td>Boxing Day</td>
</tr>
<tr>
<td>04/01/01</td>
<td>FREE</td>
<td>New Year's Day</td>
</tr>
</tbody>
</table>

Figure 287. Calendars - Description of specific dates

Status of days within the printout period

<table>
<thead>
<tr>
<th>MONTH</th>
<th>MONDAY</th>
<th>TUESDAY</th>
<th>WEDNESDAY</th>
<th>THURSDAY</th>
<th>FRIDAY</th>
<th>SATURDAY</th>
<th>SUNDAY</th>
</tr>
</thead>
<tbody>
<tr>
<td>02</td>
<td>03/02/14 W</td>
<td>03/02/15 W</td>
<td>03/02/16 W</td>
<td>03/02/17 W</td>
<td>03/02/18 F</td>
<td>03/02/19 F</td>
<td></td>
</tr>
<tr>
<td>02/03</td>
<td>03/02/20 W</td>
<td>03/02/21 W</td>
<td>03/02/22 W</td>
<td>03/02/23 W</td>
<td>03/02/24 W</td>
<td>03/02/25 F</td>
<td>03/02/26 F</td>
</tr>
<tr>
<td>03</td>
<td>03/03/06 W</td>
<td>03/03/07 W</td>
<td>03/03/08 W</td>
<td>03/03/09 W</td>
<td>03/03/10 W</td>
<td>03/03/11 F</td>
<td>03/03/12 F</td>
</tr>
<tr>
<td>03/04</td>
<td>03/03/13 W</td>
<td>03/03/14 W</td>
<td>03/03/15 W</td>
<td>03/03/16 W</td>
<td>03/03/17 W</td>
<td>03/03/18 F</td>
<td>03/03/19 F</td>
</tr>
<tr>
<td>04</td>
<td>03/04/03 W</td>
<td>03/04/04 W</td>
<td>03/04/05 W</td>
<td>03/04/06 W</td>
<td>03/04/07 W</td>
<td>03/04/08 F</td>
<td>03/04/09 F</td>
</tr>
<tr>
<td>04/04</td>
<td>03/04/17 W</td>
<td>03/04/18 W</td>
<td>03/04/19 W</td>
<td>03/04/20 W</td>
<td>03/04/21 W</td>
<td>03/04/22 F</td>
<td>03/04/23 F</td>
</tr>
</tbody>
</table>

Figure 288. Calendars - Status of days

Period reports

Start date of printout: 03/02/14
End date of printout: 04/12/31
Calendar ID: DEFAULT
Calendar description: Default calendar

Figure 289. Description of period characteristics - General information
**Workstation description reports**

**DESCRIPTION OF PERIOD CHARACTERISTICS**

<table>
<thead>
<tr>
<th>PERIOD DESCRIPTION</th>
<th>JCL VARIABLE TABLE</th>
<th>TYPE</th>
<th>INTERVAL ORIGIN DATE</th>
<th>INTERVAL END DATE</th>
<th>INTERVAL ORIGIN DATE</th>
<th>INTERVAL END DATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRSTTHU First Thursday in each month</td>
<td>NONCYCLIC</td>
<td>03/01/05 005 THU</td>
<td>03/02/02 033 THU</td>
<td>03/03/02 061 THU</td>
<td>03/04/06 096 THU</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>03/05/04 124 THU</td>
<td>03/06/01 152 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>03/07/06 187 THU</td>
<td>03/08/03 215 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>03/09/07 250 THU</td>
<td>03/10/05 278 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>03/11/02 306 THU</td>
<td>03/12/07 341 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/01/04 004 THU</td>
<td>04/02/01 032 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/03/07 067 THU</td>
<td>04/04/07 098 SUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/05/02 123 THU</td>
<td>04/06/06 158 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/07/04 186 THU</td>
<td>04/08/01 214 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/09/05 249 THU</td>
<td>04/10/03 277 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/11/07 312 THU</td>
<td>04/12/05 340 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/01/04 004 THU</td>
<td>04/02/01 032 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/03/07 067 THU</td>
<td>04/04/07 098 SUN</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/05/02 123 THU</td>
<td>04/06/06 158 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/07/04 186 THU</td>
<td>04/08/01 214 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/09/05 249 THU</td>
<td>04/10/03 277 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>04/11/07 312 THU</td>
<td>04/12/05 340 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FSSTHU7 First Thursday in July</td>
<td>NONCYCLIC</td>
<td>03/07/06 187 THU</td>
<td>04/07/04 186 THU</td>
<td>05/07/03 184 THU</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

>>> END OF PERIOD PRINTOUT <<<

**Figure 290. Description of period characteristics**

---

**Workstation description reports**

**PRINTOUT OF WORK STATION DESCRIPTIONS**

<table>
<thead>
<tr>
<th>WORK STATION NAME</th>
<th>Defaults</th>
<th>DESCRIPTION</th>
<th>TRANSPORT TIME</th>
<th>DURATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td></td>
<td>Main JES processor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>WORK STATION TYPE</th>
<th>COMPUTER</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>REPORTING ATTRIBUTE</th>
<th>AUTO.</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PRINTOUT ROUTING</th>
<th>SYSPRINT</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CONTROL ON SERVERS</th>
<th>YES</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>R1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>PLANNING</th>
<th>YES</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OPTIONS</th>
<th>CONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>NO</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SPLITTABLE</th>
<th>NO</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>JOB SETUP</th>
<th>RESOURCE 1</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>NAME</th>
<th>R2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DESTINATION</th>
<th>NAME</th>
<th>R2</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>STARTED TASK</th>
<th>NO</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>CONTROL</th>
<th>NO</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>WTO</th>
<th>NO</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>WAIT</th>
<th>NO</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>LAST UPDATED BY</th>
<th>XRAYNER</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>DAY/DATE</th>
<th>ON 94/04/29 AT 17.35</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>OPEN TIME</th>
<th>PARALLEL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>INTERVAL</th>
<th>RESOURCES</th>
<th>ALTERNATE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>R1 R2 WS</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>STA</th>
<th>00.00 - 24.00</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>END OF WORK STATION REPORT</th>
</tr>
</thead>
</table>

>>> END OF PERIOD PRINTOUT <<<

**Figure 291. Workstation description report**
Application description reports

REPORTS FOR SUBSYS EIDA

DESCRIPTION OF ALL WORK STATIONS CLOSED DATES

<table>
<thead>
<tr>
<th>DATE</th>
<th>COMMENTS</th>
<th>WORK STATION CLOSED</th>
<th>LAST UPDATED</th>
</tr>
</thead>
<tbody>
<tr>
<td>03/09/16</td>
<td>Night shift operation only</td>
<td>08.00 - 24.00</td>
<td>ANDERSM 99/04/21 15.21</td>
</tr>
<tr>
<td>03/09/17</td>
<td>Night shift operation only</td>
<td>08.00 - 24.00</td>
<td>ANDERSM 99/04/21 15.21</td>
</tr>
<tr>
<td>03/10/15</td>
<td>Hardware upgrade shutdown</td>
<td>00.00 - 24.00</td>
<td>ANDERSM 99/04/21 15.21</td>
</tr>
</tbody>
</table>

Figure 292. All workstations closed report

Application description reports

REPORTS FOR SUBSYS EIDA

PRINTOUT OF APPLICATION DESCRIPTIONS

REPORT TYPE: CROSS-REFERENCE OF JOBNAMES AND ACTIVE APPLICATIONS

<table>
<thead>
<tr>
<th>JOBNAME</th>
<th>APPL ID</th>
<th>VALID TO</th>
<th>OPERATION ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICSA</td>
<td>CICSA</td>
<td>71/12/31</td>
<td>STC1_010</td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>PAYBACKP</td>
<td>71/12/31</td>
<td>CPU1_015 WT01_030</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>PAYDAILY</td>
<td>71/12/31</td>
<td>CPU1_020 SETP_010 WT01_005</td>
</tr>
<tr>
<td>PAYMONTH</td>
<td>PAYMONTH</td>
<td>71/12/31</td>
<td>CPU1_040</td>
</tr>
<tr>
<td>PAYSLIP</td>
<td>PAYSLIP</td>
<td>71/12/31</td>
<td>CPU1_050 PRT1_099</td>
</tr>
<tr>
<td>PAYQUERY</td>
<td>PAYQUERY</td>
<td>71/12/31</td>
<td>CPU1_050</td>
</tr>
<tr>
<td>PAYRECOV</td>
<td>PAYRECOV</td>
<td>71/12/31</td>
<td>CPU1_015</td>
</tr>
<tr>
<td>PAYTAXYR</td>
<td>PAYTAXYR</td>
<td>71/12/31</td>
<td>CPU1_015</td>
</tr>
<tr>
<td>PAYTRANS</td>
<td>PAYTRANS</td>
<td>71/12/31</td>
<td>CPU1_040</td>
</tr>
<tr>
<td>PAYWEEK</td>
<td>PAYWEEK</td>
<td>71/12/31</td>
<td>CPU1_020</td>
</tr>
<tr>
<td>PAYWSLIP</td>
<td>PAYWSLIP</td>
<td>71/12/31</td>
<td>CPU1_030 PAY1_095 PRT1_090</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>APPL ID</th>
<th>VALID TO JOBNAME</th>
<th>OPERATION ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>CICSA</td>
<td>71/12/31 CICSA</td>
<td>STC1_010</td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>71/12/31 PAYBACKP</td>
<td>CPU1_015 WT01_030</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>71/12/31 PAYDAILY</td>
<td>CPU1_020 SETP_010 WT01_005</td>
</tr>
<tr>
<td>PAYM</td>
<td>71/12/31 PAYM</td>
<td>CPU1_040</td>
</tr>
<tr>
<td>PAYSLIP</td>
<td>71/12/31 PAYSLIP</td>
<td>CPU1_050 PRT1_099</td>
</tr>
<tr>
<td>PAYZ</td>
<td>71/12/31 PAYZ</td>
<td>CPU1_040</td>
</tr>
<tr>
<td>PAYQ</td>
<td>71/12/31 PAYQ</td>
<td>CPU1_050</td>
</tr>
<tr>
<td>PAYR</td>
<td>71/12/31 PAYR</td>
<td>CPU1_015</td>
</tr>
<tr>
<td>PAYWSLIP</td>
<td>71/12/31 PAYWSLIP</td>
<td>CPU1_030 PAY1_095 PRT1_090</td>
</tr>
</tbody>
</table>

Figure 293. Cross-references of job names and active applications
Application description reports

ABCD COMPANY
REPORTS FOR SUBSYS EIDA
03 MAY 03

PRINTOUT OF APPLICATION DESCRIPTIONS

REPORT TYPE: CROSS-REFERENCE OF APPLICATIONS AND EXTERNAL DEPENDENCIES

APPL ID S OP ID JOBNAME PREDECESSORS S SUCCESSORS
-------- --- -------- -------- ---------------------------------- --- ---------------------------------
PAYBACKP A CPU1_015 PAYBACKP PAYDAILY CPU1_020 PAYDAILY
PAYM1 CPU1_050 PAYSLIP
PAYM2 CPU1_040 PAYTRANS
PAYTAXYR CPU1_015 PAYTAXYR
PAYDAILY A CPU1_020 PAYDAILY
A PAYBACKP CPU1_015 PAYBACKP
A PAYM1 CPU1_040 PAYM1
A PAYM2 CPU1_040 PAYTRANS
A PAYTAXYR CPU1_015 PAYTAXYR
PAYM1 CPU1_040 PAYMONTH PAYDAILY CPU1_020 PAYDAILY
A PAYM1 CPU1_040 PAYM1
A PAYM2 CPU1_040 PAYTRANS
A PAYM2 CPU1_040 PAYMONTH
PAYW CPU1_020 PAYWEEK
PAYM2 A CPU1_040 PAYTRANS PAYM1 CPU1_040 PAYMONTH
PAYTAXYR A CPU1_015 PAYTAXYR
PAYW A CPU1_020 PAYWEEK PAYDAILY CPU1_020 PAYDAILY
A PAYM2 CPU1_040 PAYTRANS
A PAYTAXYR CPU1_015 PAYTAXYR

<<<<<< END OF APPLICATION DESCRIPTION PRINTOUT <<<<<<

Figure 294. Cross-references of applications and external dependencies

ABCD COMPANY
REPORTS FOR SUBSYS EIDA
11 JUN 03

PRINTOUT OF APPLICATION DESCRIPTIONS

REPORT TYPE: DETAILED SELECTION DATE: 03/05/11

APPL ID: PAYDAILY

COMMON DATA

APPL TEXT/ CALENDAR ID/ TYPE NUM. NUM. GROUP DEFINITION
OWNER ID OWNER DESCRIPTION VALID FROM - TO STATUS PRI RUNC OPER GROUP ID LATEST UPDATE USER
---------------- ------------------------ ------------------ -------- --- ---- ---- --------- ------------------
daily payroll jobs APPL
SAMPLE payroll application 03/01/29 71/12/31 ACTIVE 5 1 3 94/06/08 XRAYNER

OFFSET-BASED RUN CYCLE INFORMATION

VALID APPL PERIOD RC INPUT RC VALID FREE DAY RUN CYCLE DESCRIPTION
FROM STATUS NAME TYPE RUN DAYS ARRIVE DEADLINE FROM TO RULE JCL VARIABLE TABLE
-------- -------- -------- -------- -------------- ------ -------- ----------------- -------- ----------------------
03/01/29 ACTIVE DAILY NORMAL 001 12.00 16.00 03/01/29 71/12/30 CANCEL PAY
<REPEAT EVERY 00.10 FROM 19.11 UNTIL 21.00>

Figure 295. Application descriptions - Common data

Priority (PRI): 1=low, 8=high, 9=urgent
Application description reports

Run cycle type (RC TYPE): normal (times and days when run) or negative (times and days when not to run).

NUMBER PS  Number of parallel servers required by the operation.
NUMBER R1  Number of R1 workstation resources required by the operation.
NUMBER R2  Number of R2 workstation resources required by the operation.
DUR        Estimated duration of the operation at the workstation.
SPC RES    Number of special resources allocated by the operation. If present, the names will appear on following lines with the usage indicator S for shared or X for exclusive.
HRC        Highest return code that is to be treated as NOT ended-in-error.

MVS OPTIONS
A          Automatic submit, Y or N
H          Hold/release, Y or N
T          Time dependent operation, Y or N
S          Suppress time job if late, Y or N
E          Error tracking, Y or N
D          Deadline WTO, Y or N
R          Restartable operation, Y, N or blank=default
R          Reroutable operation, Y, N or blank=default
C          Clean up type, A=automatic, I=immediate, or N=none

INTERNAL OPERATION LOGIC

Figure 296. Application descriptions - Operation data

Figure 297. Application descriptions - Internal operation Logic
REPORT TYPE: OPERATIONS USING PARTICULAR WORKSTATIONS

WORKSTATION: CPU1

<table>
<thead>
<tr>
<th>OPER ID</th>
<th>OPERATION TEXT</th>
<th>NUMBER</th>
<th>SPEC.RESOURCE</th>
<th>PS</th>
<th>R1</th>
<th>R2</th>
<th>DUR</th>
<th>APPL ID</th>
<th>VALID FROM</th>
<th>VALID TO</th>
<th>STATUS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1_050</td>
<td>job1 in app1</td>
<td>1</td>
<td>Ô</td>
<td>2</td>
<td>00.05</td>
<td>APP1</td>
<td>03/03/03</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>job 2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>APP2</td>
<td>03/03/03</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>job3 in app1</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>00.05</td>
<td>APP5</td>
<td>03/03/03</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_015</td>
<td>job1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.03</td>
<td>APP6</td>
<td>03/05/06</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_015</td>
<td>job2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>03.00</td>
<td>CP</td>
<td>03/03/08</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_015</td>
<td>job1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>JOB8</td>
<td>03/01/29</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_010</td>
<td>job1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>03.00</td>
<td>LTP</td>
<td>03/03/09</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_010</td>
<td>job2</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>MAWS</td>
<td>03/04/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_010</td>
<td>job3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.01</td>
<td>MEME</td>
<td>03/05/05</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_010</td>
<td>job4</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>00.01</td>
<td>MEMEWEEK</td>
<td>03/05/08</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_015</td>
<td>Daily payroll backup</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>PAYBACKP</td>
<td>03/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_020</td>
<td>Runs pay04 and pay06</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>PAYDAILY</td>
<td>03/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_040</td>
<td>Monthly payroll job</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00.05</td>
<td>PAYM1</td>
<td>03/01/29</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Print monthly pay slips</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>PAYM2</td>
<td>03/01/29</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Create bank giro tape</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00.10</td>
<td>PAYM2</td>
<td>03/01/29</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Run as required</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00.05</td>
<td>PAYQUERY</td>
<td>03/02/02</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_015</td>
<td>Recover payroll database</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>PAYRECOV</td>
<td>03/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_015</td>
<td>Third Thursday in July</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00.05</td>
<td>PAYTAXYEAR</td>
<td>03/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_020</td>
<td>pay07, pay10, and pay16</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00.05</td>
<td>PAY</td>
<td>03/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_030</td>
<td>pay14, pay15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>00.05</td>
<td>PAY</td>
<td>03/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_030</td>
<td>pay16</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>00.20</td>
<td>TEST3</td>
<td>03/05/01</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 298. Application descriptions - Operations using particular work stations

NUMBER: PS = parallel servers; R1, R2 = workstation resources 1 and 2

DUR: Duration

The special resource name is followed by the quantity and whether it is exclusive (X) or shared (S). A quantity of * means ALL.
Run the EQQAXR00 program to produce the cross-reference list, selecting up to five fields for the sort sequence. Using the AD DATABASE panel, select 6 (XRF OF ITEMS) from the PRINTING APPLICATIONS panel.
JCL variable table report

JCL variable table report

ABCD COMPANY
REPORTS FOR SUBSYS EIDA
03 MAY 03
17:48

PRINTOUT OF JCL VARIABLES

JCL VARIABLE TABLE ID : PAY
OWNERID : SAMPLE

VARIABLE NAME

DEPT
DESCRIPTION :
DEFAULT VALUE : EXTRA
UPPER CASE : N
SETUP : N
SUBSTITUTION
EXIT :
VALUE REQUIRED : N
DEFAULT POS : 00
DIALOG TEXT :

VALIDATION CRITERIA

 verification type :
variable length : 00
numeric :
comparison oper :
validation pattern :
valid ranges/values :

VALIDATION CRITERIA

variable dependencies

INDEPEND. VARIABLE VALUE SETS CURRENT VARIABLE
NONE TO

Figure 300. JCL variable table report
**Mass update reports**

**EXPLANATION OF CONTENTS OF ‘UPDATED FIELD’ USED IN THE REPORT**

- **A. COMMON FIELDS**
  - A - 1: APPLICATION TEXT
  - A - 2: OWNER ID
  - A - 3: OWNER TEXT
  - A - 4: PRIORITY
  - A - 5: AUTHORITY GROUP ID
  - A - 6: CALENDAR ID
  - A - 7: APPLICATION GROUP ID

- **B. RUNCYCLE DEFINITION FIELDS**
  - B - 1: RUNCYCLE TEXT
  - B - 2: PERIOD NAME/RULE NAME
  - B - 3: INPUT ARRIVAL TIME
  - B - 4: DEADLINE DAY
  - B - 5: DEADLINE TIME
  - B - 6: FREE DAY RULE
  - B - 7: IN EFFECT FROM
  - B - 8: OUT OF EFFECT FROM
  - B - 9: VARIABLE TABLE
  - B - 10: RULE NAME
  - B - 11: REPEAT EVERY
  - B - 12: REPEAT END TIME
  - B - 13: AUTOMATIC RELEASE
  - B - 14: TIME DEPENDENT
  - B - 15: SUPPRESS IF LATE
  - B - 16: FORM NUMBER
  - B - 17: SYSOUT CLASS
  - B - 18: SMOOTHING FACTOR
  - B - 19: FEEDBACK LIMIT
  - B - 20: INPUT ARRIVAL DAY
  - B - 21: INPUT ARRIVAL TIME
  - B - 22: DEADLINE DAY
  - B - 23: DEADLINE TIME
  - B - 24: DEADLINE WTO
  - B - 25: RESTARTABLE
  - B - 26: REROUTEABLE
  - B - 27: CLEAN UP TYPE
  - B - 28: USER SYSOUT
  - B - 29: EXPANDED JCL
  - B - 30: EXTERNAL MONITOR
  - B - 31: CENTRALIZED SCRIPT
  - B - 32: USE EXTENDED JOB NAME
  - B - 33: EXTENDED JOB NAME
  - B - 34: USER SYSOUT
  - B - 35: EXPANDED JCL
  - B - 36: EXTERNAL MONITOR
  - B - 37: CENTRALIZED SCRIPT
  - B - 38: USE EXTENDED JOB NAME
  - B - 39: EXTENDED JOB NAME

- **C. OPERATION DATA FIELDS**
  - C - 1: OPERATION TEXT
  - C - 2: WORKSTATION NAME
  - C - 3: DURATION
  - C - 4: NUMBER OF SERVERS
  - C - 5: WORK STATION RESOURCE 1
  - C - 6: WORK STATION RESOURCE 2
  - C - 7: SPECIAL RESOURCE NAME
  - C - 8: JOB NAME
  - C - 9: JOB CLASS
  - C - 10: AUTOMATIC ERROR TRACKING
  - C - 11: HIGHEST RETURN CODE
  - C - 12: SUBMIT
  - C - 13: AUTOMATIC RELEASE
  - C - 14: TIME DEPENDENT
  - C - 15: SUPPRESS IF LATE
  - C - 16: FORM NUMBER
  - C - 17: SYSOUT CLASS
  - C - 18: SMOOTHING FACTOR
  - C - 19: FEEDBACK LIMIT
  - C - 20: INPUT ARRIVAL DAY
  - C - 21: INPUT ARRIVAL TIME
  - C - 22: DEADLINE DAY
  - C - 23: DEADLINE TIME
  - C - 24: DEADLINE WTO
  - C - 25: RESTARTABLE
  - C - 26: REROUTEABLE
  - C - 27: CLEAN UP TYPE
  - C - 28: USER SYSOUT
  - C - 29: EXPANDED JCL
  - C - 30: EXTERNAL MONITOR
  - C - 31: CENTRALIZED SCRIPT
  - C - 32: USE EXTENDED JOB NAME
  - C - 33: EXTENDED JOB NAME

- **D. INTERNAL PREDECESSOR FIELDS**
- **E. EXTERNAL PREDECESSOR FIELDS**

---

**Figure 301. Mass update report - Application descriptions**

**Figure 302. Mass update report - Cleanup type updated**
### APPL ID OP VALID FROM VALID TO WSID | LAST UPDATE

<table>
<thead>
<tr>
<th>NO</th>
<th>DATE TIME USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>----</td>
<td>------------------</td>
</tr>
<tr>
<td>015</td>
<td>71/12/31 24.00 CPU1</td>
</tr>
</tbody>
</table>

### INSTRUCTION TEXT

The daily backup for the payroll database. Run this job after all the payroll jobs (daily, weekly, monthly, or yearly) that are scheduled on the day. When the job finishes, OPC/ESA sends a message to open the CICSA payroll datasets.

---

<table>
<thead>
<tr>
<th>APPL ID OP VALID FROM VALID TO WSID</th>
<th>LAST UPDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYDAILY 010 71/12/31 24.00 SETP</td>
<td>03/01/30 18.36 XRAYNER</td>
</tr>
</tbody>
</table>

### INSTRUCTION TEXT

This job can run only when the resource 'PAYROLL.DATABASE' is freed by CICSA. To do this, run the CICS transaction PAYC.

---

<table>
<thead>
<tr>
<th>APPL ID OP VALID FROM VALID TO WSID</th>
<th>LAST UPDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYDAILY 020 71/12/31 24.00 CPU1</td>
<td>03/01/30 18.43 XRAYNER</td>
</tr>
</tbody>
</table>

### INSTRUCTION TEXT

This job can run only when the CICSA database is closed. Close it with the PAYC transaction. The job runs two programs: PAY04 transfers payroll transactions from the CICSA database to a sequential dataset. PAY06 validates the transactions and updates the payroll database if there are no errors. The recovery procedure is as follows:

- If PAY04 fails, the job can be rerun.
- If PAY06 fails before the database has been updated (RC=4), you can rerun the job after payroll have corrected the data.
- If PAY06 has updated the database, you must run the PAYRECOV job before rerunning this job.

---

Figure 303. Operator instructions report
The following data appears on the heading page:

<table>
<thead>
<tr>
<th>Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report produced by</td>
<td>Long-term report function used to produce report</td>
</tr>
<tr>
<td>Report type</td>
<td>FULL or DEPENDENCIES/NORMAL or TEMPORARY</td>
</tr>
<tr>
<td>Latest user update</td>
<td>Date when long-term plan was last updated</td>
</tr>
<tr>
<td>The LTP covers period</td>
<td>Start day and end day selected in long-term plan panel</td>
</tr>
<tr>
<td>Printout is for period</td>
<td>Start of first occurrence to end of last occurrence in plan</td>
</tr>
<tr>
<td>Printout is for application(s)</td>
<td>ALL, or an application ID</td>
</tr>
<tr>
<td>Sort order of printout</td>
<td>Requested or default sort order</td>
</tr>
</tbody>
</table>
Long-term plan reports

This data appears on an long-term plan full report:

<table>
<thead>
<tr>
<th>Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appl ID</td>
<td>Application ID of the occurrence</td>
</tr>
</tbody>
</table>

---

### Long-term plan report for applications, sorted by run date

<table>
<thead>
<tr>
<th>APPL ID</th>
<th>INPUT</th>
<th>OWNER ID/OP ID</th>
<th>INPUT ARRIVAL</th>
<th>APPL/OPERATION TEXT</th>
<th>DEPEND.</th>
<th>DEPENDING OCCURRENCES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CP</td>
<td></td>
<td>SAMPLE3</td>
<td>12.00</td>
<td>16.00</td>
<td>7</td>
<td>current plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU1_050</td>
<td>P</td>
<td>CPU1_050 12.00</td>
</tr>
<tr>
<td>LTP</td>
<td></td>
<td>SAMPLE3</td>
<td>12.00</td>
<td>15.00</td>
<td>7</td>
<td>long-term-plan</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU1_050</td>
<td>S</td>
<td>CPU1_050 08/05/03 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CPU1_050 08/05/04 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CPU1_050 08/05/05 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CPU1_050 08/05/06 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CPU1_050 08/05/07 12.00</td>
</tr>
<tr>
<td>PAYBACKP</td>
<td></td>
<td>SAMPLE</td>
<td>12.00</td>
<td>08/05/03 06.00</td>
<td>5</td>
<td>backup payroll database</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU1_015</td>
<td>P</td>
<td>CPU1_020 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>P</td>
<td>PAY daily job and other payroll jobs</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td></td>
<td>SAMPLE</td>
<td>12.00</td>
<td>16.00</td>
<td>5</td>
<td>daily payroll jobs</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU1_020</td>
<td>S</td>
<td>CPU1_015 08/05/04 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CPU1_015 08/05/05 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CPU1_015 08/05/06 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>S</td>
<td>CPU1_015 08/05/07 12.00</td>
</tr>
</tbody>
</table>

---

### Long-term plan report for applications, sorted by owner

<table>
<thead>
<tr>
<th>APPL ID/OP ID</th>
<th>INPUT ARRIVAL</th>
<th>Deadline</th>
<th>APPL/OPERATION TEXT</th>
<th>DEPEND.</th>
<th>DEPENDING OCCURRENCES:</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYBACKP</td>
<td>08/05/04 12.00</td>
<td>08/05/05 06.00 5</td>
<td>backup payroll database</td>
<td>CPU1_015</td>
<td>P PAYDAILY CPU1_020 08/05/04 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>wait for the daily job and other payroll jobs</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>08/05/04 12.00</td>
<td>08/05/04 16.00 5</td>
<td>daily payroll jobs</td>
<td>CPU1_020</td>
<td>S PAYBACKP CPU1_015 08/05/04 12.00</td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>08/05/05 12.00</td>
<td>08/05/06 06.00 5</td>
<td>backup payroll database</td>
<td>CPU1_015</td>
<td>P PAYDAILY CPU1_020 08/05/05 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>wait for the daily job and other payroll jobs</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>08/05/05 12.00</td>
<td>08/05/05 16.00 5</td>
<td>daily payroll jobs</td>
<td>CPU1_020</td>
<td>S PAYBACKP CPU1_015 08/05/05 12.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU1_020 08/05/05 12.00</td>
</tr>
</tbody>
</table>

Figure 305. Long-term plan report for applications, sorted by run date

Figure 306. Long-term plan report for applications, sorted by owner

Appendix C. Report examples 717
Long-term plan reports

- **Owner ID/op ID**: Application owner ID and the operation ID of any operations that have changed operation date on the long-term plan file.
- **Input arr**: Application input arrival time.
- **Deadline**: Application deadline date and time.
- **PI**: Application priority.
- **Appl/operation text**: Application or operation text.
- **+ LTP Comment**: Comments generated from Tivoli Workload Scheduler for z/OS.
- **+ Var table**: Variable tables.
- **Depend. op ID**: ID of operation with external dependency.
- **Depending occurrences**: All external predecessors and successors listed (for reports on only one application, external dependencies not listed).
- **Type**: P (non-conditional predecessor), S (non-conditional successor), PC (conditional predecessor), or SC (conditional successor).
- **Appl ID**: Application ID of dependent occurrence.
- **Op ID, input arrival interval**: ID of dependent operation occurrence input interval.

### TOTAL DURATION PER WORKSTATION

<table>
<thead>
<tr>
<th>WS</th>
<th>DURATION</th>
<th>NO OF OPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU1</td>
<td>6.10</td>
<td>4</td>
</tr>
<tr>
<td>SETP</td>
<td>0.03</td>
<td>1</td>
</tr>
<tr>
<td>WTO1</td>
<td>0.02</td>
<td>2</td>
</tr>
</tbody>
</table>

*Figure 307. Long-term plan report - Total duration per workstation*

**Data**

**Explanation**

**Total duration per workstation**

Lists all workstations used.

**WS, duration and no of ops**

Workstation name, total time of all operations at the workstation, number of operations at the workstation.
Long-term plan reports

GRAND TOTAL WORKLOAD FOR PERIOD 03/04/28 TO 03/07/01 FOR WORKSTATION CPU1 : 151 HOURS

Figure 308. Long-term plan report - Grand total workload for period
The following data appears on the general information page:

### Data

**Requested planning period**  
Start and end times as specified in panel.

**Plan covers**  
Start time is the time the latest backup of the current plan was taken; end time is the end of the tail-end period. If no current plan exists, start time is the earliest input arrival time.

**Type of planning**  
Name of daily planning function used to produce this set of plans.

**Long-term plan used**  
Time the long-term plan was last updated; appears only if the long-term plan was used as input to this daily planning run.

**Old current plan used**  
Time the last current plan backup was taken; appears only if the current plan was used in this planning run.

**New current plan created**  
YES if new current plan created; NO if no plan was created.

**Workstation summary**  
Report produced, YES or NO.

**Daily operating plan**  
Report produced, YES or NO.

**Workstation plans**  
Report produced, YES or NO.
Daily planning reports

Input arrival lists
Input arrival lists are reported, YES or NO.

Nonreporting workstation plans only
Nonreporting workstation plans are reported, YES or NO.

Current period results
Current period results include start and end time for the period covered; NO if no report produced.

Previous period results
Previous period results include start and end time for the period covered; NO if no report produced.

Missed feedback report
Missed feedback report is reported, YES or NO.

Planned resource utilization
Planned resource utilization report is produced, YES or NO.

Actual resource utilization
Actual resource utilization report is produced, YES or NO.

Critical path
Critical path report is produced, YES or NO.

Number of planned applications
Number of planned applications indicates the number of occurrences in the daily plan.

Number of planned operations
Number of planned operations indicates the number of operations in the daily plan.

Number of messages
Number of messages indicates the number of error, warning, and information messages.

The daily operating plan shows all work to be done during the period covered by the plan. It is a printed copy of the current plan. All applications are listed in alphanumeric order. Where there are several occurrences of the same application, these are listed in order of their input arrival times.

The following data appears in the daily operating plan:

Data | Explanation
--- | ---
Application ID | Application name
Application owner | Owner ID of the application
Operation ID | Operation number and workstation name
P | Application priority
Daily planning reports

The report in Figure 311 lists the work to be done at the workstation in order of planned operation start times. You can request this report either for all workstations or only for nonreporting workstations.
Daily planning reports

On the PLAN FOR WORKSTATION report, Tivoli Workload Scheduler for z/OS leaves the input arrival day for the predecessor application blank if it is the same day as the start of the planning interval (shown in the report title).

On the INPUT ARRIVAL LIST report, Tivoli Workload Scheduler for z/OS leaves the input arrival day blank if it is the same day as the start of the planning interval (shown in the report title).

Daily planning produces workstation utilization histograms, which show how many operations are processed by the workstation.

The level of resource available during the interval; that is, the number of parallel servers as shown by the horizontal line. The histogram shows the number of operations planned concurrently during each 15-minute interval. A histogram is produced for each 24-hour interval included in the daily planning period.

Figure 312. Daily planning reports - Workstation utilization (parallel operations)
If R1 and R2 resources are specified to be planned for on the workstation; histograms will be produced showing utilization of these resources. These have a similar layout to the workstation utilization histograms described above. The number of operations using the special resource is shown for every 15-minute interval.

You see this data on the planned resource utilization report:

Data Explanation

---
IBM Tivoli Workload Scheduler for z/OS: Managing the Workload
Daily planning reports

Dynamically added
If this is YES, the resource was dynamically added during daily planning (it is not present in the database and was added because an operation needs it, and the DYNAMICADD keyword of the BATCHOPTS statement is set to YES).

Available
This is the planned availability, taken from the resource database interval values or the default availability. It does not show the overriding (global) availability, because daily planning does not use this for planning purposes.

Quantity
This is the planned quantity, taken from the resource database interval values or the default quantity. It does not show the overriding (global) quantity, because daily planning does not use this for planning purposes.

Planned utilization
This is the maximum number of the resource that is planned to be in use in the interval.

Allocation connect failures
This is the number of times that the planning program could not allocate this resource to an operation because the required workstation was not connected to the resource.

Allocation quantity failures
This is the number of times that the planning program could not allocate this resource to an operation because there was not enough quantity. It does not include cases where the allocation was delayed (because of contention) but was successful later in the interval.

Contentions
This is the number of times that the planning program could not allocate this resource to an operation, so that the operation missed its latest start time. This is counted only once for each operation, in the interval containing the latest start time of the operation.

Reports for a previous planning period
Daily planning also provides a set of management reports for a previous planning period. These reports are produced when you run Plan Next Period or Replan Current Period. These reports are created if the PREVRES keyword on the BATCHOPT statement specifies YES. Two of the reports, completed applications and operations ended in error, can also be requested when you submit a daily planning batch job.

A previous planning period is a 24-hour period starting on a fixed-hour boundary as defined by the PLANHOUR keyword of the BATCHOPT statement.

The data is kept in the current plan until the next daily planning run has produced the reports and is then deleted. For example, if you have specified the fixed-hour boundary as 0800 and you are running a Plan Next Period at 0759 on Wednesday morning, you will get reports from the interval 0800 Monday to 0800 Tuesday. If you then run a Replan Current Period at noon Wednesday, you will get the reports from 0800 Tuesday to 0800 Wednesday.

When an occurrence has been reported in any of the previous period reports, it will not be reported again in any subsequent daily plans for the same planning period. This means that the reporting period might not cover a full previous 24
Reports for a previous planning period

hours when several daily plans have been executed within a planning period.

The report shown in Figure 315 shows the number of applications processed in the period and gives the number of applications:

- With late input arrival, showing the average input delay
- That missed their deadlines, showing the average deadline delay
- That completed before their deadlines, showing the average deadline earliness
- That were rerun
- That were deleted

It provides a summary of events in the latest daily planning period.

Figure 315. Daily planning reports - Summary of completed applications

The report shown in Figure 315 shows the number of applications processed in the period and gives the number of applications:

- With late input arrival, showing the average input delay
- That missed their deadlines, showing the average deadline delay
- That completed before their deadlines, showing the average deadline earliness
- That were rerun
- That were deleted

Figure 316. Daily planning reports - Completed applications

The report shown in Figure 316 shows all applications completed or deleted in the given period. Also, each operation with a specified input arrival or deadline is printed in the report.
Reports for a previous planning period

An error code is present if specified when adding an occurrence into the long-term or current plan. This is defined as a rerun of a whole application, whereas a rerun of one or more operations in an application is reported in the error statistics report.

The report is produced automatically if specified at installation time or it might also be requested each time a Plan Next Period, a Replan Current Period, or a Print Current Period Results is run. If the report is requested at run time, a report covering the current period from the fixed-hour boundary of the current plan up to the present time will also be printed.

Note: The completion time used in the report is either the real completion time for the application or operation or DELETED for deleted applications. The real complete time for an application is defined as the time of the last completed operation with no specified input arrival or deadline. If the occurrence or operation has been manually completed, the real completion time is set to the time of the manual completion.

The report shown in Figure 317 shows applications that have had one or more operations rerun because of an error condition and which are now completed successfully.

The error duration (time lost due to errors) is printed if not zero. The rerun duration (time lost when rerunning completed applications) is printed for any application that has been added into the long-term current plans with a rerun (error) code. The report is listed in error-code order. It gives the total error duration, rerun duration, number of errors for each error code, and the total number of errors.

Notes:
1. Applications included in this report are not included in the completed applications report.
2. Only operations that are rerun after an error are included in this report. If the operation is manually completed, it is included in the completed applications report.
The report shown in Figure 318 lists all operations that have ended abnormally and are not yet taken care of.

The report is produced automatically if specified at installation time or it might be requested each time a Plan Next Period, a Replan Current Period, or a Print Current Period Results is run.

The report, if produced per shift using, for example, Print Current Period Results, could relieve operators of the task of having to record such information manually, and can also be used by the scheduler or shift supervisor to see where corrective action is necessary.

The report shown in Figure 319 is printed during a Plan Next Period or Replan Current Period run, if the scheduler has not been able to feedback application durations and deadlines.

The report lists all occurrences and operations where the feedback of the actual duration and deadline to the application description data set has not been possible.

Notes:
1. The planned deadline is the deadline set in the application description, represented in the format DD HH.MM, where DD is a day offset and HH.MM an absolute time.
2. The actual deadline is an absolute day and time, in the format DD HH.MM. For example, if the operation or occurrence completed on 29 March 2006 at 11:15, DD is 29 and HH.MM is 11:15.

The reasons for missed feedback include:

OUTSIDE LIMITS
Feedback is outside the limit set for feedback.
APPL IN USE
Application description was being updated by another user when feedback was attempted.

I/O PROBLEMS
AD record not found (if no operation ID printed) or operation not found in AD.

DL BEFORE IA
The occurrence completed before the IA was reached.

NO RUN CYCLE FOUND
It was not possible to match the occurrence with a run cycle in the application.

---

**Data**

**Explanation**

**Dynamically added**
If this is YES, the resource was dynamically added during daily planning or the life of the plan.

**Planned availability**
This is the planned availability, taken from the resource database interval values or the default availability. It does not show the overriding (global) availability, because daily planning does not use this for planning purposes.

**Actual availability**
This is the percentage of time that the resource was available as planned.

**Planned maximum utilization**
This is the maximum number of the resource that was planned to be in use in the interval.

**Actual maximum utilization**
This is the maximum number of the resource that was in use in the interval.

**Allocation failures**
It is increased by one each time Tivoli Workload Scheduler for z/OS tries unsuccessfully to allocate the resource to an operation during this interval; it can be increased many times for each waiting operation.

**Contention time**
This is the percentage of time that operations were waiting for the resource.

**Idle time**
This is the percentage of time that the resource was not allocated.
Database views

This section describes the views you can use to extract information from the DB2 database containing the historical run data that the scheduler archived to support the Dynamic Workload Console reporting feature.

For a description of the reports provided by the Dynamic Workload Console reporting feature, refer to Chapter 31, “Reporting with Tivoli Workload Scheduler for z/OS,” on page 631.

JOB_HISTORY_V

The JOB_HISTORY_V view displays historical data about jobs already run.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB_STREAM_NAME_IN_RUN</td>
<td>The name of the occurrence to which the job belonged when it run.</td>
<td>STRING</td>
<td>16</td>
</tr>
<tr>
<td>JOB_NAME</td>
<td>The name of the job.</td>
<td>STRING</td>
<td>8</td>
</tr>
<tr>
<td>WORKSTATION_NAME</td>
<td>The name of the workstation where the job was scheduled to run.</td>
<td>STRING</td>
<td>4</td>
</tr>
<tr>
<td>WORKSTATION_NAME_IN_RUN</td>
<td>The name of the workstation where the job ran.</td>
<td>STRING</td>
<td>4</td>
</tr>
<tr>
<td>JOB_OPERATION_NUMBER</td>
<td>The operation number of the job.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>JOB_STREAM_START_TIME</td>
<td>The occurrence input arrival.</td>
<td>TIMESTAMP</td>
<td>10</td>
</tr>
<tr>
<td>JOB_RUN_DATE_TIME</td>
<td>The actual start date and time.</td>
<td>TIMESTAMP</td>
<td>10</td>
</tr>
<tr>
<td>LATE_START_TIME</td>
<td>The actual start time, when it is later than the latest start time calculated by the scheduler.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>LATE_END_TIME</td>
<td>The actual end time, when it is later than the defined deadline.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>PLANNED_DURATION</td>
<td>The estimated duration of the job.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>TOTAL_ELAPSED_TIME</td>
<td>The actual duration of the job.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>LONG_DURATION</td>
<td>Indicates if the job duration exceeded the estimated duration, based on the long duration policy set in the DBOPT statement.</td>
<td>CHARACTER</td>
<td>1</td>
</tr>
<tr>
<td>JOB_STATUS</td>
<td>The status of the job. Possible values are:</td>
<td>CHARACTER</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>S=Successful</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>E=Error</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RERUN_NUMBER</td>
<td>The iteration number of the job. The first job run is identified by 1; the first job rerun is identified by 2, the second by 3, and so on.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>ERROR_CODE</td>
<td>The error code, if the status of the job is E=Error.</td>
<td>STRING</td>
<td>4</td>
</tr>
<tr>
<td>JOB_NUMBER</td>
<td>The number of the job.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>SUMISSION_DEST</td>
<td>The destination where the job was submitted. A destination marked with ***** identifies a local destination.</td>
<td>STRING</td>
<td>8</td>
</tr>
<tr>
<td>EXECUTION_DEST</td>
<td>The destination where the job was executed. A destination marked with ***** identifies a local destination.</td>
<td>STRING</td>
<td>8</td>
</tr>
</tbody>
</table>
The `JOB_STATISTICS_V` view displays runtime statistics for each job.

<table>
<thead>
<tr>
<th>Column name</th>
<th>Description</th>
<th>Type</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>JOB_NAME</td>
<td>The name of the job.</td>
<td>STRING</td>
<td>8</td>
</tr>
<tr>
<td>ABENDED_RUNS</td>
<td>The number of times the job ended in error.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>LONG_DURATION_RUNS</td>
<td>The number of times the job duration exceeded the estimated duration, based on the long duration policy set in the DBOPT statement.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>SUCCESSFUL_RUNS</td>
<td>The number of times the job ran successfully.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>LATE_START_RUNS</td>
<td>The number of times the job started later than its latest start time.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>LATE_END_RUNS</td>
<td>The number of times the job ended later than its deadline.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>TOTAL_RERUNS</td>
<td>The total number of reruns.</td>
<td>INTEGER</td>
<td>10</td>
</tr>
<tr>
<td>LAST_RUN_DATE</td>
<td>The actual start date and time of the last run.</td>
<td>TIMESTAMP</td>
<td>10</td>
</tr>
<tr>
<td>LAST_ELAPSED_TIME</td>
<td>The actual duration of the last run.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>MIN_RUN_DATE</td>
<td>The actual start date and time when the job took the shortest time to run.</td>
<td>TIMESTAMP</td>
<td>10</td>
</tr>
<tr>
<td>MIN_ELAPSED_TIME</td>
<td>The minimum duration run.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>MAX_RUN_DATE</td>
<td>The actual start date and time of the maximum duration run.</td>
<td>TIMESTAMP</td>
<td>10</td>
</tr>
<tr>
<td>MAX_ELAPSED_TIME</td>
<td>The maximum duration run.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>AVERAGE_ELAPSED_TIME</td>
<td>The average duration. The scheduler calculates it considering only the successfully completed jobs, based on the smoothing policy set in the DBOPT statement.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
<tr>
<td>TOTAL_ELAPSED_TIME</td>
<td>The sum of the times the job used the CPU and the time the job waited for other processes to release the CPU for all its runs.</td>
<td>INTEGER</td>
<td>12</td>
</tr>
</tbody>
</table>
Appendix D. Supported z/OS commands

You can start, stop, cancel, or modify Tivoli Workload Scheduler for z/OS using the following z/OS operator commands:

S  START
P  STOP
C  CANCEL
F  MODIFY

In addition, you can use the MODIFY (F) command to start and stop individual subtasks.

You can enter these commands from a multiple console support (MCS) console or from a program such as the spool display and search facility (SDSF). In both cases, the terminal or console operator must have the required authority to enter operator commands.

If Tivoli Workload Scheduler for z/OS history function is active and any DB2 maintenance is applied, stop and restart the controller to pick up the updated DB2 modules. If not, Tivoli Workload Scheduler for z/OS might experience ABEND0C4, ABEND0C1, or ABENDU3999 in the general service subtask or GS executors. Also, the controller might be unable to shut down cleanly because it is unable to close its DB2 session.

A space abend, terminating the mother task for Tivoli Workload Scheduler for z/OS, can occur in the following situations:

• If the EQQMLOG becomes full while processing START or MODIFY commands that require parsing of initialization parameters or specifying the NEWNOERR or NOERRMEM(MEMBER) options
• If the EQQMLOG is already full when START or MODIFY commands are issued

Starting the scheduler

To start Tivoli Workload Scheduler for z/OS, enter this z/OS operator command:
/S procname

where procname is the Tivoli Workload Scheduler for z/OS JCL procedure name.

If a started task with this name is already active, the second attempt to invoke it ends with an error message. If this happens, the started task in error cannot write an error message to the message log (DD name EQQMLOG) because the message log is already being used by the active started task. If Tivoli Workload Scheduler for z/OS is to run as a batch job, do not start it with an operator command. Instead, submit a batch job with the same name as Tivoli Workload Scheduler for z/OS subsystem. JES starts this job in the same manner as any ordinary job.

Note: Because Tivoli Workload Scheduler for z/OS uses JES exits, among other things, to track the progress of z/OS jobs, it does not start before JES is active.
Stopping the scheduler

To stop Tivoli Workload Scheduler for z/OS, enter the following z/OS operator command:

```
/P procname
```

where `procname` is the Tivoli Workload Scheduler for z/OS JCL procedure name.

If you are stopping a controller, the controller creates a backup copy of the current plan data set (if required) and ends all active functions.

When Tivoli Workload Scheduler for z/OS ends, it writes this message to the message log:

```
EQQZ086I NO ACTIVE OPC SUBTASKS.
OPC IS ENDING
```

Canceling the scheduler

If Tivoli Workload Scheduler for z/OS is still active 5 minutes after you enter the STOP operator command, you must cancel Tivoli Workload Scheduler for z/OS.

You might also need to cancel Tivoli Workload Scheduler for z/OS if the current plan is corrupt, because a normal shutdown causes a backup to the alternate file (refer to Customization and Tuning for details of current plan recovery). There are two ways to do this. The first is to enter:

```
/C procname,DUMP
```

where `procname` is the Tivoli Workload Scheduler for z/OS JCL procedure name.

This causes Tivoli Workload Scheduler for z/OS to end with a dump on the SYMDUMP file (if the DD name is in the started-task JCL). The second way is to enter:

```
/C procname
```

where `procname` is the Tivoli Workload Scheduler for z/OS JCL procedure name.

This causes Tivoli Workload Scheduler for z/OS to end without a dump.

If the STOP command is ineffective and you have no earlier documentation of the problem, cancel Tivoli Workload Scheduler for z/OS with a dump so that the error can be located.

Modifying the scheduler

Use the MODIFY command to supply information to Tivoli Workload Scheduler for z/OS after it has started. The syntax of the MODIFY command is:

```
/F procname,modifyoption
```

where:

`procname`

The Tivoli Workload Scheduler for z/OS JCL procedure name.

`modifyoption`

Can be one of the following:

- `S=taskname` Starts the specified subtask.
Modifying the scheduler

P=taskname  Stops the specified subtask.

*taskname* can be one of the following:

- **APPC**: APPC subtask.
- **AR**: Automatic recovery subtask.
- **A4**: APPC tracker router subtask.
- **CPH**: Critical path handler subtask.
- **DRT**: Data router subtask.
- **EMGR**: Event manager subtask.
- **ERDR**: All active event-reader subtasks.
- **EWTR**: Event writer subtask.
- **EXA**: External router subtask.
- **FL**: Fetch job log task.
- **GEN**: General service subtask.
- **IP**: TCP/IP router subtask.
- **JCC**: Job-completion-checker subtask.
- **NMM**: Normal-mode-manager subtask. The normal mode manager must be restarted as soon as possible after it has stopped. Many functions of Tivoli Workload Scheduler for z/OS require an active NMM task to execute successfully.
- **PENF53**: Deactivates the ENF mechanism implemented for the ENF 53 event. In this way, a dynamic time change for all the Tivoli Workload Scheduler for z/OS subsystems cannot be performed. By default, the ENF 53 mechanism is automatically deactivated when the controller is stopped.
- **PSU**: Pre-submit task.
- **RODM**: RODM subtask.
- **SENF53**: Activates the ENF mechanism implemented for the ENF 53 event, so that a dynamic time change for all the Tivoli Workload Scheduler for z/OS subsystems can be performed. By default, the ENF 53 mechanism is automatically activated at controller startup.
- **SUB**: Submit subtask.
- **TWS**: End-to-end with fault tolerance capabilities task.
- **VTAM**: Network communication function (NCF) subtask.
- **WSA**: Workstation analyzer subtask.

Only the tasks in Tivoli Workload Scheduler for z/OS subtask table can be activated by a MODIFY command. The subtask table is built when Tivoli Workload Scheduler for z/OS is started. This means that you can only start a task that has
stopped earlier in the current session. If you attempt to start a
started subtask or stop a stopped subtask, error message
EQQZ049W is issued, and no action is taken.

CPQSTA=ON  Activates the STATMSG(CPLOCK) message.
CPQSTA=OFF  Deactivates the STATMSG(CPLOCK) message.

DEPLOYCF  Use this keyword to manually start the deployment process
that refreshes the trackers event configuration file (member
EEQQEVLIST of the EQQJCLIB data set) according to the current
content of the controller EEQQEVLIB data set.

The process can update the event configuration file for the
trackers that are currently connected. At the end of the
deployment process, each connected tracker monitors for the
events that match the refreshed data, based on an in-storage
copy of the configuration file.

If a tracker destination is referenced in the controller routing
options (ROUTOPTS initialization statement), but it is not
currently connected, the tracker acquires the configuration file
data when the connection is restored.

DSPDEST  Lists the HTTP destinations that are currently used by the
controller. The list is stored in the message log.

DSPPRODD  Lists the DD names for data sets that cannot be discarded. The
list can be obtained in one of the following ways:
  • In the initial parameter statements: RCLOPTS DDPROT or
    DDPREM
  • With the MODIFY command /F procname,PROT/DD=member

DSPPRODS  Lists the data sets that cannot be discarded. The list can be
obtained in one of the following ways:
  • In the initial parameter statements: RCLOPTS DSNPROT or
    DSNPRMEM
  • With the MODIFY command /F procname,PROT/DS=member

DSPSTA  Displays, in message EEQQZ095, the status of statistics
messaging. The message indicates whether messaging is active
for EVENTS, CPLOCKS, GENSERV, and WSATASK. It also
gives the values currently set for EVELIM and STATIM. For
details, refer to Messages and Codes.

EVELIM=nnnn  Sets the new value of the EVELIM keyword of the JTOPTS
statement. Allowed values are 0 to 9999.

EVESTA=ON  Activates the STATMSG(EVENTS) message.
EVESTA=OFF  Deactivates the STATMSG(EVENTS) message.

GENSTA=ON  Activates the STATMSG(GENSERV) message.
GENSTA=OFF  Deactivates the STATMSG(GENSERV) message.

JCLDBG=ON  Activates the single JCL trace. For each job handled by WSA
task information, such as the elapsed time in milliseconds
needed to handle the job, retrieve the JCL, access the JS VSAM, or whatever else, will be shown.

This is a powerful trace and should be activated only for short periods of time to identify possible performance problems.

**JCLDBG=OFF** Deactivates the single JCL trace.

**LSTNOERR** The controller lists the NOERROR table content. An example of the command output follows:

```
EQQZ024I Current NOERROR table content:
EQQZ024I !------------------------------------------!----------!
EQQZ024I ! Statement ! Member !
EQQZ024I !------------------------------------------!----------!
EQQN067I ! ABC123.*.*.0016.GE ! STDCWSN !
EQQN067I ! ABC123.*.*.0012.NE ! STDCWSN !
EQQN067I ! ABC123.*.*.0016.EQ ! STDCWSN !
EQQN067I ! ABC123.*.*.0500.TO.0610 ! NOERR2 !
EQQN067I ! ABC123.*.*.0200.TO.0210 ! NOERR2 !
EQQN067I ! ABC123.*.*.0005.LT ! NOERR !
EQQZ024I !------------------------------------------!----------!
```

The information returned in the Member column can help you locate members to be updated.

**NEWDSLST** Use this keyword on the tracker side to rebuild the triggering selection table. The new table is read from member EQQEVLST (or EQQDSLST, if EQQEVLST does not exist) of the data set referenced by the EQQCLIB DD name in the started task JCL for the tracker. The new table replaces the table in ECSA.

**Note:** If EQQCLIB contains both EQQEVLST and EQQDSLST member, the resulting triggering selection table is the union of EQQEVLST and EQQDSLST. In this case, EQQEVLST data is processed first.

**NOERROR** You can use the commands NEWNOERR and NOERRMEM() only if NOERROR entries are coded under the LIST() keyword of the NOERROR init statement. For more details, refer to the *IBM Tivoli Workload Scheduler for z/OS Customization and Tuning*.

When you enter a NEWNOERR command, the program searches the controller PARMLIB member only for NOERROR init statements. If none are found, an empty table is created, thus deleting the entries which were loaded when the CONTROLLER started. Do not use the NEWNOERR command if the NOERROR list is coded under the NOERROR() keyword of the JTOPTS initialization statement. If you enter a NOERRMEM(membername) command, the current table is deleted and created again. The entries that had already been specified for membername are replaced with the entries found in the updated member of the EQQPARM library.

When using NOERROR, consider the following:
- Use this keyword only if you are sure you can stop the controller before updating the NOERROR data.
- Initialization statements cannot exceed 32 KB or 455 72-character lines. The available space for NOERROR entries in the initialization statement is more limited compared with using this keyword in the JTOPTS statement.
- Do not mix the three options for defining NOERROR entries:
Modifying the scheduler

- **JTOPTS NOERROR ()**
- **NOERROR LIST () in the main parmlib member**
- **NOERROR LIST () in separate parmlib members identified by the INCLUDE() initialization statement**

The options for dynamically updating controller data are mutually exclusive and using them incorrectly might cause the deletion of the active table.

- If you must dynamically update the NOERROR data without stopping the controller, and the NOERROR entries are defined in a separate parmlib member, place all NOERROR entries in a single parmlib member. In this way, you can use only one member name in the NOERRMEM command.

**NEWNOERR** Orders a controller to rebuild the NOERROR table, in the case NOERROR statements have been modified in the parameter library member that contains the JTOPTS statement.

**NOERRMEM(member)**

Orders a controller to rebuild the NOERROR table, in the case NOERROR statements have been modified in a parameter library member that was specified in an INCLUDE statement.

**NOERRMEM(M1)**

Order a controller to delete all NOERROR codes defined by member M1, once you have previously changed M1 to contain only comments. The modified member can contain a different number of NOERROR codes than the original member.

**Note:** The scheduler opens the EQQPARM library when Tivoli Workload Scheduler for z/OS is started and parameter library members (residing in library extents), that have been created, cannot be accessed, after have been opened. To avoid this problem, the data sets that define the EQQPARM library should be allocated without any secondary extents.

**PROT ([DD=mem1], [DS=mem2])**

Replaces the currently used list contents of the members (mem1 and mem2) inside the PDS parameter library, if DD or DSN is protected. At least one of the two keywords DD or DS must be specified. The list of protected DD and DSN is dropped when mem1 or mem2 is blank. Also, the keywords DD and DS remove the list of protected DD names and DS names.

For example, if DDPROT is used in the parameter library at startup of the controller, the command drops this list and replaces it with the contents of mem1.

**QUELEN=nnnn**

Sets the new value of the QUEUELEN keyword of the JTOPTS statement. Allowed values are 0 to 9999, but a minimum value of 5 is forced.

**RFRDEST** If you modify, add, or delete a z-centric agent destination while Tivoli Workload Scheduler for z/OS is running, makes your changes immediately effective. This command manages up to a total of 100 new destinations, regardless if you add them at once or at different times. For detailed information, see *Scheduling End-to-end with z-centric Capabilities.*
Modifying the scheduler

RFRUSER
If you modify the USRREC statement while Tivoli Workload Scheduler for z/OS is running, makes your changes immediately effective. For detailed information, see Scheduling End-to-end with z-centric Capabilities.

SKIPINC(member name)
Replaces the currently used list of INCLUDEs that are to be left at the beginning of a JCL by the JCL tailoring process of Restart and Cleanup (see the RCLOPTS SKIPINCLUDE section in the Customization and Tuning guide for details). member name can be:

• The name of a member inside the PDS parameter library
• Blank or absent

When a member name is specified, the current SKIPINCLUDE list is deleted and a new SKIPINCLUDE list is built by applying the syntax described for the RCLSKIP statement to the specified member (see the Customization and Tuning guide for reference.) When no member name is specified, the current SKIPINCLUDE list is removed.

STATIM=nn
Sets the new value of the STATIM keyword of the JTOPTS statement. Allowed values are 0 to 99.

STATUS
Returns a message on the system log with the status of Tivoli Workload Scheduler for z/OS subsystem. The status can be one of the following:

FULLY_OPERATIONAL
Everything is active and is working properly.

PARTIALLY_OPERATIONAL
The scheduler subsystem has limited functionality. For example, if a controller ER is stopped, the controller can still schedule jobs but cannot receive their statuses.

NOT_OPERATIONAL
The major subsystem functionality is not available. For example, a controller is not able to execute a plan or to submit a job.

STATUS, DD=ddname
Checks for the status of Tivoli Workload Scheduler for z/OS data set associated with the specified ddname. ddname can be a specific DD name, such as EQQWSDS, EQQCP1DS, or EQQLTDS, or it can assume the value ALL, CP, DB, LTP, or JTL. It returns the return code of the last I/O operation performed on that ddname. The status of the data set can be one of the following:

NORMAL
WARNING
SEVERE
CRITICAL
UNKNOWN

STATUS, {OP_COMP | OP_ERR}, "destination name"
Returns the number of completed operations (OP_COMP) or the number of ended-in-error operations (OP_ERR) for the specified tracker (destination name).
Modifying the scheduler

Notes:
1. If more than one workstation is defined for tracker tracker name, the number of completed or in-error operations is the sum of the operations on all the workstations defined on that tracker.
2. destination name is the destination name of a tracker, as specified in the ROUTOPTS keyword in the initialization statements.

STATUS, SUBTASK
Lists all subtasks with their statuses. The status can be ACTIVE or INACTIVE.

TAKEOVER
Orders a standby controller to take over the functions of the controller. This command is valid only when both systems are part of the same XCF group, and no controller is active. You can use this command only for Tivoli Workload Scheduler for z/OS address spaces where OPCHOST(STANDBY) is specified on the OPCOPTS initialization statement.

Note: Takeover can occur automatically if you have specified the TAKEOVER keyword on the XCOPTES initialization statement of a standby system. Refer to Customization and Tuning for more information.

TRYNOERR(member name)
Start a trial processing of the NOERROR statements contained in member name. The controller issues all the normal processing messages in EQQMLOG. At the end of the trial processing, the controller issues message EQQN099I, leaving unchanged the NOERROR table. member name is a member of the EQQPARM library.

VSTRC=START
Starts a trace on the message log of all VSAM I/O requests. In a busy scheduler system, you will need a large message-log data set, and the trace will affect the performance of Tivoli Workload Scheduler for z/OS system.

VSTRC=STOP
Stops a VSAM I/O request trace on the message log.

WSASTA=ON Activates the STATMSG(WSATASK) message.

WSASTA=OFF Deactivates the STATMSG(WSATASK) message.

After the STOP command is entered, the MODIFY command no longer functions, and issues the following message on SYSLOG:

IEE324I MODIFY REJECTED - TASK BUSY

Modifying the data store

Use the MODIFY command to supply information to Tivoli Workload Scheduler for z/OS data store after it has started. The syntax of the MODIFY command is:

/F procname,modifyoption

where:
Modifying the data store

procname

The Tivoli Workload Scheduler for z/OS JCL procedure name.

modifyoption

Can be one of the following:

S=taskname

Start the specified data store subtask.

P=taskname

Stop the specified data store subtask.

taskname can be one of the following:

ARRD Reader task.

ARCU Cleanup task.

ARCM Communication.

ARDYWR Display number of active writers.

ARDYTW Display WINTERVAL value.

ARDYNY Display MAXSTOL value.

ARDYNS Display MAXSYSL value.

ARDYTU Display CINTERVAL value.

ARDYNs Display MAXSYSL value.

ARDYPM Display all initialization parameters values.

ARSTKW O=[owner],K=[keyword]

Display the statistical data for a specific owner or single keyword.

The following describes the keyword and owner combinations:

Table 42. Keyword and owner combinations

<table>
<thead>
<tr>
<th>Owner</th>
<th>Keyword</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>JESQUEUE</td>
<td>COUNTJOB</td>
<td>Number of jobs still in the JES Queue control block</td>
</tr>
<tr>
<td>JESQUEUE</td>
<td>JOBINSQ</td>
<td>Number of jobs inserted in the JES Queue control block</td>
</tr>
<tr>
<td>JESQUEUE</td>
<td>DSIDINSQ</td>
<td>Number of DS IDs (sysout) inserted in the JES Queue control block</td>
</tr>
<tr>
<td>JESQUEUE</td>
<td>JOBRQUEU</td>
<td>Number of jobs re-queued: deleted from the JES Queue control block after the store</td>
</tr>
<tr>
<td>JESQUEUE</td>
<td>JOBDISCA</td>
<td>Number of jobs discarded: deleted from the JES Queue control block because of an error during the store in the database</td>
</tr>
<tr>
<td>DATAFILEnnnn</td>
<td>CNTDPAGE</td>
<td>Number of data pages for the data file specified</td>
</tr>
<tr>
<td>READER</td>
<td>JOBRQSDB</td>
<td>Number of jobs requested directly to the database</td>
</tr>
<tr>
<td>READER</td>
<td>JOBRQJES</td>
<td>Number of jobs requested to JES Queue because they are not stored in the database yet</td>
</tr>
<tr>
<td>COMMUNICATION</td>
<td>INPUTMSG</td>
<td>Number of input messages received</td>
</tr>
<tr>
<td>COMMUNICATION</td>
<td>OUTPTMSG</td>
<td>Number of output messages sent</td>
</tr>
</tbody>
</table>

ARSTGN Display all the statistics for the data store.

ARMDWR Modify number of active writers.
Modifying the data store

ARMDTW=n  Modify WINTERVAL value (seconds).
ARMDNY=n  Modify MAXSTOL value (number of lines).
ARMDNS=n  Modify MAXSYSL value (number of lines).
ARMDTU=n  Modify CINTERVAL value (seconds).
ARMDNS=n  Modify MAXSYSL value (number of lines).
ARDGCM=on/off
   Activate or deactivate Communication task traces.
ARDGWR=on/off
   Activate or deactivate Writer task traces.
ARDGRD=on/off
   Activate or deactivate Reader task traces.
ARDGJQ=on/off
   Activate or deactivate JES Queue task traces.
ARDGDB=on/off
   Activate or deactivate Database task traces.
Appendix E. Status, error, and reason codes

This appendix lists status, error, and operation reason codes.

The scheduler assigns a status code to every occurrence and every operation in the current plan. An error code is also assigned for any operation that ends in error. You can see the status of operations by using the WORKSTATION COMMUNICATION panel or the QUERY CURRENT PLAN panel.

The codes assigned by Tivoli Workload Scheduler for z/OS are not just for documentation purposes. They report the real status of the operation and are used by several scheduler functions to make important decisions about the running of the operation.

**Occurrence status codes**

The following is a list of the occurrence status codes:

- **C** Complete
- **D** Deleted
- **E** An operation in the occurrence has ended-in-error
- **P** A pending predecessor exists for the occurrence
- **S** Started
- **U** Undecided (the status is not known)
- **W** No operations in the occurrence have started

**Operation status codes**

When Tivoli Workload Scheduler for z/OS displays the status of an operation, it uses the format *xy*, where *x* is the status code and *y*, if present, is the extended status code. For quick recognition of the operation status, each status has a default color. You can customize the status color in the ISPF options (see “Setting options” on page 36).

The following is a list of the operation status codes and the corresponding default colors:

- **A** Arriving; the operation is ready for processing; no predecessors were defined. The default status color is green.
- **C** Complete. The default status color is blue.
- **D** Deleted. The default status color is blue.
- **E** The operation has ended-in-error. The default status color is red.
- **I** The operation is interrupted. The default status color is blue.
- **R** Ready for processing; all predecessors are complete. The default status color is green.
- **S** Started. The default status color is turquoise.
Operation status codes

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>Undecided; the operation status is not known. The default status color is red.</td>
</tr>
<tr>
<td>W</td>
<td>The operation is waiting for a predecessor to complete. The default status color is yellow.</td>
</tr>
<tr>
<td>X</td>
<td>The operation is suppressed by condition. The default status color is blue.</td>
</tr>
<tr>
<td>*</td>
<td>Ready; at least one predecessor is defined on a nonreporting workstation; all predecessors are complete. The default status color is green.</td>
</tr>
</tbody>
</table>

Extended status codes

Together with the normal status codes, Tivoli Workload Scheduler for z/OS maintains extended status codes that provide additional information about the status of operations. The extended status code is not always present.

The following is a list of the extended status codes:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>The scheduler is sending an HTTP or HTTPS request to bind the shadow job to a real instance in the remote plan.</td>
</tr>
<tr>
<td>4</td>
<td>The scheduler is waiting to receive the result of the HTTP or HTTPS request that was sent to bind the shadow job to a real instance in the remote plan.</td>
</tr>
<tr>
<td>5</td>
<td>The bind between the shadow job and a real instance in the remote plan was established.</td>
</tr>
<tr>
<td>A</td>
<td>The job is waiting for a manual cleanup action to be initiated or discarded by a panel user (the cleanup type is manual).</td>
</tr>
<tr>
<td>B</td>
<td>The job is waiting for a cleanup action to be started (the cleanup type is automatic or immediate).</td>
</tr>
<tr>
<td>C</td>
<td>A restart and cleanup process is in progress (data set cleanup or step restart, or both). The job is waiting for the process to be completed.</td>
</tr>
<tr>
<td>D</td>
<td>Close down is in progress.</td>
</tr>
<tr>
<td>E</td>
<td>An error occurred during job submission or release.</td>
</tr>
<tr>
<td>G</td>
<td>The operation is running on a WAIT workstation (it is a dummy operation waiting for the delay period to elapse).</td>
</tr>
<tr>
<td>H</td>
<td>A panel user has used the HOLD command on the operation.</td>
</tr>
<tr>
<td>L</td>
<td>The operation is a late time-dependent operation with the suppress-if-late attribute.</td>
</tr>
<tr>
<td>M</td>
<td>The status of the operation has been manually set.</td>
</tr>
<tr>
<td>N</td>
<td>A panel user has used the NOP command on the operation.</td>
</tr>
<tr>
<td>O</td>
<td>Workstation is offline.</td>
</tr>
<tr>
<td>Q</td>
<td>For z/OS jobs the job has been added to the JES job queue. For fault-tolerant workstations, it is waiting for submission.</td>
</tr>
<tr>
<td>R</td>
<td>The operation has ended in error but was automatically reset (the completion code is defined in the installation options to be automatically reset).</td>
</tr>
<tr>
<td>S</td>
<td>The job or started task is executing.</td>
</tr>
<tr>
<td>T</td>
<td>Waiting until a particular time.</td>
</tr>
</tbody>
</table>
Submitted is in progress.

The limit value of this fault-tolerant workstation was reached.

Waiting for scheduling environment.

Waiting for resource.

The job ended with an error code matching a NOERROR entry.

For operations on computer workstations, a blank extended status has a particular meaning for the following statuses:

**Arrived (A) or Ready (*) or R**

The scheduler is in the process of submitting this job. The scheduler is waiting for the availability of a parallel server or a critical resource, or the operation is not to be submitted automatically.

**Started (S)**

The job has been successfully submitted but has not yet been reported as added to the JES job queue.

---

**Error codes**

The scheduler assigns error codes to certain operations and to job and started task steps. These codes are used by the automatic job recovery function to decide a recovery action.

**CAN**

The job or started task was canceled by the operator or by a TSO user before execution. This code is also possible if the job-termination event (type 3P) is missing.

**CCUN**

The completion code is unknown. The job or started task has ended, but no completion code is available. This code is also possible if the job-end event (type 3J) is missing.

Check the job log and SYSLOG.

**CLNA**

A failure occurred when Tivoli Workload Scheduler for z/OS attempted to complete the JCL tailoring during the restart and cleanup process.

**CLNC**

A failure occurred when Tivoli Workload Scheduler for z/OS attempted to execute the data set cleanup during the restart and cleanup process.

**CLNO**

A failure occurred when Tivoli Workload Scheduler for z/OS attempted to retrieve the historical job log data during the restart and cleanup process. *mmmm* Step return code. *S xxx* System abend code. *U xxx* User abend code in hexadecimal notation. For example, user abend 2750 is represented in Tivoli Workload Scheduler for z/OS as UABE. *xxxx* User-defined error code.

**CLNP**

A failure occurred in the EQQCLEAN step, during the run of a restarted job.

**JCCE**

An error during JCC (job completion checker) processing prevented the JCC from determining an error code for the operation.

**FBND**

The request to bind the shadow job with a real instance of the remote plan failed.
**Error codes**

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>JCL</td>
<td>A JCL error was recognized after the job or started task began to execute, or a JCL error was recognized after syntax checking in the internal reader.</td>
</tr>
<tr>
<td>JCLI</td>
<td>A JCL error occurred immediately; that is, the error was detected before the job or started task began. This code is also possible when both the job-start event (type 2) and the job-end event (type 3J) are missing. On Tivoli Workload Scheduler for z/OS agent workstations, this code is returned when a parsing error occurs in the JOBREC statement.</td>
</tr>
<tr>
<td>LOOP</td>
<td>The workstation analyzer task has tried to start the same operation repeatedly and message EQQW534E has been stored in the controller MLOG to signal a loop. To stop the loop, the operation was set to error with error code LOOP.</td>
</tr>
<tr>
<td>MCP</td>
<td>The operation was manually set to error in the MCP panel.</td>
</tr>
<tr>
<td>OAUT</td>
<td>While running the System Automation command specified with the operation, the System Automation exit EQQUXSAZ issued a return code different from 0. The operation status is set to E. Check for System Automation messages in the log specified in the AUTOMATIONMSG parameter of the OPCOPTS statement.</td>
</tr>
<tr>
<td>OFxx</td>
<td>The system that the operation is defined on has gone offline. The WSOFFLINE parameter on the JTOPTS initialization statement specifies that started operations should be marked as ended-in-error. xx is the status and extended status of the failing operation. Operations that were running (status SS) have a step-code error status of OFFL.</td>
</tr>
<tr>
<td>OJCV</td>
<td>An error occurred during JCL-variable substitution when the job or started task was submitted, or Tivoli Workload Scheduler for z/OS detected an error in the RECOVER statement during automatic recovery. Browse the JCL for the operation or the EQQMLOG data set to find more information about the failure. This error code can also be issued when an error occurred during variable substitution in a System Automation command text. The operation status could be set to E, based on the SAVARFAIL parameter set in the OPCOPTS initialization statement. To identify the variables not resolved and the type of error occurred, check for the messages related to occurrence variable substitution in EQQMLOG.</td>
</tr>
<tr>
<td>OSEQ</td>
<td>A job or started task began to execute before all its predecessors have completed. This can occur only if the job was not submitted by Tivoli Workload Scheduler for z/OS and if either HOLDJOB(NO) or HOLDJOB(USER) is specified for Tivoli Workload Scheduler for z/OS event writer options. For fault-tolerant workstations, the OSEQ code can indicate that a dependency on another operation or a special resource was added after the job started, but before the event reached the controller. Refer to Customization and Tuning.</td>
</tr>
<tr>
<td>OSUB</td>
<td>A failure occurred when Tivoli Workload Scheduler for z/OS attempted to submit a job or start a started task. In the case of a started task, it could be that the started task is a subsystem that is not started by JES, or Tivoli Workload Scheduler for z/OS</td>
</tr>
</tbody>
</table>
Error codes

subsystem EQQSTC ddname is not allocated to a JES-defined procedure library. The operation should be marked as ended-in-error.

For jobs running on Tivoli Workload Scheduler for z/OS agent workstations and z/OS shadow jobs, OSUB indicates that the job submission failed.

OSUF A failure occurred when Tivoli Workload Scheduler for z/OS attempted to retrieve the JCL for a job or started task. This code is set if the SUBFAILACTION keyword of the JTOPTS initialization statement specifies that the operation should be marked as ended-in-error. This code is also caused if you have JOBCHECK(SAME) and the job name in the application description does not match the one on the job card. Another reason for this code is a job is missing JCL that was packed by ISPF in EQQJBLIB.

In end-to-end scheduling with fault tolerance capabilities, this code indicates that an error occurred while the scheduler was queuing the submission event for the job. For jobs with centralized scripts it might also show that Tivoli Workload Scheduler for z/OS could not download the script to the distributed agent.

OSUP A time operation is late, and the SUPPRESSACTION parameter of the JTOPTS initialization statement specified that the operation should be marked as ended-in-error. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed. OSxx The system on which the operation is defined has failed.

OSxx The system on which the operation is defined has failed. The WSFAILURE parameter on the JTOPTS initialization statement specifies that started operations should be marked as ended-in-error. xx is the status and extended status of the failing operation. Operations that were running (status SS) have a step-code error of OSYS.

PCAN A print operation was canceled by the operator.

nnnn Step return code.

SERC An operation submitted in a Restart and Clean up path (via dialogue but also automatically, for example, when clean up was set to AUTOMATIC) was not submitted because the required scheduling environment was not available..

SEUN An operation required a scheduling environment that is unknown to WLM; for this reason it was not submitted.

SHPF The request to bind the shadow job is rejected.

S xxx System abend code.

U xxx User abend code in hexadecimal notation. For example user abend 2750 is represented in Tivoli Workload Scheduler for z/OS as UABE.

xxxx User-defined error code.
Job log retrieval status codes

When the job log retrieval function is used, Tivoli Workload Scheduler for z/OS maintains status information to report on the retrieval of the log. The following status codes are possible:

- **C** Completed; the controller has received the log.
- **E** Error. There was an error retrieving the log.
- **I** Initiated. The controller has sent a retrieval request to the tracker, but the tracker has not yet processed the request.
- **S** Started. The controller has sent a retrieval request to the tracker, and the tracker has started to retrieve the log.
- **blank** The controller has not sent any retrieval request to the tracker.

Operation reason codes

If your ready list layout includes the RSNC field, you can see these operation reason codes. Note that the codes are listed in hierarchical order. For example, if job submission failed, and job submission is deactivated, code D is obtained, not code F.

- **A** Automatic reset error condition
- **C** Workstation is closed
- **D** Job submission deactivated
- **F** Job submission failed
- **H** Closedown in progress
- **J** No automatic job submission
- **L** Job is late
- **O** Work station is offline
- **P** All parallel servers in use
- **S** Waiting for special resource
- **T** Start time not reached
- **U** Work station is unlinked
- **W** Waiting for scheduling environment
- **1** Not enough free WS resource 1
- **2** Not enough free WS resource 2
Appendix F. Fields displayed in ready and error lists

This appendix contains a list of fields that you can display in the ready list and the error handling list. In this table, the Length column is the maximum length (in bytes) of the data in the field.

Table 43. Fields available for display in ready and error lists

<table>
<thead>
<tr>
<th>Column title</th>
<th>Length</th>
<th>Description of column contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>PR#</td>
<td>04</td>
<td>Number of predecessor operations</td>
</tr>
<tr>
<td>PS#</td>
<td>03</td>
<td>Number of parallel servers required by the operation</td>
</tr>
<tr>
<td>UPR#</td>
<td>04</td>
<td>Number of uncompleted predecessor operations</td>
</tr>
<tr>
<td>RPR#</td>
<td>04</td>
<td>Number of 1st WS resources required by the operation</td>
</tr>
<tr>
<td>R2#</td>
<td>03</td>
<td>Number of 2nd WS resources required by the operation</td>
</tr>
<tr>
<td>SR#</td>
<td>03</td>
<td>Special resources referenced by the operation</td>
</tr>
<tr>
<td>SU#</td>
<td>04</td>
<td>Number of successor operations</td>
</tr>
<tr>
<td>Arrived</td>
<td>08</td>
<td>Operation arrival date, actual if arrived</td>
</tr>
<tr>
<td>Ard</td>
<td>03</td>
<td>Operation arrival day, actual if arrived</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Operation arrival time, actual if arrived</td>
</tr>
<tr>
<td>Application</td>
<td>16</td>
<td>Application ID</td>
</tr>
<tr>
<td>Act dur</td>
<td>07</td>
<td>Actual duration of the operation</td>
</tr>
<tr>
<td>A</td>
<td>01</td>
<td>Automatic error-completion indicator, Y or N</td>
</tr>
<tr>
<td>Ended</td>
<td>08</td>
<td>End date of the operation, or blank</td>
</tr>
<tr>
<td>End</td>
<td>03</td>
<td>End day of the operation, or blank</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>End time of the operation, or blank</td>
</tr>
<tr>
<td>H</td>
<td>01</td>
<td>Automatic hold/release indicator, Y or N</td>
</tr>
<tr>
<td>Owner</td>
<td>16</td>
<td>Application owner ID</td>
</tr>
<tr>
<td>Started</td>
<td>08</td>
<td>Actual start date of the operation</td>
</tr>
<tr>
<td>Std</td>
<td>03</td>
<td>Actual start day of the operation</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Actual start time of the operation</td>
</tr>
<tr>
<td>S</td>
<td>01</td>
<td>Automatic job submission, Y or N</td>
</tr>
<tr>
<td>Application text</td>
<td>24</td>
<td>Verbal description of the application</td>
</tr>
<tr>
<td>JLOG</td>
<td>01</td>
<td>Status of job log retrieval</td>
</tr>
<tr>
<td>CLN_Ty</td>
<td>01</td>
<td>Cleanup type</td>
</tr>
<tr>
<td>CLN_Re</td>
<td>01</td>
<td>Cleanup result</td>
</tr>
<tr>
<td>RC_St</td>
<td>01</td>
<td>Internal restart and cleanup status, for debugging only</td>
</tr>
<tr>
<td>C</td>
<td>01</td>
<td>Critical path, F-1st (Y=on path, N=not)</td>
</tr>
<tr>
<td>S</td>
<td>01</td>
<td>Current status of the operation</td>
</tr>
<tr>
<td>T</td>
<td>01</td>
<td>Dependency type, P=predecessor, S=successor</td>
</tr>
<tr>
<td>W</td>
<td>01</td>
<td>Deadline WTO, Y or N</td>
</tr>
<tr>
<td>E dur</td>
<td>05</td>
<td>Estimated duration of the operation</td>
</tr>
<tr>
<td>Errc</td>
<td>04</td>
<td>Error code of the operation, otherwise blank</td>
</tr>
<tr>
<td>FTW Return Code</td>
<td>11</td>
<td>The return code for jobs on fault-tolerant workstations</td>
</tr>
<tr>
<td>Destination</td>
<td>08</td>
<td>Execution destination ID</td>
</tr>
<tr>
<td>Form no.</td>
<td>08</td>
<td>Form number or blank</td>
</tr>
<tr>
<td>Authgrp</td>
<td>08</td>
<td>Authority group</td>
</tr>
<tr>
<td>la date</td>
<td>08</td>
<td>Application input arrival date (after MCP)</td>
</tr>
<tr>
<td>la day</td>
<td>03</td>
<td>Application IA day, (&lt;) preceding month, (&gt;) next</td>
</tr>
<tr>
<td>la time</td>
<td>05</td>
<td>Application input arrival time (after MCP)</td>
</tr>
<tr>
<td>I start</td>
<td>08</td>
<td>Intermediate start date after interrupt</td>
</tr>
<tr>
<td>ISt</td>
<td>03</td>
<td>Intermediate start day after interrupt</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Intermediate start time after interrupt</td>
</tr>
<tr>
<td>Jobname</td>
<td>08</td>
<td>Job name</td>
</tr>
</tbody>
</table>
### Fields displayed in ready and error lists

Table 43. Fields available for display in ready and error lists (continued)

<table>
<thead>
<tr>
<th>Column title</th>
<th>Length</th>
<th>Description of column contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>01</td>
<td>Job, SYSOUT class, or blank</td>
</tr>
<tr>
<td>Job id</td>
<td>08</td>
<td>JES job ID</td>
</tr>
<tr>
<td>S</td>
<td>01</td>
<td>Job status (I) init, (H) held, (Q) released</td>
</tr>
<tr>
<td>L</td>
<td>01</td>
<td>Latest out time passed</td>
</tr>
<tr>
<td>Last out</td>
<td>08</td>
<td>Latest out date of the operation</td>
</tr>
<tr>
<td>Lo</td>
<td>03</td>
<td>Latest out day of the operation</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Latest out time of the operation</td>
</tr>
<tr>
<td>Last upd</td>
<td>08</td>
<td>Time stamp of last MCP update in the format MMDDHHmm (month, day, hour and minutes)</td>
</tr>
<tr>
<td>H</td>
<td>01</td>
<td>Operation manually HELD by user</td>
</tr>
<tr>
<td>N</td>
<td>01</td>
<td>Operation NOPed by user</td>
</tr>
<tr>
<td>no.</td>
<td>03</td>
<td>Operation number</td>
</tr>
<tr>
<td>Deadline</td>
<td>08</td>
<td>Deadline date of the operation</td>
</tr>
<tr>
<td>Di</td>
<td>03</td>
<td>Deadline day of the operation</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Deadline time of the operation</td>
</tr>
<tr>
<td>Plan ia</td>
<td>08</td>
<td>Planned input arrival date of operation</td>
</tr>
<tr>
<td>Ia</td>
<td>03</td>
<td>Planned input arrival day of operation</td>
</tr>
<tr>
<td>Oi</td>
<td>01</td>
<td>Operator instructions available</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Planned input arrival time of operation</td>
</tr>
<tr>
<td>Plan end</td>
<td>08</td>
<td>Planned end date of the operation</td>
</tr>
<tr>
<td>End</td>
<td>03</td>
<td>Planned end day of the operation</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Planned end time of the operation</td>
</tr>
<tr>
<td>P</td>
<td>01</td>
<td>On preparation workstation, Y or N</td>
</tr>
<tr>
<td>P</td>
<td>01</td>
<td>Priority of the operation</td>
</tr>
<tr>
<td>Pl start</td>
<td>08</td>
<td>Planned start date of the operation</td>
</tr>
<tr>
<td>St.</td>
<td>03</td>
<td>Planned start day of the operation</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Planned start time of the operation</td>
</tr>
<tr>
<td>Transp</td>
<td>06</td>
<td>Transport time, from previous WS to this WS</td>
</tr>
<tr>
<td>Rdr date</td>
<td>08</td>
<td>Date the reader recognized the job card</td>
</tr>
<tr>
<td>Rdr t</td>
<td>05</td>
<td>Time the reader recognized the job card</td>
</tr>
<tr>
<td>L</td>
<td>01</td>
<td>Time job suppressed if late, Y or N</td>
</tr>
<tr>
<td>R</td>
<td>01</td>
<td>Reroutable operation, Y or N</td>
</tr>
<tr>
<td>R</td>
<td>01</td>
<td>Restartable operation, Y or N</td>
</tr>
<tr>
<td>RSNC</td>
<td>01</td>
<td>Reason code</td>
</tr>
<tr>
<td>St</td>
<td>02</td>
<td>Operation status, current and extended status</td>
</tr>
<tr>
<td>T</td>
<td>01</td>
<td>Time job indicator, Y or N</td>
</tr>
<tr>
<td>Operation text</td>
<td>24</td>
<td>Verbal description of the operation</td>
</tr>
<tr>
<td>User data</td>
<td>16</td>
<td>Operation user data</td>
</tr>
<tr>
<td>U</td>
<td>01</td>
<td>Urgency indicator</td>
</tr>
<tr>
<td>RE</td>
<td>01</td>
<td>Operation rerouted, Y or N</td>
</tr>
<tr>
<td>ws</td>
<td>04</td>
<td>Workstation</td>
</tr>
<tr>
<td>T</td>
<td>01</td>
<td>Workstation type</td>
</tr>
<tr>
<td>WTO ws</td>
<td>01</td>
<td>Workstation is a WTO</td>
</tr>
<tr>
<td>Dur.</td>
<td>07</td>
<td>Actual duration, if completed, otherwise estimated</td>
</tr>
<tr>
<td>Inp arr</td>
<td>08</td>
<td>Operation IA date, actual if arrived</td>
</tr>
<tr>
<td>Ia</td>
<td>03</td>
<td>Operation IA day, actual if arrived, otherwise planned</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>Operation IA time, actual if arrived</td>
</tr>
<tr>
<td>X</td>
<td>01</td>
<td>Extended status of the operation</td>
</tr>
<tr>
<td>Ended</td>
<td>08</td>
<td>End date, actual if ended, otherwise planned</td>
</tr>
<tr>
<td>End</td>
<td>03</td>
<td>End day, actual if ended, otherwise planned</td>
</tr>
<tr>
<td>time</td>
<td>05</td>
<td>End time, actual if ended, otherwise planned</td>
</tr>
<tr>
<td>Start</td>
<td>08</td>
<td>Operation start date, actual if started</td>
</tr>
<tr>
<td>St</td>
<td>03</td>
<td>Operation start day, actual if started</td>
</tr>
</tbody>
</table>
## Fields displayed in ready and error lists

Table 43. Fields available for display in ready and error lists (continued)

<table>
<thead>
<tr>
<th>Column title</th>
<th>Length</th>
<th>Description of column contents</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>05</td>
<td>Operation start time, actual if started</td>
</tr>
<tr>
<td>Actual dur</td>
<td>10</td>
<td>Actual duration of the operation in seconds</td>
</tr>
<tr>
<td>Est dur</td>
<td>08</td>
<td>Estimated duration of the operation in seconds</td>
</tr>
<tr>
<td>Duration</td>
<td>10</td>
<td>Actual duration, if completed, otherwise estimated in seconds</td>
</tr>
</tbody>
</table>
Fields displayed in ready and error lists
Appendix G. Defining event rules and invoking EQQLSENT macro

This chapter contains the following sections:

- “Syntax to define event rules”
- “Event rule examples” on page 758
- “Invoking the EQQLSENT macro” on page 760

Syntax to define event rules

This section describes the syntax to edit the EVLIB.XML data set, that is referenced as SYSIN in the EQQTRBSLs skeleton (produced when running EQQJOBS).

While editing the EVLIB.XML data set, consider using the HILITE ON XML or HILITE LOGIC XML editor commands. They require a record length equal to 255, as specified in the EQQPCS01 sample.

The following is a list of all the language elements used for defining an event rule. Table 44 explains the meaning of the notation that follows each language element. n represents an unbounded number.

Table 44. Valid number of occurrences for a language element

<table>
<thead>
<tr>
<th>Notation</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0, 1)</td>
<td>0 indicates that the language element is optional. 1 indicates that if coded, only 1 occurrence is allowed.</td>
</tr>
<tr>
<td>(0, n)</td>
<td>0 indicates that the language element is optional. n indicates that if coded, multiple occurrences are allowed.</td>
</tr>
<tr>
<td>(1, 1)</td>
<td>The first 1 indicates that the language element is required. The second 1 indicates that only 1 occurrence is allowed.</td>
</tr>
<tr>
<td>(1, n)</td>
<td>1 indicates that the language element is required. n indicates that multiple occurrences are allowed.</td>
</tr>
</tbody>
</table>

- `<eventRule name=" " ruleType=" " isDraft=" "> (1, 1)
  - `<description> (0, 1)
  - `<eventCondition eventProvider=" " eventType=" "> (1, 1)
    - `<filteringPredicate> (1,1)
      - `<attributeFilter name=" " operator="eq"> (1, n)
        - `<value> (1, 15)
    - `<action actionProvider=" " actionType=" " responseType=" "> (1,1)
      - `<description> (0, 1)
      - `<parameter name=" "> (0, n)
        - `<value> (0, 1)

Event rule definitions are grouped into event rule sets.

- `<eventRuleSet> (1, 1)
  - `<eventRule> (1, n)

Use the eventRuleSet language element also if you have to enclose a single rule definition.
Arguments

To include comments in the XML file, use the form <!--text-->.

**Arguments**

The keywords describing an event rule are the following XML tags:

*eventRule*

It includes:
- A number of required and optional rule attributes.
- One event condition.
- One action.

The rule attributes are:
- Required attributes:
  - **name**
    
    The name of the event rule. You can specify up to 40 characters, blank and special characters included. It is a label making easier to identify each rule in the XML file. Configuration files do not contain this data.
  - **ruleType**
    
    Always set to filter.
  - **isDraft**
    
    Values can be yes or no. Specify no to activate the corresponding rule definition. Specify yes to deactivate the corresponding rule definition or to check its syntax without activating it. The default is no.

- Optional attributes:
  - **description**
    
    A description of the rule. It can be of up to 120 characters.

*eventCondition*

The event condition is made up by the following attributes:

*eventProvider*

Identifies the event monitoring provider that can capture a type of event. SMF only is currently supported.

*eventType*

Specifies the type of event that is to be monitored. Table 45 lists the valid types. Click on the event types to see their properties.

**Table 45. SMF events**

<table>
<thead>
<tr>
<th>Event type</th>
<th>Event trigger</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReadCompleted</td>
<td>A data set is closed after it was opened in read mode.</td>
</tr>
<tr>
<td>ModificationCompleted</td>
<td>A data set is closed after it was opened in write mode. This event is sent also when you create an empty data set.</td>
</tr>
</tbody>
</table>

*filteringPredicate*

To filter the event conditions that are to be monitored. It is made up by:

*attributeFilter*

The attribute filter is a particular attribute of the event that is to be monitored:
- It is defined by the following elements:
Arguments

name
The attribute, or property name, of the event that is to be monitored. Refer to Table 46 for a list of the supported property names.

operator
Always set to eq (equal).

• It includes a list of:

value
The value on which the operator must be matched. You can specify up to 15 values.

Items in the value list works with the same behavior as the OR operator in the boolean logic.

Items in the attributeFilter list works with the same behavior as the AND operator in the boolean logic.

action
The action that is to be triggered if the event is detected. Event rule definitions with events but no actions are syntactically accepted, although they may have no practical significance. You may save such rules as draft, by specifying isDraft="yes", and add actions later before they are deployed.

• Is defined by the following required attributes:

actionProvider
The name of the action provider that can implement one or more configurable actions. Tracker only is currently supported.

actionType
Specifies the type of action that is to be triggered when a specified event is detected. SpecialResourceEvent only is currently supported.

responseType
Specifies when the action is to run. Always set to onDetection, meaning that the action starts as soon as all the events defined in the rule have been detected.

• Includes the following optional attributes:

description
A description of the action. It can be of up to 120 characters. It is a label making easier to identify each action in the XML file. Configuration files do not contain this data.

• Includes a list of one or more parameters, or property names. Every parameter is defined by:

parameter name=" 
See Table 47 on page 757 for a list of the supported parameters, or property names.

value
See Table 47 on page 757 for a list of possible values or value types.

Table 46 shows the relationship between name and value of attributeFilter.

<table>
<thead>
<tr>
<th>attributeFilter name</th>
<th>Type</th>
<th>Required</th>
<th>Wildcard allowed</th>
<th>Length (min-max)</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FileName</td>
<td>string</td>
<td>☑</td>
<td>☑</td>
<td>1</td>
<td>44</td>
</tr>
<tr>
<td>Destination</td>
<td>string</td>
<td>☑</td>
<td></td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Userid</td>
<td>string</td>
<td>☑</td>
<td></td>
<td>1</td>
<td>*</td>
</tr>
</tbody>
</table>
Table 46. Parameters of ReadCompleted and ModificationCompleted event types (continued)

<table>
<thead>
<tr>
<th>attributeFilter name</th>
<th>Type</th>
<th>Required</th>
<th>Wildcard allowed</th>
<th>Length (min-max)</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jobname</td>
<td>string</td>
<td></td>
<td>✔</td>
<td>1</td>
<td>*</td>
</tr>
</tbody>
</table>

Note: For parameters with wildcard allowed, you can use the following wildcards:

- * To match any sequence of characters.
- ? To match any single character. For example, if you specify AB?, ABC is a match, AB or ABCD are not a match.
- % For backward compatibility, it is supported for the same function as ?.

The following is a detailed description of the parameters:

**FileName**

Specifies the data set name to be monitored for actions on special resources. For details about how the tracker requests to change the resource availability, based on the specified FileName value, refer to “Effects on special resource availability” on page 429.

**Destination**

The destination of the action provider, that is the tracker where the specified data sets reside. Specify $$$$$$$$ to identify a local destination.

**Userid**

Specifies a generic character string to be compared with the SMFxxUID field, which contains the user identification associated with the job, started task, or TSO user that requested the activity against the data set that resulted in the data set close.

Note: The SMF user ID field may contain a blank value. Refer to z/OS System Management Facilities for more information about the SMFxxUID or SMFxxUIF field.

If you need to control SR availability events based on the user ID and the SMF value is blank in your installation, consider using the IEFUJI exit to insert the user ID. You are recommended to specify SETUID=YES on the EQQEXIT macro when you generate the IEFUJI exit: this sets the JMRUSEID field, which SMF then copies to the SMF user ID field.

If you want to update the JMRUSEID field yourself, the user ID is most easily taken from the ACEEUSRI field in the ACEE, pointed to from the ASXB, pointed to from the ASCB. This can be located as follows:

- PSAAOLOD ===> ASCB
- ACSBASXB ===> ASXB
- ASXBSENV ===> ACEE
- ACEEUSRI ===> userid

The DSECTs needed are mapped by these macros:

<table>
<thead>
<tr>
<th>Area</th>
<th>Macro</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSA</td>
<td>IHAPSA</td>
<td>SYS1.MACLIB</td>
</tr>
<tr>
<td>ASCB</td>
<td>IHAASCB</td>
<td>SYS1.MACLIB</td>
</tr>
<tr>
<td>ASXB</td>
<td>IHAASXB</td>
<td>SYS1.MODGEN</td>
</tr>
</tbody>
</table>
Arguments

<table>
<thead>
<tr>
<th>Area</th>
<th>Macro</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACEE</td>
<td>IHAACEE</td>
<td>SYS1.MACLIB</td>
</tr>
</tbody>
</table>

The JMR, mapped by IEFJMR, is already available in the EQQEXIT expansion in IEFUJ1.

Jobname

Specifies a generic character string to be compared with the SMF14JBN, SMF15JBN, or SMF64JMN field, which contains the name of the job, started task or TSO user that requested the activity against the data set that resulted in the data set close.

If the data set is to be processed by FTP, JOBNAME corresponds to the **USERID** under which the dataset is received. That is, the USERID supplied when the remote host opened the FTP session to PUT the dataset, or when a local user (or batch job) opened the FTP session to GET the dataset.

Table 47 shows the parameters for SpecialResourceEvent action type. All of them are optional.

Table 47. Parameters of SpecialResourceEvent action type

<table>
<thead>
<tr>
<th>Property name</th>
<th>Type</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Availability</td>
<td>string</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td>Value can be Yes or No</td>
<td></td>
</tr>
<tr>
<td>LifeSpanAction</td>
<td>string</td>
<td>Reset</td>
</tr>
<tr>
<td></td>
<td>Value can be Yes, No, or Reset</td>
<td></td>
</tr>
<tr>
<td>LifeSpanTime</td>
<td>numeric (1-999999)</td>
<td></td>
</tr>
</tbody>
</table>

Detailed parameter descriptions follow:

**Availability**

Specifies that the special resource, which the action refers to, must be set to available (Yes) or unavailable (No).

**LifeSpanAction**

Specifies the value to which the global availability of the special resource is reset, after the interval of time specified by LifeSpanTime has expired. Allowed values are:

- **Yes**  ― Sets the global availability to Yes
- **No**   ― Sets the global availability to No
- **Reset**  ― Sets the global availability to blank

This keyword is valid only if LifeSpanTime is specified.

**LifeSpanTime**

Specifies the interval of time, in minutes, after which the global availability of the special resource is reset to the value specified by LifeSpanAction.
Event rule examples

The following examples show how to combine language elements and use wild cards.

Basic scenario

It applies to the scenario described in “Business scenario” on page 425.

```xml
<?xml version="1.0"?><eventRuleSet>
  <eventRule name="Monitor_FTP" ruleType="filter" isDraft="no">
    <description>To monitor data sent by FTP</description>
    <eventCondition eventProvider="SMF" eventType="ModificationCompleted">
      <FilteringPredicate>
        <attributeFilter name="FileName" operator="eq">
          <value>TWSDEV.FALSI.TWS</value>
        </attributeFilter>
        <attributeFilter name="Destination" operator="eq">
          <value>destX</value>
        </attributeFilter>
      </filteringPredicate>
    </eventCondition>
    <action actionProvider="Tracker" actionType="SpecialResourceEvent" responseType="onDetection">
      <description>Set TWSDEV.FALSI.TWS special res to available</description>
      <parameter name="Availability">
        <value>Yes</value>
      </parameter>
      <parameter name="LifeSpanAction">
        <value>No</value>
      </parameter>
      <parameter name="LifeSpanTime">
        <value>60</value>
      </parameter>
    </action>
  </eventRule>
</eventRuleSet>
```

Using wild cards

Suppose that:

- TWSDEV.AB.TWS, TWSDEV.AB1.TWS, TWSDEV.OLDAB.TWS, TWSDEV.NEWAB.TWS, TWSDEV.OLDAB1.TWS, and TWSDEV.NEWAB1.TWS are data sets shared among trackers with DEST1, DEST2 and DEST3 destinations.
- TWSDEV.AB.TWS and TWSDEV.AB1.TWS are not defined in the current plan or special resource database.

The controller RESOPTS statement specifies DYNAMICADD(EVENT).

Consider the following XML data:

```xml
<?xml version="1.0"?><eventRuleSet>
  <eventRule name="Monitor_MultipleTriggers1" ruleType="filter" isDraft="no">
    <description>Monitor data set changes at DEST1 or DEST2</description>
    <eventCondition eventProvider="SMF" eventType="ModificationCompleted">
      <FilteringPredicate>
        <attributeFilter name="FileName" operator="eq">
          <value>TWSDEV.AB*.TWS</value>
        </attributeFilter>
        <attributeFilter name="Destination" operator="eq">
          <value>dest1</value>
          <value>dest2</value>
        </attributeFilter>
      </filteringPredicate>
    </eventCondition>
    <action actionProvider="Tracker" actionType="SpecialResourceEvent" responseType="onDetection">
      <description>Set TWSDEV.AB*.TWS special res to available</description>
      <parameter name="Availability">
        <value>Yes</value>
      </parameter>
      <parameter name="LifeSpanAction">
        <value>No</value>
      </parameter>
      <parameter name="LifeSpanTime">
        <value>60</value>
      </parameter>
    </action>
  </eventRule>
</eventRuleSet>
```
Using wild cards

According to the previous event rules:

- At DEST1 or DEST2 destination, when SMF writes records that trace TWSDEV.AB.TWS or TWSDEV.AB1.TWS as closed data sets, the tracker requests the controller to define TWSDEV.AB.TWS or TWSDEV.AB1.TWS as available special resources.

- At DEST3 destination, when SMF writes records that trace TWSDEV.OLDAB.TWS, TWSDEV.NEWAB.TWS, TWSDEV.OLDAB1.TWS, and TWSDEV.NEWAB1.TWS as closed data sets, the tracker requests the controller to update the definition of TWSDEV.OLDAB.TWS, TWSDEV.NEWAB.TWS, TWSDEV.OLDAB1.TWS, and TWSDEV.NEWAB1.TWS special resources, setting them to available.

Setting isDraft to yes

Setting to 'yes' the isDraft attribute in all the rules defined in - Using wild cards on page 758 deactivates the trigger definitions for DEST1, DEST2, or DEST3 destinations.

Selecting EQQEVLIB members to be updated

To update specific EQQEVLIB members, run EQQRXTRG by using an input XML file that references only those Destination values corresponding to the members to be updated. For example, suppose that EQQEVLIB contains DEST1, DEST2, and DEST3 members defined according to "Using wild cards" on page 758. To update the LifeSpanTime value for DEST2 only, use an XML file with the following data:

```xml
<?xml version="1.0"?>
<eventRuleSet>
  <eventRule name="Monitor_MultipleTriggers2" ruleType="filter" isDraft="no">
    <description>Monitor data set changes at DEST3</description>
    <eventCondition eventProvider="SMF" eventType="ModificationCompleted">
      <FilteringPredicate>
        <attributeFilter name="FileName" operator="eq">
          <value>TWSDEV.????AB*.TWS</value>
        </attributeFilter>
        <attributeFilter name="Destination" operator="eq">
          <value>dest3</value>
        </attributeFilter>
      </FilteringPredicate>
    </eventCondition>
  </eventRule>
</eventRuleSet>
```
Selecting EQQEVLIB members to be updated

```xml
<attributeFilter name="FileName" operator="eq">
  <value>TWSDEV.AB*.TWS</value>
</attributeFilter>
<attributeFilter name="Destination" operator="eq">
  <value>dest2</value>
</attributeFilter>
</filteringPredicate>
</eventCondition>
<action actionProvider="Tracker" actionType="SpecialResourceEvent"
  responseType="onDetection"
  <parameter name="Availability">
    <value>Yes</value>
  </parameter>
  <parameter name="LifeSpanAction">
    <value>No</value>
  </parameter>
  <parameter name="LifeSpanTime">
    <value>30</value>
  </parameter>
</action>
</eventRule>
</eventRuleSet>

A rule definition specifying isDraft="yes" produces members containing no data set list.

Invoking the EQQLSENT macro

The following procedure is supported for backward compatibility only. To exploit the current support for data set triggering, refer to Chapter 21, “Running event-driven workload automation,” on page 425.

When the data set triggering function is used, you specify the data sets for which you want events generated by building the data set selection table EQQDSLST. The EQQDSLST is created by invoking the EQQLSENT macro. The following sections describe how you invoke the EQQLSENT macro. This appendix contains General-use Programming Interface and Associated Guidance Information.

Note: The current support for data set triggering is based on the EQQEVLIST configuration file. If EQQJCLIB contains both EQQEVLIST and EQQDSLST, the resulting triggering selection table is the union of EQQEVLIST and EQQDSLST. In this case, EQQEVLIST data is processed first. If EQQJCLIB contains only EQQDSLST, the tracker loads it as triggering selection table.

Invoking EQQLSENT to create EQQDSLST

The EQQLSENT macro is used to create entries in the data set triggering selection table. The selection table is loaded into ECSA when the Tivoli Workload Scheduler for z/OS event writer is started.

The sample EQQLSJCL in the SEQQSAMP library can be used to invoke the EQQLSENT macro.
Macro invocation syntax for EQQLSENT

Macro invocation syntax for EQQLSENT

Purpose: EQQLSENT produces an entry in the data set triggering selection table, EQQDSLST. EQQDSLST is used in SMF exit IEFU83 by the data set triggering function to decide which SMF records to process. When an SMF 14, 15, or 64 record matches a condition in EQQDSLST, a special resource availability event is created and broadcast to all Tivoli Workload Scheduler for z/OS subsystems defined on the system where the SMF record was created.

Syntax:

STRING= string | LASTENTRY
POS= numeric position
USERID= user ID filter criteria
JOBNAME= jobname filter criteria
AINDIC=[Y|N]
LFFACT=[Y|N | R]
LIFTIM=interval

Parameters:

STRING=string | LASTENTRY
   Required keyword specifying the character string to be searched for. The string can be 1 to 44 characters long. To identify the fully-qualified last level of a data set name, add a space as the last character and enclose the string in single quotes. Consider this example. You have two data sets:

   DSN.NAME.AB
   DSN.NAME.ABC

   Specify STRING=DSN.NAME.AB, POS=1 if you want SR availability events created for both data sets. Specify STRING='DSN.NAME.AB ', POS=1 if you want events created only for the first data set.

   When EQQLSENT is invoked with STRING=LASTENTRY it generates an end of table indicator. After having invoked EQQLSENT with keyword parameters STRING and POS a number of times, EQQLSENT must be invoked one last time with STRING=LASTENTRY in order to complete the table.

   To create an empty EQQDSLST, just invoke EQQLSENT once, with STRING=LASTENTRY. When an empty list is used by IEFU83, no SR events are created.

POS=numeric position
   A required keyword specifying the numeric position where the string begins.

USERID=string
   Optional keyword specifying a generic character string to be compared with the SMFxUID field, which contains the user identification associated with the job, started task, or TSO user that requested the activity against the data set that resulted in the data set close. The string can be 1 to 8 characters long.

Note: The SMF user ID field may contain a blank value. Refer to z/OS System Management Facilities for more information about the SMFxUID or SMFxUIF field.

If you need to control SR availability events based on the user ID and the SMF value is blank in your installation, consider using the IEFUJI exit to insert the
Macro invocation syntax for \texttt{EQQLSENT}

user ID. You are recommended to specify \texttt{SETUID=YES} on the \texttt{EOQEXIT} macro when you generate the \texttt{IEFUJI} exit: this sets the JMRUSEID field, which SMF then copies to the SMF user ID field.

If you want to update the JMRUSEID field yourself, the user ID is most easily taken from the ACEEUSRI field in the ACEE, pointed to from the ASXB, pointed to from the ASCB. This can be located as follows: \texttt{PSAAOLD \implies ASCB \implies ASCBASXB \implies ASXB \implies ASXBSENV \implies ACEE \implies ACEEUSRI \implies userid}

The DSECTs needed are mapped by these macros:

<table>
<thead>
<tr>
<th>Area</th>
<th>Macro</th>
<th>Library</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSA</td>
<td>IHAPSA</td>
<td>SYS1.MACLIB</td>
</tr>
<tr>
<td>ASCB</td>
<td>IHAASCB</td>
<td>SYS1.MACLIB</td>
</tr>
<tr>
<td>ASXB</td>
<td>IHAASXB</td>
<td>SYS1.MODGEN</td>
</tr>
<tr>
<td>ACEE</td>
<td>IHAACEE</td>
<td>SYS1.MACLIB</td>
</tr>
</tbody>
</table>

The JMR, mapped by \texttt{IEFJMR}, is already available in the \texttt{EOQEXIT} expansion in \texttt{IEFUJI}.

\texttt{JOBNAME=string}

Optional keyword specifying a generic character string to be compared with the SMF14JBN, SMF15JBN, or SMF64JMN field, which contains the name of the job, started task or TSO user that requested the activity against the data set that resulted in the data set close. The string can be 1 to 8 characters long.

If the data set is to be processed by FTP, \texttt{JOBNAME} corresponds to the **USERID** under which the dataset is received. That is, the USERID supplied when the remote host opened the FTP session to PUT the dataset, or when a local user (or batch job) opened the FTP session to GET the dataset.

\texttt{AINDIC=\{Y\mid N\}}

Optional keyword specifying that the special resource is available (Y) or unavailable (N). The default is that the resource available.

\texttt{LIFACT=\{Y\mid N\mid R\}}

Optional keyword specifying the value to which the global availability of the special resource is reset, after the interval of time specified by \texttt{LIFTIM} has expired. Allowed values are:

\begin{itemize}
  \item \texttt{Y} \quad \text{Sets the global availability to Yes}
  \item \texttt{N} \quad \text{Sets the global availability to No}
  \item \texttt{R} \quad \text{Sets the global availability to blank}
\end{itemize}

This keyword is valid only if \texttt{LIFTIM} is specified. The default is \texttt{R}.

\texttt{LIFTIM=interval}

Optional keyword specifying the interval of time, in minutes, after which the global availability of the special resource is reset to the value specified by \texttt{LIFACT}. The allowed range is from 1 to 999999.

Notes:

1. The output from assembling the \texttt{EQQLSENT} macro must be placed in the \texttt{EOQDSLST} member in the data set referenced by the \texttt{ddname} \texttt{EOQICLIB}.
Macro invocation syntax for EQQLSENT

2. Generation Data Group data sets are specified by the group name. For example, when a GDG data set with the name 'DSN.OPCSUB.S.GDG.G0001V00' is closed the special resource event contains resource name 'DSN.OPCSUB.S.GDG'.

3. For a partitioned data set, the member name is not part of the resource name in the SR event.

4. For VSAM data sets the resource name in the SR event is the cluster name (without the DATA or INDEX suffix).

Example:

```plaintext
EQQLSENT STRING=SYS1.MAN,POS=1
EQQLSENT STRING='TEST.DSCLOSE ',POS=1,USERID=SYSOP
EQQLSENT STRING=CP2,POS=12
EQQLSENT STRING=EQQDATA.EXCL,POS=5
EQQLSENT STRING='DSN.OPCSUB.S.GDG ',POS=1
EQQLSENT STRING=LASTENTRY
END
```

In this example, SMF records with:
- A data set name beginning with SYS1.MAN, or
- Data set name TEST.DSCLOSE and user ID SYSOP
- Records with CP2 in position 12, such as DSN.OPCSUB.CP2, or
- Records that have EQQDATA.EXCL starting in position 5
- The root of a GDG dataset name

will cause SR availability events to be generated.

Return codes: The following return code can be generated at assembly time:

12 Input invalid, check error messages.

Messages: The following messages can be generated at assembly time:
- KEYWORD STRING IS REQUIRED
- KEYWORD POS IS REQUIRED
- POSITION MUST BE BETWEEN 1 AND 43
- NULL NAME NOT VALID
- NAME (STRING) GREATER THAN 44 CHARACTERS
- POSITION INVALID FOR NAME (STRING)
- USERID STRING NOT VALID
- JOBNAME STRING NOT VALID
- AINDIC MUST BE EITHER Y OR N
- POSITION NOT VALID FOR NAME (STRING)
- LIFACT MUST BE Y, N, OR R
- LIFTIM LENGTH NOT VALID
- LIFTIM VALUE NOT VALID
- LIFTIM VALUE 0 NOT ALLOWED
Return codes
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 publication 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 publication. The furnishing of this publication does not give you any license to these patents. You can send license inquiries, in writing, to:

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

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

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

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

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

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

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

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

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

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.
Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows 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 product and service names might be trademarks or service marks of others.
Index

Special characters

? (question mark) variables 448
* (asterisk) global search character 485
* (asterisk) operation status code 743
* (asterisk) use in generic search arguments 46
% (percent) global search character 485
% (percent) use in generic search arguments 46
% (percent) variables 447
& (ampersand) variables 446
= (equals) ISPF quick return command 42

A
A extended status code 744
A operation status code 743
abend codes, grouping together 369
accessibility xxiii
ACTION keyword 203, 205, 213, 219, 220, 222, 229, 237
activating
automatic recovery 369
event-triggered tracking (ETT) 291, 433
JCL variable substitution 444
job submission 289
active, application status 122
actual quantity 610
AD (application description) database 190
AD NOT FOUND ON AD FILE message 261
AD/OI checks, setting options 40
ADCNC, batch-loader control statement 203
ADCNs, batch-loader control statement 205
ADDEP, batch-loader control statement 208
adding
dependencies 555
operations 569
adding an application to the current plan panel 551
adding an occurrence group to the CP panel 559
adding applications to the current plan panel 550
ADGROUPID keyword 234
ADID keyword 234, 240
adjusted quantity 610
ADOP, batch-loader control statement 211
ADOPEXTN, batch-loader control statement 219
ADOPPWTO keyword 214
ADOPEXAI, batch-loader control statement 220
ADOPWLMCLASS, keyword 214
ADRE, batch-loader control statement 222
ADRULE, batch-loader control statement 224
ADRUNE, batch-loader control statement 228
ADSR batch-loader control statement 231
ADSTART, batch-loader control statement 233
ADSTAT keyword 234
ADTYPE keyword 234
ADUSF, batch-loader control statement 237
ADVALFROM keyword 234
ADVALFROMCHG keyword 242
AEC keyword 214
alert when deadline passes 155
all dependencies of an operation panel 538
all open time intervals panel 76
all workstations closed 707, 708
all-days cyclic periods 106
allocated special resources 613
allocation status of special resources 90
ALPHA value in variable validation 462
alternate current plan data set, EQQCP2DS 280
alternate workstations 56, 586
automatic recovery 350
automatic rerouting to 153
recovery with cleanup 337
ampersand (&) variables 446
AOC 95
APAR (authorized program analysis report)
tape function 291
APARs
PK15640 332, 593
PK15761 539
PK17041 446
PK19236 495, 497
PK22521 242
PK24559 748
PK25979 340
PK28077 298
PK28888 335
PK30586 195, 201
PK35081 749
PK37583 233, 234, 242, 244
PK40561 284
PK41519 730
PK49503 529
PK50941 345
PK64650 762
PK69119 625
PK69328 670
PK71539 735
PK73549 740
PK79509 156
PK79909 243
PK80600 476
PK81179 763
PK81402 588
PK82176 61, 498, 504
PK83269 242
PK83834 466, 585
PK87319 748
PK88065 436
PK88734 314, 563, 566
PK93917 692
PK96348 625
PM01090 475, 636, 647, 737
PM01425 684, 686, 687, 688, 689, 692
PM04927 334, 746
PM06648 531
PM08778 234
PM10598 579
PM11388 146
PM19724 300, 343, 344, 436
PM21607 310, 428, 467, 469
PM23805 555
PQ6977 451, 453, 474, 478
PQ79808 335
PQ80124 324, 332, 349, 350, 358, 359
PQ83269 242

© Copyright IBM Corp. 1991, 2011
APARs (continued)
PQ84104 295, 532, 533
PQ84233 295, 532, 533
PQ85645 313, 578
PQ87473 335
PQ87576 305, 340
PQ87904 151, 152, 309, 314, 347, 745
PQ89557 313, 331, 349, 353, 356
PQ89566 446
PQ93442 670
PQ95706 476
PQ96888 340
PQ99317 272, 675, 702
PQ99903 343, 344, 482, 484
application description
database 190, 267
Application Description panel 119
application groups
guidelines 118
specifying 119
application groups, adding to the current plan 550
applications
adding to the current plan 550
common data 709
considerations 115
creating 24
cross-reference list 711
cross-references of job names and active applications 708
EVERY options 133
excluding 560
ID 117, 121
in the long-term plan 264
internal operation logic report 710
introduction 115
JCL variable tables 442
mass update report 714
naming standards 117
operation data 710
operations using particular workstations report 710
scheduling 123
status 122
updating in batch 185
valid-from date 122
valid-to date 122
APRIL keyword 225
ARC row command 367, 603
AROPTS initialization statement 290
AUTHUSER keyword 367
ENDTIME keyword 357
PREDWS keyword 365
STARTTIME keyword 357
USERREQ keyword 367
arrival time, specifying 158
associating variable tables with applications 442
asterisk (*) global search character 485
asterisk (*) operation status code 743
ATTR command 48
attributes, reporting for workstations 65
audit trails
failed jobs 368
recovery actions 368
reports 276
AUGUST keyword 225
AUTFUNC keyword 220
AUTHDEF initialization statement, security 367
authority group ID 121
authorization, batch programs 674
authorized program analysis report (APAR)
tape function 291
AUTHUSER keyword of AROPTS 367
automatic
job submission 153
recovery 306
reporting attribute 66
automatic job and started-task recovery
activating 290
deactivating 290
examples 32
introduction 347
automatic restart 602
automation options 151
automation using ETT 437
automation, option 74
avail on complete 149
availability
controlling with event-triggered resource definition 98
controlling with max usage limit option 96
controlling with NetView 94
controlling with on complete option 95
controlling with RODM 95
controlling with SRSTAT LIFESPAN 97
intervals 610
special resources 81, 605, 665
special resources and LOOKAHEAD 94
workstation 74
Availability of a Workstation panel 74
avoiding loops in variable substitution 489
B
B extended status code 744
BACKUP command 648
BACKUP keyword of JTOPTS 283, 648
backup workstation 56, 350
backward references in JCL 371
batch jobs to update the LTP 256
batch loader 119
close statements 192, 200
derror messages 196
example of statements 193
introduction 189
reasons for using 185
security 194
batch programs 50
BATCHOPT initialization statement
DYNAMICADD keyword 98
DYNAMICDEL keyword 98
PLANHOUR keyword 274
PREVRES keyword 274
BEGIN JCL directive 480
best restart step 324
bind
definition 406
bind process
for a shadow job 411
BIT value in variable validation 462
blank, extended status code 745
browsing
system information 540, 591
browsing active critical jobs 542
browsing critical jobs panel 541
browsing critical path panel 542
browsing general current plan information panel 540
browsing most critical occurrences panel 536
browsing summary of activities at a workstation panel 539
BULKDISC command 651
BULKDISC keyword of MONOPTS 651

C
C extended status code 744
C occurrence status code 743
C operation status code 743
C row command 566, 596
calendar 38
creating 21
defining 103
extra 106
for variable substitution 490
general information report 705
period characteristics report 706
search order 103
specific dates report 705
specifying 121
status of days report 706
CALENDAR keyword 234
CAN error code 745
CANCEL command, OS/390 734
case codes 369
CCUN error code 745
CDATE, JCL variable 453
CDAY, JCL variable 452
CDD, JCL variable 452
CDDDD, JCL variable 452
CFREEDAY, JCL variable 452
CHANGED MANUALLY (ONLINE) message 261
changing
application descriptions in batch 185
current plan 271, 549, 550
dependencies in the current plan 567
dependencies in the long-term plan 265
error-list layouts 596
group definitions 186
occurrences added to the current plan 561
operation status 79
operations in the current plan 568
operator instructions 186
resource availability 581, 611
special resources 615
workstation availability 584
workstation details 586
CHECK keyword 243
checking
current plan status 540
dependencies 539
if a job has failed 312
JCL variable substitution 461
job names 166
occurrences 535
operations 537
workstation status 539
checkpoint data set, EQQCKPT 280, 359
CHH, JCL variable 452
CHHM, JCL variable 452
CHHMMS, JCL variable 452
CHHMMSX, JCL variable 452
CICS, tracking jobs submitted by 434
CLASS option 152
cleanup
actions taken 334
cleanup (continued)
and automatic recovery 349
data sets 331
event-triggered tracking 340
generating copies 340
operations 161
options 331
starting 330
starting with AR 331
viewing results 331
when performed 334
CLEAR command 187
CLNA error code 745
CLNC error code 745
CLNO error code 745
closest to loop entry 512
closing down online systems 168
closing workstations 76
CMM, JCL variable 452
CMMYY, JCL variable 452
codes
error 745
extended status codes 744
job log retrieval 748
status codes for occurrences 743
status codes for operations 743
coding JCL variables 445
color and highlight attributes for panels 38
command delimiter, specifying 39
commands
concatenating 42
ISPF in dialog 43
OS/390
CANCEL 734
MODIFY 734
START 733
STOP 734
TSO 647
commands, system automation
customizing 457
enabling workstation 74
comments in long-term plan reports 261
COMMTEXT keyword 220
communicating with workstations menu 519
COMP keyword on JCL directives 485
COMPINFO keyword 220
COMPLETE status 251
completed applications
daily planning report 274, 726
error statistics (daily planning report) 727
summary of, daily planning report 726
completing an ended-in-error operation 596
completing an occurrence 566
completion codes 309, 311
completion-only reporting attribute 66
compound variables 447
compact computer automatic workstations
dynamic workstations 55
compact computer workstations
alternate 53, 56
job statements 165
job type 53, 54
started task (STC) type 54
using 53
congatating commands 42
COND RECOVERY JOB option 156
CONDCOUNT keyword 203
Index 771
Condition definitions panel 145

Condition dependency

definition 374

how is defined 374

how is evaluated 375

status evaluation 375

status False 376

status True 376

status Undefined 376

when is evaluated 375

Conditional

dependencies 9

logic 373

predecessor definition 9

successor definition 9

Conditional dependencies

automatic resolution 402

daily plan handling 402

Conditional dependency

at job-level 139, 144

at step-level 383

Automatic recovery 395

COND RECOVERY JOB option 392

Condition 374

Condition dependency 374

Critical path handling 395

Example of handling recovery 393

Examples at job-level 377

Examples at step-level 386

Highest return code 395

How to handle recovery 392

Job-level 373

MCP consistency 395

NOERROR 395

Operations ended-in-error 397

overview 373

Recovered by COND output flag 392

Restart and cleanup 395
CRITICAL option 152
critical path 537
  calculating 141
  jobs in current plan 541
  managing 142
  monitoring 141
  setting 141
  targets in current plan 541
Critical path
  setting 152
  WLM policy and class 152
cross
  dependencies 9
  as dependency from shadow job 409
  bind 406
  defining 140
  definition in database 406
  how to add to the plan 410
  HTTP or HTTPS destination 407
  introduction 405
  managing 405
  matching criteria 141
  monitoring resolution in plan 410
  remote engine workstation 405
  remote job 405
  resolution 141
  setup 58
  shadow job 406
  steps to define 406
cross reference
  application descriptions 711
  job names and active applications 708
cross-system coupling facility (XCF) 347
CSCRIPT keyword 215
CTIME, JCL variable 453
current plan
  analyzing problems 277
  automatic backup process 283
  catch up 283
  changing 278, 547
  checking occurrences 535
  creation date and time 540
  critical jobs 541
  critical path targets 541
  description 7, 249
  end date and time 540
  event-triggered tracking 435
  extending 270
  first logged event after last backup 540
  integrity 280
  introduction 245, 267
  last backup 540
  length 252
  NCP takeover 283
  new current plan 284
  new plan in production 540
  on-request BACKUP 648
  on-request BULKDISC 651
  on-request JSUACT 653
  organization 280
  recreating 271
  replanning 582
  reports 273
  status 540, 586
  trial 272
  turnover 283

current plan and status inquiry menu 535
Current Plan and Status Inquiry menu 279
current plan extension data set, EQQCXDS 280
CWW, JCL variable 452
CWWD, JCL variable 452
cycle specification, for rules 127
cyclic periods 106
CYMD, JCL variable 452
CYY, JCL variable 452
CYYDDD, JCL variable 452
CYYMM, JCL variable 452
CYYMDD, JCL variable 452
CYYYYY, JCL variable 452
CYYYYYY, JCL variable 452

D
D extended status code 744
D occurrence status code 743
D operation status code 743
D row command 566, 569, 614
daily plan consistency checks
  on remote engine workstations 409
  on shadow jobs 409
daily planning 267
  analyzing problems 277
  completed applications report 726
  detecting a loop condition 509
  error statistics on completed applications 727
  missed feedback report 728
  new current plan process 284
  operation in error report 727
  report on completed applications 273
  reports
    actual resource utilization 729
    general information report 720
    operating plan 721
    parallel operations, daily planning report 723
    plan for workstation 722
    planned resource utilization 724
    resource 1, daily planning report 723
    summary of completed applications report 726
    workstation resource utilization reports 723
    workstation utilization reports 723
data lookaside facility (DLF) 100
data set triggering selection table macro (EQQLSENT)
  invoking 760
  syntax 761
data sets
  EQQAD2DS 195
  EQQADDSS 195
  EQQDUMP 195
  EQQBLIB 296
  EQQLIB 195
  EQQMLOG 195
  EQQIDS 196
  EQQOIPDS 195
  EQQSTC 297
  EQQWSDS 196
  for long-term plan 256
  JCL repository (EQQJSnDS) 296
  SYSUDUMP 195
data store 304
date-related supplied variables 452
date, specifying format 36
DAY keyword 225
day specification, for rules 127
day status 104
daylight savings time, automatic update 301
DB2 578
DB2 database, using 306
DBCS (double-byte character set) 46
DBCS (double-byte character set) and job descriptions 116, 170
ddnames being used 540
deactivating automatic recovery 369
deactivating event-triggered tracking (ETT) 433
deactivating job submission 289
deactivating variable substitution 445, 464
deadline feedback 122
DEADLINE - START > 24 HRS message 261
deadline date 552
deadline feedback options 122
deadline smoothing factor 123
DEADLINE WTO option 155
deadline, specifying 126
deallocating a resource 611
DECEMBER keyword 225
default calendar 21, 38
default time for an operation 74
defining
  applications 24, 115, 119
  calendars 21, 103
  ETT criteria 433
  JCL variable tables 108
  job descriptions 169
  operation as critical path target 141
  operations 137
  periods 106
  predecessors 139
  reporting attributes 65
  resource usage 147
  rules 125
  run cycles 124
  special resources 20, 86
  workstations 17, 53
defining dependencies in the current plan panel 557
delaying operations 531
delays 533
deleting
  dependencies 555
  occurrences 566
  operations 569, 577
deleting data sets, for job recovery 331
DEP primary command 567
dependencies
  adding 555
  changing 555, 567
  changing in the long-term plan 265
  checking 539
  displaying graphically 43
  in AD database 190
  in the long-term plan 257
  long-term plan default 263
  resolving 146, 276
  specifying 553
  with GRAPH command 43
dependency (continued)
  specifying 139
DEPENDENCY CHANGED message 261
dependency loop 277, 509
dependency resolution with ETT 433
dependent variable 457
DESCR keyword 209, 215, 229
destination of workstations 73
DETAILS command 173
determining the success or failure of a job 312
development of special resources 83, 666
dialog
  commands, ISPF in dialog 43
  main menu 35
  options, setting 36
  overview 35
  parameters, setting 36
  query current plan 535
  Ready List 519
directives in JCL 466
  disabling variable substitution 445, 464
disaster recovery, Tivoli Workload Scheduler 584
discarding actions 601
  displaying
    critical path targets 541
distributed jobs
  scheduling in z-centric end-to-end configuration 71
DILAY keyword 229
DLF
  See data lookaside facility 100
DLIMFDBK keyword 234
DLTIME keyword 229
double-byte character set (DBCS) 46
  and job descriptions 116, 170
DSMOOTHING keyword 235
DSNAME value in variable validation 462
dual job-tracking-log data set, EQQDLnn 280
dummy application 549
dummy operation 61
dummy workstation 61
duplicate occurrences 257
duration 74
  feedback 156
  duration feedback options 156
DURATION keyword 215
duration per workstation report 718
duration smoothing factor 157
DURUNIT keyword 243
dynamic inclusion of JCL 441
dynamic jobs
  scheduling in z-centric end-to-end configuration 56
dynamic resource creation 98
dynamic type
  setting 69
Dynamic Workload Console 50
  accessibility xxiii
dynamic workstation
  dynamic type 69
  settings 69
dynamic workstations 55
dynamic-format supplied variables 453, 472
DYNAMICADD keyword of BATCHOPT 98
DYNAMICADD keyword of RESOPTS 99
DYNAMICDEL keyword of BATCHOPT 98
E extended status code 744
E occurrence status code 743
E operation status code 743
editing JCL for an MVS job panel 602
editing JCL for an operation 527
editing JCL for an operation panel 528, 530
editing JCL variable tables 454
education
See Tivoli technical training
EIADAYS keyword 229
embedded blanks in JCL variables 466, 486
enabling JCL variable substitution 444
END JCL directive 480
end time of a work day 105
ended-in-error
list layout 594
operations 593
status 527
ended-in-error operations 311
ENDTIME keyword of AROPTS 357
ENTERED MANUALLY (ONLINE) message 261
ENUM value in variable validation 462
EQACCOPP panel 120
EQAD2DS data set 195
EQADVCP batch program 676
EQADDEP batch program 677
EQADDSD, application description data set 195
EQADMUP batch program 677
EQADPRT batch program 678
EQADXRF batch program 679
EQALML panel 159
EQAMAIP panel 163
EQAMCMCL panel 144
EQAMCMCP panel 145
EQAMFBDP panel 156
EQAMJBP panel 29, 151, 620, 629
EQAMOPL panel 137
EQAMOSL panel 140
EQAMPDLP panel 144
EQAMRCL panel 161
EQAMRDP panel 165
EQAMRNLP panel 130
EQAMRPL panel 125
EQAMRZLP panel 165
EQAMSDDP panel 143, 173
EQAMSRL panel 147
EQAMTMP panel 158
EQAMUFL panel 164
EQAMWBP panel 150
EQAMXDP panel 162
EQAOIDP panel 160
EQAUDT batch program 679
EQAUFP panel 187
EQAXDPL panel 186
EQAXUCL panel 187
EQBATCH program 673
EQCASEC, case-code-list definition macro 369
EQCKPT, checkpoint data set 280
EQCLEAN 601
EQCLPRC batch program 682
EQCLPRP batch program 683
EQCP1DS, primary current plan data set 280
EQCP2DS, alternate current plan data set 280
EQCXDS, current plan extension data set 280
EQDLFX exit sample 100
EQDLnn, dual job-tracking-log data set 280
EQDNTOP batch program 683
EQDDOTOP batch program 686
EQDPLNP panel 669
EQDDRTOP batch program 687
EQDSTOP batch program 688
EQDTTOP batch program 689
EQDUMP, diagnostic data set 195
EQDELDEF member 596
EQDLOUT member 596
EQDELICL panel 595
EQDELYL panel 595
EQQEVPCM batch program 647, 691
changing operation status 57
in event-triggered tracking 432
EQDHIPUP panel 581
EQDHISTL panel 580
EQQJBLIB, job library data set 24, 165, 296, 367
EQQJCOPP panel 170
EQQJTCPL panel 433
EQQJSnDs (JCL repository data set) 24, 265
EQQJSnDS, JCL repository data set 24, 265
EQQJSNBP panel 170
EQQJTARC, job-tracking-archive data set 280
EQQJTN, job-tracking-log data set 280
EQQJVL panel 457
EQQJVMAP menu 454
EQQJVPRT batch program 691
EQQJVML panel 454
EQQJVVL panel 455
EQQJVVPN panel 461
EQQJVMPP panel 455
EQQKCRTE panel 159
EQQLBPAT panel 256
EQQLBWP panel 263
EQQLCDPL panel 266
EQQLCHGP panel 265
EQQLCRPE panel 258
EQQLDSS, long-term plan work data set 256
EQQLEXTP panel 259
EQQLPRAP panel 260
EQQPLAND (data set triggering selection table macro) invoking 760
syntax 761
EQQLSTAP panel 262
EQQLLSTOL panel 265
EQQLTPKBP, long-term-plan backup data set 256
EQQLTRC batch program 692
EQQLTDS, long-term plan data set 256
EQQLTEXP panel 262
EQQLTMOA batch program 693
EQQLTMEO batch program 694
EQQLTOPP panel 255
EQQLTPRT batch program 694
EQQLTRY batch program 696
EQQMAADL panel 551
EQQMAAGL panel 559
EQQMAADD panel 550
EQQMAMOL panel 561
EQQMAOCP panel 551
EQQMCMDL panel 601
EQQMEP panel 397, 593
EQQMMERP panel 594
EQQMERSI panel 321
EQQMERSL panel 320, 600
EQQMERTP panel 598
EQQMJCPL panel 602
IBM Tivoli Workload Scheduler for z/OS: Managing the Workload

ERROR CODE = xxxx message 261
error codes
automatic recovery 745
from completion codes 309
introduction 309
job log retrieval 748
jobs on distributed agents 311
specifying 553
error list 593
error messages
batch loader 196
generated in long-term plan reports 261
error statistics on completed applications 275, 727
ERROR TRACKING option 151
errors in variable substitution 463
ERRRES, keyword of JTOPTS 312
estimated deadline, feedback 122
estimated duration, feedback 156
ETT keyword of JTOPTS 291, 433
ETTRCY1 period use by ETT 436
event data set, time zones 301
event rule definition
keywords
actionProvider 755
actionType 755
description 754, 755
eventCondition 754
eventProvider 754
eventRule 754
eventType 754
eventRuleType 754
name 754
responseType 755
ruleType 754

event-driven workload automation (EDWA) 425
HIGHRC keyword of JTOPTS 215, 311
Hiperbatch 100
HIST command 577
histogram showing workstation utilization 261
history database 525, 578
history function 578
history function, using 306
HOLD status 294
HOLD/RELEASE option 154
holding operations 531
HTTP or HTTPS destination
for cross dependencies 407
remote engine workstation 58
ROUTOPTS statement 407
I
I operation status code 743
I row command 569
IA time check 512
IADAYS keyword 229
IATIME keyword 229
IBM Tivoli Monitoring
description
Tivoli Enterprise Portal 623
scheduler start options 624
Tivoli Enterprise Portal
setting monitors 628
IMS, tracking jobs submitted by 434
in-effect dates 132
in-progress operation 655
in-use list 613
inactive workstation, effect on operations 153
including dependencies 553
including JCL 466, 483
independent variable 458
information about jobs 527
initialization statements
determining success or failure of a job 312
JTOPTS
HIGHRC keyword 152
JOBSUBMIT keyword 153
NOERROR keyword 152
SUPPRESSACTION keyword 169
WSFAILURE keyword 153
WSOFFLINE keyword 153
initiating deferred actions 601
initiators 150
input arrival date 552
input arrival time (IAT)
importance for rules 125
in long-term plan 247
occurrences 133
operations 158
resolving dependencies 146
input field pad character, specifying 39
input, batch loader 189
instruction database 192
instructions 596
Edit JCL 160
operator 159
internal dependency 140
internal operation logic report 710
interrupting workstation activity 72
interval origin 108
intervals
for special resources 92
intervals (continued)
when workstations are open 75
intervals, for special resources 610
invoking JCL variable substitution 444
IPL, avoiding lost events 300
ISPF (Interactive System Productivity Facility)
cmdon delimit 42
dialog commands 43
dialog manager 35
select service 521
setting options 35, 39
table services 35
using 35
J
J event, in event-triggered tracking 432
J row command 597
JANUARY keyword 225
Japanese language environment 46
JCCE error code 745
JCL
backward references 371
checking 166
editing 597
error code 746
including and excluding 466
on-request BACKUP of repository file 648
on-request BULKDISC 651
on-request JSUACT of repository file 653
preparing 22, 527
rebuild parameters 357
RECOVER statement 351
repository 24, 165, 265, 296
tailoring 165, 441
variable substitution
in JCL procedures 464
introduction 441
invoking 444
restrictions 488
supplied variables 449
suppressing 445, 464
syntax 445
validation 461
variables with embedded blanks 466
ý (logical not) 441
variable table 108
creating 455
specifying 126, 470, 479
variable table report 713
variables 529
JCL setup workstations 57
JCL Variable Dependency Value List panel 457
JCLI error code 746
JCLPREPA record, simulating JCL substitution 441
JES initiators 150
controlling with parallel servers 72
job
conditional dependency 373
JOB CLASS option 154
job completion checker (JCC)
considerations 304
setting error codes 310
job computer workstations 54
Job Description panel 119
job descriptions
creating 116
job descriptions (continued)
  mass updating 186
job log
  retrieval status codes 748
job log retrieval, automatic 304
job logs
  retrieving and storing 304
job name, changing 568
job restart 577
job setup workstations
  description 57
preparing JCL 166
job statement information for system printer, specifying 39
job submission
  criteria 533
delays 533
job-level
  conditional dependency 377
  conditional dependency examples 377
  conditional predecessor 377
  conditional successor 377
job-level condition
  application consistency using FOP 382
  EQQE141W message 381, 390
  EQQE142W message 389
  key concepts 381
  restrictions 381
  Unexpected RC 380
Job, WTO, and Print Options panel 29, 151, 620, 629
JOBCHECK keyword 166
JOBCLASS keyword 216
JOBLOGRETRIEVAL keyword 304
JOBN keyword 216, 240
jobs
  and started-task recovery 347
  activating 290
  deactivating 290
  cleanup 333
  delaying the start 531
  executing 533
  holding 531
  names
  checking 166
  description 117
  ETT replace function 433
  replace function of ETT 433
NOPing 532
options 151, 154
preparation 22
priority 293
recovery example 32, 362
releasing 531
repository 165, 296
data set 24, 265
submission
  activating and deactivating 289
  order 293
tailoring 165
tracking
  archive data set (EQQJTARC) 280
  log data set (EQQJTran) 280
variable table
  creating 455
  specifying 126
JS file backup 648
JS file jsuact 653
JSUACT command 653
JSUACT keyword of JTOPTS 653
JTOPTS
  SUBFALIACTION keyword 533
  WSFAILURE keyword 585
  WSOFFLINE keyword 585
JTOPTS initialization statement
  BACKUP keyword 283, 648
  ETT keyword 291, 433
  HIGHRC keyword 152, 311
  JOBCHECK keyword 166
  JOBSUBMIT keyword 153, 289
CACT keyword 653
MAXJSFILE keyword 648
NOERROR keyword 152, 311
OPINFOSCOPE keyword 655
QUEUELEN keyword 295
SHUTDOWNPOLICY keyword 76
SUBFALIACTION keyword 747
SUPPRESSACTION keyword 169, 747
SUPPRESSPOLICY keyword 169
TRACK keyword 435
WSFAILURE keyword 153, 351, 747
WSOFFLINE keyword 153, 351, 746
JULY keyword 225
JUNE keyword 225
K
Kanji characters 46
keep on error 148
keeping
  sequential data set with application descriptions 185
KEEPONERR keyword 231
key concepts
  job-level condition 381
KEYS command 48
killing a job on a standard agent 577
L
L extended status code 744
language support, REINIT 36
LAST keyword 225
last operation, definition 509
late operations 168
latest start time 293, 536
layout of OI data set 699
layouts 519
LIFESPAN
  controlling availability of a resource 97
lifespan of special resources 666
LIMFDBK keyword of JTOPTS 216
limit
  for deadline feedback 123
  for duration feedback 157
  specifying 121
linking applications together 118
linklist 195
list criteria, specifying 44
list data set options, specifying 39
list of calendar periods panel 106
List of Generated Dates panel 127
list of JCL preparation variables to be set panel 529
List of JCL Variable Tables panel 454
List of Operator Instructions panel 159
List of Special Resources panel 89
mass update 185
mass update reports
  application descriptions 714
  update performed for owner ID    714
mass updating of application description panel 186
matching criteria
  cross dependency 141
  remote job 141
  shadow job 141
max usage limit 82, 606
  controlling availability of a resource 96
  updated by daily plan batch processes 97
max usage type 82, 606
  updated by daily plan batch processes 97
MAXJSFILE keyword of JTOPTS 648
MAY keyword 225
MCP error code 746
MCP panel 547
MEMBER keyword 240
message retrieval tool, LookAt xxii
MH command 531, 577
migrated data sets 336
minimal net distortion 512
missed feedback report 275, 728
missing step end
  step-level dependency 385
MODIFY command, OS/390 734
modify current plan (MCP) panel
  adding applications 550
  adding group applications 550
Modify Current Plan (MCP) panel 547
modifying
  fault-tolerant workstation 583
  modifying a JCL Variable panel 455
  modifying a Rule panel 126
  modifying a Special Resource panel 615
  modifying a workstation in the current plan panel 586
  modifying all workstations closed panel 77
  Modifying an Occurrence panel 265
  modifying an operation in the current plan menu 556
  modifying clean up actions panel 601
  Modifying Connected Workstations for a Special Resource panel 618
  Modifying Connected Workstations panel 93
  modifying dependencies in the current plan 556
  Modifying Dependencies panel 266
  Modifying ETT Tracking Criteria panel 433
  modifying extended info in the current plan panel 569
  Modifying Intervals for a Special Resource panel 92, 617
  modifying occurrences added to the current plan panel 561
  modifying occurrences in the current plan panel 562
  modifying open time intervals in the CP panel 587
  modifying operations in the current plan panel 555, 577, 578
  modifying resources 615
  Modifying the Current Plan menu 278, 549
  modifying virtual workstation destinations panel 64
  modifying workstation status in the current plan 588, 589
  modifying workstation status in the current plan panel 587
MONDAY keyword 225
MONITOR keyword 216
monitoring
  resources 87, 610
  special resources 605
workload, a user scenario 543

M
M extended status code 744
M row command 567
macros
  EQQLSENT (data set triggering selection table macro) 760
main menu 35
maintaining a sequential data set with application descriptions 185
Maintaining Job Descriptions menu 170
Maintaining Special Resources panel 89
Maintaining the Long-Term Plan menu 255
Maintaining TWSz Databases panel 88
Maintaining TWSz JCL Variable Tables menu 454
Making a Trial Long-Term Plan panel 262
making changes conditionally 186
making plans smaller 174
manually holding operations 531, 577
manually releasing operations 531, 577
MARCH keyword 225
Index 781
OPCOPTS initialization statement (continued)
  RODMTASK keyword 85
  VARSUB keyword 444
open intervals for workstations 75, 584
open time intervals for one day panel 75
OPER command 121, 568
operating plan, daily planning report 721
operation data report 710
Operation Details panel 143, 173
operation in error, daily planning report 727
operation number 117
operation-related supplied variables 451
operations
  automatic restart 602
  browsing 577
  changing 568, 569
  checking 537
  creating 24, 137
  deadline feedback options 122
  defining details 143
  delay start 577
  delaying the start 531
  deleting 569
  diagnosing delays 533
  dummy 61
  duration feedback options 156
  ended-in-error 311
  ended-in-error status 527
  EX command 577
  EXECUTE command 533
  extended information 569
  extended status codes 744
  feedback of information 655
  history 577, 578
  holding 531, 577
  in error 593
  interrupting 527
  MH command 577
  MR command 577
  NOPing 532
  NOPing an operation 577
  NP command 532, 577
  options 151
  parallel servers 150
  print 167
  querying 537
  reason codes 748
  releasing 531, 577
  removing a resource 611
  removing from group 577
  removing from schedule 577
  requesting immediate execute 533, 577
  rerunning 563
  resources 294
  resources used 147
  restarting 597
  restarting and cleaning up 161
  reversing a NO-OP 532, 577
  setting status 526
  special resources and LOOKAHEAD 94
  started task 167
  starting immediately 533, 577
  status (OPSTAT) command 659
  status codes 743
  step restart 600
  submission criteria 533
  time-dependent 155
operations (continued)
  UN command 532, 577
  user data 655
  using special resources 613
  waiting for resources 613
  write to operator (WTO) 295
  WTO 167
Operations panel 137, 140
operations using particular workstations, report 710
operator commands, OS/390 710
operator instructions 527, 596
  data set 699
  database 190, 192
  editing JCL 160
  loading in batch 189
  report example 715
  specifying 159
  OPIADATE, JCL variable 453
  OPIATIME, JCL variable 453
  OPINFO command
    description 655
    security considerations 656
    syntax 656
  OPINFOSCOPE keyword of JTOPTS 655
  OPLSDATE, JCL variable 453
  OPLSTIME, JCL variable 453
  OPNO keyword 216, 240
  OPSTAT command 432
    general workstation status 57
    introduction 659
    updating the current plan 249
  options and parameters, defining 36
  OPTIONS, batch-loader control statement 242
  order of operations 293
  origin date 108
  origin of periods 106
  origin shift, in rules 127
  original return code
    condition 375
  ORIGINSHIFT keyword 225
OS/390
  CANCEL command 734
  MODIFY command 734
  operator commands 733
  START command 733
  STOP command 734
  OSEQ error code 599, 746
  OSUB error code 746
  OSUF error code 747
  OSUP error code 747
  OSxx error code 747
  out-of-effect dates 132
  output group, tracking print operations 167
  output, batch loader 190
  owner ID
    for applications 117
    mass update report 714
  OWNER keyword 236, 244
  OWSID, JCL variable 452
  OWW, JCL variable 450
  OWWDD, JCL variable 450
  OWWM, JCL variable 450
  OXJOBNAM, JCL variable 451
  OYM, JCL variable 451
  OYMD, JCL variable 451
  OYMD1, JCL variable 451
  OYMD2, JCL variable 451
  OYMD3, JCL variable 451
OYMD2, JCL variable 451
OYMD3, JCL variable 451
OYY, JCL variable 451
OYYDDD, JCL variable 451
OYYMM, JCL variable 451
OYYYY, JCL variable 451

P
P occurrence status code 743
pad character, specifying 39
panel highlight attributes 38

adding an application to the current plan 551
adding an occurrence group to the CP 559
adding applications to the current plan 550
all dependencies of an operation 538
browsing active critical jobs 542
browsing condition 400
browsing condition dependencies 401
browsing critical jobs 541, 542
browsing general current plan information 540
browsing Most Critical Occurrences 536
browsing summary of activities at a workstation 539
communicating with workstations 519
confirm restart 598
creating a dependency in the current plan 557
creating a ready list layout 521
creating an error list layout 595
current plan and status inquiry 535
defining dependencies in the current plan 557
editing JCL for an MVS job 602
editing JCL for an operation 528, 530

EQQACGP 120
EQQALSM 159
EQQAMAIP 163
EQQAMCCCL 144
EQQAMCCCP 145
EQQAMFBP 156
EQQAMJPB 29, 151, 620, 629
EQQAMOPL 137
EQQAMOSL 140
EQQAMPDLP 144
EQQAMRCCL 161
EQQAMRDLP 165
EQQAMRNLL 130
EQQAMRPPL 125
EQQAMRZIP 165
EQQAMSDDP 143, 173
EQQAMSRLL 147
EQQAMTMBP 158
EQQAMUFL 164
EQQAMWARP 150
EQQAMXDP 162
EQQAOIDP 160
EQQAFUFL 187
EQQAPUPL 186
EQQAUULCL 187
EQQDPLPNP 269
EQQELYCL 595
EQQELYLL 595
EQQHIPUP 581
EQQHISTL 580
EQQICGP 170
EQQIMTCL 433
EQQISUBP 170
EQQIVDVL 457

panels (continued)
EQQIVMAP 454
EQQIVTML 454
EQQIVVCL 455
EQQIVVEP 461
EQQIVVMP 455
EQQQKCRTE 159
EQLABATP 256
EQLBDW 263
EQLCDP 266
EQLCHGP 265
EQLCLRP 258
EQLCXTP 259
EQLPRAP 260
EQLSTAP 262
EQLSTCL 265
EQLTEXP 262
EQLTOPP 255
EQQMAADL 551
EQQMAGL 559
EQQMADDP 550
EQQMAOML 561
EQQMAOCPL 551
EQQMCMDL 601
EQQMEPL 397, 593
EQQMERRP 594
EQQMERSI 321
EQQMERSL 320, 600
EQQMERTP 598
EQQMJCLE 602
EQQMMADP 557
EQQMMCLL 558
EQQMMDDL 557
EQQMMDPL 556
EQQMMODP 556
EQQMMOPL 555
EQQMMXDP 569
EQQMOCCL 562
EQQMOPRL 577
EQQMOPRP 578
EQQMRCL 563
EQQMTOPP 278, 549
EQQMDES 589
EQQMWSIL 590
EQQMWSIP 588
EQQMWSLP 589
EQQMWSLL 583
EQQMWSOL 587
EQQMWSRP 586
EQQMWSRV 589
EQQMWSVP 587
EQQQDSBP 88
EQQQPCAP 35
EQQQDCRP 89
EQQQDEM 92, 617
EQQQDSL 89
EQQQDTOP 89
EQQQDWM 93, 618
EQQQMMIL 613
EQQQMLSL 612
EQQQMMRP 615
EQQQMSFP 611
EQQQMWML 613
EQQQRCSE 313, 564, 597
EQQRCL 528, 530
EQQRLRM 525
EQQRLVAL 529

Index 783
Generating JCL for a Batch Job 50
handling operations ended in error 593
list of JCL preparation variables to be set 529
Modify Current Plan 547
Modifying a Special Resource 615
Modifying a Virtual Workstation Destination in the CP 589
modifying a workstation in the current plan 586
modifying an operation in the current plan 556
modifying clean up actions 601
modifying conditional dependencies in the CP 558
Modifying Connected Workstations for a Special Resource 618
modifying dependencies in the current plan 556
modifying extended info in the current plan 569
Modifying Intervals for a Special Resource 617
modifying occurrences added to the current plan 561
modifying occurrences in the current plan 562
modifying open intervals of a virtual WS destination 590
modifying open time intervals in the CP 587
modifying operations in the current plan 555, 577, 578
Modifying the Current Plan 549
modifying workstation status in the current plan 587, 588, 589
ready list 525
ready list layouts 521
rerunning an occurrence in the current plan 563
selecting an error list layout 595
selecting application occurrence and operation information 534, 537
Selecting application occurrence information 536
selecting applications to add to the CP 551
selecting operations 400
Special Resource Monitor 612
Special Resource Monitor - In Use List 613
Special Resource Monitor - Waiting Queue 613
specifying ended in error list criteria 594
specifying ready list criteria 520
Specifying Resource Monitor List Criteria 611
step information list 321
step restart selection list 320, 600
parallel operations for workstations, daily planning report 723
parallel servers 71, 81, 150
parameters and options, defining 36
path for progression condition 375
Paymore example 13
PCAN error code 747
pending application status 122
pending occurrences, resolving 250
percent (%) global search character 485
percent (%) variables 447
period characteristics report for calendars 706
PERIOD keyword 229
periods
  defining 106
  JCL variable table 108
  offsets 108
  work-days-only 111
  PF keys, assigning 48
  PF keys, specifying number of 39
  PFSHOW command 49
  phase of substitution 445
  PICT value in variable validation 462
plan
  example of creating 30
  reducing the size 174
  types 245
plan execution, managing critical path 142
plan for workstation, daily planning report 722
PLANHOUR keyword of BATCHOPT 274
planning system outages 76
POLICY option 152
potential predecessor 554
PREADID keyword 209
PRED command 140
predecessor
  definition 9
predecessors
  changing 567
  checking 539
  in the current plan 276
  including 553
  potential 554
  specifying 139
Predecessors panel 144
PREDSW keyword of AROPTS 365
PREJOBN keyword 210, 216
PREOPNO keyword 209, 216
preparing
  JCL 22, 166
  job statements 57
preparing JCL 529
PREVRES keyword of BATCHOPT 274
PREWSID keyword 209, 216

primary commands 567, 568
primary current plan data set, EQQCP1DS 280
print of application descriptions 709
print operations 154, 167
print process options, specifying 39
printer workstations 56
printing the long-term plan 260
Printing the Long-Term Plan - All Applications panel 260
PRIORITY keyword 236
priority of operations 121, 293
problems with the current plan 277
procedure library for started tasks 54
procedure library, started tasks 165
procedures, variable substitution 464
producing daily plans 269
program function (PF) keys 49
program interface (PIF)
   for creating applications 119
   JCL preparation 441
promptable variables 441, 529
PSNUM keyword 216
publications xxii
Q
Q extended status code 744
QCP dialog 535
QUANTITY keyword 231
quantity of resources 612
quantity of special resources 81, 148, 605, 667
Query Current Plan (QCP) dialog 535
querying the current plan 279
question mark (?) variables 448
QUEUELEN keyword of JTOPTS 295
R
R event, in event-triggered tracking 432
R extended status code 744
R operation status code 743
R row command 563, 569
R1 and R2 resources 78, 149
RNUM keyword 216
R2NUM keyword 216
RACF, resource profiles 291
RANGE value in variable validation 462
RCLEANUP parameter 304
Ready
   conditional successor status 376
ready list
   resources 294
ready lists
   layout, user exit 522
   panels 521, 525
   using 525
reason codes 748
reasons for job failure 312
rebuild parameters 357
rebuilding the current plan 582
recalling the last command 48
RECOMPL keyword 222
RECOVER statement 290, 310, 347, 351
recovery
   activating 290
   deactivating 290
   example 32
recovery (continued)
   job and started task 347
   modify current plan panel 367
   with data set cleanup 349
   RECOVERY keyword of OPCOPTS 369
   recovery when automatic restart fails 602
   redirecting work to an alternate workstation 586
   reducing the size of plans 174
   reference to a broker job 71
   reference to a Broke job 56
   refreshing
      long-term plan 258, 290
      RACF resource profiles 291
   REINIT option 36
   REJOBNAME keyword 222
   REJSNAME keyword 222
   REJSWS keyword 222
   releasing operations 531, 577
   remote engine
      workstation type 58
   remote engine type
      setting 71
   remote engine workstation
      daily plan consistency checks 409
      defining 58, 408
      definition 405
      destination 71
      HTTP or HTTPS destination 58
      modify in the CP 423
      remote engine failover 58
      remote engine type 71
      settings 71
      shadow job 58
      to support remote failover scenario 423
      when switching master domain manager 423
   remote job
      definition 405
      how it is bound 412, 415
      matching criteria 141
      status transition during recovery 422
      status transition during rerun 423
   Remote Job Info panel 164, 165
   remote job information, defining 222
   removing a resource from an operation 611
   removing an operation from the schedule 532, 577
   REOPNO keyword 222
   REPLAN option 271, 582
   reporting attributes, defining 65
   reports
      actual resource utilization 729
      application description 709
      cross-reference of applications and external dependencies 708
      cross-references of job names and active applications 708
   calendars
      period characteristics 706
      specific dates 705
      status of days 706
   completed applications 726
   cross-reference of applications 711
   current plan 273
   daily planning
      daily operating plan 721
      general information 720
      plan for work station 722
      summary of completed applications 726
Index 785
reports (continued)
  daily planning (continued)
    workstation resource utilization 723
    workstation utilization 723
    duration per workstation 718
    error statistics on completed applications 727
examples 705
internal operation logic 710
JCL variable table 713
long-term plan 260
  general information examples 716
  grand total workload for period 718
mass update
  application descriptions 714
  owner ID, updates performed for 714
missed feedback 728
operation data 710
operations in error 727
operations using particular workstations 710
operator instructions 715
planned resource utilization 724
workstation description 707
REROUTABLE keyword 216
REROUTABLE option 153
rerunning a job 313
rerunning an occurrence in the current plan panel 563
rerunning occurrences 563
resetting operations 312
resolving dependencies 146, 276
RESOPTS initialization statement
  DYNAMICADD keyword 99
RESOURCE keyword 231
Resource Monitor 610
Resource Object Data Manager (RODM) 85, 609
  controlling availability of a resource 95
resource profiles, refreshing RACF 291
resources
  allocation status 90
  changing 615
  changing the status 664
  data lookaside facility (DLF) 100
description database
  input to current plan 267
development 612
  fixed R1 and R2 78
  monitoring 605
  parallel servers 150
  quantity 612
  R1 and R2 149
  special 20, 147
  unplanned changes 581, 611
  usage 147
  workstation fixed 149
resources for a workstation panel 78
RESSTEP keyword 349
restart and cleanup (continued)
  operations 597
  overview 313
  rerunning occurrences 564
  status, pending processes 602
  uncatalog data sets 331
RESTARTABLE keyword 217
RESTARTABLE option 153
RESTARTINFORETRIEVAL keyword 304
restarting
  from a step 600
  jobs 330
  jobs, overview 313
  occurrences 562
  operations 161
  operations automatically 602
restarting an ended-in-error operation 597
  managing cleanup action 597
restrictions
  job-level condition 381
  restrictions on variable substitution 488
RETCODE keyword of EWUROPTS 309, 311
RETRIEVE command 48
retrieving and storing job logs 304
return code check
  condition 374
return codes
  grouping together 369
  HIGHEST RETURNCODE option 152, 171
RG command 577
RODM
  See Resource Object Data Manager 85
RODM (Resource Object Data Manager) 609
RODMOPTS initialization statement
  RODMTASK keyword 85, 609
RODMTASK keyword 85, 609
row commands
  adding or deleting operations 569
  failed jobs 602
  failed operations 596
  for failed operations 596
  for special resources 614, 615
RPTENDT keyword 230
RPTEVRY keyword 230
rule
  condition 374
RULE keyword 230
rules 106
  creating 124
  creating with batch loader 224
  in run cycles 125
RUN command 121, 173
run cycle
  creating 124
  EVERY options 133
  examples using offsets 135
  negative 131
  offsets 129
  periods 110
  specifying with batch loader 228
Run Cycles panel 125
run days 131
Run Days panel 130
run order of operations 293
running OPC commands in batch 691
S
S extended status code 744
S occurrence status code 743
S operation status code 743
sample application implementation 13
SATURDAY keyword 225
saving error-list layouts 596
SCAN JCL directive 469
SCHEDULED ON FREE DAY message 261
scheduling job closedown 168
scope of OPINFO 655
screen format, specifying 39
SEARCH JCL directive 470
search order
  calendar 103
  JCL variables 442
search order, calendar 38
searching 46
searching application descriptions 186
SECELEM keyword 221
security
  automatic job and started-task recovery 367
  BACKUP command 649
  batch loader 194
  of batch programs 674
  OPINFO command 656
  OPSTAT command 660
  refreshing RACF resource profiles 291
  SRSTAT command 664
  WSTAT command 669
selecting
  restart steps 319
  work for submission 293
selecting an error list layout panel 595
selecting application occurrence and operation information menu 537
selecting application occurrence and operation information panel 534
selecting application occurrence information panel 536
selecting applications to add to the CP panel 551
Selecting Long-Term Plan Batch Job panel 256
selection criteria 549
SEPTEMBER keyword 225
SERC error code 747
servers
  parallel 71
service functions 289
Service Functions panel 289
SET CLOCK command 301
SETFORM JCL directive 472
setting an occurrence to complete 566
Setting Default for Browse panel 263
setting up
  for step-level dependency 384
setting up JCL 22, 166, 527
setup variables 456
SETVAR JCL directive 474
shadow job (continued)
  modify in the CP 421
  remote engine workstation 58
Remote job info 164
status 141, 411
status transition 411, 418
shared and exclusive use of special resources 148
shared spool 59
shift of origin, in rules 127
SHPF error code 747
shutdown
  avoiding lost events 300
  online systems 168
SHUTDOWNPOLICY keyword of JTOPTS 76
side information data set, EQQSIDS 280
simple variables 447
simulating JCL substitution 441
smoothing factor
  specifying 121
SMOOTHING keyword of JTOPTS 217
SMS 336
SOME NODES COULD NOT BE CHECKED loop 510
SORT command 43, 46
sorting items in a list 46
Special Resource Monitor 87, 610
Special Resource Monitor - In Use List panel 613
Special Resource Monitor - Waiting Queue panel 613
Special Resource Monitor panel 612
special resources
  actual utilization report 729
  allocation status 90
  availability 81, 605
  changing global availability 95, 96, 97, 98
  connecting to workstations 93
  data lookaside facility (DLF) 100
  deviation 83
  example 20
  how to use them 85
  intervals 90
  introduction 81
  max usage limit 82, 91, 606
  max usage type 82, 91, 606
  on complete 82, 91, 605
  on-error action 81, 605
  parallel servers 81
  planned utilization report 724
  quantity 81, 605
  reports 101
  shared and exclusive 148
  status (SRSTAT) command 664
  understanding 81
  updating database 86
  used by operations 147
  used for 81, 605
  using to trigger events 101
  when an operation fails 90
  workstation fixed resources 81
Special Resources panel 147
specific dates report for calendars 705
specifying
  JCL variable tables 108
  specifying ended in error list criteria panel 594
  specifying filter criteria panel 187
  specifying list criteria 44
  specifying ready list criteria panel 520
Specifying Resource Monitor List Criteria panel 611
Specifying Verification Criteria panel 461
successor and predecessor workstations
  in long-term plan 263
successor dependency resolution in ETT 436
successors
  changing 567
  checking 539
    in the current plan 276
    including 553
    specifying 139
SUNDAY keyword 225
supplied variables 449
SUPPRESS IF LATE option 155, 169
SUPPRESSACTION keyword of JTOPTS 747
Suppressed by condition
  conditional successor status 376
suppressed by condition status 146
suppressing variable substitution 445, 464
Sxxx error code 747
Symphony files 273, 540, 584
  renew 688
syntax diagrams, how to read xxiv
SYSLPARMLIB 195
SYSCHK data set 359
SYSOUT CLASS option 154
SYSOUT data set 167
system automation
  customizing commands 457
  enabling workstation to send commands 74
  specifying information 162
  system automation commands, customizing 457
SYSUDUMP data set 195

T
T extended status code 744
TABLE JCL directive 479
tabular variables 448
tail plan 250
technical training
  See Tivoli technical training
temporary variables 453
terminal type, specifying 39
TEXT command 155
THURSDAY keyword 225
time
  default for an operation 74
  end of a work day 105
  input arrival 158
  modifying time options 568
  specifying format 36
  transport time between workstations 73
TIME DEPENDENT option 155
Time Specifications panel 158
time stamps 301
time zones 37, 301
time-dependent operations 168
Tivoli Business Systems Manager 155, 216
description 619
discovery 622
  process, monitoring 621
  scheduler start options 619
Tivoli Enterprise Portal 155
description 623
  process, monitoring 626
Tivoli technical training xxiii
Tivoli Workload Scheduler (continued)
status of current plan 540
Tivoli Workload Scheduler for z/OS agent workstations 55
  settings 67
  TOD clock, changing 301
  TRACK keyword of JTOPTS 435
track log 276
tracking
  jobs 299
  non-OPC work using ETT 432
tracking jobs
  submitted by CICS or IMS 434
  track log 540
  tracklog 291
training
  See also Tivoli technical training
technical xxiii
transport time 73, 144
TRANSPT keyword 210
trial current plan 272
trial long-term plan 261
triggering event 101, 433
triggering operations 167
TSO commands
  BACKUP 648
  BULKDISC 651
  introduction 647
  JSUACT 653
  OPINFO 655
  OPSTAT 659
  SRSTAT 664
  WSSSTAT 669
TUESDAY keyword 225
TYPE keyword 230
type of resource allocation 613

U
U extended status code 744
U occurrence status code 743
U operation status code 743
UFNAME keyword 237
UFVALUE keyword 237
UN command 532, 577
uncataloging data sets for job recovery 331
UNDECEIVED status 252
Unexpected RC
  job-level condition 380
  step-level condition 388, 389
  step-level dependency 388, 389
UNKNOWN workstation status 584
unwished deletion
  recovery with cleanup 337
update performed for owner ID 714
updating data item panel 187
updating in batch 186
updating the current plan 582
USAGE keyword 231
user data 655
user exit, Ready List layout 522
user-defined variables 454
USERREQ keyword of AROPTS 367
using ISPF 35
utilities 50
utilization of workstation resources
  daily planning report 723
utilization of workstations
  daily planning report 723
Uxxx error code 747

V
V extended status code 744
VALFROM keyword 230
VALFROMMD keyword 240
VALFROMT keyword 240
valid-from date 122
validating JCL 166
validation of variable values 461
VALTO keyword 230
variable dependency 457
variable substitution 441, 461
variable table 108
creating 455
default values 441
global 441
specifying 126, 470, 479
VARSUB keyword of OPCOPTS 444
viewing
  operator instructions 527
virtual workstations 55

W
W extended status code 744
W occurrence status code 743
W operation status code 743
W row command 562
WAIT option
  general workstations 57
Waiting
  conditional successor status 376
  waiting for connection 585
  waiting for manual intervention 585
  waiting for resources 613
  WEDNESDAY keyword 225
  WEEK keyword 226
wildcard characters 46
WLM
  See Workload Manager 493
work days
  cyclic period 111, 130
  end time 22, 104, 105
work run on request 549
work-days-only cyclic period 111
work-days-only cyclic periods 106
WORKDAY keyword 225
workload manager (WLM) function
  selecting assistance policies 493
  setting a job as critical 493
workstation (continued)
  defining 53, 81
  defining reporting attributes 65
description database
  input to current plan 267
destination 73
dummy 61
duration 74
equivalent 17, 18
fixed resources 78, 149
job setup 166
modifying fault-tolerant 583
open intervals 75
parallel servers 71
R1 and R2 resources 78
recovery with cleanup 337
resource utilization report 275
selecting for operation 137
setup 527
splittable 72
status
  active and inactive 584
  checking 539
  offline actions pending 586
  status set by EQQUX009 586
  status set by WSSTAT 586
  status was set manually 586
  waiting for connection 585
  waiting for manual intervention 585
  transport time 73
types 53
WTO 167
workstation fixed resources 81
Workstation Resources and Servers panel 150
workstation type
  remote engine 58
workstations
  utilization 261
  write to operator (WTO)
    deadline 155
    general workstations 57
    operations 167, 295
WSFAILURE keyword 337
WSFAILURE keyword of JTOPTS 351, 747
WSOFFLINE keyword 337
WSOFFLINE keyword of JTOPTS 351, 746
WSSTAT command 584, 669
WTO
  See write to operator (WTO) 295

X
X extended status code 744
xxxx error code 747

Y
Y extended status code 744
YEAR keyword 226

Z
z-centric end-to-end configuration
  scheduling distributed jobs 71
  scheduling dynamic jobs 56
z/OS Remote Job Info panel 165