Note

Before using this information and the product it supports, be sure to read the general information under “Notices” on page xix.

ISO 9001 Certification

This product was developed using an ISO 9001 certified quality system.
Certification has been awarded by the Italian quality system certification group, CSQ (Certification No. CISQ/CSQ 9150.IBM7).
CSQ is a member of the mutually recognized organization of European assessors, ITQS, which assesses and certifies quality systems in the field of information technology enterprises.

Third Edition (December 1999)

This is a major revision of, and obsoletes, SH19-4376-01.

This edition applies to Version 2 Release 3 Modification Level 0 of Tivoli Operations Planning and Control, Program Number 5697-OPC, and to all subsequent releases and modifications until otherwise indicated in new editions or technical newsletters. See the “Summary of Tivoli OPC Version 2 Release 3 Enhancements” on page xxxi for the changes made to this manual. Technical changes or additions to the text to describe the Tivoli Job Scheduling Console Support are indicated by a vertical line to the left of the change. Make sure you are using the correct edition for the level of the product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address below.

IBM welcomes your comments. A form for readers’ comments appears at the back of this publication. If the form has been removed, address your comments to:

Tivoli OPC Information Development
Rome Tivoli Laboratory
IBM Italy S.p.A.
Via Sciangai, 53
00144 Rome
Italy

Fax Number (+39) 06 5966 2077

Internet ID: ROMERCF at VNET.IBM.COM

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1991, 1999. All rights reserved.
Note to U.S. Government Users — Documentation related to restricted rights — Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.
## Contents

**Notices** ................................................................. xix
Programming Interface Information .............................. xx
Trademarks ................................................................. xx

**Preface** .................................................................. xxiii
Who Should Read This Book ........................................... xxiii
Required Product Knowledge .......................................... xxiii
How to Use This Book ................................................... xxiii
Tivoli OPC Publications .................................................. xxiv
  Tivoli OPC Online Books ............................................ xxv
  Online Message Facility ............................................. xxv
Books about Related Products ......................................... xxvi
How to Read the Syntax Diagrams ..................................... xxviii

**Summary of Tivoli OPC Version 2 Release 3 Enhancements** ... xxxi
Job Scheduling Console .................................................. xxxi
Catalog Management — Data Availability ......................... xxxi
OS/390 Workload Manager Support ................................... xxxi
OS/390 Automatic Restart Manager Support ....................... xxxii
Program Interface (PIF) Enhancements .............................. xxxii
Enhancements for Non-OS/390 Tracker Agents ................. xxxii
Usability Enhancements .................................................. xxxiii
New and Changed Installation Exits .................................. xxxiii
New and Changed Initialization Statements ....................... xxxiv
Version 2 Release 2 Summary ........................................... xxxv
Version 2 Release 1 Summary ........................................... xxxviii

**Chapter 1. Introduction to Tivoli OPC** .......................... 1
Implementing a Payroll System ......................................... 1
Designing the Payroll Applications ................................... 3
  Creating Workstations ................................................ 5
  Controlling General Workstations ................................ 7
  Using Servers ........................................................... 7
  Creating Special Resources ......................................... 8
  Creating the Default Calendar ...................................... 10
Preparing the Payroll Jobs ............................................. 11
Creating the Groups, Applications, and Operations ............. 13
Creating Plans ............................................................ 19
What Does Tivoli OPC Do if Jobs or Started Tasks Fail? ....... 20
  Coping with More Complex Problems ............................ 21
Summary of What You Need to Do .................................... 22

**Chapter 2. Creating Workstations** ............................... 23
What Types of Workstation Are There? ............................ 23
  Computer Workstations .............................................. 23
  Job Computer Workstations ....................................... 24
  Started-Task Computer Workstations ............................ 24
  Alternate Workstations ............................................ 24
  Printer Workstations ................................................ 25
  General Workstations .............................................. 25

© Copyright IBM Corp. 1991, 1999
## Chapter 9. Overview of the Long-Term and Current Plans

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Long-Term Plan</td>
<td>166</td>
</tr>
<tr>
<td>Current Plan</td>
<td>168</td>
</tr>
<tr>
<td>Resolving Pending Occurrences</td>
<td>169</td>
</tr>
<tr>
<td>Creating the Plans for the First Time</td>
<td>170</td>
</tr>
<tr>
<td>Deciding What to Do when You Make Changes</td>
<td>173</td>
</tr>
</tbody>
</table>

## Chapter 10. Producing and Modifying the Long-Term Plan

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Long-Term Plan Batch Jobs</td>
<td>176</td>
</tr>
<tr>
<td>The Long-Term Plan Datasets</td>
<td>176</td>
</tr>
<tr>
<td>How Tivoli OPC Creates a Long-Term Plan</td>
<td>177</td>
</tr>
<tr>
<td>Creating the Long-Term Plan</td>
<td>178</td>
</tr>
<tr>
<td>Extending the Long-Term Plan</td>
<td>179</td>
</tr>
<tr>
<td>Modifying the Long-Term Plan in Batch</td>
<td>179</td>
</tr>
<tr>
<td>Creating Reports about the Long-Term Plan</td>
<td>180</td>
</tr>
<tr>
<td>Tivoli OPC-Generated Comments in LTP Reports</td>
<td>181</td>
</tr>
<tr>
<td>Creating a Trial Long-Term Plan</td>
<td>182</td>
</tr>
<tr>
<td>Displaying the Status of the LTP</td>
<td>182</td>
</tr>
<tr>
<td>Setting Default Successor and Predecessor Workstations</td>
<td>183</td>
</tr>
<tr>
<td>Amending Applications in the LTP Online</td>
<td>184</td>
</tr>
<tr>
<td>Modifying Occurrences in the Long-Term Plan</td>
<td>184</td>
</tr>
<tr>
<td>Modifying Dependencies in the Long-Term Plan</td>
<td>185</td>
</tr>
</tbody>
</table>

## Chapter 11. Producing the Current Plan

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Information Used by Daily Planning</td>
<td>187</td>
</tr>
<tr>
<td>Creating the Current Plan</td>
<td>189</td>
</tr>
<tr>
<td>Extending the Current Plan</td>
<td>190</td>
</tr>
<tr>
<td>Recreating the Current Plan</td>
<td>191</td>
</tr>
<tr>
<td>Creating a Trial Current Plan</td>
<td>192</td>
</tr>
<tr>
<td>Producing Reports Using Daily Planning</td>
<td>193</td>
</tr>
<tr>
<td>Plan Reports</td>
<td>193</td>
</tr>
<tr>
<td>Management Reports</td>
<td>193</td>
</tr>
<tr>
<td>Current Period</td>
<td>193</td>
</tr>
<tr>
<td>Previous Period</td>
<td>194</td>
</tr>
<tr>
<td>Using the Track Log</td>
<td>195</td>
</tr>
<tr>
<td>Resolving Dependencies between Operations</td>
<td>195</td>
</tr>
<tr>
<td>Analyzing Problems Reported by Daily Planning</td>
<td>196</td>
</tr>
<tr>
<td>Modifying the Current Plan</td>
<td>198</td>
</tr>
<tr>
<td>Querying the Current Plan</td>
<td>198</td>
</tr>
<tr>
<td>Current Plan Reference Information</td>
<td>199</td>
</tr>
<tr>
<td>Organization and Integrity of the Current Plan</td>
<td>199</td>
</tr>
<tr>
<td>The Current Plan during Normal Tivoli OPC Processing</td>
<td>200</td>
</tr>
<tr>
<td>The Current-plan Backup Process</td>
<td>202</td>
</tr>
<tr>
<td>Creating and Activating the NCP</td>
<td>203</td>
</tr>
</tbody>
</table>

## Chapter 12. Using Tivoli OPC Service Functions

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activating and Deactivating Job Submission</td>
<td>205</td>
</tr>
<tr>
<td>Activating and Deactivating Automatic Job and Started-Task Recovery</td>
<td>206</td>
</tr>
<tr>
<td>Refreshing the Long-term Plan</td>
<td>206</td>
</tr>
<tr>
<td>Refreshing RACF Resource Profiles</td>
<td>207</td>
</tr>
<tr>
<td>Activating and Deactivating ETT</td>
<td>207</td>
</tr>
<tr>
<td>Chapter 13. How Tivoli OPC Selects Work for Automatic Submission</td>
<td>209</td>
</tr>
<tr>
<td>-----------------------------------------------------------------</td>
<td>-----</td>
</tr>
<tr>
<td>Identifying the Order of Submission</td>
<td>209</td>
</tr>
<tr>
<td>The Submission Process</td>
<td>210</td>
</tr>
<tr>
<td>Submitting Work on Nonreporting Workstations</td>
<td>211</td>
</tr>
<tr>
<td>Submitting Work on WTO Workstations</td>
<td>211</td>
</tr>
<tr>
<td>Submitting Started Tasks</td>
<td>212</td>
</tr>
<tr>
<td>Submitting Jobs</td>
<td>213</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 14. MVS Job Tracking Overview</th>
<th>215</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to Make Sure that Events Are Not Lost</td>
<td>216</td>
</tr>
<tr>
<td>Time Zones and Tivoli OPC</td>
<td>217</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 15. Planning for Recovery and Restart</th>
<th>219</th>
</tr>
</thead>
<tbody>
<tr>
<td>What You Need to Run the Functions</td>
<td>220</td>
</tr>
<tr>
<td>What You Need for the Job Completion Checker</td>
<td>220</td>
</tr>
<tr>
<td>Where to Find More Information</td>
<td>220</td>
</tr>
<tr>
<td>What You Need for Job Log Retrieval</td>
<td>220</td>
</tr>
<tr>
<td>Considerations for MVS Job Logs</td>
<td>221</td>
</tr>
<tr>
<td>Considerations for Immediate Retrieval</td>
<td>221</td>
</tr>
<tr>
<td>Considerations for Delayed Retrieval</td>
<td>221</td>
</tr>
<tr>
<td>Consideration for Data Store Delayed Retrieval</td>
<td>221</td>
</tr>
<tr>
<td>Where to Find More Information</td>
<td>222</td>
</tr>
<tr>
<td>What You Need for Step Restart</td>
<td>222</td>
</tr>
<tr>
<td>Where to Find More Information</td>
<td>222</td>
</tr>
<tr>
<td>What You Need for Catalog Management</td>
<td>222</td>
</tr>
<tr>
<td>Where to Find More Information</td>
<td>223</td>
</tr>
<tr>
<td>What You Need for Automatic Recovery</td>
<td>223</td>
</tr>
<tr>
<td>Where to Find More Information</td>
<td>223</td>
</tr>
<tr>
<td>What You Need for the History Function</td>
<td>223</td>
</tr>
<tr>
<td>Where to Find More Information</td>
<td>223</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 16. Setting Error Codes</th>
<th>225</th>
</tr>
</thead>
<tbody>
<tr>
<td>How Tivoli OPC Error Codes Are Set</td>
<td>225</td>
</tr>
<tr>
<td>Setting Error Codes Using Completion Codes</td>
<td>225</td>
</tr>
<tr>
<td>Setting Error Codes Using the Job Completion Checker</td>
<td>226</td>
</tr>
<tr>
<td>Using Error Codes to Set Operations to Ended-in-Error</td>
<td>227</td>
</tr>
<tr>
<td>Resetting Operations Based on Error Codes</td>
<td>227</td>
</tr>
<tr>
<td>Determining the Success or Failure of a Job</td>
<td>227</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 17. Restarting an MVS Job at a Step</th>
<th>229</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selecting the Steps</td>
<td>229</td>
</tr>
<tr>
<td>How Tivoli OPC Gets the Job Log</td>
<td>230</td>
</tr>
<tr>
<td>How Tivoli OPC Tailors the Job</td>
<td>230</td>
</tr>
<tr>
<td>How Tivoli OPC Validates the Job</td>
<td>231</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Chapter 18. Catalog Management</th>
<th>235</th>
</tr>
</thead>
<tbody>
<tr>
<td>Selecting Catalog Management Options</td>
<td>235</td>
</tr>
<tr>
<td>Immediate and Deferred Catalog Management</td>
<td>235</td>
</tr>
<tr>
<td>Job and Step-Level Catalog Management</td>
<td>236</td>
</tr>
<tr>
<td>Catalog Management Exits</td>
<td>236</td>
</tr>
<tr>
<td>When Does Tivoli OPC Perform Catalog Management Actions?</td>
<td>236</td>
</tr>
<tr>
<td>What Action Does Tivoli OPC Take for Affected Datasets?</td>
<td>237</td>
</tr>
</tbody>
</table>
Appendix B. MVS Operator Commands Supported by Tivoli OPC .......................... 337
  Starting Tivoli OPC ........................................................................................................ 337
  Stopping Tivoli OPC ........................................................................................................ 337
  Canceling Tivoli OPC ....................................................................................................... 338
  Modifying Tivoli OPC ..................................................................................................... 338
  Modifying Tivoli OPC Data Store ..................................................................................... 342

Appendix C. Status, Error, and Reason Codes ................................................................. 345
  Occurrence Status Codes ............................................................................................... 345
  Operation Status and Extended Status Codes .............................................................. 345
    Operation Status Codes ............................................................................................... 345
    Extended Status Codes ............................................................................................... 346
  Error Codes ................................................................................................................... 347
  Catalog Management Status Codes .............................................................................. 348
  Job Log Retrieval Status Codes .................................................................................... 349
  Operation Reason Codes ............................................................................................... 349

Appendix D. Batch Programs ............................................................................................. 351
  DD Statements Referenced by Tivoli OPC Batch Jobs .................................................... 351
    DD Statements Used by EQQBATCH ........................................................................... 351
    DD Statements Describing Tivoli OPC Data ............................................................... 352
  Security and Batch Programs ......................................................................................... 352
  Batch Programs and JCL Examples ................................................................................. 353
    EQQADCOPI—Print Applications: Calculate and Print Application Run Days ............ 354
      SYSIN Requirements ................................................................................................. 354
      JCL Example ............................................................................................................. 354
    EQQADDUP—Print Applications: Cross-Reference External Dependencies .................. 355
      SYSIN Requirements ................................................................................................. 355
      JCL Example ............................................................................................................. 355
    EQQADMUP—Perform Mass Update of Application Descriptions ............................... 356
      SYSIN Requirements ................................................................................................. 356
      JCL Example ............................................................................................................. 356
    EQQADPRT—Print Applications: Detailed ................................................................... 357
      SYSIN Requirements ................................................................................................. 357
      Notes ....................................................................................................................... 357
      JCL Example ............................................................................................................. 358
    EQQADXR00—Print Applications: Cross-Reference Different Items ............................ 359
      SYSIN Requirements ................................................................................................. 359
      JCL Example ............................................................................................................. 359
    EQQACLPRC—Print Calendars ..................................................................................... 361
      SYSIN Requirements ................................................................................................. 361
      JCL Example ............................................................................................................. 361
    EQQCLPRP—Print Periods ............................................................................................. 362
      SYSIN Requirements ................................................................................................. 362
      JCL Example ............................................................................................................. 362
    EQQDNTOP—Create or Extend the Current Plan ............................................................. 363
      SYSIN Requirements ................................................................................................. 363
      JCL Example ............................................................................................................. 364
    EQQDOTOP—Print Statistics for Current Planning Period .............................................. 366
      SYSIN Requirements ................................................................................................. 366
      JCL Example ............................................................................................................. 366
Figures

1. Payroll Jobs at Paymore Incorporated ........................................... 2
2. Grouping of Payroll Applications ..................................................... 5
3. EQQWCGEP—Creating General Information about a Workstation .......... 6
4. Creating Workstations for Paymore .................................................. 8
5. Specifying the Payroll Database as a Resource .................................. 9
6. Changing the Default Calendar ....................................................... 10
7. Creating a Variable ........................................................................... 11
8. Creating the PAYDAILY Application .................................................. 13
9. Creating the Run Cycle for PAYDAILY .............................................. 14
10. Creating the Rule for PAYDAILY ..................................................... 15
11. Creating Operations in PAYDAILY ................................................... 15
12. Specifying the Predecessors for PAYDAILY Operations ....................... 16
13. Specifying the Payroll Database as a Resource .................................. 17
14. Specifying that the WTO is Time-dependent ..................................... 17
15. Creating the Long-term Plan ............................................................ 19
16. Listing Occurrences in the Long-Term Plan ..................................... 19
17. Creating the Current Plan ................................................................ 20
18. Dependent Operations with Complex Dependencies ............................ 27
19. Using a Dummy Operation to Simplify Complex Dependencies .......... 27
20. EQQWMLSL—List of Workstation Descriptions .................................. 28
21. EQQWCGEP—Creating General Information about a Workstation .... 29
22. Computer Workstation with Server Usage Set to B or C, Controlling Job Submission ............................................................ 32
23. EQQWMAVL—Availability of a Workstation ...................................... 34
24. EQQWMOTL—Open Time Intervals For One Day ............................... 35
25. EQQWMATL—All Open Time Intervals ............................................. 36
26. EQQWMACL—Modifying All Workstations Closed ............................ 37
27. EQQWMREP—Resources for a Workstation ...................................... 38
28. EQQWAMP—Work Station Access Method Information Panel ............ 39
29. How Resource Quantity and Availability Can Be Changed ................. 47
30. EQQODBSP—Maintaining OPC/ESA Databases ................................. 48
31. EQQQDTOP—Maintaining Special Resources .................................... 48
32. EQQQDLSL—List of Resources ....................................................... 49
33. EQQQDCRP—Creating a Special Resource ....................................... 49
34. EQQQDWML—Modifying Connected Work Stations for a Special Resource ................................................................. 50
35. EQQQDIML—Modifying Intervals for a Special Resource ..................... 51
36. EQQQDWML—Modifying Connected Work Stations for a Special Resource ................................................................. 52
37. EQQTCCAL—Creating a Calendar .................................................... 60
38. EQQTPERL—List of Calendar Periods .............................................. 63
39. EQQTCRPL—Creating a Calendar Period .......................................... 63
40. A 5-day Work-days-only Cyclic Period, MYWEEK ............................ 67
41. How Tivoli OPC Counts Offsets with Work-Days-Only Cyclic Periods .... 68
42. Results When the Work-Days-Only Cyclic Period Origin Is a Free Day ... 68
43. Differences between Applications and Group Definitions .................... 72
44. EQQASUBP—Maintaining Application Descriptions ........................... 75
45. EQQACGPP—Creating an Application .............................................. 76
46. How to Use the RUN and OPER Commands and the GROUP DEFINITION Field ................................................................. 76

© Copyright IBM Corp. 1991, 1999
155. LTP Report—Grand Total Workload for Period .............. 407
156. Daily Planning Reports—General Information ............... 408
158. Daily Planning Reports—Plan for Workstation ............... 410
159. Daily Planning Reports—Workstation Utilization (Parallel Operations) . 411
160. Daily Planning Reports—Workstation Utilization (Resource 1) .... 412
161. Daily Planning Reports—Planned Resource Utilization ....... 413
162. Daily Planning Reports—Summary of Completed Applications . 414
163. Daily Planning Reports—Completed Applications ............. 415
164. Daily Planning Reports—Error Statistics on Completed Applications 416
165. Daily Planning Reports—Operations in Error ............... 417
166. Daily Planning Reports—Missed Feedback Report .......... 417
167. Daily Planning Reports—Actual Resource Utilization ....... 418

Tables

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>The INCLUDE Statement .................................. xl</td>
</tr>
<tr>
<td>2.</td>
<td>The INIT Statement .................................... xl</td>
</tr>
<tr>
<td>3.</td>
<td>Changes to Installation Exits .......................... xli</td>
</tr>
<tr>
<td>4.</td>
<td>Where Values Are Taken from for Each Interval .......... 52</td>
</tr>
<tr>
<td>5.</td>
<td>Symbols that Mark the End of Variables ................ 276</td>
</tr>
<tr>
<td>6.</td>
<td>Occurrence-Related Tivoli OPC-Supplied Variables ....... 281</td>
</tr>
<tr>
<td>7.</td>
<td>Operation-Related Tivoli OPC-Supplied Variables ........ 282</td>
</tr>
<tr>
<td>8.</td>
<td>Date-Related Tivoli OPC-Supplied Variables .............. 283</td>
</tr>
<tr>
<td>9.</td>
<td>Dynamic-Format Date-Related Tivoli OPC-Supplied Variables 283</td>
</tr>
<tr>
<td>10.</td>
<td>Dynamic-Format Substitution Results .................... 300</td>
</tr>
<tr>
<td>11.</td>
<td>Date Formats Allowed in the SETVAR Directive ............ 302</td>
</tr>
<tr>
<td>15.</td>
<td>Tivoli OPC Batch Programs ................................ 353</td>
</tr>
</tbody>
</table>
Notices

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM’s product, program, or service may be used. Subject to IBM’s valid intellectual property or other legally protectable rights, any functionally equivalent product, program, or service may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user’s responsibility.

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

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

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
P.O. Box 12195
3039 Cornwallis
Research Triangle Park, NC 27709-2195
U.S.A.

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

This book is intended to help you customize and tune Tivoli Operations Planning and Control (Tivoli OPC).

This book also documents General-use Programming Interface and Associated Guidance Information, and Diagnosis, Modification, or Tuning Information provided by Tivoli OPC.

General-use programming interfaces allow the customer to write programs that obtain the services of Tivoli OPC.

General-use Programming Interface and Associated Guidance Information is identified where it occurs by an introductory statement to a chapter or section.

Diagnosis, Modification, or Tuning Information is provided to help you customize Tivoli OPC.

Attention: Do not use this Diagnosis, Modification, or Tuning Information as a programming interface.

Diagnosis, Modification, or Tuning Information is identified where it occurs by an introductory statement to a chapter or section.

Trademarks

The following terms are trademarks of Tivoli Systems or IBM Corporation in the United States or other countries or both:

ACF/VTAM  AIX  AIX/6000  
AS/400  BookManager  CICS  
DATABASE 2  DB2  DFSMS/MVS  
DFSMShsm  DFSORT  Extended Services for OS/2  
GDDM  Hiperbatch  Hiperspace  
IBM  IBMLink  IMS  
LoadLeveler  MVS/DFP  MVS/ESA  
MVS/SP  MVS/XA  NetView  
OPC  Operating System/2  OS/2  
OS/390  OS/400  Personal System/2  
PS/2  RACF  RISC System/6000  
RS/6000  SAA  Sysplex Timer  
Systems Application Architecture  Tivoli  TME  
TME 10  VM/ESA  VTAM  
Workplace Shell  

In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S

Microsoft, Windows, Windows NT, and the Windows logo are trademarks or registered trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.
Java and all Java-based trademarks or logos are trademarks of Sun Microsystems, Inc.

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

ActionMedia, LANDesk, MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States and other countries.

Other company, product, and service names which may be denoted by a double asterisk (**), may be trademarks or service marks of others.

DEC (Digital Equipment Corporation)
Hewlett-Packard (Hewlett-Packard Corp.)
HP-UX (Hewlett-Packard Corp.)
Solaris (Sun Microsystems, Inc.)
SPARC (SPARC International, Inc.)
SPARCstation (SPARC International, Inc.)
Sun (Sun Microsystems, Inc.)
SunOS (Sun Microsystems, Inc.)
SunOS (Sun Microsystems, Inc.)
VMS (Digital Equipment Corporation)
Preface

Deciding when your workload runs, and what it depends on, is planning, and is the main subject of this book. It is also called scheduling—the two terms are used interchangeably in OPC/ESA.

Who Should Read This Book

This book is intended for those involved in planning and scheduling the work in the production department of a computer installation. It explains how you can use IBM Operations Planning and Control/ESA (Tivoli OPC) in performing these tasks. Throughout this book, Tivoli Operations Planning and Control is referred to as Tivoli OPC.

Required Product Knowledge

You do not need to know any programming languages to use this book, but you must know the work that you will automate with Tivoli OPC—the jobs and the dependencies between them.

To make the most of Tivoli OPC, work together with the system programmer who installs and customizes Tivoli OPC. Work closely with the job preparation and operations staff that control the work with Tivoli OPC—their tasks will change considerably when Tivoli OPC controls your work, so consider their new tasks, which are described in Tivoli OPC Controlling and Monitoring the Workload.

How to Use This Book

Read Chapter 1, “Introduction to Tivoli OPC” on page 1 for an example of how to implement a batch system using Tivoli OPC. This chapter introduces the features of Tivoli OPC that are described in more detail in Chapter 2, “Creating Workstations” on page 23 through Chapter 11, “Producing the Current Plan” on page 187.

The remaining chapters of this book describe the more advanced features of Tivoli OPC, which you will probably not use for the first systems that you automate. As you get more confidence with Tivoli OPC, take the time to learn about JCL variable substitution, catalog management, automatic recovery, and the other Tivoli OPC functions that improve the automation of your work.
# Tivoli OPC Publications

This book is part of an extensive Tivoli OPC library. These books can help you use Tivoli OPC more effectively:

<table>
<thead>
<tr>
<th>Task</th>
<th>Publication</th>
<th>Order number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Evaluating Tivoli OPC</td>
<td>General Fact Sheet</td>
<td>GH19-4370</td>
</tr>
<tr>
<td>Evaluating Tracker Agents</td>
<td>Tracker Agent Features Fact Sheet</td>
<td>GH19-4371</td>
</tr>
<tr>
<td>Planning Tivoli OPC</td>
<td>Licensed Program Specifications</td>
<td>GH19-4373</td>
</tr>
<tr>
<td>Understanding Tivoli OPC</td>
<td>General Information</td>
<td>GH19-4372</td>
</tr>
<tr>
<td>Learning Tivoli OPC concepts and terminology</td>
<td>Getting Started with Tivoli OPC</td>
<td>SH19-4481</td>
</tr>
<tr>
<td>Using the Java GUI</td>
<td>Tivoli Job Scheduling Console Guide for OPC Users</td>
<td>GC32-0402</td>
</tr>
<tr>
<td>Using the Java GUI</td>
<td>Tivoli Job Scheduling Console Release Notes</td>
<td>GI10-9233</td>
</tr>
<tr>
<td>Interpreting messages and codes</td>
<td>Messages and Codes</td>
<td>SH19-4480</td>
</tr>
<tr>
<td>Installing Tivoli OPC</td>
<td>Installation Guide</td>
<td>SH19-4379</td>
</tr>
<tr>
<td>Customizing and tuning Tivoli OPC</td>
<td>Customization and Tuning</td>
<td>SH19-4380</td>
</tr>
<tr>
<td>Planning and scheduling the workload</td>
<td>Planning and Scheduling the Workload</td>
<td>SH19-4376</td>
</tr>
<tr>
<td>Controlling and monitoring the current plan</td>
<td>Controlling and Monitoring the Workload</td>
<td>SH19-4377</td>
</tr>
<tr>
<td>Using Workload Monitor/2</td>
<td>Workload Monitor/2 User’s Guide</td>
<td>SH19-4482</td>
</tr>
<tr>
<td>Writing application programs</td>
<td>Programming Interfaces</td>
<td>SH19-4378</td>
</tr>
<tr>
<td>Quick reference</td>
<td>Quick Reference</td>
<td>GH19-4374</td>
</tr>
<tr>
<td>Diagnosing failures</td>
<td>Diagnosis Guide and Reference</td>
<td>LY19-6405</td>
</tr>
<tr>
<td>Controlling the AIX, UNIX**, VMS, OS/390 Open Edition workload</td>
<td>Tracker Agents for AIX, UNIX, VMS, OS/390 Open Edition Installation and Operation</td>
<td>SH19-4484</td>
</tr>
<tr>
<td>Controlling the OS/2 and NT workload</td>
<td>Tracker Agents for OS/2 and Windows NT Installation and Operation</td>
<td>SH19-4483</td>
</tr>
<tr>
<td>Controlling the OS/400 workload</td>
<td>Tracker Agent for OS/400 Installation and Operation</td>
<td>SH19-4485</td>
</tr>
</tbody>
</table>

A Master Index, SH19-4375, is published for the Tivoli OPC library.

Maximizing Your OPC Throughput, SG24-2130, contains useful information for tuning the OPC installation.
Tivoli OPC Online Books

All the books in the Tivoli OPC library, except the licensed publications, are available in displayable softcopy form on CD-ROM in the following Softcopy Collection Kit:

- OS/390, SK2T-6700

You can read the softcopy books on CD-ROMs using these IBM licensed programs:

- BookManager READ/2 (program number 5601-454)
- BookManager READ/DOS (program number 5601-453)
- BookManager READ/6000 (program number 5765-086)

All the BookManager programs need a personal computer equipped with a CD-ROM disk drive (capable of reading disks formatted in the ISO 9660 standard) and a matching adapter and cable. For additional hardware and software information, refer to the documentation for the specific BookManager product you are using.

Updates to books between releases are provided in softcopy only.

Online Message Facility

The Online Message Facility (OMF) is an OS/2 program that provides online access to information from BookManager softcopy books. It helps you diagnose problems without interrupting your work. You can retrieve the description of a message by clicking on a message number in a Communications Manager emulator window. Additional information about OMF is available on the Messages and Codes CD-ROM.
### Books about Related Products

These publications are referred to in the text or contain information that you might need:

<table>
<thead>
<tr>
<th>Short title</th>
<th>Publication</th>
<th>Order number</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DFP Services for ICF</strong></td>
<td>MVS/DFP: Access Method Services for the Integrated Catalog Facility</td>
<td>SC26-4562</td>
</tr>
<tr>
<td></td>
<td>DFSMS/MVS Version 1 Access Method Services for the Integrated Catalog Facility</td>
<td>SC26-4905</td>
</tr>
<tr>
<td><strong>DFP Utilities</strong></td>
<td>MVS/DFP: Utilities</td>
<td>SC26-4559</td>
</tr>
<tr>
<td></td>
<td>DFSMS/MVS Version 1 Utilities</td>
<td>SC26-4926</td>
</tr>
<tr>
<td><strong>Performance Reporter for MVS</strong></td>
<td>Performance Reporter for MVS Fact Sheets</td>
<td>GBOF-6022</td>
</tr>
<tr>
<td><strong>GDDM User’s Guide</strong></td>
<td>Graphical Data Display Manager Guide for Users</td>
<td>SC33-0327</td>
</tr>
<tr>
<td></td>
<td>Graphical Data Display Manager Version 3 User’s Guide</td>
<td>SC33-0875</td>
</tr>
<tr>
<td><strong>Info Planning and Installation</strong></td>
<td>Information/Management Version 6 Planning and Installation Guide and Reference</td>
<td>SC34-4455</td>
</tr>
<tr>
<td><strong>ISPF Reference Summary</strong></td>
<td>ISPF/PDF Version 3 MVS Reference Summary</td>
<td>SC34-4214</td>
</tr>
<tr>
<td></td>
<td>ISPF Version 4 Reference Summary</td>
<td>SC34-4445</td>
</tr>
<tr>
<td><strong>ISPF Planning and Customization</strong></td>
<td>ISPF and ISPF/PDF Planning and Customization</td>
<td>SC34-4257</td>
</tr>
<tr>
<td></td>
<td>ISPF Version 4 Planning and Customization</td>
<td>SC34-4433</td>
</tr>
<tr>
<td><strong>ISPF/PDF Guide and Reference</strong></td>
<td>ISPF/PDF Guide and Reference</td>
<td>SC34-4258</td>
</tr>
<tr>
<td></td>
<td>ISPF Version 4 User’s Guide</td>
<td>SC34-4484</td>
</tr>
<tr>
<td><strong>ISPF Examples</strong></td>
<td>ISPF Dialog Management Services Examples</td>
<td>SC34-4215</td>
</tr>
<tr>
<td></td>
<td>ISPF Version 4 Examples</td>
<td>SC34-4415</td>
</tr>
<tr>
<td><strong>JCL Reference</strong></td>
<td>MVS JCL Reference</td>
<td>GC28-1654</td>
</tr>
<tr>
<td></td>
<td>MVS SP5 JCL Reference</td>
<td>GC28-1479</td>
</tr>
<tr>
<td></td>
<td>MVS SP5 JCL User’s Guide</td>
<td>GC28-1473</td>
</tr>
<tr>
<td><strong>JES2 Commands</strong></td>
<td>MVS/ESA JES2 commands</td>
<td>GC23-0084</td>
</tr>
<tr>
<td></td>
<td>MVS/ESA SP5 JES2 commands</td>
<td>GC23-1443</td>
</tr>
<tr>
<td><strong>JES3 Commands</strong></td>
<td>MVS/ESA JES3 commands</td>
<td>GC23-0090</td>
</tr>
<tr>
<td></td>
<td>MVS/ESA SP5 JES3 commands</td>
<td>SC28-1444</td>
</tr>
<tr>
<td><strong>RACF Command Reference</strong></td>
<td>Resource Access Control Facility Command Language Reference</td>
<td>SC28-0733</td>
</tr>
<tr>
<td><strong>System Commands</strong></td>
<td>MVS/ESA System commands</td>
<td>GC23-1626</td>
</tr>
<tr>
<td></td>
<td>MVS/ESA SP5 System commands</td>
<td>GC23-1442</td>
</tr>
<tr>
<td><strong>TSO/E Commands</strong></td>
<td>TSO Extensions Command Language Reference</td>
<td>SC28-1307</td>
</tr>
<tr>
<td>Short title</td>
<td>Publication</td>
<td>Order number</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Maestro Supplemental Documentation Set</td>
<td>Unison Maestro Supplemental Documentation Set</td>
<td>SK3T-3566</td>
</tr>
</tbody>
</table>
How to Read the Syntax Diagrams

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

```
SRSTAT command
  ┌─────────────┐
  │             │
  │ SRSTAT ─            resource name ─            │
  │ ────────────┐    ────────────┐
  │            │        │            │
  │ SUBSYS(     │        │ AVAIL(      │
  │          subsystem name ─        │ KEEP                    │
  │        ┌┐─OPCA ──┬ ┬─KEEP─ │ NO ──┐─ SUBSYS( ─ ┘─RESET─ ──        │
  │        └┘ ──MSTR ──┼ ┼─AVAIL─ ─ ┘ ──RESET─── ─ └ ┘ ──YES─── ──        │
  │ ────────────┐    ────────────┘
  │            │        │            │
  │ DEVIATION(  │        │ QUANTITY(   │
  │          amount ─        │ KEEP                   │
  │        ┌┐─KEEP ──┬ ┬─KEEP─ ─ ┘ ──QUANTITY ─ ──        │
  │        └┘ ──DEVIATION ─ ┼ ┼─RESET─ ─ ┘ ──RESET─── ──        │
  │ ────────────┐    ────────────┘
  │            │        │            │
  │ CREATE(     │        │ TRACE(     │
  │          YES ─        │ KEEP ─        │ trace level ─
  │        ┌┐─KEEP ──┴ ┴─KEEP─ ─ ┘ ──QUANTITY ─ ──        │
  │        └┘ ──DEVIATION ─ ┵ ┵ ──QUANTITY ─ ─ ┘ ──RESET─── ──        │
  │ ────────────┐    ────────────┘
  │            │        │            │
  │            │        │            │
  │            │        │            │
  │ ────────────┘
```

The symbols have these meanings:

- The statement begins here.
- The statement is continued on the next line.
- The statement is continued from a previous line.
- The statement ends here.

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:
    
    ![Diagram](image1)

  - If choosing one of the items is optional, the entire stack appears below the main path:
    
    ![Diagram](image2)

  - A repeat arrow above a stack indicates that you can make more than one choice from the stacked items:
    
    ![Diagram](image3)

- Parameters that are above the main line are default parameters:
  
  ![Diagram](image4)

- 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:
  
  ![Diagram](image5)

  **option 1:**

  ![Diagram](image6)

  **option 2:**

  ![Diagram](image7)
Summary of Tivoli OPC Version 2 Release 3 Enhancements

Job Scheduling Console

The new Tivoli Job Scheduling Console (JSC) is a Java-based, client/server application. The key advantages of the JSC are the ability to perform administration and operation tasks in a graphical manner and the ability to access multiple OPC controllers from a single console.

The JSC can:

- Display lists of objects already defined to OPC, from the database and from the current plan, by using flexible filtering criteria
- Work with application descriptions including jobs and their dependencies, time restrictions (input arrival time, deadline, duration), and run cycles
- Work with special resource and workstation definitions
- Modify occurrences, workstation status, and special resource information from the current plan.

The JSC retains the OPC security model. Each data access request is validated by the controller as it is done currently for ISPF users.

The JSC is a real-time interface with OPC and can be used concurrently with the ISPF interface. It is available for various UNIX platforms, Windows NT, and Windows 98. The OPC Connector, which is a backend component supporting the JSC, is available for various UNIX platforms and Windows NT.

Catalog Management — Data Availability

The new Catalog Management – Data Availability feature improves OPC performance for job restart and job log retrieval functions. Job runtime information, for example, the sysout datasets, is maintained locally on the tracked system. The controller retrieves this information only when needed for catalog management actions, eliminating the network and processing overhead associated with the transmission of superfluous data. The runtime information at the tracked system is managed by a new component, the OPC Data Store. Using the OPC Data Store, OPC Tracker processes are bypassed and are dedicated to the time-critical job submission and tracking tasks. A new feature is provided to selectively determine how long job runtime information is kept in the Data Store. This new feature is especially useful when a joblog archiving product is used concurrently with OPC.

OS/390 Workload Manager Support

OS/390 Workload Manager, when used in goal mode, provides a new, policy-based management of deadlines for critical jobs. Some CPU-type operations can now be marked as critical in OPC. When such a critical operation is late, according to the specified policy, OPC interfaces with Workload Manager to move the associated job to a higher performance service class. Thus the job receives appropriate additional system resource to reduce or eliminate the delay. Several policies are available to
decide when a job is late, considering characteristics such as duration, deadline time, and latest start time.

**OS/390 Automatic Restart Manager Support**

OS/390 Automatic Restart Manager increases the availability of OPC components. In the event of program failure, OPC components, for example, the Controller, the OS/390 Tracker and the Server can now be restarted automatically by the Automatic Restart Manager.

**Program Interface (PIF) Enhancements**

The Program Interface (PIF) has been extended to increase the flexibility of OPC, allowing users to have extended access to OPC data from other application programs. Tivoli OPC Version 2 Release 3 significantly enhances the ability to access current plan data from the PIF by providing:

- Full support for special resources data
- Read access to special resource usage information for operations
- The ability to modify the workstation open intervals
- The ability to modify the successor information for an operation.

New resource codes have been added to the Program Interface (PIF):

- **CPOPSRU**: Current plan operation segment with information for the operation in relation to a special resource
- **CPSUC**: Current plan successor segment
- **CSR**: Current plan special resources
- **CSRCOM**: Current plan special resource common segment
- **IVL**: Current plan workstation interval segment

**Enhancements for Non-OS/390 Tracker Agents**

The OPC Tracker Agents for non-OS/390 platforms have been enhanced:

- A new version of the OPC Tracker Agent for OpenVMS is available. This new version runs in the native OpenVMS environment, thus removing the requirement to install the POSIX shell.
- The security features for the UNIX OPC Tracker Agents have been enhanced. Stricter file permissions are now used for temporary work files.
- The installation process of the OPC Tracker Agent for OS/390 UNIX System Services has been simplified.
Usability Enhancements

New features increase the overall usability of the product, thus increasing user productivity:

- OPC can perform variable substitution within inline procedures, thus increasing the flexibility of the job setup feature. It is possible to customize OPC so that jobs are submitted also when variables are not defined in the OPC variable tables. This means that, when variables are substituted outside OPC, duplicate variable definitions are avoided.

- During Catalog Management actions, OPC can delete datasets with an expiration date.

- A new Modify command (JSUACT) has been provided to start or stop the job submission function. This feature enables automation products, for example, Tivoli NetView to have control over the OPC job submission activity.

- The Mass Update utility has been enhanced with a new sample job. This downloads all the applications belonging to a group in a sequential file for use as input to the Batch Loader utility, thus easing the management of group applications from the batch administration.

- The sample library now contains the DSECT sections for the Program Interface (PIF) data areas. This eases the process of writing PIF applications and the migration of existing PIF applications to new OPC releases.

New and Changed Installation Exits

User exit EQQUX001 has three new parameters:

**NEWREC**
- Number of JCL lines in new JCLAREA

**NEWJCL**
- New JCLAREA

**USDREC**
- Number of JCL lines used in new JCLAREA

User exit EQQUX007 has the new extended status (PEXSTAT) as part of its parameters set.

The Job Submission exit (installation exit 1) now allows changes to the size of JCL being processed. This enhancement gives users more flexibility to customize their operating environment.

The Operation Status Change exit (installation exit 7) has been enhanced to receive extended status information. This means that full status information is available within this exit to allow more detailed processing.

The samples set has two new samples: EQQCMX01 and EQQCMX05.
New and Changed Initialization Statements

Two initialization statements have been added to enhance the JCL variable substitution:

**VARFAIL**
If VARFAIL is specified, JCL variable substitution error is bypassed for the specified types and variables are left unresolved in the submitted JCL.

**VARPROC**
Specifies whether or not the variables must be resolved also in the inline procedures.

Three initialization statements have been added to handle the OPC Data Store options:

**FLOPTS**
Defines the options for the FL (Fetch Job Log) task. A Controller uses this statement when OPCOPTS DSTTASK (YES) is specified.

**DSTOPTS**
Specifies options for the OPC Data Store.

**DSTUTIL**
Specifies options for the Data Store batch utilities and the clean up subtask.

Parameters have been added to, or changed in, the JOBOPTS statement so as to handle the new Data Store options:

**JOBLOGRETRIEVAL**
A new value DELAYEDST has been added to this keyword for specifying that the job log is to be retrieved by means of the OPC Data Store.

**DSTCLASS**
A new parameter to define the reserved held class that is to be used by the OPC Data Store associated with this tracker.

**DSTFILTER**
A new parameter to specify if the job-completion checker (JCC) requeues to the reserved Data Store classes only the sysouts belonging to these classes.

Parameters have been added to, or changed in, the OPCOPTS statement so as to handle the new catalog management functions:

**DSTTASK**
Specifies whether or not the OPC Data Store is to be used.

**JCCTASK**
A new DST value has been added to specify if the JCC function is not needed, but the Data Store is used.

A parameter has been added to the OPCOPTS and the SERVOPTS statements:

**ARM**
Activates automatic restart (via the Automatic Restart Manager) of a failed OPC component.

A parameter has been added to the OPCOPTS statement for the Workload Manager (WLM) support:
WLM
Defines the WLM options, that is, the generic profile for a critical job. The profile contains the WLM service class and policy.

Version 2 Release 2 Summary

Instrumentation for Tivoli Global Enterprise Manager
Tivoli Global Enterprise Manager (GEM) is the industry's first solution for unifying the management of cross-platform business applications that run businesses and make them competitive. Tivoli GEM helps you to manage strategic applications from a unique business systems perspective, focusing your IT resources on keeping these systems working properly and productively. Tivoli OPC has been enhanced to support the Job Scheduling Business System of the Tivoli GEM Systems Management Business System. From the Tivoli GEM console, which provides a single point of management, a Tivoli OPC user has complete control of all the Tivoli OPC components, regardless of the platform on which they run. In more detail, the Tivoli OPC instrumentation for Tivoli GEM enables you to do the following:

- Show all the Tivoli OPC components, including controllers, stand-by controllers, OS/390 trackers, AS/400 tracker agents, TCP/IP connected tracker agents.
- Show the different links between the above components. This provides, at a glance, a check on the health of the connections. For example, an OS/390 tracker might be running but might have no connection to the controller.
- For each component, manage a set of status parameters (monitors) specific to that component. These monitors might report the status of some vital OPC controller data sets such as database, current plan, and long-term plan.
- Manage this set of monitors graphically. You can:
  - Ask for value of the monitor
  - Be notified when the value of the monitor changes
  - Associate a severity (such as normal, warning, severe, or critical) with each monitor value
- Start or stop Tivoli OPC trackers without logging them on.
- Know at a glance, in a sysplex environment, which is the active controller and which the stand-by.
- Execute commands on Tivoli OPC components, from a single point of control, regardless of the platform and operating system used for that component.

SAP R/3 support
Tivoli OPC has been enhanced to exploit the Extended Agent technology of the Tivoli Workload Scheduler product. This technology enables Tivoli OPC to interface with a number of third party applications that can perform scheduling. By using this technology, you can now start and track a SAP R/3 job from Tivoli OPC. You can also retrieve and display the job log at the Tivoli OPC controller. This function requires the Tivoli OPC Tracker Agent for one of the following platforms:

- AIX
TCP/IP communication improvements
The TCP/IP communication component that enables the controller to communicate with the TCP/IP connected tracker agents has been restructured to use the standard TCP/IP C–Socket interface. This change enables Tivoli OPC for the latest OS/390 releases and provides for the use of the standard TCP/IP features, such as the KEEPALIVE option.

Catalog management enhancements
The logic that Tivoli OPC uses when determining which catalog management actions to perform has been extended to manage the following situations:

- Some steps in a job are not executed, but are flushed. The datasets referred to in those steps are ignored by the catalog management function.
- A dataset referred to with disposition NEW in one step is also referred to in other steps. Logic to determine the action to perform in these cases has been added to the Catalog Management function.

Dataset Delete function (EQQDELD) improvements
The Dataset Delete function has been enhanced to determine the correct action when a dataset referred to with disposition NEW in one step is also referred to in other steps. Logic to determine the correct action to perform in these cases has been added to the Dataset Delete function. The Dataset Delete function has also been improved to do the following:

- Delete datasets for which an expiration date was specified.
- Issue diagnostic information when the IDCAMS DELETE command or the DFHSM ARCHDEL command fails to delete a dataset.

Current plan occurrence limit removal
The maximum number of occurrences in the current plan has been increased from 32767 to 9999999. This enhancement enables you to manage the current plan more flexibly when you have large workloads.

Operations in AD limit removal
You can now define up to 255 operations in each Application Description. This enhancement provides for more flexibility in the definition of the workload.

AD and OI consistency check
The consistency between the Application Description and the Operator Instruction OPC databases is now enforced by OPC. For instance, whenever an operation is deleted the associated operator instructions is also deleted. Some usability enhancements have also been implemented in the Application Description dialogs when defining operator instructions. For instance, you can now also access temporary operator instructions.

JCL editing from Application Description dialogs
You can now customize the Tivoli OPC dialogs so that a library management application used in the customer's environment to manage the production jobs can be invoked from the Application Description OPC dialogs, thus increasing user productivity. New row commands have been added to
invoke such an application from the Operation List panel while working with an Application Description.

**OPC Control Language tool**
The OPC Control Language (OCL) tool enables you to access and manipulate Tivoli OPC data by using a REXX-like language. Several macro-functions are made available that perform, in a single action, what would require several invocations of the OPC Program Interface functions. The OCL tool acts as an extension to the REXX language processor. Therefore, normal REXX statements can be coded together with OCL statements. This tool runs in a batch TSO session.

**Tracker agents**
New Tracker Agents are provided to control the workload on:
- Digital UNIX
- OS/390 Open Edition

**SmartBatch coexistence**
Tivoli OPC has been extensively tested to make sure that all the features continue to work correctly when the production workload is under SmartBatch control.

**Other enhancements to functions**
- **EQQZSUBX 16 MB limit removal**: because it is no longer necessary to move the JCL buffer below the 16 MB line before submitting it to JES2 or JES3, this processing has been removed from Tivoli OPC.
- To improve the robustness of Tivoli OPC, the STIMERM macro is now invoked, wherever the STIMER macro was previously invoked.
- Tivoli OPC Job-Submit user exit (EQQUX001) has been improved by adding two new parameters: WorkstationID and ErrorCode. When ErrorCode is set, Tivoli OPC will not submit the job.
- Tivoli OPC Operation-Status-Change user exit (EQQUX007) has been improved by adding the procstep name to the JOBAREA parameter. This enhancement provides for fully automated problem management.
- Debugging aids for performance problems: new statistics are now produced by Tivoli OPC to trace all the activities performed during the job submission process. These statistics are especially useful when you tune your systems to maximize job throughput in Tivoli OPC. You can dynamically activate and deactivate these statistics by means of new MODIFY commands.

**New and changed installation exits**
User exit EQQUX001 has two new parameters:
- **RETCO** The error code
- **WSNAME** The workstation name of submission process

User Exit EQQUX007 has a new field in the JobArea called procedure step name.

**Changes to commands**
The following modify commands have been added:
- **CPQSTA** Activates or deactivates CP lock statistic messaging
- **EVELIM** Sets a new value for the EVELIM keyword of the JTOPTS statement
EVESTA       Activates or deactivates EVENT statistic messaging
GENSTA       Activates or deactivates GS task statistic messaging
HB           Issues a heartbeat message for an OPC controller or for all trackers connected to that controller
JCLDBG       Activates or deactivates the JCL debugging trace
QUELEN       Sets a new value for the QUEUELEN keyword of the JTOPTS statement
STATIM       Sets a new value for the STATIM keyword of the JTOPTS statement
STATUS       Returns status information about the OPC controller and the tracker agents connected to it.
WSASTA       Activates or deactivates WSA task statistic messaging

New and changed initialization statements
The following values have been added to the STATMSG keyword of the JTOPTS statement:

EVELIM       Makes customizable the event number criterion for statistic messaging.
STATIM       Uses an interval time criterion to issue statistics messaging.
WSATASK      Activates new statistics for WSA task.

The following new values have been added to the SUBFAILACTION keyword of the JTOPTS statement:

XC, XE and XR
To specify how OPC must handle values returned by the Job Submission Exit (EQQUX001) for the RETCO parameter.

A new keyword has been added to the BATCHOPT statement:

MAXOCCNUM    Set the maximum number of occurrences in the current plan for the daily planning function.

A new keyword has been added to the JTOPTS statement:

MAXOCCNUM    Set the maximum number of occurrences in the Current Plan for the dialog, ETT, Automatic Recovery and PIF functions.

Changes to programming interfaces
The OPC Programming Interface (PIF) has been extended as follows:

- A new subsegment has been added to the Workstation record called the Workstation Access Method Information (WSAM).
- A new keyword, ADOICHK, has been added to the OPTIONS request to activate the consistency check between Application Description and Operator Instruction records.

Version 2 Release 1 Summary
Tivoli OPC Version 2 Release 1 became generally available in March 1997. Major enhancements compared to OPC/ESA Release 3.1 are described in the following sections.

Tracker agents
New Tracker Agents are provided to control the workload on:

- Digital OpenVMS
• Pyramid MIPS ABI

Shared parm library in Sysplex environment
MVS controllers and trackers can share common controller and tracker initialization statements and started task JCLs, making it easier to install many OPC subsystems inside an MVS/ESA sysplex environment.

Controller configuration in Sysplex environment
Tivoli OPC support of MVS/ESA sysplex (base and parallel) has been extended to enable any one of many cloned controllers on different MVS images to switch from standby to active status. An OPC controller is started on each member of the XCF group. The first potential controller that becomes active is the active controller and the others are standby controllers. If the active controller fails, a standby controller running in another MVS/ESA image of the sysplex environment takes over automatically as the active controller.

Single system image
This enhancement allows OPC TSO dialog users and PIF users to be on a different MVS/ESA image from the OPC controller. Dialog users and PIF applications can also be on MVS systems outside the sysplex where the controller runs. Remote communication is based on APPC.

Extended dialog filter
The dialog filter has been extended to allow more specific search arguments and to define the interpretation of wildcard characters.

Reparseing of NOERROR
New operator commands allow the operator to dynamically update the NOERROR table using the NOERROR initialization statements defined by the OPC PARMLIB member, and to read the statements from a member of the EQQPARM DD concatenated libraries. In addition a new initialization statement allows the inclusion of NOERROR statements from members of the EQQPARM DD concatenated libraries.

PIF extension
Program Interface has been greatly extended to support almost all OPC database record types.

Job tracking log
This enhancement provides to user exit 11 job tracking records on which an effective disaster recovery procedure can be based. The customer through exit 11 receives job tracking records, and can send this data to a remote controller that, in case of failure of the active controller, will take over as controller.

GMT clock support improvement
The GMTOFFSET keyword in the OPCOPTS statement lets the user define an offset between the GMT time set in the MVS system and the actual GMT time. The OPC controller uses the GMT clock to validate an OPC Tracker Agent trying to connect; this improvement addresses the need of some users to have the MVS GMT clock independent of the actual GMT time, while keeping the ability to use Tracker Agents.

Batch command interface tool
A batch command interface tool is supplied to perform most of the actions supported by the PIF interface by means of a batch command interface.
New and changed initialization statements

Initialization statements have been added and changed in Tivoli OPC Version 2. The following sections summarize the differences.

The INCLUDE statement

Added in Tivoli OPC Version 2, the INCLUDE statement lets you reduce the size of the parameter library member that contains the OPCOPTS and JTOPTS statements and reduce the associated maintenance activities.

Table 1. The INCLUDE Statement

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NOERROR</td>
<td>Specifies to read NOERROR information from other members of the EQQPARM library.</td>
</tr>
</tbody>
</table>

The INIT statement

Added in OPC/ESA Release 3.1, the INIT statement lets you define run-time options for processing requests from a PIF application. These settings override the values set by the INTFOPTS statement in EQQPARM. The statement is defined in a second parameter file that is identified by the EQQYPARM DD statement in the JCL procedure of the PIF application. In Tivoli OPC Version 2 the LUNAME keyword has been added.

Table 2. The INIT Statement

<table>
<thead>
<tr>
<th>Keyword</th>
<th>Short description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CWBASE</td>
<td>Specifies the origin for the century window used by the PIF application.</td>
</tr>
<tr>
<td>HIGHDATE</td>
<td>Specifies the high date presented to the PIF application in valid-to fields.</td>
</tr>
<tr>
<td>LUNAME</td>
<td>Specifies a server or controller LU name for the PIF application.</td>
</tr>
<tr>
<td>SUBSYS</td>
<td>Identifies the Tivoli OPC subsystem controller.</td>
</tr>
<tr>
<td>TRACE</td>
<td>Specifies the level of trace information to write to the diagnostic file.</td>
</tr>
</tbody>
</table>

Changes to commands

These modify commands have been added:

- **NEWNOERR** Requests that the NOERROR statements be reprocessed.
- **NOERRMEM (member)** Requests that the NOERROR information be read from the specified member.

The MODIFY command has been extended to accept stop and start of the server started tasks:

- **F ssname, P=SERV**
- **S ssname, P=SERV**
Changes to programming interfaces
The Programming Interface is extended as follows:

UPDATE is supported for calendars, periods, workstations, and all workstations closed.

BROWSE and UPDATE are supported for ETT and special resources.

The LIST request has been extended to support a new keyword, MATCHTYP, to specify whether generic search arguments (* and % are to be treated as normal characters.

A new keyword, ADVERS, has been added to the OPTIONS request, to activate the support of AD versioning.

New and changed installation exits
Table 3 summarizes the changes to installation exits in Tivoli OPC Version 2.

Table 3. Changes to Installation Exits

<table>
<thead>
<tr>
<th>Exit name</th>
<th>Short description of change</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQQUX001</td>
<td>Tivoli OPC Version 2 now also supports the addressing modes RMODE(24) and AMODE(31).</td>
</tr>
<tr>
<td>EQQUX011</td>
<td>Sample job tracking log write exit.</td>
</tr>
</tbody>
</table>

Messages
Messages have been changed, deleted, and added in Tivoli OPC Version 2. Refer to Tivoli OPC Messages and Codes for the complete message text and descriptions. Note that in Version 2 the message text and explanations refer to the product as OPC/ESA.
Chapter 1. Introduction to Tivoli OPC

This chapter introduces you to Tivoli OPC by describing the steps that you would take to run a payroll system under the control of Tivoli OPC.

If you are the Tivoli OPC scheduler, 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 OPC can reduce manual work and can handle the exceptions as well as the routine.

2 Specify your data processing environment to Tivoli OPC.

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 datasets, tape drives, and initiators
   e. How the JCL is processed before each job is submitted
   f. What actions Tivoli OPC must take if each job fails

5 Build the high-level schedule, which is called the long-term plan (LTP). The LTP 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.

You will find many terms that are new to you, but by the time you finish this chapter, you should understand the main building blocks of Tivoli OPC and the way they fit together. Each step is explained in outline, without the details that belong to the chapters that follow.

Do not try to understand every field on the panels just yet; that belongs to the later chapters. You can work through the chapter using your Tivoli OPC system. You will not break anything, though you might want to recreate your databases before you specify your own systems.

Implementing a Payroll System

To help you understand Tivoli OPC and plan the definition of your systems to Tivoli OPC, this book bases its examples on a fictional corporation, Paymore Incorporated, that has a system running MVS, and is converting its applications to run under the control of Tivoli OPC. It first converts the payroll system.
The payroll analyst describes the process for the Tivoli OPC 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 dataset 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 dataset. 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.
When PAYWEEK completes successfully, the job PAYWSLIP runs to print pay slips. It includes these programs:

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

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.

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.

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

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 may 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 OPC, there is a good chance that you can automate recovery, so consider all procedures, even manual ones. One of the benefits of implementing Tivoli OPC 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

Figure 2 on page 5 shows how the payroll jobs can be grouped to run under Tivoli OPC. Here are other problems to solve:

How do you get PAYDAILY to run after the CICSA dataset 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 dataset 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 dataset. Include Tivoli OPC-supplied code in your IEFU83 SMF exit to generate a special resource event when the dataset is closed, allowing PAYDAILY to start. Refer to Customization and Tuning for information on dataset 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.
Trigger the CICS transaction with a WTO operation, but use an automatic reporting WTO general workstation (see “Specifying Workstation Reporting Attributes” on page 30) 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.

How do you automate the running of the CICS transaction?
Use Tivoli OPC 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 dataset.

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 dataset?
Include a final WTO operation in PAYBACKP that triggers, via NetView, a CICS transaction that reopens the payroll dataset 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 OPC 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 OPC terms, this means that an occurrence of the PAYQUERY application is added to the schedule, or current plan, by an operator using the Tivoli OPC Modify Current Plan dialog. 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 OPC waits until the allocating operation has freed it.
Creating Workstations

Specify your environment to Tivoli OPC in terms of workstations and resources. A workstation is not necessarily hardware. It is a stage in the processing that is controlled by Tivoli OPC. The processor complex that runs your Tivoli OPC 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 run-time parameters on request. Often a run-time parameter is just the day's date. Why can the program not simply get the date from the time-of-day clock? Sometimes the date must be when the group of jobs starts, and must not change, even if the jobs 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 OPC is usually able to substitute the correct date into the jobs automatically. Where manual intervention is unavoidable, Tivoli OPC can control when the job is available for variable substitution and submission. Tivoli OPC 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 OPC is notified when a print output group has finished printing, so Tivoli OPC 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. Specify a print operation only when you want to track it.

Briefly, 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 OPC what is going on, but you can have a terminal where the operator changes the Tivoli OPC status of an operation through the ISPF dialog. 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 OPC of the new status, and Tivoli OPC 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
- A computer workstation for started tasks, if CICSA is a started task and you want it to run under Tivoli OPC
- A general workstation for job preparation
- A printer workstation
- A general workstation for the payroll office, if you need to keep track of manual operations there, such as the decollation of pay slips and preparation of pay packets

See Chapter 2, "Creating Workstations" on page 23 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 Tivoli OPC main menu) to list workstations. The command CREATE displays this panel:

```
EQQWCSE Ep -----> CREATING GENERAL INFORMATION ABOUT A WORK STATION "---------- Command ===> Enter the command R for resources or A for availability or M for access method above, or enter/change data below:
WORK STATION NAME ===> CPU1
DESCRIPTION ===> Local system processing
WORK STATION TYPE ===> C General, C Computer, P Printer
REPORTING ATTR ===> A A Automatic, S Manual start and completion
C Completion only, N Non reporting
PRINTOUT ROUTING ===> SYSPRINT The ddname of daily plan printout data set
SERVER USAGE ===> B Parallel server usage, C, P, B or N
Options:
SPLITTABLE ===> N Interruption of operation allowed, Y or N
JOB SETUP ===> N Editing of JCL allowed, Y or N
STARTED TASK, STC ===> N Started task support, Y or N
WTO ===> N Automatic WTO, Y or N
DESTINATION ===> ________ Name of destination
Defaults:
TRANSPORT TIME ===> 00.00 Time from previous workstation HH.MM
DURATION ===> 00.05 Duration for a normal operation HH.MM
```

**Figure 3. EQQWCSE Ep—Creating General Information about a Workstation**

Figure 3 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 \texttt{R} command to specify workstation fixed resources, called R1 and R2. In Release 3 of Tivoli OPC, these fixed resources are less useful, because you can do more with special resources, which are described in Chapter 3, “Creating Special Resources” on page 41. So the Paymore application does not use any workstation fixed resources.

Use the \texttt{M} command to specify the access method.

Figure 4 on page 8 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.

\section*{Controlling General Workstations}

You control general workstations such as SETP and PAY1 using the Ready List dialog, which you access using option 4 (WORK STATIONS) from the Tivoli OPC main menu. Refer to \textit{Controlling and Monitoring the Workload} for details of this dialog.

Using the Ready List dialog, the job preparation team can see what operations are waiting for setup, initiate setup by setting \textit{next status}, and complete the setup operation when they finish editing the JCL. The process is similar for other manual operations.

So 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).

\section*{Using Servers}

For each workstation, you specify \textit{parallel servers} on the availability panels. This is another resource, usually representing (for MVS computer workstations) JES initiators. If CPU1 has 15 servers, Tivoli OPC 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 OPC 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 OPC 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.
Creating Special Resources

The payroll system has several jobs that update the payroll database, and these cannot run together. If you submit them together, MVS 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 MVS 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 in Tivoli OPC is to create a special resource, which in this case is the database. See Chapter 3, “Creating Special Resources” on page 41 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 TIVOLI OPC DATA BASES menu (Figure 30 on page 48).
2 Select option 3 (LIST) on the MAINTAINING SPECIAL RESOURCES menu (Figure 31 on page 48). 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), you see the SPECIFYING SPECIAL RESOURCE LIST CRITERIA panel, where you can 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. You see the CREATING A SPECIAL RESOURCE panel (Figure 5):

```
EQQQDCRP ---------------- CREATING A SPECIAL RESOURCE -------------------
Option ===>
Select one of the following:
1 INTERVALS - Specify intervals
2 WS - Modify default connected work stations
SPECIAL RESOURCE ===> PAYROLL.DATABASE____________________________
TEXT ===> serializes access to the Paymore database________
SPECRES GROUP ID ===> SAMPLE__
Hiperbatch ===> N DLF object Y or N
USED FOR ===> B Planning and control C , P , B or N
ON ERROR ===> K_ On error action F , FS , FX , K or blank

Defaults
QUANTITY ===> 1_____ Number available 1-999999
AVAILABLE ===> Y Available Y or N
```

Figure 5. Specifying the Payroll Database as a Resource

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 OPC 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.
Introduction

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.

Create the default calendar like this:

1. Select option 1.2.2 from the Tivoli OPC main menu.
2. On the MODIFYING CALENDARS panel, enter the CREATE command. You see the CREATING A CALENDAR panel, (Figure 6).

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 OPC plans the work according to the free-day rule on each application run cycle. See “Selecting a Free-day Rule” on page 87 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 work-day end time, which is the time that Tivoli OPC 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 OPC 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 dataset, which must be concatenated with the head office dataset, 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 dialog (1.9.2 from the Tivoli OPC main menu) to create a variable table called PAY.

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

3. Code the JCL for PAYWEEK like this:

```clike
//PAYWEEK JOB ...
///c5197%OPC SCAN
///c5197%OPC BEGIN ACTION=INCLUDE,COMP=(&ODAY..EQ.1)
///c5197%OPC SETVAR TFRIDAY=(OCDATE+4CD)
///c5197%OPC END ACTION=INCLUDE
//PAY/zerodot7 EXEC PGM=PAY/zerodot7,PARM='&TFRIDAY.'
///c5197%OPC BEGIN PHASE=SUBMIT,ACTION=INCLUDE,COMP=(&DEPT..NE.N)
///c5197%OPC END ACTION=INCLUDE
//PAYDB DD DSN=XRAYNER.PAYROLL.DATABASE,DISP=SHR
//SYSPRINT DD SYSOUT=/c5197
```

---

**Figure 7. Creating a Variable**

---
This is an explanation of the marked lines:

1. /SF5800001/SF590000 is a statement that tells Tivoli OPC to perform variable substitution on the following lines. You need this unless you have VARSUB set to YES on the OPCOPTS initialization statement.

2. /SF5800002/SF590000 tells Tivoli OPC to use the format MM/DD/CCYY for OCDATE. If the input arrival date is 16 March 1997, for example, Tivoli OPC substitutes 03/16/1997 for &OCDATE.

3. /SF5800003/SF590000 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 OPC.

4. /SF5800004/SF590000 passes Friday’s date to the PAY07 program.

5. /SF5800005/SF590000 tests whether the variable DEPT has its default value N, and if not, it adds the extra line of JCL, which is the extra dataset for another department.

6. /SF5800006/SF590000 marks the end of the lines to be included.

4. Put the JCL into the partitioned dataset that is allocated to the ddname EQQJBLIB. Tivoli OPC 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 ddnames in the format EQQJSnDS). Tivoli OPC 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 OPC changes JCL:

- When you set up the JCL for an operation. You use the Ready list (described in Controlling and Monitoring the Workload) to perform setup and complete the operation on the job setup workstation.

- On request from the Modify Current Plan (MCP) dialog. This is independent of the job setup operation. If you change the JCL for an occurrence, Tivoli OPC 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 dialog. 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 OPC removes the JCL for the previous occurrence from the job repository.

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

See Chapter 21, “Job Tailoring” on page 273 for more details of variable substitution.

Note: PAYWEEK is an MVS 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 MVS system that runs the Tivoli OPC 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 dialogs, and using the batch loader program. See Chapter 8, “Defining Applications in Batch” on page 123 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 OPC dialogs. You can create all applications and groups using the Application Description dialog, 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 dialog.

First, create the PAYDAILY job, which is part of an application description of the same name, with three operations, as shown in Figure 2 on page 5. Follow these steps:

1. Select option 1.4 from the Tivoli OPC main menu to get the Application Description dialog menu, shown in Figure 44 on page 75.

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

3. Type the fields as shown and press Enter. See Chapter 6, “Standard Applications and Group Definitions” on page 75 for more information about each field.

4. To get Tivoli OPC to schedule PAYDAILY, enter the RUN command to specify a run cycle. You see the RUN CYCLES panel, shown in Figure 9 on page 14.
Specify an input arrival time of 12.00. This time has two main purposes in Tivoli OPC:

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 OPC 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 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 dataset at midday.

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 OPC 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.

Type R for a normal rule.

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 OPC needs to know what to do if Friday is Christmas Day, for example.

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

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 OPC looks for values to substitute into the JCL in a variable table.

Enter the S row command to specify the days that this rule will select. You see the MODIFYING A RULE panel, shown in Figure 10 on page 15.
Introduction

Figure 10. Creating the Rule for PAYDAILY

12 Select the fields shown so that Tivoli OPC 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.

14 Enter the OPER command. You see the OPERATIONS panel, shown in Figure 11.

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 DATASET) 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 OPC uses for planning purposes and also to judge whether an operation might finish late. Tivoli OPC can use the actual run times of PAYDAILY to adjust this estimate.

17 Specify the job name. For batch and setup operations, Tivoli OPC uses this to find the JCL in the EQQJBLIB partitioned dataset. 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 OPC tracks.

18 Enter the PRED command to specify the internal predecessors (the dependencies within this application). You see the panel in Figure 12.

![Figure 12. Specifying the Predecessors for PAYDAILY Operations](image)

19 Specify the predecessors shown to tell Tivoli OPC 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 dataset after getting the command from NetView. This dependency, however, is handled using resources. While CICSA has the dataset open, it has exclusive use of the Tivoli OPC resource PAYROLL.DATABASE. When the PAYX transaction successfully closes the dataset, it executes the Tivoli OPC EQQUSIN 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. You see the OPERATION DETAILS panel.

21 Specify option 3 (SPECIAL RES) to specify that the batch job must have exclusive use of the PAYROLL.DATABASE resource (see Chapter 3, “Creating Special Resources” on page 41 for details of the use of resources). Fill in the values shown in Figure 13 on page 17 and press PF3 (End).
Introduction

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.

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. Select option 4 (AUTOMATIC OPTIONS) from the OPERATION DETAILS menu. You see the JOB, WTO, AND PRINT OPTIONS panel, shown in Figure 14.

Specify y in the TIME DEPENDENT field. Now Tivoli OPC will not try to start the WTO operation until 12.00.

Return to the CREATING AN APPLICATION panel by pressing PF3 (End), and press PF3 (End) again to save the new application description in the Tivoli OPC 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 Figure 2 on page 5, 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 dataset**

The last operation of PAYBACKP is a WTO that triggers a CICS transaction to reopen the dataset.

**Job setup for other operations**

There are no job setup operations other than for PAYDAILY. Normally, you do not need job setup, because most JCL variable substitution can be handled automatically with JCL variable tables, and in these cases Tivoli OPC 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 OPC 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 OPC uses a key of job name, form number, and SYSOUT class to track the event, and this key must be unique, or Tivoli OPC might track the wrong spool dataset. See “Options that Apply to Print Operations” on page 104.

**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 to Tivoli OPC, you are ready to create the plans.

Follow these steps to create the plans:

1. Select option 2 (LTP) from the Tivoli OPC main menu. You see the MAINTAINING THE LONG-TERM PLAN menu.

2. Select option 2 (BATCH). You see the SELECTING LONG-TERM PLAN BATCH JOB menu.

3. Select option 7 (CREATE). You see the CREATING THE LONG-TERM PLAN panel, shown in Figure 15.

   **Figure 15. Creating the Long-term Plan**

   EQQLCREP ---------------- CREATING THE LONG TERM PLAN -------------------
   Command ==> Enter/Change data below:
   Long term plan:
   START ===> 97/2/3 Date in format YY/MM/DD
   END ===> 97/2/28 Date in format YY/MM/DD

4. Start the LTP 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. You see the GENERATING JCL FOR A BATCH JOB panel.

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

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

   **Figure 16. Listing Occurrences in the Long-Term Plan**

   EQQLSTOL ---------------- LONG TERM PLAN OCCURRENCES ------ ROW 1 TO 5 OF 189
   Command ==> Scroll ==> PAGE
   Enter the CREATE command above to create a new occurrence or enter the GRAPH command above to view occurrences graphically, or, enter any of the commands below:
   B - Browse, D - Delete, J - Job setup, M - Modify, RG - Remove from Group
   Row Application id Owner id Input arrival Deadline P C Pre Suc cmd date time date time
   *** PAYBACKP SAMPLE 97/2/2 12.00 97/2/3 06.00 5 N 0 0
   *** PAYDAILY SAMPLE 97/2/2 12.00 97/2/2 16.00 5 N 0 0
   *** PAYBACKP SAMPLE 97/2/3 12.00 97/2/4 06.00 5 N 0 1
   *** PAYDAILY SAMPLE 97/2/3 12.00 97/2/3 16.00 5 N 0 0
   *** PAYW SAMPLE 97/2/3 12.00 97/2/3 16.00 5 N 1 0

Chapter 1. Introduction to Tivoli OPC 19
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 re-create the LTP. You cannot recreate the LTP once you have a current plan; you have to delete the current plan first with the REFRESH function.

9 If the LTP looks good, put the jobs that you need into the EQQJBLIB dataset. The member name must be the same as the operation name.

10 Create the current plan. Select option 3 (DAILY PLANNING) from the Tivoli OPC main menu. You see the PRODUCING TIVOLI OPC DAILY PLANS menu.

11 Select option 2 (EXTEND). You see the EXTENDING CURRENT PLAN PERIOD panel, shown in Figure 17.

```
EQQDPEXP --------------- EXTENDING CURRENT PLAN PERIOD ------------------
Command ===> 
Enter/change data below and press ENTER

Current plan end date :
START DATE ===> 97/02/03 YY/MM/DD If no current plan exists
TIME ===> 19.02 HH.MM
END DATE ===> ________ YY/MM/DD Specific END date
TIME ===> _____ HH.MM Specific END time
EXTENSION LENGTH ===> 02400 HHHHM Extend plan by
TYPE ===> A A - includes all days
          W - includes only work days in the extension length above
Report selection :
  WS SUMMARY ===> Y Y if report wanted, otherwise N
  OPERATING PLAN ===> Y Summary for all work stations
  WS PLANS ===> Y Daily operation plan
  INPUT ARRIVAL ===> Y Plans for all work stations
  NON REPORTING ===> Y List of input arrival operations
  CURRENT PERIOD ===> Y Plans for non reporting work station
  PLANNED RESOURCE ===> Y Print current period results
  ACTUAL RESOURCE ===> Y Planned resource utilization
```

Figure 17. Creating the Current Plan

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 Does Tivoli OPC Do 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 OPC 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 dataset to a sequential dataset. 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 OPC handles it:

```plaintext
//PAYDAILY JOB (890122,N080),'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=/c5197
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.CICS.PAYDB,DISP=SHR
PAYOUT DD DSN=XRAYNER.DAY.TRANS,DISP=SHR
SYSPRINT DD SYSOUT=* 
```

This is an explanation of the marked lines:

1. Tells Tivoli OPC what to do if PAY04 fails for whatever reason. The default, with no action on the RECOVER statement, is that Tivoli OPC reruns the job. But before rerunning the job, it turns this RECOVER statement into a Tivoli OPC 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 OPC 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 OPC 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 19, “Automatic Recovery of Tivoli OPC-Controlled Jobs and Started Tasks” on page 243 for a full description of automatic recovery.

**Coping with More Complex Problems**

Sometimes,

- A job creates a dataset, and the NEW disposition gives an error on a rerun. Tivoli OPC handles problems like this with catalog management, which is described in Chapter 18, “Catalog Management” on page 235. It can delete datasets that have been created by failing jobs.

- You have to look at the SYSOUT dataset 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 OPC, implement it in stages. Specify just one of your applications to Tivoli OPC, and wait for this to run smoothly before implementing the others. Leave the optional functions, such as catalog management 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 OPC (refer to Controlling and Monitoring the Workload).
- 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 2. 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
- Specifying workstation access method parameters
- Controlling workstations

Each operation that Tivoli OPC 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

There are three types of workstation:

- Computer workstations
- Printer workstations
- General workstations

Create one or more workstations for each activity that you want to schedule and control with Tivoli OPC. 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.

Computer Workstations

The majority of Tivoli OPC operations are batch jobs and started tasks. These operations are specified to run on computer workstations where they are automatically started by Tivoli OPC 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 by Tivoli OPC. To automate jobs and started tasks, create at least one job computer workstation and one started-task workstation, even though these may be the same physical processor.
Job Computer Workstations
A job computer workstation has the STC option on the panel in Figure 21 on page 29 set to N. Tivoli OPC cannot submit a job unless the computer workstation used by the operation representing the job is available.

When you create your computer workstation open intervals (see “Specifying Workstation Open Intervals” on page 35), you can specify an alternate workstation where Tivoli OPC reroutes operations if the normal workstation becomes unavailable. You can also set the status of a workstation from the dialog. See “Recovery of Operations When a Workstation Becomes Inactive” on page 246 for more details on rerouting computer workstation operations and workload restart.

A computer workstation can also submit operations to non-MVS 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 110.

Note: When Tivoli OPC submits a job, it gives control to MVS and JES, or some other operating environment. Tivoli OPC 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 OPC cannot affect how the job or started task executes.

Started-Task Computer Workstations
A started-task computer workstation has the STC option on the panel in Figure 21 on page 29 set to Y. Operations you specify to run on this workstation are treated as started tasks, not as jobs.

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 110). But, instead of submitting the job to the internal reader on the destination system, Tivoli OPC puts it into the procedure library allocated to the EQQSTC ddname, and issues the MVS START command to invoke it.

If you specify the deadline-WTO option, described in “Started-Task Operations” on page 113, Tivoli OPC 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.

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 OPC 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 Tivoli OPC-tracked output group stops printing, Tivoli OPC 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 OPC of its status, or an operator can report the status manually. For Tivoli OPC operations and workstations that are associated with MVS jobs and started tasks, these events are created automatically by JES and SMF exits. On computer workstations with other Tivoli OPC-supported operating systems such as AIX/6000, events are also created 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 OPC of the operation status:

- Tivoli OPC ISPF dialogs.
- The Tivoli OPC-supplied 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 Tivoli OPC-supplied EQQUSIN subroutine. This lets you create events from your own programs.
- The Workload Monitor/2 feature dialog.

Tivoli OPC also uses two special general workstations to perform these tasks:

- Set up JCL for jobs and started tasks
- Issue WTO messages, possibly to trigger action by NetView.

Job Setup General Workstations

A job setup general workstation has the JOB SETUP option on the panel in Figure 21 on page 29 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 OPC databases or manually using the Tivoli OPC dialog. You do not need a job setup workstation if Tivoli OPC 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. Refer to Controlling and Monitoring the Workload for details of controlling operations using the ready list.

In Tivoli OPC, 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 OPC displays a list of the operations that you can choose from.
WTO General Workstations

A WTO general workstation has the WTO option on the panel in Figure 21 on page 29 set to Y. A WTO workstation lets you use Tivoli OPC’s 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 OPC builds a WTO message containing the text for the operation. NetView can then intercept the WTO message and take necessary action.

Tivoli OPC 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.

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.

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 option 1.4.4.3; DATABASE.AD.PRINT.OPERATIONS.

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 MCP dialog.

Which Workstation Types Are Needed?

Although you must create at least one workstation to enable Tivoli OPC 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 OPC is to control jobs and started tasks, at least two computer workstations will probably be required at any installation running Tivoli OPC: one for jobs and one for started 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 MVS system in your complex (one for started tasks and one for jobs). Make the computer workstations available when the MVS 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 MVS 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 MVS system.

If you create individual computer workstations for the unique components of your MVS batch workload, you have access to the Tivoli OPC 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 MVS system remains operational.

**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 18 shows this.

![Figure 18. Dependent Operations with Complex Dependencies](image)

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 19.

![Figure 19. Using a Dummy Operation to Simplify Complex Dependencies](image)

To create such an operation, first set up a dummy workstation: a general workstation with the REPORTING ATTR field on the panel in Figure 21 on page 29 set to N. Any operation that becomes ready on a nonreporting workstation
Workstations

will immediately be 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.

Creating a Workstation

Follow these steps to create a workstation:

1. Access the workstation dialog by entering option 1.1 from the Tivoli OPC main menu. The MAINTAINING WORK STATION DESCRIPTIONS menu is displayed.

2. Select option 2 (LIST). You see the LIST OF WORKSTATION DESCRIPTIONS panel, shown in Figure 20.

3. Enter the CREATE command. You see the CREATING GENERAL INFORMATION ABOUT A WORKSTATION panel, shown in Figure 21 on page 29.
EQQWCGEP  -----  CREATING GENERAL INFORMATION ABOUT A WORK STATION  -----  
Command ===>

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

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>WORK STATION NAME</td>
<td>cpu1</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>Main JES processor</td>
</tr>
<tr>
<td>WORK STATION TYPE</td>
<td>c</td>
</tr>
<tr>
<td>REPORTING ATTR</td>
<td>a</td>
</tr>
<tr>
<td>PRINTOUT ROUTING</td>
<td>SYSPRINT</td>
</tr>
<tr>
<td>SERVER USAGE</td>
<td>b</td>
</tr>
<tr>
<td>SPLITTABLE</td>
<td>N</td>
</tr>
<tr>
<td>JOB SETUP</td>
<td>N</td>
</tr>
<tr>
<td>STARTED TASK, STC</td>
<td>N</td>
</tr>
<tr>
<td>WTO</td>
<td>N</td>
</tr>
<tr>
<td>DESTINATION</td>
<td></td>
</tr>
<tr>
<td>TRANSPORT TIME</td>
<td>00:00</td>
</tr>
<tr>
<td>DURATION</td>
<td>00:05</td>
</tr>
</tbody>
</table>

Options:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>REPORTING ATTR</td>
<td>Automatic, S Manual start and completion</td>
</tr>
<tr>
<td>SERVER USAGE</td>
<td>Parallel server usage, B, N, P, or C</td>
</tr>
<tr>
<td>SPLITTABLE</td>
<td>Interruption of operation allowed, Y or N</td>
</tr>
<tr>
<td>JOB SETUP</td>
<td>Editing of JCL allowed, Y or N</td>
</tr>
<tr>
<td>STARTED TASK, STC</td>
<td>Started task support, Y or N</td>
</tr>
<tr>
<td>WTO</td>
<td>Automatic WTO, Y or N</td>
</tr>
<tr>
<td>DESTINATION</td>
<td>Name of destination</td>
</tr>
</tbody>
</table>

Defaults:

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRANSPORT TIME</td>
<td>Time from previous work station HH.MM</td>
</tr>
<tr>
<td>DURATION</td>
<td>Duration for a normal operation HH.MM</td>
</tr>
</tbody>
</table>

Figure 21. EQQWCGEP—Creating General Information about a Workstation

4 Fill in the fields as described in “Specifying Workstation Attributes and 
Options.”

5 Enter the A command to specify:

Workstation availability and open time intervals
See “Specifying Workstation Availability” on page 34.

The number of parallel servers in each time interval
See “Specifying Workstation Parallel Servers” on page 31.

The number of fixed resources in each time interval
See “Specifying Workstation Fixed Resources” on page 37.

The alternate workstation in each time interval
Refer to Controlling and Monitoring the Workload for details of how to 
redirect work.

6 Enter the M command if you want to specify access method parameters for 
that workstation. When you specify access method parameters, all the jobs 
submitted on that workstation will be run by the specified access method 
program. If you want that workstation to be able to submit jobs to a SAP R/3 
system, set the access method name as described under “Specifying 
Workstation Access Method Parameters” on page 39.

7 If you are using workstation fixed resources, described in “Specifying 
Workstation Fixed Resources” on page 37, enter the R command to specify 
them.

Specifying Workstation Attributes and Options

You also specify these workstation attributes and options:

- Reporting attributes
- Parallel servers
- Splittable attribute
Workstations

- Destination
- Default transport time
- Default duration.

These are described in the following sections. See also Figure 4 on page 8 for the attributes used for the Paymore example.

Specifying Workstation Reporting Attributes

Every operation in the Tivoli OPC 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 OPC changes the status of the operation in response to:

- JES and SMF exits
- Tivoli OPC dialog requests
- The EQQUSIN subroutine
- The program interface
- The OPSTAT and WSSTAT TSO commands
- The Workload Monitor/2 feature dialog

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 list of status codes is given in Appendix C, “Status, Error, and Reason Codes” on page 345.

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 Tivoli OPC JES and SMF exits. You can also change the status of these operations using the Tivoli OPC dialogs, 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 OPC 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 OPC to submit the job, started task or WTO that the operation represents. Tivoli OPC 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
dialog. You can also change the status of operations with the OPSTAT
command, the EQUSIN subroutine, the EQEVPGM 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 OPC 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 OPC automatically sets the status to C (completed), so
successor operations can run immediately.

N  Nonreporting.
Operations on a nonreporting 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. For example, the
operations can be used to hold the dependencies for successors, or the
operation may represent milestones in the processing. This is a good way to
keep track of the important processing points in large installations.

This type of workstation is often used for dummy operations created to simplify
the sequencing of other operations. (See “Dummy Workstations” on page 27.)

S  Start and completion.
The status change of operations is normally reported from the Ready List
dialog. You can also change the status of operations with the OPSTAT
command, the EQUSIN subroutine, the EQEVPGM 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 Workstation Parallel Servers
When you create a Tivoli OPC operation, you specify how many parallel servers it
requires (see “Using Parallel Servers” on page 100). The workstation that the
operation is using must have that number of parallel servers available before the
operation can run.

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 25 on page 36), but you can change it using the Modify
Current Plan dialog, described in Controlling and Monitoring the Workload.

You can specify, by setting server usage = P (planning only), that the number of
parallel servers should not be considered when Tivoli OPC 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 OPC produces cannot accurately
predict the behavior of real work in your system, because Tivoli OPC 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 OPC 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 OPC.

In Figure 22, 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 Tivoli OPC Selects Work for Automatic Submission” on page 209.

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 preparer 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 Tivoli OPC tracker. The Tivoli OPC tracker can communicate with the controller in several ways:

- Shared DASD containing a Tivoli OPC SUBMIT/RELEASE dataset
- MVS cross-system coupling facility (XCF) communication links
- A VTAM connection and the network communication function (NCF) of the Tivoli OPC tracker
- A user-defined method invoked from the operation-initiation exit, EQQUX009.

Therefore, you can specify the destination as:

- The ddname of a SUBMIT/RELEASE dataset
- The XCF member name of a Tivoli OPC tracker
- The NCF receiving LU name of a Tivoli OPC tracker
- A user-defined name

When you create a workstation, specify a destination that corresponds to a destination specified on a ROUTOPTS initialization statement (refer to Customization and Tuning). The default destination is the system where the Tivoli OPC 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 OPC. 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 minute.
Tivoli OPC 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 OPC can adjust this figure dynamically from its experience of the actual durations (see “Using Feedback Options” on page 105).

### Specifying Workstation Availability

Some workstations are not available 24 hours every day; Tivoli OPC can run work on the workstation only when it is available. When it is unavailable, either as scheduled or unexpectedly, Tivoli OPC can reroute work to an alternate workstation.

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 21 on page 29). This panel is displayed:

![EQQWMAVL — Availability of a Workstation](image)

In Figure 23, 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 23), you see the OPEN TIME INTERVALS FOR ONE DAY panel, shown in Figure 24 on page 35.
Workstations

Specifying Workstation Open Intervals

Before Tivoli OPC 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 OPC establishes the availability of a workstation by using the open intervals in the workstation description database. These are the times 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 OPC, 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 MVS system in a Tivoli OPC complex. See “How Many Workstations of Each Type?” on page 26. So, the availability of the computer workstation should reflect the availability of the MVS system it represents. This prevents Tivoli OPC from submitting work to an MVS system that is not physically available. Also, the accuracy of any planning predictions that Tivoli OPC produces for you depends on how accurately you have described your installation to Tivoli OPC.

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 23 on page 34). The ALL OPEN TIME INTERVALS panel will then be displayed.

---

**Figure 24. EQQWMOTL—Open Time Intervals For One Day**

If you press F3 (Exit) on 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.
Closing Workstations

Work and free days, as specified in the Tivoli OPC calendar, are used by Tivoli OPC to decide whether applications should be scheduled to start on a particular day. Tivoli OPC must also consider the availability of workstations. It takes this information from the workstation description database. Before it schedules an operation, Tivoli OPC checks that the workstation on which the operation will run is available throughout the estimated run time of the operation. Whether Tivoli OPC 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 (refer to Customization and Tuning).

When you have a holiday, which is specified in the calendar as a free day, Tivoli OPC 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 may be what you want.

But when you have a MVS 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 OPC 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:

- Specify it as a free day.
- Check that no applications with free day rule 3 are scheduled on that day.
- Do not have any open intervals for the workstation on the holiday, so that Tivoli OPC 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 26 on page 37),...
which you reach by selecting option 4 from the MAINTAINING WORKSTATION DESCRIPTIONS panel.

Figure 26. EQQWMACL—Modifying All Workstations Closed

You can specify when workstations are open with the Workstation Description dialog, 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 OPC 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 dialog, you must run daily planning for the change to take effect. But a change to the 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 dialog, 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 OPC can keep track of these limited resources and take account of them in its plan.

When you set up a workstation, you can specify the quantity of two resources available to operations at the workstation, R1 and R2. You can also assign a 2-character name to these resources.

If you are new to Tivoli OPC, you are recommended not to use these fixed resources, but to use the special resources described in Chapter 3, “Creating
Special Resources” on page 41 instead, because these do not have the restrictions associated with workstation fixed resources, the most important being that:

- You may 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 21 on page 29). You see the RESOURCES FOR A WORKSTATION panel, shown in Figure 27.

```
EQQWMREP ---------------- RESOURCES FOR A WORKSTATION ------------------------
Command ===>

Enter/change data below:
Work station : CPU1 Main JES processor

Resource 1
NAME ===> R1
PLANNING ===> Y Used at planning, Y or N
CONTROL ===> N Used at control, Y or N

Resource 2
NAME ===> R2
PLANNING ===> Y Used at planning, Y or N
CONTROL ===> N Used at control, Y or N
```

Figure 27. EQQWMREP—Resources for a Workstation

Workstation resources can be used for scheduling purposes. When you create an operation, you can specify how many of the workstation resources (R1, R2, or both) the operation will use. If that quantity of the resources is not available, the operation will not be started by Tivoli OPC (see “Specifying Workstation Fixed Resources” on page 37 for exceptions to this rule).

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 an MVS 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 OPC will not schedule the operation unless its required number of tape drives is available.

**Note:** Tivoli OPC 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 OPC-controlled operations are using a resource. If a resource is being used outside the control of Tivoli OPC or you have not specified to Tivoli OPC that an operation uses a resource, Tivoli OPC has no means of knowing that the physical resource is not available.

When Tivoli OPC 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 OPC might need to reassess the start time of some of its operations. At this point, the Tivoli OPC CONTROL option becomes important. If you specified Y for this option, Tivoli OPC considers the workstation resources when rescheduling its operations. Otherwise, the workstation resources are ignored.

**Specifying Workstation Access Method Parameters**

You can define some special workstations to handle particular kinds of jobs. On such workstations, you can submit and track jobs that are not executed by the Tivoli OPC tracker but by other, external products. An example is a SAP R/3 job. Using Tivoli OPC, you can define a workstation so that all Tivoli OPC jobs submitted on it are recognized as SAP R/3 jobs and are sent from the tracker to the SAP R/3 system to be executed. When the operation has been submitted to the external product—in this case, SAP R/3—Tivoli OPC is notified of all changes in the operation’s status. Tivoli OPC maps the specific product states to Tivoli OPC states, so that a Tivoli OPC user can monitor an external job as though it were a Tivoli OPC job.

To define a workstation as able to handle such jobs, you must associate with it some specific values, called “access method parameters.” To specify access method parameters for a workstation, enter M on the Creating General Information about a Work Station panel (see Figure 21 on page 29). The panel shown in Figure 28 is displayed.

![EQQWMAMP—Work Station Access Method Information Panel](image)

The access method is an optional software component, running on a Tivoli OPC tracker agent machine, which the tracker agent can use to interface with an external product. You can thus submit and track jobs on that system.

To define a workstation as able to submit and track jobs that are executed by an external product, set the access method name to the name to the access method that handles that particular product. In particular, if you want all jobs to be handled as SAP R/3 jobs and be submitted to a SAP R/3 system, set the name of the access method for that workstation to **r3batch**.

To interface with the SAP R/3 system from a Tivoli OPC tracker agent, you need to configure on it the Tivoli Workload Scheduler (the **r3batch** access method). This feature is not part of the Tivoli OPC product; it can be ordered separately.
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 dialog. Refer to *Controlling and Monitoring the Workload* for these tasks that dynamically control the availability of workstations:

- Modifying the workstation open intervals in the current plan from the Modify Current Plan dialog (option 5.5 from the Tivoli OPC main menu)
- Setting the workstation to *active*
- Looking at the workstation status
- Closing a workstation
- Redirecting work from one workstation to an alternate workstation
Chapter 3. Creating Special Resources

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

**Special resources**
These resources are the most flexible. Create them using the dialog 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 may have problems if a computer workstation and a started task workstation share the same tape drives. Specify them using the workstation dialog.

**Parallel servers**
These resources normally represent JES initiators for computer workstations running the MVS operating system. Specify them using the workstation dialog.

If you are new to OPC, use the special resources described in this chapter instead of workstation fixed resources. If you used early releases of OPC, note that the implementation of special resources changed in Release 3.

Understanding Special Resources

You can use Tivoli OPC 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 dialog, which is described in this chapter. The Special Resource dialog updates the resource database, which has these details of each resource:

- **Name**: Up to 44 characters. This identifies the resource.
- **Availability**: Yes (Y) or no (N).
- **Connected workstations**: A list of the workstations where operations can allocate the resource.
- **Quantity**: 1 to 999999.
- **Used for**: How Tivoli OPC is to use the resource: for planning (P), control (C), both (B), or neither (N).
- **On-error action**: Free all (F), free exclusively-held resources (FX), free shared resources (FS), and keep all (K). Tivoli OPC 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.

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

- **Quantity**: 1 to 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 to 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 option 7 (SPECRES) in the Modify Current Plan dialog. 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 *Controlling and Monitoring the Workload* for details of the Special Resource Monitor.

Other ways of changing resource attributes are:

- **EQQUSIN subroutine** Refer to *Customization and Tuning*.
- **SRSTAT command** See “SRSTAT” on page 330.

If the availability of a resource is known to the Resource Object Data Manager (RODM), Tivoli OPC 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.

### Example Using Datasets

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

To prevent Tivoli OPC 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 OPC makes sure the jobs are not scheduled to run together or run at a time when the resources are unavailable. This is how Tivoli OPC makes use of the resource at the planning stage. When Tivoli OPC is ready to submit each job, it checks that the resource is available. This is how Tivoli OPC 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 dataset; for Paymore, this is CPU1 and STC1.

**Intervals**   The dataset 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.

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 MVS, 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:

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

How Does Tivoli OPC Use Special Resources?

Tivoli OPC just keeps a record of the state of each resource and its allocation. Tivoli OPC 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 Tivoli OPC knows the true availability of the objects that the resources represent.

VSAM will not tell Tivoli OPC when the payroll database is opened, and MVS is not going to tell Tivoli OPC when you vary a tape drive offline. This is your responsibility.

The best way to tell Tivoli OPC 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 OPC about changes in resources by intercepting messages, for example, and issuing the Tivoli OPC SRSTAT command, or calling the EQQUSIN subroutine. If you cannot automatically notify Tivoli OPC 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 OPC/ESA uses the data: ────────────────┐
Where? │
CP PRD │
┌────────────────────────────────────────────────────────────────┐ │
│Overriding (global) Quantity Availability Deviation x │ │
├────────────────────────────────────────────────────────────────┤ │
│Interval: Quantity Availability Connected workstations x x │ │
│Interval: Quantity Availability Connected workstations x x │ │
│Interval: Quantity Availability Connected workstations x x │ │
│... (there can be many intervals) │ │
└────────────────────────────────────────────────────────────────┘

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 dialog 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</td>
<td>CPU1 STC1</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 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 dialog, 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 Chapter 6, “Standard Applications and Group Definitions” on page 75). 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 OPC continues to allocate the broken tape drive to a job that needs one. The job will wait, because MVS 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 dialog to set a resource unavailable. Refer to Controlling and Monitoring the Workload for details of the Special Resource Monitor dialog.

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 OPC subscribes to it, Tivoli OPC 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.

Figure 29 on page 47 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 OPC revert to a scheduled value after a manual alteration, you must reset the value, as at 11.20.
<table>
<thead>
<tr>
<th>Start of interval / time of event</th>
<th>Planned values</th>
<th>Actual values</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Planned quantity</td>
<td>Planned availability</td>
</tr>
<tr>
<td>08.00</td>
<td>8 N</td>
<td>Interval specifies quantity 8, not available</td>
</tr>
<tr>
<td></td>
<td>8 N</td>
<td>0</td>
</tr>
<tr>
<td>08.40</td>
<td>You set the availability to Y with the EQUSIN subroutine</td>
<td>8 Y</td>
</tr>
<tr>
<td></td>
<td>8 Y</td>
<td>0</td>
</tr>
<tr>
<td>09.00</td>
<td>A new interval specifies the resource unavailable</td>
<td>8 Y</td>
</tr>
<tr>
<td></td>
<td>8 Y</td>
<td>0</td>
</tr>
<tr>
<td>09.40</td>
<td>You set a deviation of -1 with the SRSTAT command</td>
<td>8 Y</td>
</tr>
<tr>
<td></td>
<td>8 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 Y</td>
</tr>
<tr>
<td></td>
<td>8 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 Y</td>
</tr>
<tr>
<td></td>
<td>8 Y</td>
<td>-1</td>
</tr>
<tr>
<td>10.00</td>
<td>Extend CP, and interval specifies 9 available</td>
<td>9 Y</td>
</tr>
<tr>
<td></td>
<td>9 Y</td>
<td>-1</td>
</tr>
<tr>
<td>10.20</td>
<td>You set the quantity to 6 with the SRSTAT command</td>
<td>6 Y</td>
</tr>
<tr>
<td></td>
<td>6 Y</td>
<td>-1</td>
</tr>
<tr>
<td>11.00</td>
<td>Interval specifies 8 available</td>
<td>8 Y</td>
</tr>
<tr>
<td></td>
<td>8 Y</td>
<td>-1</td>
</tr>
<tr>
<td>11.20</td>
<td>You reset the quantity with the SRSTAT command</td>
<td>8 Y</td>
</tr>
<tr>
<td></td>
<td>8 Y</td>
<td>-1</td>
</tr>
</tbody>
</table>

*Figure 29. How Resource Quantity and Availability Can Be Changed*

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 dialog 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 TIVOLI OPC DATA BASES menu (Figure 30).

2. Select option 3 (LIST) on the MAINTAINING SPECIAL RESOURCES menu (Figure 31). You can select option 2 (CREATE) instead, but option 3 gives you a chance to see the resources that already exist.

When you select 3 (LIST), you see the SPECIFYING SPECIAL RESOURCE LIST CRITERIA panel, where you can 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 (Figure 32), enter the CREATE command.

```
EQQQDLSL -------------- LIST OF SPECIAL RESOURCES ------- ROW 1 TO 4 OF 4
Command ===> create   Scroll ===> PAGE

Enter the CREATE command above to create a new resource, or,
enter any of the row commands below:
B - Browse, M - Modify, C - Copy, D - Delete

<table>
<thead>
<tr>
<th>Row</th>
<th>Special Resource</th>
<th>Specres Group ID</th>
<th>Qty</th>
<th>Num</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HEIDE.ISPF.PROFILE</td>
<td>Y</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>HEIDE.OPCESA.EQQDUMP</td>
<td>Y</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>PAYROLL.DATABASE</td>
<td>Y</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>RITZMAN.DSCLOSE.TEST</td>
<td>Y</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Figure 32. EQQQDLSL—List of Resources

You see the CREATING A SPECIAL RESOURCE panel:

```

```
EQQQDCRP -------------- CREATING A SPECIAL RESOURCE --------------
Option ===> 

Select one of the following:
1 INTERVALS - Specify intervals
2 WS - Modify default connected work stations

<table>
<thead>
<tr>
<th>SPECIAL RESOURCE</th>
<th>TEXT</th>
<th>SPECRES GROUP ID</th>
<th>Hiperbatch</th>
<th>USED FOR</th>
<th>ON ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>tapes</td>
<td>tape drives</td>
<td>sample</td>
<td>N</td>
<td>B</td>
<td></td>
</tr>
</tbody>
</table>

Defaults
```

```
QUANTITY   ==> 8  Number available 1-999999
AVAILABLE  ==> Y  Available Y or N

Figure 33. EQQQDCRP—Creating a Special Resource

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 OPC uses these for filtering and searching in the dialogs.

**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 dialog (a list filter).

**Hiperbatch**
Whether the resource represents a Data Lookaside Facility (DLF) object, Y or N. See “Hiperbatch and the Dataset Lookaside Facility” on page 56.
Special Resources

**USED FOR** Whether the resource is used for
- P (planning, when the current plan is extended)
- C (control, when Tivoli OPC starts an operation)
- B (both)
- N (neither)

**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):
- F (free its full allocation of this resource, both those allocated exclusive and those allocated shared)
- FS (free its full shared allocation of this resource)
- FX (free its full exclusive allocation of this resource)
- K (keep its full allocation of this resource)
- Blank (use the default specified in the ONERROR keyword of the RESOPTS statement. Refer to *Customization and Tuning*.)

You may 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 next two values, QUANTITY and AVAILABLE, 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.

**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. You see the MODIFYING CONNECTED WORK STATIONS FOR A SPECIAL RESOURCE panel, shown in Figure 34.

![Figure 34. EQQQDWML—Modifying Connected Work Stations for a Special Resource](image)

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 34.
Note: When an operation is switched to an alternate workstation, when the primary workstation is offline, Tivoli OPC still allows the operation 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. You see the MODIFYING INTERVALS FOR A SPECIAL RESOURCE panel, shown in Figure 35.

![EQQQDIML -- MODIFYING INTERVALS FOR A SPECIAL RESOURCE](image)

Figure 35. EQQQDIML—Modifying Intervals for a Special Resource

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

**Day of week or Date**

Specify a date in the format specified in the Options dialog, or one of these:

- STANDARD (meaning default for the days not mentioned)
- MONDAY
- TUESDAY
- WEDNESDAY
- THURSDAY
- FRIDAY
- SATURDAY
- SUNDAY

**From time and To time**

Specify a time range, with times in the format specified in the Options dialog.

**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 49.

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.
Enter the S command beside the Saturday interval row to specify that STC1 operations cannot use the resource on that day. You see the MODIFYING CONNECTED WORK STATIONS FOR A SPECIAL RESOURCE panel, shown in Figure 36 on page 52.

Figure 36. EQQDWML—Modifying Connected Work Stations for a Special Resource

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

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

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

Table 4 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 4. 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>
<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 OPC 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 dialog 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 dataset, you can make this resource available when the dataset is created and loaded with valid data. If the dataset is created and loaded by a TSO user, the user can set the availability status with the SRSTAT command immediately after the dataset is loaded. If a batch job creates and loads the dataset, 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 dialog. Refer to Controlling and Monitoring the Workload for details of these and other operator tasks.

By default, Tivoli OPC 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 Tivoli OPC 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 Tivoli OPC operations as requiring shared access to the resource.

Using RODM to Set the Availability of a Resource

If you have the Resource Object Data Manager (RODM) on your MVS 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 OPC changes in resource status detected by such products as AOC/MVS.

Creating a Resource Dynamically

If an operation in one of your applications needs a resource but the resource it requires is not created, Tivoli OPC 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 OPC 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 OPC 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 OPC 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.”

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 OPC ignores the CREATE keyword on the SRSTAT command, and the DYNAMICADD keyword on the RESOPTS initialization statement.

- You add an occurrence to the current plan using the MCP dialog. One of the operations in the occurrence references the resource. Tivoli OPC 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 OPC does not wait until it starts the operation.

Tivoli OPC 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 OPC 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 OPC 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 dialog. One of the operations in the occurrence references the resource. Tivoli OPC 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 may 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:
   ```
   SRSTAT 'TAPES' SUBSYS(OPC1) AVAIL(YES)
   ```
3. Tivoli OPC adds TAPES to the current plan, taking the values from the resource database.
4. Tivoli OPC sets the overriding (global) availability to YES. It leaves the overriding quantity and deviation fields blank, so that Tivoli OPC 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:
   ```
   SRSTAT 'TAPES' SUBSYS(OPC1) AVAIL(RESET)
   ```
8. Tivoli OPC 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 for deletion.
**Hiperbatch and the Dataset Lookaside Facility**

`Hiperbatch` is an MVS performance enhancement that works with the data lookaside facility (DLF) to allow batch jobs and started tasks to share access to a dataset, or *data object*. Tivoli OPC provides control information to DLF concerning which operations are allowed to connect to which DLF object, and which datasets are eligible for Hiperbatch.

Within Tivoli OPC, a dataset that is eligible for Hiperbatch is treated as a resource. Using the Resources dialog, you can define datasets with the DLF attribute. The DLF exit sample, EQQDLFX, can then make the following decisions about DLF processing:

- Is this dataset eligible for Hiperbatch?
- Should this operation be connected to this data object?

Tivoli OPC issues enqueues on the job and dataset name to notify the DLF exit that the job to be scheduled will use Hiperbatch. When the job ends, Tivoli OPC checks if the same dataset will be used by the immediate successor operation or by any other ready operation. If so Tivoli OPC does not purge the data object. Otherwise, Tivoli OPC initiates purge processing of the data object (that is, Tivoli OPC removes it from Hiperspace). For details about installing Tivoli OPC Hiperbatch support, refer to `Customization and Tuning`.

**Note:** The Tivoli OPC 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 OPC 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 dialog print menu, or option 1.4.4.6 from the Tivoli OPC main menu.

**Using Availability Changes to Trigger Events**

Tivoli OPC can start an operation automatically when some resource becomes available. Consider this example: an online system closes a dataset, and you want this to trigger some batch processing. Follow this procedure:

1. Create a special resource that represents the dataset. It is not really necessary to create the resource in the resource database—if you do not do this, the event causes the resource to be created dynamically in the current plan. See “Dynamically Adding Resources that Are Not in the Database” on page 54 for details.

2. Specify that Tivoli OPC adds an occurrence to the current plan when the special resource becomes available (an R-type event). This is an

---

**Special Resources**

**Reporting on Special Resources**

When you run the daily planning job (EXTEND or REPLAN), Tivoli OPC 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 dialog print menu, or option 1.4.4.6 from the Tivoli OPC main menu.
event-triggered tracking (ETT) function, which is described in Chapter 20, “Event-Triggered Tracking” on page 267.

3 Cause Tivoli OPC to be notified when the real resource becomes available. There are two methods of doing this:

   a. Have the online application set the resource available when it closes the dataset, using the EQQUSIN subroutine or the SRSTAT command.

   b. Use the dataset triggering function, which is included in the IEFU83 SMF exit. The advantage of this is that the exit invokes the Tivoli OPC interface, so you do not have to change your applications. Refer to the Installation Guide for details of this function.
Special Resources
Chapter 4. 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 78, where you specify these as part of the application definition.

When you run work manually, without Tivoli OPC, 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 OPC is to be able to automate this, you must specify when jobs are to run. To achieve this, Tivoli OPC has three important objects:

Calendars
The calendar specifies normal working days and public holidays. Tivoli OPC 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 OPC 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 Tivoli OPC OPTIONS dialog (0.2 from the Tivoli OPC 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 may have several calendars, but always call your default calendar DEFAULT, and specify the same calendar name on BATCHOPT and in the dialog.

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 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 OPC 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 Tivoli OPC calendar defines the status of each day and the work-day end time.
Tivoli OPC 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 dialog to change the workstation open
intervals as described in Controlling and Monitoring the Workload.

To create the default calendar:

1 Select option 1.2.2 from the Tivoli OPC 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. You see
the CREATING A CALENDAR panel (Figure 37).

3 Type DEFAULT for the calendar ID.
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:

**Work day**  When Tivoli OPC schedules work as normal

**Free day**  When Tivoli OPC plans work according to the free-day rule on each application run cycle. See “Selecting a Free-day Rule” on page 87 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.

5 Specify the work-day 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 work-day end time for your installation: the time that a normal work day ends. The work-day 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 work-day end time on the work day. If the work-day end time is after midnight, the work-day end time determines when work is planned to run. If the work-day end time is before midnight on the free day, Tivoli OPC waits for the next work-day 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 work-day 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 work-day 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 LTP 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 work-day 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.

**End time  Effect on scheduling**

<table>
<thead>
<tr>
<th>Time</th>
<th>Description</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 OPC assumes that you mean midnight between that day and the next day—times before the work-day end time come at the end of the work day.</td>
</tr>
</tbody>
</table>
23.59 This work day end time is not recommended. The free day, 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 work-day end time has no effect on the daily planning process; when Tivoli OPC 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 OPC 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 work-day 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 Tivoli OPC OPTIONS dialog 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 a 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 OPC 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 dialog. From the Tivoli OPC main menu, select option 1.3, which will display the MAINTAINING THE PERIODS menu.
2 Select option 2 to display the LIST OF CALENDAR PERIODS panel:

```
EQQTPERL ------------------- LIST OF CALENDAR PERIODS --------  ROW 1 TO 5 OF 5
     Command ===> Scroll ===> CSR

Enter the CREATE command above to create a new period or enter any of the row commands below:
B - Browse, C - Copy, D - Delete, M - Modify

Row  Period Description Origin Cyc Per Last update
      cmd name   date int typ user date
 1  ADVENT Before Christmas 97/11/30 0 N XCHAS 97/01/30
 1  SEMESTER University term 97/01/06 0 N XCHAS 97/01/02
 1  TAXYEAR British tax year 97/04/03 0 N XMAWS 97/01/30
 1  BACKUP A cyclic interval of 4 days 97/01/01 4 W XMAWS 97/01/30
 1  QUARTER Three calendar months 97/01/01 0 N XMAWS 97/01/30

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

Figure 38. EQQTPERL—List of Calendar Periods

3 To create a new period, enter CREATE at the command prompt. You see this panel:

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

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

PERIOD NAME ===> ________ Name of the period
PERIOD TYPE ===> _ A = cyclic based on all days, W = cyclic based on work days, N = non-cyclic
DESCRIPTION ===> ______________________________ Descriptive text of the period
INTERVAL ===> /zerodot/zerodot/zerodot Number of days between cyclic run period.
VARIABLE TABLE ===> ________________ JCL variable table id

Row  Interval Interval
     cmd origin end
     ****

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

Figure 39. EQQTCRPL—Creating a Calendar Period

4 Fill in the fields on 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 OPC counts both work days and free days when calculating period dates.

- **W** Work-days-only cyclic. Tivoli OPC 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 67.

- **N** Noncyclic.
Calendars and Periods

For example, if the calendar you are using specifies that 25 December 1997 is a free day, as are every Saturday and Sunday, and you create a type \( W \) period with an origin of December 1 and a length of 10 days, cyclic intervals will begin on these dates:

- 1 December, 1997
- 15 December, 1997
- 30 December, 1997
- 13 January, 1998, 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 OPC 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 OPC uses the JCL variable table that you specify here. If you use this period with a rule-based run cycle, Tivoli OPC ignores any table name specified here. See Chapter 21, “Job Tailoring” on page 273 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) dialog (described in Controlling and Monitoring the Workload), the Long-Term Plan dialog, or by using Tivoli OPC 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 OPC 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 OPC assumes that the interval ends the day before the next interval.

When a run cycle specifies a negative offset in the period, or 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
Calendars and Periods

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 OPC outside the specified interval.

2. From OPC/ESA Release 3.1 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 OPC. 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 67 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 LTP.

If an explicit interval end date is defined for an interval and a specified offset is greater than the number of days in the interval, then the obtained date will not be considered and the message EQQ0527W will be issued.

However, with open intervals, that is, with intervals which have no explicitly defined end date, offsets greater than the number of days between origin dates will be accepted and rundates may be generated outside the defined intervals.

Thus, OPC checks that the generated date is within the period interval only for intervals with explicitly defined end dates.

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.
Calendars and Periods

This example shows a lunar calendar month, assumed to be 28 days:

<table>
<thead>
<tr>
<th>Cyclic period: work days and free days</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD NAME = MOON</td>
</tr>
<tr>
<td>PERIOD TYPE = A</td>
</tr>
<tr>
<td>INTERVAL ORIGIN = 7 February 1997</td>
</tr>
<tr>
<td>INTERVAL = 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:

<table>
<thead>
<tr>
<th>Cyclic period: work days only</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD NAME = BACKUP</td>
</tr>
<tr>
<td>PERIOD TYPE = W</td>
</tr>
<tr>
<td>INTERVAL ORIGIN = 2 January 1997</td>
</tr>
<tr>
<td>INTERVAL = 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:

<table>
<thead>
<tr>
<th>Noncyclic period</th>
</tr>
</thead>
<tbody>
<tr>
<td>PERIOD NAME = Semester</td>
</tr>
<tr>
<td>PERIOD TYPE = N</td>
</tr>
<tr>
<td>INTERVAL ORIGIN INTERVAL END</td>
</tr>
<tr>
<td>26 August 1996 13 December 1996</td>
</tr>
<tr>
<td>13 January 1997 16 May 1997</td>
</tr>
<tr>
<td>9 June 1997 28 June 1997</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 OPC 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:

<table>
<thead>
<tr>
<th>Run cycles that use rules</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Day in Semester</td>
</tr>
<tr>
<td>Last Sunday in Semester</td>
</tr>
<tr>
<td>Seventh Sunday in Semester</td>
</tr>
<tr>
<td>Last Day in Semester</td>
</tr>
</tbody>
</table>

(This may result in an error)
These are the equivalent selections using offsets:

<table>
<thead>
<tr>
<th>Run cycles that use offsets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Semester, offset 1     First day in semester</td>
</tr>
<tr>
<td>Semester, offset 105   Last Sunday in semester (for the semester</td>
</tr>
<tr>
<td>beginning on 26 August 1996)</td>
</tr>
<tr>
<td>Semester, offset 38    Seventh Sunday in semester (for the</td>
</tr>
<tr>
<td>semester beginning on 18 January 1994)</td>
</tr>
<tr>
<td>Semester, offset -1    Last day in semester</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 OPC 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 Chapter 20, “Event-Triggered Tracking” on page 267 for more information.

Using Work-Days-Only Cyclic Periods

A work-days-only cyclic period (type W) has only work days. Figure 40 shows a 5-day period MYWEEK that starts on a Monday.

<table>
<thead>
<tr>
<th>Free days:</th>
<th>F</th>
<th>F</th>
<th>F</th>
<th>F</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sat</td>
<td>Sun</td>
<td>Mon</td>
<td>Tue</td>
<td>Wed</td>
<td>Thu</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Periods:</th>
</tr>
</thead>
<tbody>
<tr>
<td>1:    x x x x x</td>
</tr>
<tr>
<td>2:    x x x x x x</td>
</tr>
<tr>
<td>3:    x x x x</td>
</tr>
</tbody>
</table>

Figure 40. A 5-day Work-days-only Cyclic Period, MYWEEK

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 OPC can select different days depending on the type of run cycle. Figure 41 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 87 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 OPC jumps over the free days, counting only the work days in the period.

![Diagram showing how Tivoli OPC counts offsets with work-days-only cyclic periods](image)

**Greater Consistency with Tivoli OPC**

When you create a work-days-only cyclic period where the interval origin is a free day, the Tivoli OPC long-term planning program ignores this day and starts the first period on the first work day after this. This process was introduced in OPC/ESA Release 3.1 OPC/ESA Release 3 and earlier releases, on the other hand, counted this day as the first day even if it was free, so that the first period contained one work day less than others.

Figure 42 shows an example of this, where the origin of the 5-day period is a Saturday, and Saturdays and Sundays are free days.

![Diagram showing results when the work-days-only cyclic period origin is a free day](image)
Chapter 5. Things You Should Consider before Creating Applications

This chapter 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 Chapter 6, “Standard Applications and Group Definitions” on page 75 and Chapter 7, “Creating Job Descriptions” on page 117 (online dialogs), and in Chapter 8, “Defining Applications in Batch” on page 123.

What Are Applications?

The tasks that you want Tivoli OPC 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:

- General data—application name, description, and owner representative
- Run schedules—period references, run time, and deadlines
- Operation data—job names, workstations, and resources
- Dependencies—between operations within the application and other applications

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 dialog to create and modify standard applications. Standard applications can contain more than one job.

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

You can browse or modify job descriptions using both the Job Description and the Application Description dialogs, but if you change a job description using the
Applications

Application Description dialog 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 dialog.

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 OPC classes it as a job description, whether you create it with the Application Description dialog, the Job Description dialog, 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 OPC 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 OPC and in ongoing maintenance to the applications. You can also use groups in the Modify Current Plan dialog, 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.
Applications

Naming Standards in Applications

Naming standards are important when you are creating Tivoli OPC 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 OPC dialog 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 OPC jobs should also be different from those of non-Tivoli OPC jobs.

Tivoli OPC imposes some restrictions on your choice of job names:

- Print operations must have the same job name as the job operations they refer to.
- 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.
Applications

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 within Tivoli OPC. For example, all Paymore applications have an owner ID of SAMPLE. The owner ID of SAMPLE can be used as selection criteria in the Tivoli OPC dialogs and reports.

Rules for Creating Applications

This section outlines the basic rules that govern standard applications and group definitions. The Tivoli OPC dialog prevents 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></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 OPC 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>
<tr>
<td>What fields must you specify?</td>
<td>Application ID Type, which must be A Owner ID Valid-from date Status Priority At least one operation</td>
<td>Application ID (the group ID) Type, which must be G Owner ID Valid-from date Status</td>
</tr>
</tbody>
</table>

Figure 43. Differences between Applications and Group Definitions

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 Operation Predecessors” on page 94.
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.

Tivoli OPC scheduling capabilities let you 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 OPC 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 27 for more details.

Methods of Creating Applications

You can use any of these methods to create your application:

- **Application Description dialog**
  You enter this dialog by selecting the AD option from the MAINTAINING OPC/ESA DATA BASES panel. See Chapter 6, “Standard Applications and Group Definitions” on page 75.

- **Job Description dialog**
  The Job Description dialog compresses most of the function of the Application Description dialog into one panel by making some assumptions about the application that you are creating. See Chapter 7, “Creating Job Descriptions” on page 117.

- **Batch loader**
  This Tivoli OPC-supplied program lets you enter your application descriptions in batch. See Chapter 8, “Defining Applications in Batch” on page 123 for details.

- **Program interface**
  This standard interface to Tivoli OPC lets you write your own programs to update information held by Tivoli OPC. Refer to Programming Interfaces for details.
Chapter 6. Standard Applications and Group Definitions

This chapter describes how you use the Application Description dialog 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 Tivoli OPC operations
• Using print operations
• Using WTO operations
• Creating started-task operations
• Creating time-dependent operations

Creating an Application and its Operations

1. Select option 1.4 from the Tivoli OPC main menu to get the Application Description dialog menu, shown in Figure 44.

Figure 44. EQQASUBP—Maintaining Application Descriptions
2 Select option 2 to show the CREATING AN APPLICATION panel, shown in Figure 45.

![CREATING AN APPLICATION panel](image)

**Figure 45. EQQACGPP—Creating an Application**

3 Specify the Application ID (see “Specifying the Application ID” on page 77) 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 Figure 46.

<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</td>
<td>Use the RUN command if you want Tivoli OPC 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>TYPE = G</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>An application that is part of a group</td>
<td>Do not use the RUN command, because Tivoli OPC 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>TYPE = A</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>An application that stands alone</td>
<td>Use the RUN command if you want Tivoli OPC to schedule the application.</td>
<td>Use the OPER command to specify the operations in the application.</td>
<td>Leave blank.</td>
</tr>
<tr>
<td>TYPE = A</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

See “Specifying When Your Application Should Be Scheduled” on page 78 for detailed information about run cycles and the RUN command.

See “Creating Operations” on page 92 for detailed information about operations and the OPER command.

**Figure 46. How to Use the RUN and OPER Commands and the GROUP DEFINITION Field**

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.
Applications

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 (see “Specifying the Valid-From Date”).

8 Specify A or P in the STATUS field as required (see “Specifying the Status” on page 78).

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.

Specifying the Application ID

The name of the application, the date to which it is valid, and the status uniquely identify each application to Tivoli OPC. 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:

MYAPPLICATION+1, IMYAPPL, &

Specifying the Valid-From Date

You can create several applications with the same ID but with different valid-from dates. Tivoli OPC 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 OPC uses the first version for the period 98/01/01 to 98/01/04 and the second version from 98/01/05.

Note: You can tailor the date format in the dialog to suit your installation. The default format is used here.

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 Tivoli OPC 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 Tivoli OPC 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 OPC 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.

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 OPC 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:

- Dialog 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 dialog:

1. Add a member to it.
2. Specify option G on panel EQQMAADL.

For details, refer to the section “Adding an Application Group to the Current Plan” in Controlling and Monitoring the Workload.

Creating Run Cycles

Run cycles use the calendar and periods that are described in Chapter 4, “Creating Calendars and Periods” on page 59. 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 OPC actions when this day falls on a free day (holiday), you do not
need to create any periods. These are 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 OPC 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, 3, 5, 9, 13, 17 etc. The rule combines these and removes duplicates, giving January 1, 3, 5, 7, 9, 10, 13, 15, 17 etc.</td>
</tr>
<tr>
<td>Every last Monday in the month</td>
<td>This generates every Monday, because Tivoli OPC 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>
<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.

Figure 47. Examples of Rules
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. From the CREATING AN APPLICATION panel (Figure 45 on page 76), enter the RUN command. You see the RUN CYCLES panel, shown in Figure 48.

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. See “Specifying the Input Arrival Time” on page 89 for additional information.

   Note: See “How Days are Generated Using Rules” on page 84 for details of the effect of specifying an input arrival time earlier than the work-day end time.

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 OPC 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 86 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 87 for a full description.

6 Specify the in-effect and out-of-effect dates. If you leave these blank, Tivoli OPC fills the valid-from date and 71/12/31, 31 December 2071. See “Using In-Effect/Out-of-Effect Dates” on page 89 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 21, “Job Tailoring” on page 273.

Note: When you use rule-based run cycles with periods that you have created, Tivoli OPC 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 dialog.
- You specify the table name on the Modify Current Plan (MCP) dialog 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 49 on page 82.
EQQRULEP --------------------- MODIFYING A RULE -------------------------------

Command ===>
Enter the GENDAYS command to display the dates generated by this rule
Enter $ and user data in the fields below to define a rule

Application : PAYTAXYR YEARLY PAYROLL RUN
Rule : TAXYEAR Run on the third Thursday in July

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

Figure 49. EQQRULEP—Modifying a Rule

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 GENDAYS command to check that the rule will cause Tivoli OPC to select the days 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:
Notes:

a. If you have not specified a calendar for the application, GENDAYS uses the calendar specified under the Tivoli OPC OPTIONS dialog (0.2 from the Tivoli OPC main menu), or the calendar called DEFAULT if no calendar is specified. If the DEFAULT calendar does not exist, all days are considered work days. For long-term planning batch jobs, however, Tivoli OPC uses the calendar specified on the CALENDAR keyword of the BATCHOPT initialization statement, or the DEFAULT calendar. So make sure that you specify the same calendar in the dialog and BATCHOPT (if you do not specify the calendar for the application on the CREATING AN APPLICATION panel in Figure 45 on page 76) or you may find that GENDAYS shows you different days to those generated when you extend the long-term plan.

b. The interval displayable 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 80 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 OPC uses both the free-day rule and the work-day 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 OPC:

1. First generates the days regardless of the free-day rule and work-day end time.

2. Adds one day to the generated dates, if the application input arrival time is before the work-day end time. (If an application input arrival time is 02.00 and the work-day end time is 03.00; when you specify that the application runs on Monday, Tivoli OPC assumes you mean Tuesday at 02.00, because 02.00 on Monday belongs to Sunday.)

3. Takes the work-day 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 work-day end time in the calendar is 06.00. You have specified EVERY DAY in WEEK 21 on the MODIFYING A RULE panel. Figure 50 on page 85 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 work-day 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, Sunday, Tuesday (Saturday and Sunday are transferred to Tuesday at 04.00, considered part of Monday)</td>
</tr>
</tbody>
</table>

Figure 50. Effect of the Input Arrival Time and the Free-day Rule

See “Creating the Default Calendar” on page 60 for details of the work-day end time.

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 OPC 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

In earlier releases of Tivoli OPC, you could 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 48 on page 80), fill in the fields as described in “Creating Run Cycles with Rules” on page 80. 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 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 51.

   Specify a positive or negative offset from the period origin dates.

   For example, using the SEMESTER period on page 66, 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.

   For example, using the SEMESTER period on page 66, 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. See “Using Work-Days-Only Cyclic Periods” on page 67.

Selecting a Free-day Rule

Inform Tivoli OPC whether the rule or offset-based run cycle selects work days only or calendar days (that is, work days and free days). Tivoli OPC 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 48 on page 80).

The possible values of the free-day rule are:

E  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.
1  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.
2  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.
3  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.
4  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 52 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></td>
<td>1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18</td>
</tr>
<tr>
<td>Free days in calendar</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Shaded areas above are free days in the calendar.</td>
</tr>
<tr>
<td>Rule</td>
<td></td>
</tr>
<tr>
<td>Free days excluded</td>
<td>E</td>
</tr>
<tr>
<td>Closest work day before</td>
<td>1</td>
</tr>
<tr>
<td>Closest work day after</td>
<td>2</td>
</tr>
<tr>
<td>Schedule on the free day</td>
<td>3</td>
</tr>
<tr>
<td>Do not schedule at all</td>
<td>4</td>
</tr>
</tbody>
</table>

| Free days excluded      | E       |
| Closest work day before | 1       |
| Closest work day after  | 2       |
| Schedule on the free day| 3       |
| Do not schedule at all  | 4       |

Shaded areas below are the days that Tivoli OPC selects.

Figure 52. The Effect of the Free-Day Rule
Specifying When Your Application Should Not Run

You might want to prevent Tivoli OPC 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 OPC 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 to Tivoli OPC 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 53. Negative Run Cycles

Figure 53 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 THURSDAY of WEEK</td>
<td>4</td>
</tr>
<tr>
<td>E</td>
<td>EVERY (or ONLY) FRIDAY of WEEK</td>
<td>1</td>
</tr>
<tr>
<td>Weekly job run cycles:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>R</td>
<td>EVERY (or ONLY) FRIDAY of WEEK</td>
<td>1</td>
</tr>
</tbody>
</table>

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 Figure 54.

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

Application : SAMPLEA An application_desc

<table>
<thead>
<tr>
<th>Row</th>
<th>Period name</th>
<th>Input Deadline</th>
<th>F day effect</th>
<th>Effect</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td>Text</td>
<td>HH.MM day HH.MM</td>
<td>Type rule</td>
<td>YY/MM/DD</td>
</tr>
<tr>
<td></td>
<td></td>
<td>HH/MM/DD</td>
<td></td>
<td>YY/MM/DD</td>
</tr>
<tr>
<td>DAILY</td>
<td>21.01 01</td>
<td>08.00 N</td>
<td>4</td>
<td>97/01/01</td>
</tr>
<tr>
<td>WEEK</td>
<td>21.01 01</td>
<td>08.00 N</td>
<td>4</td>
<td>97/07/01</td>
</tr>
</tbody>
</table>

Figure 54. Using In-Effect Dates to Switch between Run Cycles

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 OPC attempts to start the occurrence, unless you specify the first operation as time-dependent (see “Creating Time-Dependent Operations” on page 114). Tivoli OPC also uses the input arrival time to resolve external dependencies. See “External Dependencies between Multiple Occurrences” on page 96.
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 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

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.

Figure 55 shows the period definitions that you need, if you specify the Paymore run cycles using periods with offsets.

<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>

Figure 55. Period Definitions Needed for Offset Examples
Daily Run Cycle Example
The payroll jobs required daily (PAYDAILY and PAYBACKP) can use this run cycle:

<table>
<thead>
<tr>
<th>Daily run cycle example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = NORMAL</td>
</tr>
<tr>
<td>Period = WEEK</td>
</tr>
<tr>
<td>Offset = 1,2,3,4,5</td>
</tr>
<tr>
<td>Free-day rule = 4</td>
</tr>
<tr>
<td>Input arrival = 17.00</td>
</tr>
<tr>
<td>Deadline = 01 06.00</td>
</tr>
<tr>
<td>Description = Run Monday to Friday except public holidays</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Weekly run cycle example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = NORMAL</td>
</tr>
<tr>
<td>Period = WEEK</td>
</tr>
<tr>
<td>Offset = 4</td>
</tr>
<tr>
<td>Free-day rule = 1</td>
</tr>
<tr>
<td>Input arrival = 17.00</td>
</tr>
<tr>
<td>Deadline = 01 06.00</td>
</tr>
<tr>
<td>Description = Run Thursday or day before if Thursday is free</td>
</tr>
<tr>
<td>Type = NEGATIVE</td>
</tr>
<tr>
<td>Period = FIRSTTHU</td>
</tr>
<tr>
<td>Offset = 14</td>
</tr>
<tr>
<td>Free-day rule = 1</td>
</tr>
<tr>
<td>Input arrival = 17.00</td>
</tr>
<tr>
<td>Deadline = 01 06.00</td>
</tr>
<tr>
<td>Description = Exclude third Thursday of the month</td>
</tr>
</tbody>
</table>

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.

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.

<table>
<thead>
<tr>
<th>Monthly run cycle example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = NORMAL</td>
</tr>
<tr>
<td>Period = FIRSTTHU</td>
</tr>
<tr>
<td>Offset = 14</td>
</tr>
<tr>
<td>Free-day rule = 1</td>
</tr>
<tr>
<td>Input arrival = 17.00</td>
</tr>
<tr>
<td>Deadline = 01 06.00</td>
</tr>
<tr>
<td>Description = Run third Thursday of the month or day before if free</td>
</tr>
</tbody>
</table>

The yearly processing PAYTAXYR also requires a unique period definition to be sure the application is always scheduled correctly.

<table>
<thead>
<tr>
<th>Yearly run cycle example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type = NORMAL</td>
</tr>
<tr>
<td>Period = FSTTHU7</td>
</tr>
<tr>
<td>Offset = 14</td>
</tr>
<tr>
<td>Free-day rule = 1</td>
</tr>
<tr>
<td>Input arrival = 17.00</td>
</tr>
<tr>
<td>Deadline = 01 06.00</td>
</tr>
<tr>
<td>Description = Run third Thursday in July or day before if free</td>
</tr>
</tbody>
</table>
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 Tivoli OPC 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.
- 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.
- 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 Tivoli OPC operations usually run on general workstations.

Follow these steps to specify the operations in an application:

1. From the CREATING AN APPLICATION panel (Figure 45 on page 76), enter the OPER command. You see the OPERATIONS panel, shown in Figure 56.

   ![Figure 56. EQQAMOPL—Operations](image)

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. See “Using Feedback Options” on page 105 for details of duration feedback.

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 OPC uses this to find the JCL for job or started-task operations. See “Associating Job Statements with Tivoli OPC Operations” on page 110. 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. See “Using WTO Operations” on page 112 and the description of the DEADLINE WTO option on page 105 for more details. 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 predecessors to the operation you are creating. See “Specifying Operation Predecessors” on page 94 for more details of operation predecessors and successors.

7 Specify the details of each operation by entering $ next to the operation you want to specify. See “Specifying Operation Details” on page 95.

8 Check the JCL of the associated jobname 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. See “Editing JCL Operations” on page 110 for more details.

Examples of Operations

There is no need to restrict Tivoli OPC 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 in Tivoli OPC.

The operations in the example for Figure 56 on page 92 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, Tivoli OPC 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, Tivoli OPC 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 Tivoli OPC 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:

```plaintext
OPSTAT JOBNAME(PAYDAILY) STATUS(C) WNAME(WTO1)
```

The EQQEVPGM program creates a job-tracking event that is communicated to Tivoli OPC by the same means that job start and job end events are. When the Tivoli OPC 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 Operation Predecessors

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 termed a successor of A1. An operation can have many predecessors and successors. These relationships between operations are called dependencies.

Dependencies are specified between operations. You cannot specify a dependency between applications or application occurrences using the Application Description dialog. You can create dependencies between occurrences with the long-term plan dialog, 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 automatically generated when you create job descriptions using the Job Description dialog, 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 57 on page 95.
Applications

Enter/Change data in the rows, and/or enter any of the following row commands:

- I(nn) - Insert, R(nn), RR(nn) - Repeat, D(nn), DD - Delete
- S - Select operation details, J - Edit JCL

Enter the PRED command above to include predecessors in this list, or, enter the GRAPH command to view the list graphically.

---

Figure 57. EQQAMOSL—Operations

You can specify external predecessors, or additional internal predecessors, by selecting an operation using the S row command.

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 dialogs.

Specifying Operation Details

If you select an operation from the OPERATIONS panel (Figure 56 on page 92 or Figure 57) using the row command S, the OPERATION DETAILS panel is displayed (Figure 58 on page 96).

This section describes the following operation details in the order in which they appear on the OPERATION DETAILS panel.

1. External predecessors or additional internal predecessors
2. Workstation resources and servers
3. Special resources
4. Automation options
5. Feedback options
6. Time specifications
7. Operator instructions
8. Edit JCL
Applications

Specifying External Predecessors

You specify external predecessors and additional internal predecessors by selecting option 1 from the OPERATION DETAILS panel. The PREDECESSORS panel (Figure 59) is displayed:

![Figure 59. EQQAMPDL—Predecessors](image)

External Dependencies between Multiple Occurrences

If you have specified an external dependency between two operations, Tivoli OPC 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 Tivoli OPC terminology, the dependency is resolved) by Tivoli OPC during the LTP planning process.

To resolve an external dependency, Tivoli OPC uses the input arrival times of the occurrences or, if they have been specified, the input arrival times of the individual operations. See “Specifying the Input Arrival Time” on page 89 and “Specifying...”
Input Arrival Times and Deadlines for Operations” on page 108 for an explanation of input arrival times. 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 OPC 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 Tivoli OPC dialogs), 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 Tivoli OPC 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 Tivoli OPC operations. See Chapter 3, “Creating Special Resources” on page 41 for information about resources.

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 dataset, it is good practice to make the resource name the dataset name.
When you create an operation that uses a special resource, select option 3 (SPECIAL RESOURCES) from the OPERATION DETAILS menu. You see the SPECIAL RESOURCES panel, shown in Figure 60.

```
EOQAMSRL --------------------- SPECIAL RESOURCES ------------ ROW 1 TO 1 OF 1
Command ===>
Scroll ===>
PAGE

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

Operation : CPU 030

Row Special Qty Shr Keep On
cmd Resource Ex Error
'''' payroll.database____________________________ 1_____ x _

Figure 60. EQQAMSRL—Special Resources

Type values in the fields on the SPECIAL RESOURCES panel:

**Special Resource**
The name of the resource, up to 44 characters. You can use the global search characters % and * if you are unsure of the name—Tivoli OPC displays a list of matching names. For example, if you specify PAY*, Tivoli OPC 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 OPC does not start another operation that requires the allocated units until the first operation ends.

This is like dataset 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

Whether the operation keeps the resource when it fails:

Y (keep it)
N (free it)
Blank (take the default for that resource)

Use the Special Resource dialog to create and modify special resources (option 1.6 from the Tivoli OPC 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 OPC knows how many of each resource is available on that workstation. See “Specifying Workstation Fixed Resources” on page 37. When deciding whether to start the operation, it considers the available workstation resources and the resources required for this operation. Normally, Tivoli OPC starts an operation only if the needed quantity of a resource is available at the workstation. However, when creating the workstation, you can specify that resource usage will not be considered when Tivoli OPC is either producing its schedule or starting operations.

When you create workstations (see “What Types of Workstation Are There?” on page 23), you specify the quantity of workstation resources available. Tivoli OPC 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 OPC has no knowledge of the actual state of the resources. That is, Tivoli OPC can only keep track of resource users that are operations in the current plan. In addition, Tivoli OPC is only aware of the total value specified on the workstation definition, or the modified amount in the current plan. Tivoli OPC 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 told Tivoli OPC in the application description that operations A, B, and C (all jobs) use 2 tape drives each. Tivoli OPC 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 OPC in the order A, B, C, this means that C is waiting for a tape drive.

Tivoli OPC 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 OPC 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 OPC 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 WORKSTATION RESOURCES AND SERVERS panel (Figure 61), displayed by selecting option 2 on the OPERATION DETAILS panel (Figure 58 on page 96).

![EQQAMWRP -- WORK STATION RESOURCES AND SERVERS](image)

Figure 61. EQQAMWRP—Workstation 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 OPC will start the operation. You can choose whether Tivoli OPC takes parallel servers into consideration when it plans the schedule, when it starts an operation, at both 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 37.

Tivoli OPC assumes that operations on computer workstations always use a single server.

Specify the number of servers for an operation on the WORKSTATION RESOURCES AND SERVERS panel (Figure 61). You can display this panel by selecting option 2 on the OPERATION DETAILS panel (Figure 58 on page 96).
Specifying Options for Automation

Selecting AUTOMATIC OPTIONS, option 4, from the OPERATION DETAILS panel takes you to the JOB, WTO, AND PRINT OPTIONS panel (Figure 62).

![EQQAMJB empty](https://example.com/eqqamjbe.png)

**Figure 62. EQQAMJB—Job, WTO, and Print Options**

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.

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 dialog 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. 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 227 for more details.

**CATALOG MANAGEMENT (default: N)**

If you specify Y, Tivoli OPC automatically performs catalog management actions for operations that end in error:

- Datasets created in the job are automatically deleted and uncataloged.
- Generations of GDG datasets created in the job are reset.
- Datasets uncataloged in the job are cataloged.

If you specify D (deferred), Tivoli OPC does not automatically perform catalog management actions for operations that end in error, but these actions can be requested for the operation from the Modify Current Plan dialog.

N specifies that catalog management actions will not be performed for this operation.

**Note:** Specifying catalog management actions for an operation running on a workstation with a user-defined destination ID has no effect.

For a description of the Tivoli OPC catalog management function, see Chapter 18, “Catalog Management” on page 235.

**CRITICAL (default: N) and POLICY (default: ‘ ’)**

If you specify Y in the CRITICAL field, Tivoli OPC automatically sends a request to the Workload Manager (WLM) to promote a job or started task in the high-performance service class, appropriately defined for batch jobs in the WLM environment, whenever the conditions of the specified assist policy are reached. Valid policies that you can enter in the POLICY field are:

- **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 OPCOPTS.

For a description of the Tivoli OPC WLM function, see Chapter 22, “How Tivoli OPC Selects Work to be Assisted by WLM” on page 313.

**SUBMIT (default: Y)**

If you specify Y, Tivoli OPC 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 OPC 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 dialog (see Chapter 12, “Using Tivoli OPC Service Functions” on page 205).

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).

\( Y \) 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 will be reset to ready status only when:
1. You specify this in the MCP dialog 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 dialog, 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).

\( \text{blank} \)

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 OPC 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.

\( Y \) The operation is eligible to be rerouted if the workstation becomes inactive, and:
1. You specify this in the MCP dialog 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 dialog, 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 refer to *Controlling and Monitoring the Workload*.

**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.

**HOLD/RELEASE (default: Y)**
Use this option to control held jobs that are not submitted by Tivoli OPC.

If you place non-Tivoli OPC-submitted jobs in HOLD status (for example, by specifying TYPRUN=HOLD on the job card), and HOLD/RELEASE is set to Y, Tivoli OPC 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 OPC releases a held job immediately without reference to its schedule.

**Note:** Specifying a HOLD/RELEASE value Y for a job operation does not cause Tivoli OPC to put the job into HOLD status. If the job is already in HOLD status, however, Tivoli OPC 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 Tivoli OPC 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.

Ensure that the combination of job or started-task name, form number, and SYSOUT class is unique to the print dataset that you want Tivoli OPC to monitor. If this is not the case, the print operation might be marked as complete when another print dataset, which matches the selection criteria, is printed or purged.

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. Refer to *Customization and Tuning* for more information about the PRTCOMPLETE keyword.
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 OPC will not start the operation until the operation input arrival time is reached. If Tivoli OPC 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 114.

If you specify N, the operation will not be time-dependent. Tivoli OPC 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 OPC 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 114 for details of what happens to late time-dependent operations.

If you specify N, Tivoli OPC 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 OPC issues the operator message EQQW776I when an MVS 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 Tivoli OPC message log (EQQMLOG). The WTO is issued only for MVS 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 56 on page 92).

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 113 for details.

Specifying Y for an operation on a workstation that is not an MVS destination will have no effect. That is, a deadline WTO will not be routed to the workstation destination.

Using Feedback Options

Tivoli OPC 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 dataset 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 in Tivoli OPC, the smoothing factor and the limit for feedback, control how measured durations are used. Any value you specify here overrides the installation default specified on the JTOPTS keyword.
Note: The feedback limit used when selecting operations for which a long duration alert is issued is the value of the LIMFDBK keyword of JTOPTS. The limit for feedback value that can optionally be entered in the application description is not used.

Select option 5 from the OPERATIONS DETAILS panel. Specify the feedback options on the panel shown in Figure 63 to automatically adjust the estimated duration in the database after the job completes.

```
EQQAMFBP ---------------------- FEEDBACK OPTIONS ----------------------
Command ==>

Enter/Change data below:
Operation : CPU1 020
SMOOTHING FACTOR ==》 050 A value 0 to 999 where 0=no smoothing
FEEDBACK LIMIT ==》 200 A value 100 to 999 where 100=no feedback
```

Figure 63. EQQAMFBP—Feedback Options

**Smoothing Factor**

The 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 dataset will not be updated.

The new estimated duration is calculated as follows:

```
New estimated duration

ND = OD + ((AD - OD) * SF/100)
```

where:

- ND is the new estimated duration to be stored in the AD database.
- OD is the old estimated duration stored there.
- AD is the measured duration.
- SF is the smoothing factor.

Figure 64 shows some examples of how the smoothing factor algorithm works.
### Limit for Feedback

The limit for 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:

#### Limits for feedback

\[
\text{Lower limit} = \frac{\text{OD} \times 100}{\text{LF}} \\
\text{Upper limit} = \frac{\text{OD} \times \text{LF}}{100}
\]

where:

- \(\text{OD}\) is the old estimated duration stored in the application description database.
- \(\text{LF}\) is the limit for feedback.

Figure 65 shows some examples of how the limit for feedback algorithm works.

<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>

*Figure 65. Examples of Limits for Feedback*
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 66) which you can reach by selecting option 6 from the OPERATION DETAILS panel (Figure 58 on page 96).

```
EQQAMTMP  ------------------- TIME SPECIFICATIONS  -------------------
Command ===>

Enter/Change data below:

Application time specifications:
Input arrival time : 21.00
Deadline day/time : 01 04.00

Operation : CPU1 010

Operation input arrival:
DAY ===> __ The day the input arrives for operation, relative to application start day (0 means the start day).
TIME ===> ____ Arrival time of the input in the format HH.MM

Operation deadline:
DAY ===> __ Deadline day for operation completion, relative to application start day (0 means the start day).
TIME ===> ____ Deadline time of deadline day in the format HH.MM
```

Figure 66. EQQAMTMP—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 OPC 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 OPC to include free days when calculating deadline dates—refer to the description of the OPERIALL and OPERDALL keywords of the BATCHOPT statement in 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 dialog or the Operator Instruction dialog.

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 69 on page 110).
If you select option 7 from the OPERATION DETAILS panel, (Figure 58 on page 96), then the LIST OF OPERATOR INSTRUCTIONS panel (Figure 67 on page 109) is displayed.

<table>
<thead>
<tr>
<th>Command</th>
<th>List of Operator Instructions</th>
<th>Row 1 to 3 of 3</th>
</tr>
</thead>
</table>

Enter the CREATE command to create a new instruction, or enter any of the row commands below:

- B - Browse, M - Modify, C - Copy, D - Delete

<table>
<thead>
<tr>
<th>Row</th>
<th>Application id</th>
<th>Operation</th>
<th>Valid from</th>
<th>Valid to</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>cmd</td>
<td>number</td>
<td>date</td>
<td>time</td>
<td>date</td>
<td>time</td>
</tr>
<tr>
<td>PAYINOUT</td>
<td>010</td>
<td>19980101 10.00</td>
<td>19980131 10.00</td>
<td>003</td>
<td></td>
</tr>
<tr>
<td>PAYINOUT</td>
<td>010</td>
<td>19980301 10.00</td>
<td>19980331 10.00</td>
<td>002</td>
<td></td>
</tr>
</tbody>
</table>

Figure 67. EQQALSML—List of Operator Instructions

From here it is possible to create/update/delete/browse operator instructions in the same way as from the Operator Instruction dialog (option 1.5 on the Tivoli OPC 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 on the CREATING AN OPERATOR INSTRUCTION panel (Figure 68).

<table>
<thead>
<tr>
<th>Command</th>
<th>Creating an Operator Instruction</th>
<th>Row 1 to 3 of 3</th>
</tr>
</thead>
</table>

Edit instruction text below:

<table>
<thead>
<tr>
<th>APPLICATION ID</th>
<th>OPERATION</th>
<th>VALID FROM</th>
<th>VALID TO</th>
<th>TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYINOUT</td>
<td>010</td>
<td>19980130 10.00</td>
<td>Format: CCYMMDD HH.MM</td>
<td></td>
</tr>
<tr>
<td>PAYINOUT</td>
<td>010</td>
<td>19981231 10.00</td>
<td>Format: CCYMMDD HH.MM</td>
<td></td>
</tr>
</tbody>
</table>

In the unlikely event that this job should fail, please refer to the call roster for PAYROLL systems and page as soon as possible.

Figure 68. EQQACRTE—Creating an Operator Instruction

While handling operator instructions from the Application Description dialog menu, and returning from the LIST OF OPERATOR INSTRUCTIONS panel, the operator
Applications

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 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 69).

<table>
<thead>
<tr>
<th>Application id</th>
<th>Operation</th>
<th>Valid From</th>
<th>Valid To</th>
<th>Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYINOUT</td>
<td>010</td>
<td>19980101</td>
<td>10.00</td>
<td>19980131 10.00 003</td>
</tr>
<tr>
<td>PAYOUT</td>
<td>010</td>
<td>006</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 69. EQQAOIDP—Confirm the Deletion of OI

**Editing JCL Operations**

You can edit the JCL of an operation if a jobname exists, and if you have a tool for editing JCL and have specified its name in the 0.6 OPTION panel.

See Chapter 2. Using the Tivoli OPC Dialogs - Setting Options in the *Controlling and Monitoring the Workload* manual.

**Associating Job Statements with Tivoli OPC Operations**

The main purpose of Tivoli OPC is to start work according to a predefined schedule, the current plan. The current plan is produced from information in the Tivoli OPC database and LTP. Before Tivoli OPC can submit a job or start a started task, it must have job statements (JCL in the case of the MVS operating system). Tivoli OPC 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**

Store job statements for Tivoli OPC-submitted jobs in the partitioned datasets that are allocated to the ddname EQQJBLIB. Tivoli OPC associates an operation with a member of one of these libraries, using the *job name* field on the panel in Figure 57 on page 95. That is, the job statements for a job or started task are stored in the member identified by the *job name* field.
Tivoli OPC 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 Tivoli OPC job repository (a cycle of VSAM datasets with the ddname 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 OPC 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 by Tivoli OPC at job or started-task submission time, if you require. Alternatively, operators can be prompted for the values of variables before submission. See Chapter 21, “Job Tailoring” on page 273 for more details.

Notes:

1. Tivoli OPC 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 MVS. 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 OPC cannot track the job. You must not specify JOBCHECK(SAME). This applies only to jobs destined for MVS.

3. Do not use JCL that has been packed by ISPF in EQQJBLIB, because Tivoli OPC does not use ISPF routines to read it.

4. Do not include JCL with TYPRUN=SCAN in EQQJBLIB. Tivoli OPC does not track these jobs. To test JCL, do this outside Tivoli OPC, or add TYPRUN=SCAN using the MCP dialog and type the SUBMIT command; then remove TYPRUN=SCAN or cancel the edit. This applies only to jobs destined for MVS.

If you use Tivoli OPC to submit jobs to non-MVS 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 Tivoli OPC 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 dataset 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 OPC. Tivoli OPC 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 dataset, 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 104.

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 56 on page 92), is also issued as part of the WTO.

You can schedule a WTO operation like any other Tivoli OPC operation: it can have predecessors, specify resources, and be time-dependent. The WTO is issued only if the operation is automatically started by Tivoli OPC, 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 25 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 OPC 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 110). 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 OPC procedure, and a START command is issued to invoke it. Refer to Installation Guide for details of the procedure libraries required.

---

**Scheduling the Closedown of Online Systems and Started Tasks**

Tivoli OPC can initiate and track a job or started task but cannot directly stop it. The recommended method for scheduling the stop of an MVS 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 108).

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 105).

   After the operation is started, it will reach its deadline for completion, and Tivoli OPC issues message EQQW776I as a WTO (if the workstation is an MVS 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 EQQNETW1 in the Tivoli OPC sample library for an example in the REXX language.

When the started task has terminated normally, Tivoli OPC sets the operation to complete.
Creating Time-Dependent Operations

The input arrival time of an operation is usually not the time that Tivoli OPC will start an operation. Tivoli OPC 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 OPC will usually start the operation regardless of its input arrival time or the time of day.

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 62 on page 101). See “What Happens to Late Operations?” on page 115 for a map showing when Tivoli OPC starts a time-dependent operation, and what happens if it is late.
What Happens to Late Operations?

001

Is the operation time-dependent?

Yes  No

002

Tivoli OPC starts the operation as soon as it is added to the current plan, or as soon as its dependencies are met.

003

Is it the input arrival time?

Yes  No

004

– Wait for the input arrival time. Continue at Step 003.

005

Is the operation ready?

Yes  No

006

– Wait until the operation is ready. Continue at Step 005.

007

Does the operation have the suppress-if-late attribute (see “Options that Apply to All Operations” on page 105)?

Yes  No

008

– Tivoli OPC starts the operation.

009

Does the SUPPRESSPOLICY allow more time for this operation?

Yes  No

010

– Give the operation a status according to the SUPPRESSACTION option (C, E, or RL).

011

– Tivoli OPC starts the operation.

An operation with the status RL can be started only after manual intervention. Refer to Customization and Tuning for details of the SUPPRESSPOLICY and SUPPRESSACTION keywords of the JTOPTS statement.
Chapter 7. Creating Job Descriptions

This chapter describes the Job Description dialog, which lets you create job, started-task, and WTO operations using a faster path than the one available with the Application Description dialog.

A job description is a Tivoli OPC 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 Chapter 6, “Standard Applications and Group Definitions” on page 75. 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 dialog.

You can change a job description with the Application Description dialog, 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 Dialog

The information that you must provide to create a job description is described more fully in Chapter 6, “Standard Applications and Group Definitions” on page 75, 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 OPC/ESA DATABASES panel, or by entering 1.8 from the Tivoli OPC main menu. You see the MAINTAINING JOB DESCRIPTIONS panel:

   EQQJSUBP ------------ MAINTAINING JOB DESCRIPTIONS ---------------
   Option ===>
   Select one of the following:
   1 BROWSE - Browse jobs
   2 CREATE - Create a job
   3 LIST - List jobs for further processing
             (browse, modify, copy, delete, print, calculate and print rundays, modify LTP)
   4 PRINT - Perform printing of jobs
   5 MASS UPDATE - Perform mass updating of jobs

   Figure 70. EQQJSUBP—Maintaining Job Descriptions
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:

![EQQJCGRP — Creating a Job]

3 On the CREATING A JOB panel, specify the characteristics of your main operation and up to two predecessor operations:

**JOBNAME – TEXT**
This is the name of the main operation. It 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 dialog 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 dialog instead if you need a double-byte character set application ID.

**OWNER: ID – TEXT**
This is the owner ID and a description.

**CALENDAR ID**
If you leave this blank, Tivoli OPC 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 Tivoli OPC OPTIONS dialog (0.2 from the Tivoli OPC 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 may have several calendars, but always call your default calendar DEFAULT, and specify the same calendar name on BATCHOPT and in the dialog.

**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
Specifies 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 OPC 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 114 for details of time-dependent operations.

WORK STATION
Specifies the workstation of the main operation.

PRIORITY
Specifies 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.

HIGHEST RETURN CODE
This field specifies 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 227 for more details.

MANUAL INTERACTION and MANUAL WS
Specifies a text for a manual operation that will precede this computer operation, and specifies 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 120 for more information.

PREDECESSORS
 Specifies names of other job descriptions that will be 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 Operation Predecessors” on page 94 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), an allocation type of S (shared) or X (exclusive), and a keep-on-error value of N (free), Y (keep), or blank (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:

```
PREDECESSORS ===> ________ ________ ________ ________ ________
SPECIAL ===> PAYROLL.DATABASE____________________________ /zerodot/zerodot/zerodot/zerodot/zerodot1 X Y
RESOURCES LINES.TO.LONDON_____________________________ /zerodot/zerodot/zerodot/zerodot25 X N
GROUP DEFINITION ===> ______________________________
```

**GROUP DEFINITION**

Specifies a group name, if this job description is to be part of a group.

See Chapter 6, “Standard Applications and Group Definitions” on page 75 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:

```
RUN CYCLES ===> MON_____- .001 TUE_____- .001 WED_____- .001 THU_____- .001 FRI_____- .001 SAT_____- .001 +
```

*Figure 72. 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 78. You can specify up to six periods and offsets, as shown in Figure 72, 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 48 on page 80) 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 72.
Specifying Additional Operation Details for Job Descriptions

If you enter the DETAILS command in the Job Description dialog, 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 95 for more information.

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

Select one of the following:
1 PREDECESSORS - List of predecessors
2 WS RES AND SERVERS - Work station resources and servers
3 SPECIAL RESOURCES - List of special resources
4 AUTOMATIC OPTIONS - Job, WTO, and print options
5 FEEDBACK - Feedback options
6 TIME - Time specifications
7 OP INSTRUCTIONS - Operator instructions

Application : PAYDAILY  daily payroll jobs
Operation : CPU1 020  Runs pay04 and pay06
Duration : 0.05
Jobname : PAYDAILY
Number of int preds : 1
Number of ext preds : 0
```

Figure 73. EQQAMSDP—Operation Details

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.
Chapter 8. Defining Applications in Batch

This chapter describes:

- 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

Mass Update or Batch Loader?

There are two utilities that process applications in batch; the batch loader and the mass update utility. Use Figure 74 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 dataset with your application descriptions,</td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>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</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>original values of options and attributes.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce a report showing all applications that have specified</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>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 Dataset” on page 384 for details.

Figure 74. Comparison of Batch Loader (BL) and Mass Update (MU)

Maintaining a Sequential Dataset

You may want to maintain a sequential dataset 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 OPC for the first time, if you already have application descriptions in some format that you can convert.
3. You may 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 OPC and are developing naming standards.
Defining Applications in Batch

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.
- 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 Dataset” on page 384 for details of EQQOIBLK.

Making Changes Conditional on the Original Values
You may want to change all priority 5 applications to priority 4, for example.

Searching Applications for Values in Specified Fields
You may 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 Y, but in trial mode. (If you change from Y to Y, you don't embarrass yourself if you forget to specify 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 dialog 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 75 on page 125.
Defining Applications in Batch

EQQAUPDL -------- MASS UPDATING OF APPLICATION DESCRIPTION  ROW 1 TO 10 OF 51
Command ===> Scroll ===> CSR

Enter the row command S to create pending updates for a data item.
Enter the CLEAR command above to delete all pending updates.

Number of pending updates in this session: 0

TYPE OF BATCH JOB ===> _ T - Trial run, U - Updating run

<table>
<thead>
<tr>
<th>Row</th>
<th>cmd Data item</th>
<th>NEW</th>
<th>GEN - APPLICATION TEXT</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>' GEN - OWNER ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>' GEN - OWNER TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>s GEN - PRIORITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>' GEN - AUTHORITY GROUP ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>' GEN - CALANDER ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>' GEN - APPLICATION GROUP ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>' RUN - TEXT</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>' RUN - PERIOD/RULE NAME</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>' RUN - INPUT ARRIVAL TIME</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 75. EQQAUPDL—Mass Updating of Application Description

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. You see the UPDATING DATA ITEM panel, shown in Figure 76:

   ```plaintext
   Figure 76. EQQAUUGL—Updating Data Item
   ```
5 Enter the $ row command to select a subset of applications. You see the SPECIFYING FILTER CRITERIA panel, shown in Figure 77:

<table>
<thead>
<tr>
<th>EQQAUFIP ---------------- SPECIFYING FILTER CRITERIA -------------------</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command ==&gt;&gt; Specifying filter criteria below:</td>
</tr>
<tr>
<td>OWNER ID ===&gt; sample__________  Restrict the updating of applications to applications with a certain owner.</td>
</tr>
<tr>
<td>PERIOD/RULE NAME ===&gt; ________  Restrict the updating of runcycles to runcycles with a certain period name.</td>
</tr>
<tr>
<td>JOB NAME ===&gt; ________  Restrict the updating of operations to operations with a certain job name.</td>
</tr>
<tr>
<td>WORK STATION NAME ===&gt; ____  Restrict the updating of operations to operations with a certain work station name.</td>
</tr>
</tbody>
</table>

Figure 77. EQQAUFIP—Specifying Filter Criteria

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:
- The target field is 8 bytes long
- Three instances of the target field contain PAYUPD, PAYUPDAT, and UPDATE
- Old data is UPD
- New data is BKUP

After the mass update:
- The updated fields contain PAYBKUP, PAYBKUPA, and BKUPATE respectively.
8 Press PF3 (End). You return to the MASS UPDATING OF APPLICATION DESCRIPTION panel.

9 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 all changed fields. Type t in the TYPE OF BATCH JOB field to specify a trial run, and press PF3 (End). You see the GENERATING JCL FOR A BATCH JOB panel, shown in Figure 78:

```
EQQXSUBP -------------- GENERATING JCL FOR A BATCH JOB ------------------
Command ===>
Enter/change data below and press ENTER to submit/edit the JCL.
JCL to be generated for: MASS UPDATE OF APPLICATIONS
SYSOUT CLASS ===> _ (Used only if output to system printer)
LOCAL PRINTER NAME ===> ________ (Used only if output on local printer)
DATASET NAME ===> XRAYNER.EIDA.ADMUP.LIST (Used only if CLASS and LOCAL PRINTER are both blank). If blank default name used is XRAYNER.EIDA.ADMUP.LIST
SUBMIT/EDIT JOB ===> S S to submit JOB, E to edit
Job statement :
  ===> //XRAYNERE JOB (890122,NOBO),'SIMON RAYNER',
  ===> // MSGCLASS=H,NOTIFY=XRAYNER,CLASS=A
  ===> //OUTPUT1 OUTPUT DEST=LAB21,DEFAULT=YES
  ===> //*
```

Figure 78. EQQXSUBP—Generating JCL for a Batch Job

10 Check the report to see the applications that have been found with priority 5.

11 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 Tivoli OPC batch-loader program 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 OPC dialog.

This chapter does not describe fully all the application options. Read Chapter 5, “Things You Should Consider before Creating Applications” on page 69 and Chapter 6, “Standard Applications and Group Definitions” on page 75 before using the batch loader.
Defining Applications in Batch

Input

Input to the batch loader is a set of control statements in an input dataset. 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 OPC subsystem
- A database that you set up yourself, that is, a VSAM key-sequenced dataset (KSDS)

If the batch-loader output is directed to an active Tivoli OPC subsystem, you specify which Tivoli OPC 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 dataset, the batch-loader program is run independently of any Tivoli OPC subsystem; it is not necessary to start a Tivoli OPC subsystem. This independently created database can then be allocated to an active Tivoli OPC subsystem at a later time.

Validity Checking

The batch loader checks the validity of the information you provide in the input dataset. However, validity checking is optional if the output is directed to a VSAM dataset. 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 OPC 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

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 3, “Creating Special Resources” on page 41 for information about special resources.

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.
Defining Applications in Batch

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 dataset, 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 and ADSR must follow the ADOP that they belong to, because they build subsections of an operation section.
- An ADRULE statement must immediately follow its associated ADRUN statement.

The example in Figure 79 on page 131 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—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')
      ODESCR('SAMPLE APPLICATION') PRIORITY(5)
      OWNER(SAMPLE)
ADRULE ACTION(SETDEFAULT) TYPE(R)
ADSTART ADID(PAYDAILY) DESC('DAILY PAYROLL JOBS')
ADRUN NAME(DAILY) RULE(1) DESC('RUN EVERY WORK DAY')
      IATIME(1200) DLTIME(1600)
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)
ADSR  RES(PAYROLL.DATABASE) USAGE(X)
ADSTART ADID(GPAYW) DESC('WEEKLY PAYROLL GROUP') ADTYPE(G)
ADRUN NAME(THURS) RULE(1) DESC('RUN ON THURSDAY')
      IATIME(1200) DLTIME(1600)
ADRULE EVERY DAY(THURSDAY) YEAR
ADSTART ADID(PAYW) DESC('WEEKLY PAYROLL JOBS') ADGROUP(GPAYW)
ADOP  WSID(SETP) OPNO(10) JOBN(PAYWEEK)
      DESC('SETUP FOR PAYWEEK') DURATION(5)
ADOP  WSID(CPU1) OPNO(20) JOBN(PAYWEEK)
      DESC('PAYWEEK JOB') DURATION(5) PREOP(10)
ADDEP PREWSID(CPU1) PREOP(/zerodot2/zerodot) PREADID(PAYDAILY)
      DESC('WAIT FOR PAYDAILY')
OISTART ADID(PAYW) OPNO(/zerodot2/zerodot)
OIT  'Please note...'
OIT  'If this job (PAYWEEK) fails, automatic recovery'
OIT  'will be attempted for some situations.'

Figure 79. Example of Batch-Loader Control Statements

You can find simple examples at the end of each control statement section and in
"Batch-Loader Sample" on page 160.
### Batch-Loader Output—Tivoli OPC System or VSAM Dataset?

<table>
<thead>
<tr>
<th>If you update the active subsystem:</th>
<th>If you use independent VSAM datasets:</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Tivoli OPC subsystem must be active.</td>
<td>The Tivoli OPC subsystem need not be active.</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 OPC subsystem, allocate the new datasets, and restart the Tivoli OPC subsystem, before the new application descriptions and operator instructions can be used.</td>
</tr>
<tr>
<td>You may 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 datasets and rerun the batch loader if things go wrong.</td>
</tr>
<tr>
<td>There can be performance problems if your Tivoli OPC 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 OPC subsystem.</td>
</tr>
</tbody>
</table>

*Figure 80. Deciding Whether to Update the Active Subsystem*

### Batch-Loader Security Considerations

If you are updating the databases of a Tivoli OPC subsystem, you need the same authority that would be required if you were to make the updates using the Tivoli OPC dialogs. You will need access to these resource codes:

- **AD** For the application descriptions
- **OI** For the operator instructions

### Batch-Loader Control Statements

The batch-loader control statements are:

- **OPTIONS** Defines execution options for the batch loader
- **ADSTART** Starts the creation of a new application description
- **ADRUN** Describes when the application is to run
- **ADRULE** Provides additional information to the ADRUN statement for rule-based run cycles
- **ADOP** Defines an operation in the application
- **ADSR** Specifies special resources required by an operation
- **ADDEP** Specifies an operation that is a predecessor to the operation being built
Defining Applications in Batch

OISTART               Starts the creation of a new operator instruction
OIT                  Specifies a line of text in the operator instruction that is under construction

Syntax and Construction

Batch loader control statements follow the same syntax rules as TSO commands.

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:

ccontrolstatementname  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. 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. Refer to 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 OPC 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 dataset, or until you specify another ACTION(SETDEFAULT). For example:

ADOP  ACTION(SETDEFAULT)  OPNO(1/zerodot)  DURATION(5)

In this example, the default values set for the keywords OPNO and DURATION are used for all following ADOPs in the input dataset. 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.
Defining Applications in Batch

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:
   a. A character value defaults to blank.
   b. An integer value defaults to 0.

OPTIONS

Use the OPTIONS control statement to define execution options for the batch loader. None of the keywords are required. If present, the OPTIONS statement must be the first statement in the input dataset.

Syntax

```
OPTIONS control statement
   ACTIONS
   SUBSYS(subsystem name)
   ADID
   EBCDIC
   OWNER
   MSGLEVEL
```

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 dataset. The batch loader does not produce any output other than error messages.

ACTION(ADD) specifies that you are adding new ADs and Ols. 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 Ols defined in the input dataset can replace existing ones in the database if they have the same name.

SUBSYS (subsystem name)
The SUBSYS keyword indicates which Tivoli OPC subsystem the output will be directed to. If this keyword is not specified, output will be directed to the VSAM datasets 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 OPC subsystem, they are ignored.

**Note:** Do not use the datasets defined by EQQADDS and EQQOIDS if they are allocated to an active subsystem. Otherwise, an error message is issued and batch-loader processing ends.

**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 dataset is produced, so do not specify the CHECK keyword.

**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.

**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.

**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.
Defining Applications in Batch

Examples

```plaintext
OPTIONS
OPTIONS SUBSYS(OPC1)
```

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), and MSGLEVEL(2).

**ADSTART**

Use the ADSTART control statement to signal the start of an application description. When this statement is found in the input dataset, 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 dataset.

**Syntax**

```
ADSTART control statement
  ADSTART
    ACTION(ADD, SETDEFAULT)
    ADID(application ID)
    ADTYPE(A, G)
    ADGROUPID(group definition ID)
    ADVALFROM(yymmdd)
    CALENDAR(default calendar name)
    DESCR('descriptive text')
    GROUP(authority group)
    OWNER(owner ID)
    ODESCR('owner description')
    PRIORITY(priority)
```
Values Not Set by SETDEFAULT
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.

ADTYPE (A | G)
Application type indicator. A is for applications, G is for group definitions.

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 parameter is valid only for an ADTYPE of A and should not be specified
with CALENDAR.

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 OPC 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.

GROUP (authority group)
The name of the application authority group to be used for additional authority
checking. You can specify up to 8 characters.
Defining Applications in Batch

**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.

**ODESCR** (*'owner description'*)

A free-format description of the application owner up to 24 characters and contained in single quotation marks.

**PRIORITY** (*priority | 5*)

The scheduling priority of the application. Must be a single digit in the range 1–9. This parameter is valid only for an ADTYPE of A.

### Examples

**Setting defaults**

```
ADSTART ACTION(SETDEFAULT) OWNER(PAYGRP) PRIORITY(6)
    ODESCR('PAYROLL GROUP')
```

This example sets defaults for all following ADSTART statements.

```
ADSTART
    ADID(REORG7) PRIORITY(9) DESCR('Reorganize IMS databases')
    ADVALFROM(99/1/1) OWNER(XDARVOD)
```

In this example, the batch loader will create application REORG7, which is an urgent application. It will be valid for inclusion in Tivoli OPC plans on the first day of 1999.

### ADRUN

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 78 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.
Syntax

**ADRUN control statement**

```
ADRUN NAME( rule name ) ACTION( SETDEFAULT | ADD ) PERIOD( period name )
```

```
RULE(1) VALFROM( yymmdd ) VALTO( yymmdd )
```

```
DESCR('run cycle description') TYPE( X | N | R | E )
```

```
IADAYS( start day from start of period ) EIA DAYS( start day from end of period )
```

```
IATIME( hhmm ) EITIME( deadline day ) DLTIME( hhmm )
```

```
ADRJTAB( variable table name )
```

**Values Not Set by SETDEFAULT**

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.

**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.
**Defining Applications in Batch**

**RULE (1 | 2 | 3 | 4 | E)**
*Defines which free-day rule is in effect. See “Selecting a Free-day Rule” on page 87.*

**VALFROM (yyymmd | current date)**
The start date of validity of this run cycle, in the format yyymmd. See the note under VALTO.

**VALTO (yyymmd | 711231)**
The end date of validity of this run cycle, in the format yyymmd.

**Note:** Tivoli OPC interprets the yy part in the VALTO and VALFROM keywords as follows:

**YY Year**
- 72 - 99 1972 - 1999
- 00 - 71 2000 - 2071

**DESCR ('run cycle description')**
A free-format description of the run cycle, up to 50 characters and contained in single quotation marks.

**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.

**IADAYS (start day from start of period | 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.

**EIADAYS (start day from end of period | 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.

**IATIME (hhmm)**
The time, in the format hhmm, that the application is to arrive at the first workstation.

**DLDAY (deadline day | 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.
Defining Applications in Batch

**DLTIME (hhmm)**

The time on the deadline day that the application should be completed by, in the format *hhmm*.

**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 OPC ignores any JCL variable table associated with the period. The first character must be alphabetic.

### Examples

```
ADRUN
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)
```

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.

### ADRULE

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.
Defining Applications in Batch

Syntax

```
ADRULE control statement

EVERY
ONLY

DAY
WORKDAY
FREEYDAY
MONDAY
TUESDAY
WEDNESDAY
THURSDAY
FRIDAY
SATURDAY
SUNDAY

MONTH
JANUARY
FEBRUARY
MARCH
APRIL
MAY
JUNE
JULY
AUGUST
SEPTEMBER
OCTOBER
NOVEMBER
DECEMBER

YEAR

PERIOD

ORIGINSHIFT
```

Values Not Set by SETDEFAULT

You cannot use ACTION(SETDEFAULT) to set default values for this control statement.

Parameters

**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 LAST(3), read “third last,” so ONLY LAST(3) DAY(DAY) JANUARY specifies JANUARY 29.

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.

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.

WEEK *(week number)*

Specifies the week number or numbers. The number may 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.

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.

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.

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 OPC looks for a user-defined period JULY, and gives an error if one does not exist.

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.

Examples

<table>
<thead>
<tr>
<th>Example 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRUN NAME(Ex1A) TYPE(R) RULE(1) IATIME(1800) DLTIME(2300)</td>
</tr>
<tr>
<td>ADRULE ONLY(4) DAY(THU) MONTH</td>
</tr>
<tr>
<td>ADRUN NAME(Ex1B) TYPE(E) RULE(1) IATIME(1800) DLTIME(2300)</td>
</tr>
<tr>
<td>ADRULE ONLY LAST DAY(THU) YEAR</td>
</tr>
</tbody>
</table>

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.
Defining Applications in Batch

**Example 2**

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)

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.

**Example 3**

ADRUN NAME(EX3A) TYPE(R) RULE(4) IATIME(1800) DLTIME(2300)
ADRULE EVERY(2,3) DAY(DAY) YEAR

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 OPC 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).

**Example 4**

ADRUN NAME(EX4A) TYPE(R) RULE(3) IATIME(0100) DLTIME(2300)
ADRULE EVERY DAY(MONDAY) 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.

**Example 5**

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

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.

**Example 6**

ADRUN NAME(EX6A) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)
ADRULE ONLY DAY(FREE) WEEK YEAR

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 OPC would have generated the closest work day before the first free day in each week and the first free day in the year.

**Note:** You are not recommended to build complex rules that are hard to understand. It is better to create two simple rules:

<table>
<thead>
<tr>
<th>Example 6 (improvement)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRUN NAME(EX6A) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)</td>
</tr>
<tr>
<td>ADRULE ONLY(1) DAY(FREE) WEEK</td>
</tr>
<tr>
<td>ADRUN NAME(EX6B) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)</td>
</tr>
<tr>
<td>ADRULE ONLY(1) DAY(FREE) YEAR</td>
</tr>
</tbody>
</table>

EX6A (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 OPC 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.

<table>
<thead>
<tr>
<th>Example 7</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRUN NAME(EX7A) TYPE(R) RULE(3) IATIME(0800) DLTIME(2300)</td>
</tr>
<tr>
<td>ADRULE ONLY(29) DAY(DAY) MONTH(FEB)</td>
</tr>
</tbody>
</table>

EX7A (type R) selects February 29. In years that are not leap years, no day is selected.

<table>
<thead>
<tr>
<th>Example 8</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADRUN NAME(EX8A) TYPE(R) RULE(1) IATIME(0800) DLTIME(2300)</td>
</tr>
<tr>
<td>ADRULE O D(DAY) W(4) Y /* USING KEYWORD ABBREVIATIONS */</td>
</tr>
</tbody>
</table>

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 OPC 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.

**ADOP**

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.
Syntax

ADOP control statement

- ADOP
- ACTION(SETDEFAULT)
- WSID(workstation name)
- OPNO(operation number)
- JOBN(job name)
- DESCR(operation description)
- DURATION(operation duration)
- SMOOTHING(smoothing factor)
- LIMFDBK(feedback limit)
- AEC
- JOBCLASS(jobclass)
- HIGHRC(return code)
- PRTCLASS(print class)
- FORM(form number)
- STARTDAY(start day)
- STARTTIME(hhmm)
- DLDAY(deadline day)
- DLTIME(hhmm)
- AJSUB
- AJR
- TIME(YN)
- CLATE(YN)
- ADOPCATM
- RNUM
- R2NUM
- PSNUM
- PRESID(predecessor workstation)
- PREOPNO(predecessor operation number)
- PREJOB(predecessor job name)
- PREPWTO
- REROUTABLE
- RESTARTABLE

Values Not Set by SETDEFAULT

You cannot use ACTION(SETDEFAULT) to set default values for these keywords:

- WSID
- OPNO
- JOBN
- PREWSID
- PREOPNO
- PREJOB

146  Tivoli OPC Planning and Scheduling the Workload
Defining Applications in Batch

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.

**WSID (workstation name)**
The name of the workstation at which this operation takes place. A WSID
must be specified.

**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, incremented by 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.

**JOBN (job name)**
The job name for this operation, if applicable.

**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.

**DURATION (operation duration)**
The estimated duration of this operation in minutes. It must be an integer
greater than 0. The default is the duration specified for the workstation.

**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.

**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.

**AEC (N | Y)**
For operations on automatic reporting workstations, Tivoli OPC 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 OPC
will not check for errors and will give the operation completed status,
regardless of any error reported by job tracking.

**JOBCLASS (jobclass)**
A single character that appears on workstation ready lists for information only.
This should be the MVS job class from the JCL.

**HIGHRC (return code)**
If this operation is an MVS 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 OPC as completed. This must be an integer less than
Defining Applications in Batch

4096. If you leave out the parameter, Tivoli OPC takes the value you specified in the JTOPTS statement.

**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 OPC identify the different print operations belonging to a specific job. This is a single character, and must be specified for print operations.

**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 OPC identify the different print operations belonging to a specific job. This can be up to 8 characters.

**TIME (Y | N)**
If you specify Y, the job is made time-dependent. See “Creating Time-Dependent Operations” on page 114 for more details.

**CLATE (Y | N)**
Specify Y to cancel this operation if it is time-dependent and late.

*Note:* Tivoli OPC never cancels a job that has already started running.

**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.

**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*.

**AJSUB (N | Y)**
Automatic job submission.

**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 OPC scheduling conditions are met, or be released immediately. AJR(Y) means that Tivoli OPC should control the scheduling. AJR(N) means that Tivoli OPC 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.

**ADOPCATM** *(Y | D | N)*
The catalog management action for this operation:

- **N** Catalog management actions for this operation are bypassed. This is the default.
- **Y** The operation is eligible for catalog management actions, if the operation ends in error in the current plan, or when a rerun is requested.
- **D** The catalog management actions for this operation are deferred. You must manually start the catalog management actions for this operation if the operation ends in error or is rerun.

**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.

**PSNUM** *(number of servers | 0)*
The number of workstation parallel servers required by this operation. This must be an integer.

**PREWSID** *(predecessor workstation)*
The workstation name of an internal predecessor operation to this operation.

**PREOPNO** *(predecessor operation number)*
The operation number of an internal predecessor operation to this operation.

**PREJOBN** *(predecessor job name)*
The job name of an internal predecessor operation to this operation.

**ADOPPWTO** *(Y | N)*
If you specify Y, a WTO message is issued if the operation passes its deadline and is in started status.

**RESTARTABLE** *(Y | N )*
This parameter specifies the restart option for the operation. The default is that the operation is restartable if the WSFAILURE initialization statement RESTART keyword is set to RESTART.

- **Y** The operation is always restartable.
- **N** The operation is never restartable.

**REROUTABLE** *(Y | N )*
This parameter 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 never reroutable.
Defining Applications in Batch

Examples

```
ADOP
ADOP WSID(CPU1) JOBN(SMFCHK) OPNO(060) PREOP(030)
```

In this example, a job SMFCHK is given an operation number of 060. It has an internal predecessor—operation 030.

```
A time-dependent main operation
ADOP WSID(SETP) JOBN(BACKUP) OPNO(010)
ADOP WSID(CPU1) JOBN(BACKUP) TIME(Y) STARTDAY(0) STARTTIME(2000)
    PREOPNO(010) OPNO(015) DLTIME(2100) ADOPWTO(Y) ADOPCATM(Y)
ADOP WSID(PRT1) JOBN(BACKUP) PRTCLASS(H) PREOPNO(015)
```

This example creates three operations. 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 OPC issues a DEADLINE WTO. Tivoli OPC performs immediate catalog management if the job fails.

The last operation has no number specified, but Tivoli OPC 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 OPC 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.

ADSR

Use the ADSR control statement to specify the use of special resources by an operation.

Syntax

```
ADSR control statement
ADSR
    ACTION(SETDEFAULT)
    RESOURCE(resource name)
    USAGE(SX)
    QUANTITY(quantity required)
    KEEPONERR(NY)
```
Values Not Set by SETDEFAULT

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.

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).

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.

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.

Examples

<table>
<thead>
<tr>
<th>Exclusive allocation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADOP ...</td>
</tr>
<tr>
<td>ADSR RESOURCE(LINE.LONDON) USAGE(X) Q(2)</td>
</tr>
</tbody>
</table>

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.

<table>
<thead>
<tr>
<th>Taking the whole quantity</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADOP ...</td>
</tr>
<tr>
<td>ADSR RESOURCE(PAYROLL.DATABASE) KEEP(Y)</td>
</tr>
</tbody>
</table>

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.
ADDEP

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 OPC 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 dataset, the predecessor operation can be defined by an ADOP statement occurring later in the input dataset. This is because most validity checking occurs after all statements in the input dataset are read.

If the output is directed to an active Tivoli OPC 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.

Syntax

```
ADDEP control statement

<table>
<thead>
<tr>
<th>ADDEP</th>
<th>ACTION(SETDEFAULT)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PREWSID(—predecessor workstation—)</td>
<td></td>
</tr>
<tr>
<td>PREOPNO(—predecessor operation number—)</td>
<td></td>
</tr>
<tr>
<td>PREJOBN(—predecessor job name—)</td>
<td></td>
</tr>
<tr>
<td>PREADID(—predecessor application ID—)</td>
<td></td>
</tr>
<tr>
<td>TRANSPT(—transport time—)</td>
<td></td>
</tr>
<tr>
<td>DESCR('—predecessor description—')</td>
<td></td>
</tr>
</tbody>
</table>
```
Defining Applications in Batch

Values Not Set by SETDEFAULT

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.

PREWSID (predecessor workstation)
The 4-character workstation name of a predecessor operation to this operation.

PREOPNO (predecessor operation number)
The operation number of a predecessor operation to this operation.

PREJOBN (predecessor job name)
The job name of a predecessor operation to this operation.

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.

TRANSPT (transport time)
When Tivoli OPC 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.

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.

Tivoli OPC holds descriptions only for external dependencies. This field cannot be used to hold a description for an internal dependency.

Examples

<table>
<thead>
<tr>
<th>ADDEP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADDEP PREWSID(CPU1) PREJOBN(SMFCHK)</td>
</tr>
<tr>
<td>ADDEP PREADID(PAYDAILY) PREOP(020) DESC('WAIT FOR DAILY PAYROLL')</td>
</tr>
</tbody>
</table>
Defining Applications in Batch

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.

OISTART

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)
- Workstation name (WSID)
- 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 dataset, the operation can be defined by an ADOP statement occurring later in the input dataset. This is because most validity checking occurs after all statements in the input dataset are read.

If the output is directed to an active Tivoli OPC subsystem, the operation specified must already exist on the AD database. It cannot be defined later in the input dataset.

Syntax

```
OISTART control statement

OISTART
  ADD
  ACTION(SETDEFAULT)
  ADID(application ID)
  WSID(workstation name)
  OPNO(operation number)
  JOBN(job name)
  MEMBER(member name)
  VALFROMD(yymmdd)
  VALFROMT(hhmm)
  VALTOD(yymmdd)
  VALTOT(hhmm)
```
Values Not Set by SETDEFAULT

You cannot use ACTION(SETDEFAULT) to set default values for these keywords:

- ADID
- WSID
- OPNO
- JOBN
- MEMBER

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.

**WSID (workstation name)**

The workstation of the operation that this OI is for.

**OPNO (operation number)**

The operation number of the operation that this OI is for.

**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 dataset (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.

**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.
Defining Applications in Batch

Notes:
1. If you do not supply any of these VAL keywords, Tivoli OPC assumes that the operator instruction is permanent.
2. Tivoli OPC interprets the yy part as follows:
   - YY Year
   - 72 - 99 1972 - 1999
   - 00 - 71 2000 - 2071

Examples

--- OISTART with OIT
OISTART ADID(PAYDAILY) OPNO(020)
OIT ...

In this example, the OISTART specifies that operator instruction text for operation 020 in the application PAYDAILY will follow this statement.

--- OISTART with MEMBER
OISTART ADID(PAYDAILY) OPNO(020) MEMBER(PAYDAILY)

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.

OIT

Use the OIT control statement to add one line of operator instruction text to the current operator instruction.

Syntax

OIT control statement

```
/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

--- OIT
OISTART ...
OIT 'Enter password for update'
OIT 'of IMS database'
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 OPC sample library (SEQQSAMP) contains samples to execute the batch loader. For the dataset 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.
- **EQQBVSAM** updates VSAM datasets 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.

```
//XRAYNERE JOB (890122,NOBO),'SIMON RAYNER',
// MSGCLASS=H,NOTIFY=XRAYNER,CLASS=A
//STEP1 EXEC PGM=EQQYLTOP
//STEPLIB DD DSN=OPCESA.V1R3.LOAD,DISP=SHR
//EQQMLIB DD DSN=OPCESA.V1R3.MSGLIB,DISP=SHR
//EQQMLOG DD SYSOUT=/c5197
//EQQDUMP DD SYSOUT=/c5197
//EQQUDUMP DD SYSOUT=/c5197
//EQQOIPDS DD DSN=XRAYNER.OPCESA.MESSAGES,DISP=SHR
//SYSIN DD DSN=XRAYNER.OPCESA.BATCH(PAYROLL),DISP=SHR
//EQQADDS DD DISP=OLD,DSN=OPCESA.NEW.AD
//EQQOIDS DD DISP=OLD,DSN=OPCESA.NEW.OI
```

Figure 81. Sample JCL for Executing the Batch Loader

The program name EQQYLTOP is required on the EXEC statement in your JCL. EQQYLTOP should reside in the Tivoli OPC load module library, or at least in an APF-authorized library. If this library is not defined in the active MVS LNKLSTxx member of SYS1.PARMLIB, you must provide a STEPLIB DD statement in your JCL so that EQQYLTOP can be found.

The dataset requirements depend on whether you are directing the batch-loader output to a Tivoli OPC subsystem or to VSAM datasets. If the output is directed to VSAM datasets, you must allocate them with the correct VSAM dataset characteristics for an AD or OI database. The Installation Guide and the sample library have JCL and IDCAMS control statements for creating an AD or OI database.
Defining Applications in Batch

The ddnames for the required datasets are listed here with a description of their function:

**SYSIN**  The batch-loader input dataset that contains your control statements. You can use a disk or tape dataset, with fixed blocked 80-byte records, or use SYSIN DD * and place the control statements in your JCL.

**EQQMLIB**  The Tivoli OPC message library.

**EQQMLOG**  Message log. This can be allocated to a dataset or to SYSOUT.

**EQQDUMP**  Diagnostics dataset. If batch-loader output is directed to a VSAM dataset, diagnostic information is written to this dataset during validity checking of the records in the input dataset. If output is directed to a Tivoli OPC subsystem, diagnostic information is written to the diagnostics dataset of that subsystem.

**SYSUDUMP**  For MVS dumps.

**EQQOIPDS**  Required if the MEMBER keyword is specified on an OISTART control statement. EQQOIPDS specifies a partitioned dataset, fixed blocked with a record length of 80 bytes, which contains operator instruction text.

**EQQADDS**  Required if output is being directed to a VSAM dataset. Specifies the VSAM dataset 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 dataset. If present, it is used with EQQADDS to check that applications and operations on ADDEP and OISTART statements exist. You may 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 dataset, Tivoli OPC 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 dataset. EQQOIDS specifies the VSAM dataset that is to receive OI information.

**EQQWSDS**  Required only if output is directed to a VSAM dataset. EQQWSDS specifies the dataset 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 dataset 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 OPC subsystem uses, but if you have directed the batch-loader output to a Tivoli OPC subsystem, some messages may be written to the EQQMLOG dataset 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 dataset such as the dump dataset with disposition OLD or MOD in all Tivoli OPC 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 OPC ISPF dialog to print application descriptions or operator instructions. Several reports are available.
These batch-loader sample statements create applications for the Paymore example.

```plaintext
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 */
   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 */
   DESC('RUN EVERY WORK DAY') /* */
   IATIME(12/00/00) /* INPUT ARRIVAL TIME */
   DLTIME(16/00/00) /* 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 */
   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 */
   OPNO(20) /* */
   JOBN(PAYDAILY) /* */
   DESC('PAYDAILY JOB') /* */
   DURATION(5) /* ESTIMATED JOB DURATION */
   PREOP(10) /* JCL SETUP IS PREDECESSOR */
ADR RES(PAYROLL.DATABASE) /* ASK FOR EXCLUSIVE USE OF THE */
   USAGE(X) /* RESOURCE PAYROLL.DATABASE SO THAT */
   /* PAYDAILY DOES NOT RUN ALONGSIDE OTHER*/
   /* PAYROLL JOBS */
ADSTART ADID(GPAYW) /* DEFINE THE WEEKLY PAYROLL GROUP */
   DESC('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 */
   DESC('RUN ON THURSDAY') /* RUN ON THURSDAY OR ON THE PREVIOUS */
   IATIME(12/00) /* DAY (FREE-DAY RULE 1) IF THURSDAY IS */
   DLTIME(16/00) /* 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 */
   DESC('WEEKLY PAYROLL JOBS') /* IT DOES NOT HAVE ITS OWN ADRUN. */
   ADGROUP(GPAYW) /* */
```

Figure 82 (Part 1 of 4). Batch-Loader Sample: Paymore Example
Defining Applications in Batch

ADOP WSID(CPU1) /* */
  OPNO(20) /* */
  JOB(PAYWEEK) /* */
  DESC('PAYWEEK JOB') /* */
  DURATION(5) /* */
ADDEP PREWSID(CPU1) /* */
  PREOP(020) /* */
  PREADID(PAYDAILY) /* */
  DESC('WAIT FOR PAYDAILY') /* */
ADSR RES(PAYROLL.DATABASE) /* */
  USAGE(X) /* */
ADOP WSID(CPU1) /* THE PAYSLIPS JOB. */
  OPNO(30) /* */
  JOB(PAYSLIP) /* */
  DESC('PAYSLIP JOB') /* */
  DURATION(5) /* */
  PREOP(20) /* */
ADOP WSID(PRT1) /* TRACK THE PRINTING OF THE JES OUTPUT GROUP. */
  OPNO(90) /* */
  JOB(PAYSLIP) /* SAME NAME AS THE PREDECESSOR JOB. */
  DESC('PAYSLIP PRINT') /* */
  DURATION(20) /* */
  PREOP(30) /* */
  PRTCLASS(D) FORM(SLIPS) /* TRACKS THIS SYSOUT CLASS AND FORM NO */
ADOP WSID(PAY1) /* TRACK THE DECOLLATION OF PAY SLIPS AND PREPARATION OF PAY PACKETS IN THE PAY OFFICE. */
  OPNO(95) /* */
  JOB(PAYSLIP) /* */
  DESC('PAY PACKETS') /* */
  DURATION(60) /* */
  PREOP(90) /* */
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. */
ADRULE ONLY(3) /* */
  DAY(THURSDAY) MONTH /* */
ADSTART ADID(PAYM1) /* THIS MONTHLY JOB IS PART OF GPAYM, SO IT DOES NOT HAVE ITS OWN ADRUN. */
  DESC('MONTHLY PAYROLL JOBS') /* */
  ADGROUP(GPAYM) /* */
ADOP WSID(CPU1) /* */
  OPNO(40) /* */
  JOB(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 PAYSLIPS JOB. */
  OPNO(50) /* */
  JOB(PAYSLIP) /* */
  DESC('PAYSLIP JOB') /* */
  DURATION(5) /* */
  PREOP(40) /* */

Figure 82 (Part 2 of 4). Batch-Loader Sample: Paymore Example
Defining Applications in Batch

ADOP WSID(PRT1) /* TRACK THE PRINTING OF THE JES */
OPNO(99) /* OUTPUT GROUP. */
JOBN(PAYSLIP) /* SAME NAME AS THE PREDECESSOR JOB. */
DESC('PAYSLIP PRINT') /* */
DURATION(20) /* */
PREAD(50) /* */
PRTCLASS(D) FORM(SLIPS) /* TRACKS THIS SYSOUT CLASS AND FORM NO */
ADSTART ADID(PAY2M) /* THIS MONTHLY JOB IS PART OF GPAYM, SO */
DESCR('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) /* */
PREAD(040) /* */
PREADID(PAYM2MON) /* */
DESC('WAIT FOR PAYM2MON') /* */
ADDEP PREWSID(CPU1) /* */
PREAD(020) /* */
PREADID(PAYM2WEEK) /* */
DESC('WAIT FOR PAYM2WEEK') /* */
ADSR RES(PAYROLL.DATABASE) /* ASK FOR EXCLUSIVE USE OF THE */
USAGE(X) /* RESOURCE PAYROLL.DATABASE */
ADDEP PREWSID(CPU1) /* */
PREAD(040) /* */
PREADID(PAYM2) /* */
DESC('WAIT FOR PAYM2') /* */
ADSR RES(TAPES) QUANTITY(1) /* EXCLUSIVE USE OF ONE TAPE DRIVE */
ADSTART ADID(PAYTAXYR) /* DEFINE THE YEARLY TAX RUN */
DESCR('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) /* */
PREAD(040) /* */
PREADID(PAYM2) /* */
DESC('WAIT FOR PAYM2') /* */
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) /* */

Figure 82 (Part 3 of 4): Batch-Loader Sample: Paymore Example
Defining Applications in Batch

Figure 82 (Part 4 of 4). Batch-Loader Sample: Paymore Example

```plaintext
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(S) /* */
ADSR RES(TAPES) /* */
  USAGE(X) QUANTITY(1) /* EXCLUSIVE USE OF ONE TAPE DRIVE */
ADDEP PREWSID(CPU1) /* ALL THESE */
  PREAD(PAYDAILY) /* DEPENDENCIES */
  PREOP(020) /* ARE EFFECTIVE */
    DESC('WAIT FOR PAYDAILY') /* ONLY */
  ADDEP PREWSID(CPU1) /* IF THE */
    PREAD(PAYSLIP) /* SPECIFIED */
    PREOP(030) /* OPERATION */
    DESC('WAIT FOR PAYSLIP') /* IS SCHEDULED */
  ADDEP PREWSID(CPU1) /* FOR THE SAME DAY */
    PREAD(PAYSLIP) /* AS PAYBACKP. */
    PREOP(050) /* SO, IF IT IS A NORMAL */
      DESC('WAIT FOR PAYSLIP') /* 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 */
/*
Figure 82 (Part 4 of 4). Batch-Loader Sample: Paymore Example

Chapter 8. Defining Applications in Batch 163
Chapter 9. Overview of the Long-Term and Current Plans

Once you have described your installation to Tivoli OPC and defined your work in the application description (AD) database, Tivoli OPC 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 OPC schedule, providing the detail that Tivoli OPC requires to control the processing. The current plan is derived from a section of the LTP and contains the work that Tivoli OPC 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 Tivoli OPC 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.

![Diagram showing the relationship between the database and the plans](image)
Figure 83 shows the relationship between the database information and the planning processes.

The task of creating the LTP and CP is normally performed once. After creation, the plans are continually extended using batch functions.

When you make changes to the Tivoli OPC database, such as creating a new application, this is not reflected in the Tivoli OPC plans until you have incorporated the changes into the LTP and current plan. However, last-minute or unplanned changes can be made directly to the plans using the Tivoli OPC dialog.

The Tivoli OPC 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 LTP and CP, Tivoli OPC 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 OPC is in the best position to schedule your work and optimize the use of resources.

The Tivoli OPC 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 LTP is a high-level plan that can cover a length of time from 1 day up to 4 years. You produce the LTP using Tivoli OPC batch jobs, which are usually submitted from the Tivoli OPC dialog. The batch jobs use data from:

- The application description database
- Calendar and period definitions
- The current LTP, 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 OPC examines every application and job description to determine if an occurrence should be generated in the LTP. An occurrence is generated if an active application has a run cycle with a period and calendar combination that falls within the LTP 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 LTP. 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 LTP that you have added, deleted, or changed manually are carried over to the new LTP, except when Tivoli OPC is creating a new LTP.

Figure 84 describes the data required for the long-term planning process.

![Figure 84. Production of the LTP](image)

The LTP contains an occurrence entry for every planned run of an application. Most applications generate several occurrences in the LTP; for example, applications that are required daily or weekly. Every occurrence in the LTP 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 LTP 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 LTP to cover future periods. For example, if you wanted to plan for 4 weeks, you could initially set up the LTP to cover a period of 35 days into the future. You could then extend the LTP by an additional 7 days every week. Your LTP would always stretch at least 28 days into the future.

When you create your LTP, you specify the start date. Although this information is recorded and shown as the LTP start date in the Tivoli OPC dialog, occurrences are not maintained as far back as this date. If this were the case, the LTP dataset would expand indefinitely. When you create a new current plan (NCP), part of the process is to update the LTP with any occurrences that are now complete in the CP. When you extend or modify the LTP in batch, completed occurrences are removed.

The length of the LTP is calculated from the date of the earliest occurrence it contains, that is, from the date of the earliest uncompleted occurrence. The LTP length is not calculated from the LTP start date.
Plans Overview

Tivoli OPC does not delete occurrences from the LTP that are uncompleted or any occurrences that follow an uncompleted occurrence. Tivoli OPC 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 LTP 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 OPC 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 dataset that the LTP is stored in and reduces the time and resources required to perform LTP batch and dialog functions.

The LTP contains run time and external dependency information for application occurrences. The current plan, which is the Tivoli OPC detailed schedule, is based on the occurrence information stored in the LTP.

Current Plan

The LTP does not contain all the information that Tivoli OPC needs to submit work to your system. Before Tivoli OPC 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 187).

The current plan covers a section of the LTP. Typically, the current plan covers a period of 24 hours, although it can range from 1 minute to 21 days. Tivoli OPC expands the information held in the LTP, 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 OPC 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.

When the current plan is created, it is updated in real time by events produced by the Tivoli OPC SMF and JES exits. User programs can automatically report status of operations by calling a Tivoli OPC routine or TSO users can issue the OPSTAT command. The current plan can also be updated from the Tivoli OPC dialog. Users with sufficient authority can add, change, complete, or delete operations and occurrences. Work running outside Tivoli OPC control can trigger an application occurrence to be included in the current plan. Figure 85 on page 169 shows the data used in daily planning.
Resolving Pending Occurrences

To handle occurrences that extend beyond the CP and dependencies whose predecessors are not yet in the CP, Tivoli OPC uses tail plans and pending occurrences.

**Tail Plans:** If an occurrence in the LTP has an input arrival time within the date range of the CP, it is included in the CP when the plan is created or extended. If the occurrence cannot be completed before the end of the CP, the CP 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 CP as usual when you extend the CP 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 LTP is created or extended. That is, the link is established between the two dependent occurrences.

Although the dependency has been resolved in the LTP, it can sometimes happen that the successor of a dependency relationship is included in the CP but the predecessor is not. This can occur, for example, when a dependency is added manually to the LTP. To ensure that the dependency is honored, Tivoli OPC...
Plans Overview

creates a dummy occurrence in the CP, 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 CP, the pending occurrence is replaced.

In Figure 86, 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 CP, and a pending occurrence is created in the CP until application B is included.

![Figure 86. Example of a Missing Predecessor](image)

Creating the Plans for the First Time

See “Creating Plans” on page 19 for an example of creating plans for the first time. This is a summary of the steps:

1. Decide how far the LTP will extend.

   The LTP 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, 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 LTP.

   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 LTP 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 minute-by-minute schedule of all operations. It drives all automatic Tivoli OPC 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 OPC when the plan is created, unless they are time-dependent operations.
- LTP 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 OPC does not automatically submit jobs with this status. This situation applies only when a current plan does not yet exist.
- An LTP 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 OPC to have completed. It will be marked as completed in the LTP and excluded from the current plan. This situation applies only when a current plan does not yet exist.

Avoid scheduling occurrences in the LTP before the start of your first current plan. The current plan can then start at the date and time that you want Tivoli OPC 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 OPC brings in from the LTP all occurrences with an input arrival time that is within the period you specify. Tivoli OPC then creates a detailed schedule for the operations that are contained in these occurrences.

If you are extending the current plan, Tivoli OPC 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 LTP are reflected in the current plan only after the current plan is extended.
- You cannot amend occurrences in the LTP 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.
- 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 19.

5 Consider getting Tivoli OPC 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 LTP weekly.

6 Extend the current plan a few hours before it ends, so you have time to look at the reports.
Deciding What to Do when You Make Changes

001
Is the change temporary?
Yes No

002
– Change the application database.

Do you want the change to affect the current plan?
Yes No

003
– Modify or extend the LTP in batch.

004

Are you adding or deleting occurrences that are in the CP?
Yes No

005
– Modify or extend the LTP.
– Replan or extend the CP.

006
– Modify or extend the LTP.
– Use the MCP dialog to add or delete occurrences.
– Replan or extend the CP.

007

Does the change affect the current plan?
Yes No

008
– Modify the LTP online.

009
– Use the MCP dialog.
Chapter 10. Producing and Modifying the Long-Term Plan

You can use the Long-Term Plan dialog to perform these tasks:

- Create an LTP, if one does not exist.
- Extend the LTP, if the time period covered by the plan is not far enough into the future.
- Create a trial LTP.
- Produce reports on all or part of the LTP.
- Check the contents of the LTP online.
- Change the LTP to fix errors or include last-minute changes.
- Prepare job control language (JCL) for computer operations belonging to occurrences in the LTP.
- Modify the LTP to reflect changes in the Tivoli OPC databases.

You access the Long-Term Plan dialog by selecting option 2 from the Tivoli OPC main menu. The MAINTAINING THE LONG-TERM PLAN menu is displayed (Figure 87).

Figure 87. EQQLTOPP—Maintaining the Long-Term Plan

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 dialog 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 LTP and current plans have been updated. You do not need to discard your LTP completely. Instead, you can change the LTP to reflect these changes. See “Modifying the Long-Term Plan in Batch” on page 179 for details. Then use the Daily Planning dialog to amend the current plan to include
Running Long-Term Plan Batch Jobs

For some functions, such as creating or printing the LTP, the dialog submits a batch job. You can save the job and submit it outside the Tivoli OPC dialogs. You can use Tivoli OPC to schedule and submit the LTP extend job. This ensures that the LTP is always updated at the correct time and on the correct date.

The batch jobs normally affect all occurrences in the LTP. If you specify the MODIFY ONE or PRINT ONE options, however, only the selected occurrences are modified or printed. Run LTP batch jobs by selecting option 2 (BATCH) from the MAINTAINING THE LONG-TERM PLAN menu. You see the menu shown in Figure 88.

```
   EQQLBATP  SELECTING LONG TERM PLAN BATCH JOB
Option ==>  
   1 MODIFY  Modify the long term plan for all applications
   2 MODIFY ONE Modify the long term plan for one application
   3 EXTEND  Extend the long term plan
   4 TRIAL  Make a trial long term plan
   5 PRINT  Print the long term plan for all applications
   6 PRINT ONE Print the long term plan for one application
   7 CREATE Create a new long term plan
```

Figure 88. EQQLBATP—Selecting Long-Term Plan Batch Job

The Long-Term Plan Datasets

The long-term-planning function creates or modifies the LTP dataset. The LTP is stored in a VSAM dataset that is defined when Tivoli OPC is installed. The LTP is referenced in the long-term planning jobs and by the Tivoli OPC address space by ddname EQQLTDS.

The long-term plan VSAM dataset does not require regular reorganization because the entire dataset is rewritten by LTP batch create, modify, and extend functions. A VSAM work dataset (ddname EQQLDDS) and a backup dataset (ddname EQQLTBKP) are used by the LTP batch jobs during the planning process.

LTP 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 LTP planning process detects a period definition that has no future origin dates.

Messages are written to the message-log dataset defined by the EQQMLOG DD statement.
How Tivoli OPC Creates a Long-Term Plan

Tivoli OPC examines every application and job description. If an application is a member of a group, Tivoli OPC 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 LTP 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 87 for details. Tivoli OPC also checks the work-day 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 OPC purposes until 06:00. This means that work might be scheduled on a free day. See “Creating the Default Calendar” on page 60 for more information.

Tivoli OPC 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 OPC creates an occurrence in the LTP 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 OPC detects and reports a duplicate occurrence, but does not store it in the LTP. 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 LTP; 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 LTP by selecting option 7 from the SELECTING LONG-TERM PLAN BATCH JOB menu (Figure 88 on page 176).

```
EQQLCREP ---------------- CREATING THE LONG TERM PLAN ----------------
Command ===> Enter/Change data below:
Long term plan:
START ===> 97//01/27 Date in format YY/MM/DD
END ===> 97//04/30 Date in format YY/MM/DD
```

Figure 89. EQQLCREP—Creating the Long-Term Plan

See “Creating the Plans for the First Time” on page 170 for general guidance on creating the first plan. When creating the LTP, 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 89.)

If an LTP already exists, you will receive a warning message when creating a new LTP. If occurrences in the existing LTP still exist in the current plan and are not yet complete, you cannot use the create function. To create an LTP in this situation, you must REFRESH the LTP. You perform an LTP REFRESH from the SERVICE FUNCTIONS dialog (option 9 from the Tivoli OPC main menu), not from the LTP dialog.

**Attention:** A REFRESH of the LTP 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 may 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 an LTP already exists, you can specify an end date earlier than that of the old LTP.

The create job does not use the existing LTP 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 LTP-create batch job performs the planning process as described in “How Tivoli OPC Creates a Long-Term Plan” on page 177.

To decide what time span your LTP should have, consider:

- The longer the plan, the more computing resources required to produce it.
- A long LTP probably has many occurrences and operations, so there can be a long response time when you edit the LTP with the dialog.
Extending the Long-Term Plan

Extend the LTP by selecting option 3 from the SELECTING LONG-TERM PLAN BATCH JOB menu, shown in Figure 88 on page 176. You see the EXTENDING THE LONG-TERM PLAN panel, shown in Figure 90.

```
EQQLEXTP ---------------- EXTENDING THE LONG TERM PLAN ----------------
Command ===>
Enter/Change data below:
Current end : 97/04/30
NEW END DATE ===> ________ New long term plan end date
                 in format YY/MM/DD
EXTENSION LENGTH ===> 28__ DDDD Extend plan by
```

Figure 90. EQQLEXTP—Extending the Long-Term Plan

To extend the LTP, specify a new end date, or extension length in days, for the LTP. Tivoli OPC 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 OPC uses the LTP 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 OPC; extend the LTP at regular intervals. Because Tivoli OPC can schedule the job of extending the LTP in the same way as any other job in your system, consider using Tivoli OPC to regularly schedule a job that extends the LTP by a fixed duration each time it runs.

Modifying the Long-Term Plan in Batch

You can modify the LTP to reflect the latest data available in the Tivoli OPC database by selecting option 1 from the SELECTING LONG-TERM PLAN BATCH JOB panel (see Figure 88 on page 176). This generates modify-LTP 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 dialog function are left unchanged by the modify process. This includes manually deleted occurrences. If you delete an occurrence, this is noted in the LTP. 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-LTP 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.

MODIFY can be performed for a single application by selecting option 2 from the SELECTING LONG-TERM PLAN BATCH JOB menu (Figure 88 on page 176). 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 LTP by selecting option 5 (print) or option 6 (print one) from the SELECTING LONG-TERM PLAN BATCH JOB menu (Figure 88 on page 176). Both functions extract data from the LTP 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 LTP, whereas option 6 produces a report for one application.

You can request two types of report. Specify \textit{F} for a full report, or \textit{D} (dependencies) for a report on predecessors and successors. For each dependency, the dependency report contains the optional text string called \textit{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.
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 LTP, you also get histograms showing the planned workstation utilization for the period of the printout for each workstation.

See “LTP Reports” on page 404 for examples of the various reports that are generated.

**Tivoli OPC-Generated Comments in LTP Reports**

The LTP reports include information that Tivoli OPC adds about the way each occurrence has been planned. Tivoli OPC 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 dataset. Some fields in the LTP 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 LTP sorted by owner ID.

**DEPENDENCY CHANGED**
The dependency printed on this line has been changed manually using the Tivoli OPC dialogs.

**ENTERED MANUALLY (ONLINE)**
This occurrence has been added manually using the Tivoli OPC dialogs or PIF.

**CHANGED MANUALLY (ONLINE)**
This occurrence has been modified using the Tivoli OPC dialogs or PIF.

**ERROR CODE = xxxx**
Where xxxx is an error code. This error code was added manually using the Tivoli OPC dialogs 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.

Create trial long-term plans by selecting option 4 from the SELECTING LONG-TERM PLAN BATCH JOB menu (Figure 88 on page 176).

![EQQLTEXP](image)

**Figure 92. EQQLTEXP—Making a Trial Long-Term Plan**

You can produce a trial LTP to check the validity of any database changes that you make. This process produces a print of the LTP in the same format as reports on the active LTP, but the LTP file is not updated.

Depending on the day and date information that you specify on the MAKING A TRIAL LONG TERM PLAN panel, Tivoli OPC produces one of three possible trial plans:

- If you do not specify anything in the fields, Tivoli OPC produces a modify all trial plan. This simulates modifying the LTP, which is described in “Modifying the Long-Term Plan in Batch” on page 179.
- If you specify a start date and an end date, Tivoli OPC simulates an LTP create, which is described in “Creating the Long-Term Plan” on page 178.
- If no start date is specified but an end date or an extension length is specified, Tivoli OPC simulates an LTP extend, which is described in “Extending the Long-Term Plan” on page 179.

Displaying the Status of the LTP

You can display the status of the LTP by selecting option 4–STATUS, from the MAINTAINING THE LONG-TERM PLAN menu (Figure 87 on page 175). You see the STATUS OF THE LONG-TERM PLAN panel, shown in Figure 93 on page 183.
Figure 93. **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 OPC uses to resolve external dependencies in the LTP dialog by selecting option 5 (SET DEFAULTS) from the **MAINTAINING THE LONG-TERM PLAN** menu (Figure 87 on page 175). You see the **SETTING DEFAULT FOR BROWSE** panel, shown in Figure 94.

Figure 94. **EQQLBDWP—Setting Default for Browse**

When you create a dependency in the LTP using the LTP dialog, the dependency is between two application occurrences. Tivoli OPC 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 dialog 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 OPC uses the last operation in the application for the dependency.

The values set on the panel are used only by the LTP dialog. 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**.
Amending Applications in the LTP Online

You can modify or browse individual occurrences of an application in the LTP 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
- Prepare job statements for an occurrence in the plan
- Delete, complete, or add an occurrence group

When you modify the LTP with these dialog functions, Tivoli OPC 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 LTP to verify that the changes are accurate.

A dependency in the LTP is between application occurrences. The LTP itself does not contain any operations or any operation-level information. When you browse dependencies on an operation level in the LTP dialog, 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 LTP 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 OPC does not change the manually altered occurrence when the LTP is extended. Tivoli OPC issues a warning message to indicate that this occurrence has not been changed in line with the modified application description. Tivoli OPC assumes that any manual changes you have made should override any automatically generated changes.
The Tivoli OPC long-term plan does not include all operation detail. If you want to make changes to dependencies in the LTP, Tivoli OPC 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 183.

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 LTP until they are removed from the occurrence group.

Note: If you edit a job using the LTP dialog, the job is saved in the JCL repository dataset 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 dataset.

### Modifying Dependencies in the Long-Term Plan

Specify option 1 (ONLINE) from the main menu of the LTP dialog. You see the LONG-TERM PLAN OCCURRENCES panel, shown in Figure 95.

If you want to make the 97/02/03 occurrence of PAYW not dependent on PAYDAILY, for example, type m beside the row and press Enter. You see the MODIFYING AN OCCURRENCE panel, shown in Figure 96 on page 186.
EQQLCHGP ------------------ MODIFYING AN OCCURRENCE ---------------------

Option ===>
Select one of the following:

1 OPERATIONS - Modify operation data
2 DEPENDENCIES - Modify dependencies
3 OCCURRENCE - Modify occurrence data

Application : PAYW weekly payroll jobs
Input arrival : 97/02/03 12.00
Deadline : 97/02/03 16.00
Owner : SAMPLE payroll application
Priority : 5
Error code :

Variable table : PAY
Successors : 0
Predecessors : 1
Manually created : No
Group Definition : GPAYW

Figure 96. EQQLCHGP—Modifying an Occurrence

Specify option 2 and press Enter. You see the MODIFYING DEPENDENCIES panel, shown in Figure 97.

EQQLCDPL ------------------- MODIFYING DEPENDENCIES --------- ROW 1 TO 1 OF 1

Command ===> Scroll ===> PAGE

Enter the CREATE command above to create a new dependency or enter any of the commands below:
B - Browse, D - Delete

Application : PAYW weekly payroll jobs
Input arrival : 97/02/03 12.00
Deadline : 97/02/03 16.00

Row Dep Application id Input arrival Complete Manually Deleted
  cmd Type date time Created
  P PAYDAILY 97/02/03 12.00 N N N

Figure 97. EQQLCDPL—Modifying Dependencies

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 OPC processing. It drives your production workload and provides feedback about the current status of batch work.

During normal Tivoli OPC processing, the CP can be updated by job-tracking events, Tivoli OPC dialog users, the program interface, the automatic recovery function, and the event-triggered tracking function. Depending on your workload, the CP could be updated several times per second. For this reason, and because the CP is such a critical resource, Tivoli OPC handles it in a different way from the other databases and datasets. The physical structure of the CP and the backup process used to ensure integrity are explained in “Organization and Integrity of the Current Plan” on page 199.

Daily planning is the process of producing and maintaining the CP, which is the detailed schedule of the work that Tivoli OPC will carry out on your system.

You create and maintain the CP using these options on the main menu of the Tivoli OPC dialog:

- DAILY PLANNING—Produce current plans, real and trial.
  
  This chapter describes this dialog. As with long-term planning, certain functions take effect online, whereas others are performed by batch jobs submitted from the dialog.

- MCP—Modify the Current Plan.
  
  Refer to Controlling and Monitoring the Workload for details of this dialog.

Information Used by Daily Planning

Daily planning uses data from several Tivoli OPC datasets:

- The long-term plan (LTP), which contains a list of application occurrences to run each day. The LTP 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 data is collected and used by Tivoli OPC to update the relevant datasets and produce the CP reports. The daily planning process involves:

- Update of the LTP for occurrences that are marked complete or have been deleted in the existing CP.
Current Plan

- Creation of an updated CP, called the *new current plan* (NCP), that contains:
  - Uncompleted operations from the existing CP.
  - New occurrence selections from the LTP according to the specified end date. See Chapter 10, “Producing and Modifying the Long-Term Plan” on page 175.
  - 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 CP using the Tivoli OPC dialog, the program interface, or event-triggered tracking.

- Optional copy of the job-tracking archive log.

- Creation of daily planning reports.

Figure 98 shows the required data.

![Figure 98: Data Required by the Daily Planning Process](image)

The daily planning process writes messages to the EQQMLOG and SYSPRINT datasets. 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 CP. For example, if the daily planning process detects a loop in a chain of dependent operations, a new CP is not created. “Analyzing Problems Reported by Daily Planning” on page 196 details how daily planning analyzes dependency loops.
Creating the Current Plan

You must create a CP before Tivoli OPC can schedule work. You must also create a new plan if you have performed a REFRESH of the LTP (see “Creating the Long-Term Plan” on page 178). Once you have created the CP, you continually extend it.

Follow these steps to create the CP:

1. Before you can create a CP, 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 175.

2. Select option 3 (DAILY PLANNING) from the Tivoli OPC main menu. You see the PRODUCING OPC/ESA DAILY PLANS menu, shown in Figure 99.

   Figure 99. 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 CP 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 OPC ignores this day when it calculates the end of the CP.

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.


6. Inspect the plan, and use the MCP dialog 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 OPC. In this case, Tivoli OPC might mark an occurrence as completed before full Tivoli OPC control of workload submission is in place.
Current Plan

7 Use the MCP dialog 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 LTP with the new occurrence data and lets Tivoli OPC recalculate the input arrival times of the remaining occurrences with the new current plan data.

Extending the Current Plan

The CP should always stretch for some hours or days into the future. Extend the CP at regular intervals, using the EXTEND option of the DAILY PLANNING menu. You can extend the CP up to 21 days, although one day is more usual. The dialog creates a batch job, which you can then submit. You can extend the CP to a fixed date and time in the future, or you can extend it by a period of hours and minutes.

Figure 100 shows a 48-hour CP. The initial CP extended 48 hours into the future: every morning the CP is extended by a further 24 hours.

Input is taken from both the LTP 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 CP.

Normally, you extend the CP so that it stretches 24 hours from the end of the previous plan. If you do this every 12 hours, the CP always extends at least 12 hours into the future. For instance, you could extend the plan by 24 hours at
You can automate this process by scheduling the batch job as an operation in a Tivoli OPC 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 CP 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 LTP (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 OPC 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 Tivoli OPC 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 OPC 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 OPC 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 OPC determines that a deadline is in danger. Refer to Customization and Tuning for more information about the alerts that Tivoli OPC can generate.

When the occurrences for the new plan are determined, the application database is used to generate potential predecessor records for every occurrence.

EXTEND also updates the LTP with completed occurrence information. Completed occurrences are not carried forward into the new plan.

When you extend the CP, you have the option of producing reports on the contents of the plan. See “Producing Reports Using Daily Planning” on page 193 for details.

Recreating the Current Plan

You can use the REPLAN option of the DAILY PLANNING menu to update your existing CP. This option performs the same functions as the EXTEND option, except that the CP continues to cover the same time interval as before. No new occurrences can be included in the CP by a REPLAN. This is because Tivoli OPC does not let you add to the LTP any occurrences that have an input arrival time before, or within, the period covered by the CP.
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 CP. When the list of occurrences is determined, the application database is used to generate potential predecessor records for every occurrence.

When you REPLAN the CP, you have the option of producing a report on the contents of the plan. See “Producing Reports Using Daily Planning” on page 193 for details.

Creating a Trial Current Plan

Before you create, extend, or replan the CP, 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.
Producing Reports Using Daily Planning

When creating, extending, or replanning the CP, 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 408 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 CP, you can use the QCP option on the Tivoli OPC main menu to obtain an online view of the CP.

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 nonreporting 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 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 CP. 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 CP. See “Reports for a Previous Planning Period” on page 414 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 CP if the PREVRES keyword of BATCHOPT has the value YES, which is the default value. The data is kept in the CP 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

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 CP 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 nonreporting 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 where the feedback of the actual duration to the application description database was not possible.

---

**Using the Track Log**
When you execute the daily planning process, you can create a track log dataset. The track log contains the job-tracking and audit records from the previous planning period. The track log records are written to the EQQTROUT dataset. The track log can be used as an audit trail for Tivoli OPC, as all updates to the plans and databases can be logged in the dataset.

The Tivoli OPC sample library contains an audit package, which can be used to create reports from either the track log or job-tracking datasets. 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 MVS* licensed program product to create reports using the track log. Refer to *Performance Reporter for MVS Fact Sheet* for more information.

---

**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.
Current Plan

When the CP 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 LTP when the CP is extended. An occurrence added by the daily planning process from the LTP will not be made a successor to an occurrence added by automatic job recovery, PIF, ETT, or the MCP dialog, 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 CP from the MCP dialog at 12:00. The occurrence input arrival is 12:00.
4. The daily planning batch job is run at 15:00 to extend the CP. Application B is added to the CP 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 CP at 16:15 using the MCP dialog, 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 CP, use the Modify Current Plan dialog, which is described in Controlling and Monitoring the Workload.

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 dataset defined by the EQQMLOG DD statement and to the daily plan printout dataset 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, sometimes 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 may 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 OPC 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 101 shows examples of the messages issued by the daily planning programs to assist dependency loop resolution.

#### MESSAGES

**EQQ0384E** DEPENDENCY LOOP FOUND IN AN APPLICATION NETWORK
**EQQ0384E** LIST OF OPERATIONS CONTAINED IN LOOP FOLLOWS:
**EQQ0384E** LOOP:OP 0010 NOJOB IN APPL LOPAD IA 970616 1900
**EQQ0384E** LOOP:OP 0020 LOPJOB IN APPL LOPAD IA 970616 1900
**EQQ0384E** LOOP:OP 0010 NEXJOB IN APPL NEXAD IA 970616 1900
**EQQ0384E** LOOP:OP 0005 NEXJOB IN APPL NEXAD IA 970616 1900
**EQQ0384E** LOOP:OP 0010 LASTJOB IN APPL LASTAD IA 970617 2105
**EQQ0384E** SUGGESTED DEPENDENCIES CAUSING LOOP ARE:
**EQQ0384E** LASTAD 0010 LASTJOB PRED TO LOPAD 0010 NOJOB BY IA TIME CHECK

*Figure 101. Dependency Loop Analysis—simple Loop Example*

Figure 102 shows how Tivoli OPC identifies the most likely cause of the dependency loop—the dependency that links LASTJOB 2105 as a predecessor to NOJOB 1900 on the previous day.

#### MESSAGES

**EQQ0384E** DEPENDENCY LOOP FOUND IN AN APPLICATION NETWORK
**EQQ0384E** LIST OF OPERATIONS CONTAINED IN LOOP FOLLOWS:
**EQQ0384E** LOOP:OP 0010 AJ02196 IN APPL AJ02196 IA 970616 1600
**EQQ0384E** LOOP:OP 0010 AJ02195 IN APPL AJ02195 IA 970616 1600
**EQQ0384E** LOOP:OP 0010 AJ01872 IN APPL AJ01872 IA 970616 1600
**EQQ0384E** LOOP:OP 0010 AJ01871 IN APPL AJ01871 IA 970616 1600
**EQQ0384E** SUGGESTED DEPENDENCIES CAUSING LOOP ARE:
**EQQ0384E** AJ02196 0010 AJ02196 PRED TO
**EQQ0384E** AJ01872 0010 AJ01872 CAUSES MINIMUM NET DISTORTION
**EQQ0384E** MULTIPLE LOOP PATHS ENCOUNTERED IN NETWORK
**EQQ0384E** SUGGESTED DEPENDENCIES CAUSING LOOP ARE:
**EQQ0384E** AJ02195 0010 AJ02195 PRED TO
**EQQ0384E** AJ01872 0010 AJ01872 FOR MINIMAL NET DISTORTION

*Figure 102. Dependency Loop Analysis—complex Loop Example*

This example contains two dependency loops. Coincidentally, the dependencies identified as the most likely cause of the loops have the same successor.
Modifying the Current Plan

To construct a CP, you must first set up the Tivoli OPC databases and then construct the LTP. Changes in the LTP are reflected in the current plan only after you perform a CP EXTEND or REPLAN. If you want your changes to take effect immediately or if you want to add occurrences to the existing CP, you must use the MCP (Modify Current Plan) dialog.

```
EQQMTOPP ---------------- MODIFYING THE CURRENT PLAN ----------------
Option ===> 
Select one of the following:
1 ADD - Add a new occurrence to the current plan
2 LIST - List existing occurrences for further processing
3 OPERATIONS - List existing operations for further processing
4 ERROR HANDLING - Handle operations in error
5 WORK STATIONS - Change status and open interval of work stations
6 JOB SETUP - Prepare JCL for jobs in the current plan
7 SPECRES - Special resource monitor
9 DEFINE EL - Define alternative error list layouts
```

Figure 103. EQQMTOPP—Modifying the Current Plan

You can reach the Modify Current Plan dialog from the Tivoli OPC main menu. Refer to Controlling and Monitoring the Workload for a detailed description of this dialog.

Querying the Current Plan

The QCP (query current plan) option from the Tivoli OPC main menu presents you with a list of resources within the CP that you can investigate. From here, you can obtain information on operations, workstations, and application occurrences. This information is taken directly from the CP. The dialog can also show a list of all operations that have ended in error.

```
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
```

Figure 104. EQQSTOPP—Current Plan and Status Inquiry
For more information about the functions available from the CURRENT PLAN AND STATUS INQUIRY panel, refer to Controlling and Monitoring the Workload.

Current Plan Reference Information

The remainder of this chapter describes the daily planning process in more detail. You may want to refer to this if you have a problem, or if you need to understand how Tivoli OPC builds the plan.

Organization and Integrity of the Current Plan

Tivoli OPC 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 datasets are used for current plan integrity:

- **EQQCP1DS**: Primary current-plan dataset
- **EQQCXDS**: Current plan extension dataset
- **EQQNCXDS**: New current plan extension dataset
- **EQQSIDS**: Side information dataset
- **EQQCP2DS**: Alternate current-plan dataset
- **EQQNCPDS**: New current-plan dataset
- **EQQJTnn**: Current and inactive job-tracking logs
- **EQQDLnn**: Current and inactive dual job-tracking logs
- **EQQJTARC**: Job-tracking archive log
- **EQQCKPT**: Checkpoint dataset

However, the explanation of the current plan process uses these logical terms to describe the CP and its associated datasets:

- **Current plan**: Used when describing the CP in general. The CP consists of the active current-plan dataset, the extension (CX) dataset, and the side dataset (EQQSIDS). These datasets are described later.

- **Active current plan**: Refers to the current-plan dataset that is currently in use within Tivoli OPC. It may be either EQQCP1DS or EQQCP2DS. Every time a CP backup is performed, Tivoli OPC switches the active CP to the other dataset. The CP backup process is described in “The Current-plan Backup Process” on page 202.

- **Backup current plan**: Refers to the current-plan dataset that is not currently in use. This dataset contains a backup copy of the CP. It may be either EQQCP1DS or EQQCP2DS.

- **Side information**: Refers to a file (EQQSIDS) 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 CP, which is created by one of the daily planning batch jobs. It always refers to EQQNC诸侯.

Current and inactive job-tracking log

Refers to the datasets used by Tivoli OPC 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 JT logs are used in a cyclic manner, Tivoli OPC automatically switches to the next available JT log after a CP backup. The data from the inactive dataset is copied to the archive log, and the dataset 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 OPC duplicates the JT records in the corresponding dual JT log. Dual logs are switched at the same time, and in the same sequence, as the JT logs. So the number of dual job-tracking datasets is determined by the number of normal job-tracking datasets.

Job-tracking archive log

Represents the accumulated job-tracking data since the new CP was created. When the JT log is switched, the data from the inactive dataset is appended to the archive log. The archive log is copied to the track-log dataset referenced by the EQQTROUT ddname during the daily planning process. When Tivoli OPC takes over the NCP, the archive dataset is emptied.

Checkpoint

Refers to the EQQCKPT dataset, which contains information about the current status of the Tivoli OPC system, including which CP and job-tracking datasets are currently active.

The basic principle of Tivoli OPC CP recovery is that if the active CP becomes unusable for any reason, Tivoli OPC should always be able to re-create an up-to-date current plan from the backup CP and the job-tracking logs. The way Tivoli OPC does this is described in Customization and Tuning.

The Current Plan during Normal Tivoli OPC Processing

Normal processing means that the Tivoli OPC subsystem is running, job tracking is active, and Tivoli OPC dialog users have access to the Tivoli OPC subsystem. Job tracking is active means that an active CP is updated as events occur in the operating system.

Figure 105 on page 201 shows the CP with associated datasets and how they are used during normal Tivoli OPC processing.
Tivoli OPC can update the current plan:

- With event information from the event datasets, for example *job ABC started, job XYZ ended*
- With dialog requests, for example from the Modify Current Plan dialog
- With requests from the Tivoli OPC program interface
- As a result of a triggering event recognized by the event-triggered tracking function
- As a result of a request from Tivoli OPC automatic recovery statements.

Every time the active CP is updated, a record of the change is written to the active job-tracking log. The job-tracking record is also written to the dual job-tracking log if dual logging has been requested.
The remaining datasets are not used during normal processing:

- The checkpoint dataset contains status information; for example, which physical dataset is the active CP.
- The backup CP contains a copy of the CP as it was at the last successful CP backup. In recovery situations, this is used with the job-tracking log to re-create an up-to-date CP. The CP backup process is described in “The Current-plan Backup Process.”
- 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 dataset. The NCP is also used to re-create the CP in conjunction with the various job-tracking logs, if a usable CP is not available or when you specifically request to start Tivoli OPC from the NCP.
- The inactive job-tracking logs and the inactive dual-logging datasets.

The Current-plan Backup Process

Tivoli OPC automatically backs up the active CP at certain stages in Tivoli OPC processing.

This is a step-by-step description of the current-plan backup process:

1. Tivoli OPC locks the CP to prevent it from being updated during the backup process. During the backup, events are queued in storage. Dialog users working with the CP may experience a short delay.
2. The CX data space (EQQCXDS) is backed up to DASD.
3. The backup CP is erased.
4. The active CP is copied to the backup CP. The contents of the two are now identical.
5. The datasets are switched. The backup CP becomes the active and the active becomes the backup.
6. A CP backup record is written to the job-tracking log, and the next available JT log becomes active. The corresponding dual job-tracking log is also switched.
7. The CP is unlocked. Normal processing continues. Events queued in storage start to update the active CP. Dialog users' requests are processed.
8. The data from the now inactive job-tracking log is appended to the JT archive log. The inactive job-tracking log is emptied for future use.

A CP backup is performed at these times:

- During normal Tivoli OPC processing, according to the value of the BACKUP parameter of the JTOPTS initialization statement. This parameter specifies the number of CP updates that must occur before a CP backup is performed.
- When the BACKUP command is issued specifying the CP resource. See “BACKUP” on page 318 for more information.
- Immediately before normal termination of the Tivoli OPC subsystem.
- When Tivoli OPC detects that a daily planning batch job has started.
- After a daily planning batch job has created a new CP.
- After CP recovery processing has successfully re-created an up-to-date CP.
Creating and Activating the NCP

The creation of a new CP is performed by the Tivoli OPC daily planning batch jobs. The two possible options are *extend* and *replan* the CP. The extend function is also used when creating a current plan for the first time. Both the extend and replan functions cause the creation of a new CP in the new current-plan (NCP) dataset.

The steps below describe in detail how the new CP is created and then brought into normal Tivoli OPC processing:

**A Daily Planning Job Starts and Is Recognized by the Tivoli OPC.**

1. A daily planning batch job starts, signals the Tivoli OPC subsystem using an ENQ, and then waits.
2. Tivoli OPC detects the ENQ from the batch job and locks the current plan preventing more updates.
3. The active CP 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.

**The Active CP Is Backed Up.**

4. The inactive CP is erased.
5. The active CP is copied to the inactive CP. They are now identical.
6. The inactive CP becomes the active CP, and the previously active becomes the inactive.
7. The JT logs are switched.
8. The CP is unlocked, and normal processing continues.
9. Tivoli OPC signals to the batch job that backup processing is complete.

**The Daily Planning Job Builds a New CP.**

11. The batch job starts executing again. The inactive CP is used (together with the LTP, AD, RD, and WS for a CP extend) as input, and a new current plan is created in the NCP and NCX datasets. While the batch job is building the new current plan, Tivoli OPC continues normal processing except that a current plan backup is not permitted because the batch job is using the inactive CP dataset.
12. The contents of the job-tracking archive dataset are copied to the dataset in the daily planning job referenced by the EQQTROUT ddname.
13. When the new CP is ready, the checkpoint dataset is updated to show if the new CP was successfully created. The NM subtask is notified that the daily planning process has completed.
14. The subsystem investigates the checkpoint dataset to see if a new current plan was successfully created. If not, Tivoli OPC continues as if a daily planning process had not run. If successful, processing continues with the next step.

**The New Current Plan Is Taken Over.**

15. The current plan is locked, preventing it from being updated.
16. The NCP is copied to the active current plan and the NCX is copied to the current plan extension (CX).
17. The job-tracking archive log is emptied.

18. The active job-tracking log now contains a record of the updates to the CP that were made while the daily plan job was running. These are read, and the CP is updated accordingly.

19. In-storage control blocks representing workstations and active operations are rebuilt from the active CP, and a data space is created from the current-plan-extension dataset.

The Newly Created Active CP Is Backed Up.

20. A CP backup is performed.

21. The CP is unlocked and normal processing continues.

22. The data from the now inactive job-tracking log is copied into the JT archive log.

Note: Consider that all LTP and CP batch planning jobs have to be excluded from SMARTBATCH DA (Data Accelerator) processing. When the SMARTBATCH DATA ACCELERATOR is used with the OPC LTP and CP 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 Tivoli OPC 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.
Chapter 12. Using Tivoli OPC Service Functions

This chapter describes the Tivoli OPC service functions, which let you:

- Activate or deactivate job submission
- Activate or deactivate automatic job and started-task recovery
- Refresh the LTP
- Refresh the RACF resource profiles
- Activate or deactivate event-triggered tracking
- Produce an authorized program analysis report (APAR) tape

Select option 9 on the Tivoli OPC main menu to see the SERVICE FUNCTIONS menu (Figure 106).

![EQQUTOPP SERVICE FUNCTIONS](image)

**Activating and Deactivating Job Submission**

When you deactivate job submission, operations are not started on automatic reporting computer 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) on the JTOPTS initialization statement.
2. Bring up Tivoli OPC 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 OPC 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 dialog, 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 19, "Automatic Recovery of Tivoli OPC-Controlled Jobs and Started Tasks" on page 243 and "What You Need for Automatic Recovery" on page 223.

Refreshing the Long-term Plan

Tivoli OPC has a safeguard against changing plans after they are used. Once an occurrence in the LTP has been scheduled in a current plan (CP), you cannot change it in the LTP, nor can you schedule it a second time in a CP. There might be times, however, when you need to bypass this safeguard. The refresh function is provided for that purpose.

**Attention:** 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 CP and produce a new CP for the current period, using a new version of the LTP. To do this, you must go through a complete replanning cycle, including producing a new LTP and CP. To do this, first run a refresh, which:

- Makes already scheduled occurrences available to both long-term and daily planning.
- Deletes the CP so that you can make a new CP.

If you want completely new plans, follow the refresh function with a CREATE of the LTP and then a daily planning EXTEND. The create LTP function is necessary only if the old LTP is no longer valid.
Refreshing RACF Resource Profiles

In-storage resource profiles are built for RACF-defined resources at Tivoli OPC 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 OPC start time, they are not immediately available to Tivoli OPC. Select the RACF RESOURCES option on the SERVICE FUNCTIONS panel to refresh the in-storage profiles and make them available to all dialog users.

Activating and Deactivating ETT

When Tivoli OPC 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 OPC is active, using the Activate and Deactivate ETT functions. See Chapter 20, “Event-Triggered Tracking” on page 267 for more information.

Producing an APAR Tape

You can use the authorized program analysis report (APAR) tape function when you are reporting a Tivoli OPC problem to the IBM Service Center. The APAR tape function generates a batch job that collects Tivoli OPC datasets 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 datasets are collected.

For a detailed description of the procedure for documenting and reporting Tivoli OPC problems, refer to Diagnosis Guide and Reference.
Chapter 13. How Tivoli OPC Selects Work for Automatic Submission

This chapter describes how Tivoli OPC decides the run order of operations on computer, nonreporting, and WTO workstations.

Tivoli OPC builds the current plan from information in the LTP, calendar database, application description database, workstation database, and resource database.

When the current plan is created, Tivoli OPC assigns start times to each operation. These start times are Tivoli OPC's estimates of when the operations should start. They are not the actual times that Tivoli OPC will start the operations, unless the operation has been designated as a time-dependent operation. Here, the start time is the time that Tivoli OPC will attempt to start the operation. (See “Creating Time-Dependent Operations” on page 114.) This is because Tivoli OPC 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 OPC must assess which operations, of those that are eligible, should be processed first. Tivoli OPC 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 OPC 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 OPC 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 OPC 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.
Scheduling

Tivoli OPC 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 OPC 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 OPC examines the priority. If the priority is also equal, Tivoli OPC 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 OPC 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 OPC 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 Controlling and Monitoring the Workload.

If sufficient resources are not available, or the operation is waiting for a particular time, Tivoli OPC assigns an appropriate extended status. The extended status identifies which type of resource the operation is waiting for.

---

The Submission Process

For Tivoli OPC 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. For more information, refer to Controlling and Monitoring the Workload.

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 C when the operation is ready to be started. Refer to Controlling and Monitoring the Workload for information on NOP.

Operations on computer workstations are given status S (started) once the operation, in this case a job or started task, has been submitted. The operation remains in S status until Tivoli OPC learns, by way of JES and SMF events, of the operation's fate. The status of the operation will then either be set to C (complete) or E (ended-in-error).

Because an operation can remain in S status for a long time, Tivoli OPC assigns appropriate extended status codes to help you identify exactly what stage of processing the operation has reached.
The following section describes how Tivoli OPC starts operations at computer nonreporting and WTO workstations. It describes changes to operation status and extended status codes. A complete list of operation status codes is provided in “Operation Status and Extended Status Codes” on page 345.

The submission process is initiated when Tivoli OPC 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.

### Submitting Work on Nonreporting Workstations

This is how Tivoli OPC 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 209.
- Tivoli OPC sets the operation status to complete.
- If there are more operations eligible to start, Tivoli OPC will move on to the next best candidate. Up to 5 operations can be started.
- The CP lock is released.

If the JTOPTS keyword QUEUELEN is specified, Tivoli OPC 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 OPC submits operations on WTO 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 209.
- If the operation has been marked with a NOP, Tivoli OPC immediately sets the operation to complete.
- Tivoli OPC 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 OPC will move on to the next best candidate. Up to 5 operations can be started.
- The CP lock is released.
- Tivoli OPC determines the destination, which can be a submit/release dataset, an XCF link, an NCF link, or blank. If the destination is blank, the WTO is issued on the system that the Tivoli OPC 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.
Scheduling

Operations on workstations defined with manual-complete-only reporting are immediately set to C status.

If the JTOPTS keyword QUEUELEN is specified, Tivoli OPC 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 OPC submits operations on computer workstations with the STC attribute:

- 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 209.
- If the operation has been marked with a NOP, Tivoli OPC immediately sets the operation to complete.
- Tivoli OPC 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 OPC attempts to find it (the member name must be the operation name) in the EQQJBLIB partitioned dataset 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 OPC 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.
- If there are more operations eligible to start, Tivoli OPC will move on to the next best candidate, up to 5 operations can be started.
- The CP lock is released.
- The destination is resolved, it can be either a submit/release dataset, an XCF link, an NCF 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 dataset referenced by ddname EQQSTC.
- Tivoli OPC then builds and issues the MVS START command. Once the command is issued and processed by JES, the procedure JCL is deleted from the EQQSTC dataset.
- The extended status of the operation is set to blank.
- When Tivoli OPC learns from JES that the started-task is executing, the extended status is set to S.
If the JTOPTS keyword QUEUELEN is specified, Tivoli OPC 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 OPC submits operations on computer 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 209.
- If the operation has been marked with a NOP, Tivoli OPC immediately sets the operation to complete.
- Tivoli OPC attempts to find the job in the JS file. The job will be found here only if a dialog user has updated the job statements, or if this operation occurrence has been previously submitted.
- If the job is not found on the JS file, Tivoli OPC attempts to find it (the member name must be the operation name) in the EQQJBLIB partitioned dataset 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 OPC 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 OPC will move on to the next best candidate. Up to 5 operations can be started.
- The CP lock is released.
- The destination is resolved. It can be a submit/release dataset, an XCF link, an NCF link, or blank. If the destination is blank, the job is directed to the system the controller is started on.
- (MVS 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.
- (MVS only.) When Tivoli OPC 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 MVS jobs, this happens when the initiator is available.

If the JTOPTS keyword QUEUELEN is specified, Tivoli OPC will start up to the number of operations defined by QUEUELEN. Refer to Customization and Tuning for information on the QUEUELEN parameter.
Chapter 14. MVS Job Tracking Overview

This chapter describes briefly how Tivoli OPC tracks MVS work. If an operation is not tracked, there is normally a problem with the Tivoli OPC configuration or the initialization statements.

Tivoli OPC 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 OPC TSO commands and Tivoli OPC subroutines to report event information to Tivoli OPC, 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 Tivoli OPC application programming interface (API).

This is how the event information is passed to Tivoli OPC:

- For MVS jobs and started tasks, MVS 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 OPC code in the exit collects information about the event and passes it to the Tivoli OPC event-creation module, EQQSSCMD, through the MVS subsystem interface. Relevant information for a job that has started would include the job name and number, its starting date and time, and (if catalog management is active) dataset information.

- All Tivoli OPC 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 EQQSSCMD 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 OPC 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 Tivoli OPC event-generation module, EQQSSCMD, through the MVS subsystem interface.

- You can generate an event if you write a program that passes parameters to the EQQUSIN, EQQUSINB, EQQUSINS, EQQUSINO, EQQUSINW, or EQQUSINT subroutines. The subroutine checks the parameters and passes
them to the Tivoli OPC event-generation module, EQQSSCMD, through the MVS subsystem interface.

When the events have been created, this is how they are passed to the controller:

1. EQQSSCMD 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 MVS subsystem interface is started. It is not necessary for Tivoli OPC itself to be active. If Tivoli OPC 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 MVS jobs and started tasks, even though they might not be relevant to a particular Tivoli OPC 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 OPC address space. The event creation programs reside in MVS common storage and do not belong to, or have access to, the data or resources of any Tivoli OPC address space that might be running on the same system or some other system.

2. The Tivoli OPC event-writer subtask of the tracker reads event records from the event writer queue and writes them to an event dataset. 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 or NCF 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 dataset until the required function is available.

4. The event manager subtask that is started at the controller processes the events, and Tivoli OPC 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 dataset 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 OPC is taken down and JES (JES2 commands are given in the example) is taken down:

Method 1

1. Take the tracker down (P OPCx).
2. Bring it up again under the master scheduler (S OPCx, SUB=MSTR).

   Remember that a tracker cannot use JES services (SYSOUT datasets, JCC) if it runs under the master scheduler.
3. Bring down JES.
4. Take the tracker down (P OPxC).

**Method 2**

1. Stop all JES activity (issue $T RDI,S=(SYSA,SYSC) on the other systems in the MAS, where SYSA and SYSB are the systems not being taken down).
2. Stop the tracker (P OPxC).
3. Stop JES.
4. Re-IPL.
5. Restart JES.
6. Restart the tracker.
7. Resume normal work (issue $T RDI,S=(SYSA,SYSC,SYSC) on the other systems in the MAS, where SYSC is the system that has just been brought back up).

**Method 3**

1. Abend JES2 ($PJES2,ABEND).
2. Stop the Tivoli OPC tracker (P OPxC).
3. Re-IPL.
4. Restart JES. This will be a hot start.
5. Restart the tracker.

---

**Time Zones and Tivoli OPC**

Tivoli OPC can support several MVS systems. The Tivoli OPC tracker collects information about activity on the MVS 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 Tivoli OPC controlling system. Each event contains a time stamp, which records the date and time that the event record was created.

The systems that Tivoli OPC supports might be in different time zones. The Tivoli OPC 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 by Tivoli OPC. If you intend to change the local clock GMT offset by using the MVS SET CLOCK command, all Tivoli OPC systems running on the affected MVS should be shut down and then restarted after the SET CLOCK command has been issued. The GMT offset is stored by Tivoli OPC when the subsystem is started.

The Tivoli OPC controller processes event records from all the systems in your Tivoli OPC 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 OPC databases and datasets are expressed in the local time of the controlling system.
MVS Tracking

Time values appearing on Tivoli OPC printed reports are also expressed in the local time of the Tivoli OPC 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 by Tivoli OPC to rebuild the JCL when you request a step restart.

Tivoli OPC 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:** MVS

**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. This function is also used by the catalog management and step restart functions.

**Tracker platforms supported:** All

**Step restart**

This function checks whether a job is restartable from a certain step, and tailors a job, if required, to run again from one named step to another, or to the end of the job.

**Tracker platforms supported:** MVS

**Catalog management**

This function can back out changes to catalogs and can delete datasets that are created by the failed job. This prevents jobs failing on reruns because of this JCL:

```
//OUTDS DD DSN=NEW.DATA.SET,DISP=(NEW,CATLG,CATLG),...
```

The catalog management and step restart functions work together so that, if you need to restart from a certain step, catalog management makes the necessary changes to the catalog.

**Tracker platforms supported:** MVS

**Automatic recovery**

The `//OPC RECOVER` job statement controls automatic recovery. Parameters on this statement specify whether Tivoli OPC should start other occurrences, delete steps, and so on. If the catalog management function is required, this is invoked before the rerun.

**Tracker platforms supported:** All (but step restart and catalog management functions are possible only on MVS 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
Recovery and Restart

details (including job log and job statements if available) are kept there for a
period of time that you specify in initialization statements.

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 225.

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

You can specify that the tracker always sends the job log to the controller when a
job ends, but this degrades performance on larger systems. If the retrieval is
delayed (on demand), the controller requests the log only when it needs it, which
can be for:

- Step restart, with or without catalog management
- A dialog request to browse the job log
- Automatic recovery needing step restart

The disadvantages of delayed retrieval are that:

- The log may have been archived. Tivoli OPC has an exit, EQQUX010, which
  is invoked to retrieve the archived log.
- The dialog user can experience a delay.

Another possibility is to use the delayed retrieval via Data Store (OPC Archiver
address space). In all the possible options for Job Log retrieval, the JCC task is
always involved, but you do not need to specify JCCTASK(YES) in the OPCOPTS
statement unless you also want the JCC to check the log for errors. You need to
specify JCCTASK(DST) if you are using the Data Store option. Refer to
Customization and Tuning for more details about the JCC.

To control job log retrieval when Data Store is not being used:

1. Specify the JOBLOGRETRIEVAL keyword on the JOBOPTS statement for the
   tracker where the job will run.
2. Specify the MAXNUMUSYS keyword on the JOBOPTS statement for the tracker where the job will run, to specify how much user SYSOUT data should be retrieved.

3. Specify the JOBLOGDEST keyword on the OPCOPTS statement, to specify the destination where SYSOUT can be found, if it is no longer on the original system.

To control Job log retrieval when Data Store is being used:

1. For each involved tracker specify on the JOBOPTS statement:
   - JOBLOGRETRIEVAL(DELAYEDDST) keyword
   - DSTCLASS keyword
   - DSTFILTER keyword (optionally)

2. For each involved Data Store (one for each JES spool) specify the appropriate keywords (refer to Customization and Tuning for details), for example, MAXSTOL and MAXSYSL.

**Considerations for MVS Job Logs**

Tivoli OPC cannot process SYSOUT output that is written to a nonheld class. If you have JES3, the class must be defined as an external writer SYSOUT class.

**Considerations for Immediate Retrieval**

Tivoli OPC automatically determines the message class of the job log. Do not change the SYSOUT message class or split the output into multiple classes using the JESDS JCL statement, or Tivoli OPC will not find the job log.

If your job log SYSOUT is processed by a SYSOUT archiver you can requeue the job log to the archiver class when the JCC has copied the job log by specifying the CATMDISP keyword on the JOBOPTS statement. The archiver should not use the same class as the MSGCLASS, because it will compete with JCC for the SYSOUT. If the JCC loses the race to process the SYSOUT, step-level restart of the operation is not possible.

Use the STORELOG keyword of the OPCOPTS statement to specify whether immediately-retrieved logs should be saved in the job repository dataset.

**Considerations for Delayed Retrieval**

The job log can have a new class (CATMCLAS keyword), be on another system (JOBLOGDEST keyword), or be archived (EQQUX010 exit).

**Consideration for Data Store Delayed Retrieval**

Everything the Job Log finds in its reserved held class will be archived by the OPC Data Store and requeued to the original msgclass. The Job Log is requeued to the reserved class by the JCC task. A filter criteria is provided by the JCC task to requeue to the Data Store via the DSTFILTER keyword. Do not split the output in multiple classes using the JESDS JCL statement. This could cause Job Log retrieval failure as the sysout could not be requeued to the reserved Data Store class.
Recovery and Restart

Where to Find More Information
On the JOBOPTS, OPCOPTS, and ROUTOPTS statements
Refer to Customization and Tuning.

On coding the EQQUX010 exit
Refer to Customization and Tuning.

On browsing the job log
Refer to Controlling and Monitoring the Workload.

On customizing the data store
Refer to Customization and Tuning.

What You Need for Step Restart
1. Specify CATMGT(YES) on the OPCOPTS statement for trackers where you need step restart, and on the controller.

2. Specify JOBLOGRETRIEVAL to a value other than NONE on the JOBOPTS statement, and review the CATMCLAS and CATMDISP parameters, on the trackers where you need step restart.

3. Specify MSGLEVEL(1,1), if this is not the default, for jobs where you need step restart.

4. For step restart, you need job log retrieval. See also “What You Need for Job Log Retrieval” on page 220.

5. If the job includes GDG datasets, specify immediate or deferred catalog management for the operation.

Where to Find More Information
On the OPCOPTS and JOBOPTS statements
Refer to Customization and Tuning.

On step restart
See Chapter 17, “Restarting an MVS Job at a Step” on page 229 and “Step-Level Catalog Management” on page 241.

On using the Modify Current Plan dialog
Refer to Controlling and Monitoring the Workload.

What You Need for Catalog Management
1. Specify CATMGT(YES) on the OPCOPTS statement for trackers where you need catalog management, and on the controller.

2. Specify a value other than NONE for JOBLOGRETRIEVAL on the JOBOPTS statement for trackers where you need catalog management for step restart (deleting and uncataloging, for example, only for specified steps). For step-level catalog management, see also “What You Need for Step Restart.”

3. Specify any value for JOBLOGRETRIEVAL on the JOBOPTS statement for trackers where you need catalog management only for restarting from the beginning of a job. Catalog management does not need the job log in this case.

4. Review the CATACT parameter on the OPCOPTS statement for the controller.

5. Specify deferred or immediate catalog management for each operation where you need this function. You can specify this using the application and job description dialogs, the program interface, and the batch loader.
Where to Find More Information

On the OPCOPTS and JOBOPTS statements
Refer to *Customization and Tuning*.

On catalog management
See Chapter 18, “Catalog Management” on page 235.

On specifying catalog management for an operation
See “Options that Apply to Jobs and Started Tasks” on page 101.

On using the Modify Current Plan dialog
Refer to *Controlling and Monitoring the Workload*.

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 Tivoli OPC 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 catalog management function before rerunning a job, see also “What You Need for Catalog Management” on page 222.

Where to Find More Information

On the OPCOPTS and AROPTS statements
Refer to *Customization and Tuning*.

On the automatic recovery function, and more examples
See Chapter 19, “Automatic Recovery of Tivoli OPC-Controlled Jobs and Started Tasks” on page 243.

On the syntax of the ///c5197%OPC RECOVER statement

On the SERVICE FUNCTIONS menu.
See Chapter 12, “Using Tivoli OPC Service Functions” on page 205.

What You Need for the History Function

1. Specify OPERHISTORY(YES) and the DB2SYSTEM keyword on the controller BATCHOPT and OPCOPTS statements.
2. If you want to invoke the catalog management function before rerunning a job, see also “What You Need for Catalog Management” on page 222.

Where to Find More Information

On creating a DB2 database
Refer to *Installation Guide*.

On the BATCHOPT and OPCOPTS statements
Refer to *Customization and Tuning*.

On using the history function
Refer to *Controlling and Monitoring the Workload*.
Chapter 16. Setting Error Codes

This chapter describes how Tivoli OPC 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 dialog.

When a computer workstation operation (either a job or a started task) ends, Tivoli OPC determines the conditions under which it ended by inspecting the error code of the operation. Tivoli OPC 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 Tivoli OPC 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 OPC will, upon identifying an error, set the operation status to E (ended-in-error).

How Tivoli OPC Error Codes Are Set

Tivoli OPC error codes can be set in these ways:

- Automatically by Tivoli OPC, based either on the completion codes of the job or started-task step, or on Tivoli OPC internal error detection
- By the job completion checker. See “Setting Error Codes Using the Job Completion Checker” on page 226.
- Manually from the Ready List dialog or from the MCP dialog
- Using the EQQUSIN subroutine. Refer to Customization and Tuning.
- Using the program interface. Refer to Programming Interfaces.
- Using the OSTAT command in batch or TSO. See Appendix A, “TSO Commands” on page 317.
- Using Workload Monitor/2. Refer to Workload Monitor/2 User’s Guide.

In most cases, Tivoli OPC sets error codes using the MVS user and system completion codes. In some special cases (for example, when there has been a JCL error), Tivoli OPC sets a special error code to enable it to react correctly to the situation.

Setting Error Codes Using Completion Codes

Tivoli OPC sets error codes using completion codes when they are available. This is how Tivoli OPC 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 MVS completion code 8, for example, is converted to the Tivoli OPC error code 0008. This also depends on the values specified on EWTROPTS.
- 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 50C4.
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 OPC does not use the completion code of a job or started task to set the error code. Instead, it sets one of the codes listed in “Error Codes” on page 347.

**Note:** 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.

**Attention:** Tivoli OPC 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 dataset 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 dataset of a job or started task and set an error code for the operation, depending on the results of the scan. For more details on the JCC, refer to *Customization and Tuning*.

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 Tivoli OPC 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 dataset, 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, refer to *Customization and Tuning*.

Refer to *Tivoli OPC Customization and Tuning*, Chapter 6. Using the Job Completion Checker, for information about the JCC Catalog management Interaction.
Using Error Codes to Set Operations to Ended-in-Error

Depending on the application, you might consider that not all Tivoli OPC 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 OPC to ignore certain error codes, using the NOERROR initialization statement. Here you specify a list of error codes that should not cause the operation to be marked as ended-in-error. If the error code in question is found in this list, the operation is treated as having ended normally.

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 OPC does not have the relevant step information.

If the error code is numeric, after checking the NOERROR list, Tivoli OPC checks the error code against the HIGHRC keyword of the JTOPTS initialization statement. If HIGHRC has been specified at operation level, Tivoli OPC 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.

Resetting Operations Based on Error Codes

Tivoli OPC 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 OPC 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 OPC does not restart the operation automatically.

Determining the Success or Failure of a Job

This summarizes how Tivoli OPC determines the next status of an operation that ends:

1. Tivoli OPC creates a job-end event with the highest or last return code, depending on the RETCODE keyword of the EWTROPTS 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.
Error Codes

The event reaches the event queue at the controller.

3. If the return code is 0, Tivoli OPC sets the operation status to C. Otherwise, it continues checking.

4. If the operation definition specifies no error tracking, Tivoli OPC 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 OPC 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 OPC 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 OPC 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. Restarting an MVS Job at a Step

This chapter explains how you can rerun MVS jobs and started tasks and select the steps that will be included in the restarted job. Unless otherwise stated, the description applies to both jobs and started tasks on MVS systems.

The step restart function is closely connected with the catalog management and automatic recovery functions. See Chapter 15, “Planning for Recovery and Restart” on page 219 for an overview of these functions, and a description of how to activate them.

Selecting the Steps

When you rerun a job, you select the scope of the restart. The scope of the restart is defined by the first and last step to be included in the rerun.

For jobs that have ended in error, rerun the job using the ended-in-error list in the Modify Current Plan (MCP) dialog. Use the SR (step restart) row command beside the operation that you want to rerun.

You specify the restart scope from the MCP dialog STEP RESTART SELECTION LIST panel, shown in Figure 107.

<table>
<thead>
<tr>
<th>Command</th>
<th>Scroll</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary commands:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GO - Build restart JCL</td>
<td>CAT - List catalog updates</td>
<td></td>
</tr>
<tr>
<td>JOB - Browse joblog</td>
<td>OI - Browse operator instructions</td>
<td></td>
</tr>
<tr>
<td>Selection options:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>S - Restart step</td>
<td>X - Excluded step</td>
<td>E - Restart end-step</td>
</tr>
<tr>
<td>Application data</td>
<td>PAYTAXI 97/06/12 14.44</td>
<td></td>
</tr>
<tr>
<td>Operation data</td>
<td>PAYTAXI JOB/zerodot6/zerodot8/zerodot CPU1 10</td>
<td></td>
</tr>
<tr>
<td>Current selection</td>
<td>Invoke JCL edit ==&gt; N</td>
<td></td>
</tr>
<tr>
<td>Sel</td>
<td>Stepname</td>
<td>Procstep name</td>
</tr>
<tr>
<td>s :</td>
<td>PAYTAX1A</td>
<td>PAYSTC1</td>
</tr>
<tr>
<td>:0</td>
<td>PAYTAX1B</td>
<td>PAYM07</td>
</tr>
<tr>
<td>:0</td>
<td>PAYTAXIC</td>
<td>PAY01</td>
</tr>
<tr>
<td>:0</td>
<td>PAYTAX1D</td>
<td>PAYM16</td>
</tr>
</tbody>
</table>

Figure 107. EQQMERSL—Step Restart Selection List

In Figure 107, step PAYTAX1B is selected as the restart point (with the S row command), and the restarted job will end at step PAYTAX1D (the end step defaults to the last step). The scope of the restart is steps 2 to 4. Refer to Controlling and Monitoring the Workload for details of the MCP dialog.
How Tivoli OPC Gets the Job Log

Tivoli OPC needs the job log to be able to build the JCL for step restart. When the job ends (immediate retrieval), or on demand from the controller (delayed retrieval), the tracker retrieves the job log, compresses it, and sends it to the controller, which stores it in the job repository datasets (EQQJSxDS).

To choose the job retrieval options, you must balance the performance cost of immediate retrieval and the possible inconvenience of delayed retrieval. These are the main options:

**Immediate retrieval of all jobs**
This is the most expensive option. Consider the cost of sending job logs to the controller if the tracker is on another processor, and consider the space required in the job repository datasets (the STORELOG keyword specifies whether all logs are saved). But you can use the Tivoli OPC dialog to browse the job log of all completed jobs.

**Immediate retrieval of ended-in-error jobs**
If your error procedure normally involves the operator browsing the job log, or the step restart or step-level catalog management functions, this is a good choice. The STORELOG keyword determines which logs are saved in the job repository datasets.

**Delayed retrieval**
Use this to avoid the transfer of job logs unless it is necessary. With this option, a retrieved log is always saved, regardless of the setting of STORELOG.

**Delayed retrieval via data store**
Use this to avoid the transfer of job logs unless it is necessary. The Job Log is saved in data store address space and later sent back to the Controller if requested. With this option, a retrieved log is always saved, regardless of the setting of the STORELOG keyword.

See “What You Need for Job Log Retrieval” on page 220 for more details about the retrieval options.

**Note:** "orphan-joblogs" resulting from the execution of delete occurrence commands, issued from the MCP error list (option 5.4), might remain in the job repository data seta (EQQJSxDS). These can be discarded from JS files by running a batch job such as EQQPIDJ and EQQPIFJV.

How Tivoli OPC Tailors the Job

When you select step restart, Tivoli OPC builds the necessary JCL like this:

- All relative GDG references are changed to the corresponding absolute name. References within procedures are resolved by inserting JCL overrides in the appropriate order.

- A RESTART parameter is appended to the job card. If a RESTART card already exists in the JCL, the original card is commented, the new RESTART card is inserted, and message EQQM361 is issued to indicate that a statement in the original JCL has been replaced.

- COND=ONLY parameters are inserted in all steps before the restart step if the JCL is for a started task. If a COND parameter is already defined, and does not specify COND=ONLY, the new parameter is inserted as a JCL comment.
line and message EQQM362 will be issued. If the started task JCL contains nested procedures, Tivoli OPC may not be able to insert the COND statement. If this occurs, it issues message EQQM369.

- COND=(0,LE) parameters are inserted in all steps beyond the last step to be rerun.

When the step restart is confirmed by the dialog user, Tivoli OPC:

1. Stores the JCL in the job repository dataset.
2. Takes any specified catalog management actions (see Chapter 18, “Catalog Management” on page 235).
3. Submits the job.

Notes:

1. Tivoli OPC does not recognize SYSIN data unless it is specified using:
   - DD *
   - DD DATA
   - DD dataset specification parameters

   Other SYSIN (for example, using JES2 and JES3 facilities) will not be included in the restart JCL.

2. These messages and message numbers are assumed to be present in the job log SYSOUT, and must not be suppressed by any JES initialization statements:
   - IEF142I
   - IEF236I
   - IEF272I
   - IEF285I
   - IEF373I
   - IEF374I
   - IEF377I
   - IEF472I
   - IEFC653I

3. Started tasks submitted outside Tivoli OPC control must have the string OPC in columns 73-75 of the JCL procedure to enable Tivoli OPC to rebuild the JCL procedure from the job log.

4. JES2 control statements in jobs submitted outside Tivoli OPC control are not included in the rebuilt JCL for JES2 level 4.2.2 or lower, because the statements are commented out by JES2 in the job log.

5. Results are unpredictable if you are using JES2 Version 3 or a lower version.

6. INCLUDE JCL statements are considered in the same manner as PROC JCL statements.

How Tivoli OPC Validates the Job

When Tivoli OPC has rebuilt the JCL for restart, it validates it. If problems are detected, Tivoli OPC inserts messages as JCL comment cards into the job log to direct your attention to the problem. If catalog management is requested (see Chapter 18, “Catalog Management” on page 235), Tivoli OPC inserts messages to identify the datasets that will be deleted, uncataloged or recataloged by the catalog management function before the job is restarted.
Tivoli OPC checks for the potential problems listed in Figure 108 and inserts message lines in the JCL at the place where the problem is detected:

<table>
<thead>
<tr>
<th>Potential problem</th>
<th>Message number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Referenced datasets are deleted, passed, or uncataloged in a step before the restart step. A restart can cause a JCL error.</td>
<td>EQQM372</td>
</tr>
<tr>
<td>There are DISP=MOD references to datasets within the restart scope. Restarts can cause data to be wrongly appended to these datasets.</td>
<td>EQQM371</td>
</tr>
<tr>
<td>There are backward references to a step before the restart step. Tivoli OPC cannot resolve such backward references.</td>
<td>EQQM365</td>
</tr>
<tr>
<td>There are COND or IF statements in the restart scope. MVS ignores COND or IF parameters that refer to a step before the restart step.</td>
<td>EQQM368</td>
</tr>
<tr>
<td>There is already a COND statement where Tivoli OPC needs to insert one.</td>
<td>EQQM362</td>
</tr>
<tr>
<td>There are Tivoli OPC automatic recovery statements in the JCL. See Chapter 19, &quot;Automatic Recovery of Tivoli OPC-Controlled Jobs and Started Tasks&quot; on page 243 for details of automatic job recovery. If automatic recovery has already occurred for the job, the recovery statements that have already been matched are now commented out and ignored, but Tivoli OPC will action any further recovery statements that match on subsequent reruns, which may not be what you want when you select step restart from the dialog. This is especially true when there is a RESSTEP keyword on a recovery statement, which can change the scope of the restart.</td>
<td>EQQM379</td>
</tr>
<tr>
<td>There are JES3 DJC control statements in the JCL. A JES3 dependent job control net can be destroyed.</td>
<td>EQQM360</td>
</tr>
<tr>
<td>Instream data is missing, for jobs not submitted by Tivoli OPC. This includes SYSIN data, passwords on the job statement, and data in a JES3 DATASET statement. Tivoli OPC cannot capture this data for jobs that it does not submit.</td>
<td>EQQM380</td>
</tr>
<tr>
<td>There are nested procedures, and Tivoli OPC needs to modify JCL in a nested procedure. Modifications (insertion of COND parameters and overrides for GDG references) can be made only in the immediate (level 1) procedure.</td>
<td>EQQM369</td>
</tr>
<tr>
<td>Message IEF377I was detected in the joblog. A data set was allocated but not cataloged, because a data set with the same name already exists on another volume. This could lead to the deletion of the cataloged data set instead of the data set just allocated by the job.</td>
<td>EQQM378</td>
</tr>
</tbody>
</table>

Figure 108. How Tivoli OPC Validates JCL for Step Restart

When you specify step restart in the dialog, Tivoli OPC issues a warning message for any of these potential problems, and you can edit the JCL.
Figure 109 shows a job prepared for step restart that has potential problems.

Figure 109. Modified JCL for a Step Restart

In edit, you see Tivoli OPC messages that document the inconsistencies (1). The messages are inserted in the JCL as ISPF message lines in the format:

```plaintext
///c5197> msgnumber message text
```

The message lines are temporary ISPF rows. If you want the messages to be kept with the JCL, use the ISPF row command MD to make the temporary row a dataline, which will be saved with the JCL in the job repository. If you use catalogued procedures, Tivoli OPC displays the procedure JCL statement that causes the difficulty as a comment in the job. All datasets that Tivoli OPC will attempt to perform catalog management actions for are indicated (2).
Chapter 18. Catalog Management

This chapter shows you how Tivoli OPC can help you with the recovery and restart of MVS jobs and started tasks by:

- Deleting datasets that were created in a job
- Uncataloging datasets that were cataloged in a job
- Cataloging datasets that were uncataloged in a job

In this chapter, job refers to both an MVS job and a started task, unless otherwise stated. Catalog management is available for jobs that are tracked by Tivoli OPC on the MVS operating system and are in the current plan, whether submitted by Tivoli OPC or not.

Tivoli OPC catalog management is a system that tidies up after failed jobs—it is not intended to tidy up before a job first runs. If you prefer this approach, look at the member EQQDELDS in the supplied sample library, which describes a program that deletes datasets with NEW disposition that are not referenced with a different disposition (OLD/SHR) in previous job steps. The EQQDELDS program, being a sample, is not part of the Tivoli OPC product, but you can use it if you prefer this solution.

Selecting Catalog Management Options

See “What You Need for Catalog Management” on page 222 for the details of the initialization options that you need for catalog management. Your main options are immediate or deferred catalog management, and job or step-level catalog management.

Immediate and Deferred Catalog Management

You must specify immediate or deferred catalog management for each operation where you want catalog management. Do this using the Application Description dialog (see “Options that Apply to Jobs and Started Tasks” on page 101 for more information), the Job Description dialog, the batch loader, the program interface, or the Modify Current Plan (MCP) dialog.

Specify immediate if you want automatic catalog management when a job fails. If a tracked job ends in error, and immediate catalog management is specified for the operation, the Tivoli OPC controller checks for dataset information in the current plan from the first step up to and including the failing step of that job. If there is dataset information, the Tivoli OPC controller sends instructions to the tracker to perform catalog management actions on the required datasets. Note that when an operation is marked as immediate, if a step restart is attempted it might result in a new job failure because the dataset cleanup actions were performed in the same way as for a job rerun.

Specify deferred to control catalog management from the Tivoli OPC Modify Current Plan (MCP) dialog. If a tracked job ends in error and deferred catalog management actions have been defined for the operation, or when a rerun is requested for an operation that specifies deferred catalog management, you must initiate the catalog management action from the MCP dialog. In the dialog you can see what action Tivoli OPC will take for each of the affected datasets. You can discard the action for any or all of the datasets if required. When you initiate the
catalog management action, the Tivoli OPC controller sends instructions to the tracker to perform catalog management actions on the required datasets.

In both cases, immediate and deferred, actions for datasets belonging to flushed steps are not performed. Actions are also not performed for datasets referenced in previous steps with a disposition of OLD, SHR, or MOD, whenever these steps are included in the new run.

### Job and Step-Level Catalog Management

Step-level catalog management makes it possible to take catalog management actions for a range of steps, rather than for all steps up to the last step executed (job-level catalog management). This is an initialization option; you do not specify it at operation level.

Specify a value of JOBLOGRETRIEVAL other than NONE on the JOBOPTS statement if you want to be able to do step-level catalog management. Otherwise, Tivoli OPC takes catalog management actions for all steps up to the last step executed.

Catalog management marks all datasets 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/MOD, then they are not marked for action.

In the step restart path, the dataset list panel will display any dataset that is to be excluded because it was referenced in a previous step, with the action extended field set to blank instead of an asterisk, in the same way as it identifies datasets excluded because they are outside the step restart range. Datasets excluded because they belong to flushed steps are not displayed at all.

In the job rerun path, datasets excluded because they belong to flushed steps or because they have been referenced in previous steps, are not displayed at all.

The operator can refuse the deletion if it is not appropriate in the context of the jobstream to be restarted.

### Catalog Management Exit

You can use the Tivoli OPC event filtering exit, EQQUX004, to reduce the amount of system-wide catalog management event processing that the tracker performs. You can use the Tivoli OPC pre-catalog management exit, EQQUX008, to modify the catalog management action for a dataset, discard the action for a dataset, or discard all actions for all datasets. Refer to Customization and Tuning for information on the exits.

### When Does Tivoli OPC Perform Catalog Management Actions?

If you define an operation with immediate catalog management, Tivoli OPC initiates the actions as soon as the job fails, with these exceptions:

- The job fails with a Tivoli OPC error code that is set before, or at, job submission:
  - OSEQ
  - OSUB
OSUF
OSUP
OJCV
JCLI

- 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. Tivoli OPC automatically changes catalog management action from immediate to deferred.

- The EQQUX008 exit revokes the catalog management action.

If you define an operation with deferred catalog management, Tivoli OPC takes the action only when you initiate it from the Modify Current Plan (MCP) dialog. You normally use the SR (step restart) or JR (job restart) row commands from the ended-in-error list. Refer to Controlling and Monitoring the Workload for information on the MCP dialog.

If both automatic recovery and catalog management actions are defined for an operation, Tivoli OPC takes the catalog management actions first.

---

What Action Does Tivoli OPC Take for Affected Datasets?

Catalog management actions are designed to restore the entries in the catalog that were modified by the job to the state that they were in before the job started. Catalog management deletes or uncatalogs datasets, depending on the CATACT keyword of the OPCOPTS statement. Catalog management supports a maximum of 255 concatenated datasets for each single ddname in a JCL statement.

When Tivoli OPC performs catalog management processing for a job, datasets and GDG generations that have been created in that job are either deleted or uncataloged. Uncataloged datasets are cataloged. Figure 110 on page 238 describes the action that Tivoli OPC takes for each dataset disposition.
### Catalog Management Actions for Dataset Dispositions in a Job

<table>
<thead>
<tr>
<th>DISP reference to the dataset</th>
<th>Is the dataset cataloged at job start?</th>
<th>Actions performed by Tivoli OPC</th>
</tr>
</thead>
</table>
| NEW,CATLG                    | No                                   | • Delete the dataset if CATACT is defined as DELETE.  
                                  |                                      | • Uncatalog the dataset if CATACT is defined as UNCAT. |
| NEW,KEEP                     | No                                   | No action unless it is an SMS-managed dataset, in which case:  
                                  |                                      | • Delete the dataset if CATACT is defined as DELETE. |
| OLD,UNCATLG                  | Yes                                  | Catalog the dataset.             |
| SHR,UNCATLG                  | Yes                                  | No action.                       |
| MOD,CATLG                    | Yes                                  | No action.                       |
| Mod,UNCATLG                  | No                                   | Catalog the dataset.             |
| ,,CATLG (abnormal step end)   | No                                   | • Delete the dataset if CATACT is defined as DELETE.  
                                  |                                      | • Uncatalog the dataset if CATACT is defined as UNCAT. |
| ,,UNCATLG (abnormal step end) | Yes                                  | Catalog the dataset.             |

If you restart a job from a specified step, Tivoli OPC does not try to restore a dataset catalog status that was changed before the restart step.

If the job in Figure 111 on page 239 was tracked by Tivoli OPC, failed at the third step 3, and had immediate catalog management defined:

1. This dataset would be uncataloged, and if catalog action were defined as DELETE, would be scratched.
2. This dataset would be cataloged.
3. This dataset would be cataloged.
Actions for datasets with DISP=NEW are not executed when they belong to flushed steps or when they were referenced in previous steps with DISP=SHR/OLD/MOD that are included in the new run range.

SMS Managed Datasets

SMS does not allow UNCATALOG without DELETE, so the only valid action to specify in the controller OPCOPTS statement is CATACT(DELETE). If CATACT(UNCAT) is specified, then Tivoli OPC writes a message in the message log on the controller system, and no request for action on the dataset is passed to the tracker where the job was run.

Migrated Datasets

If a dataset is migrated by DFHSM (VOLSER=MIGRAT) between creation and the time when the catalog management function attempts to take some action for the job, Tivoli OPC initiates DFHSM to request recall with the NOWAIT option. Tivoli OPC expiration time for the DFHSM recall of the dataset is calculated as: current time + the number of minutes specified in the JOBOPTS keyword CHSMWAIT (refer to Customization and Tuning). If the recall has not been performed within this time, catalog management for the dataset is set to error, and no further catalog management action is taken for the dataset.

Generation Data Groups

If you define UNCAT as the action for created generation datasets in an SMS environment, this will have no effect. In a non-SMS environment, the entry for the dataset will be removed from the GDG base, leaving the generation dataset uncataloged. In either environment, DELETE causes the dataset to be scratched, and its entry in the catalog is removed. The absolute generation name is used during catalog management processing.

The definition of the GDG base determines whether a generation dataset is removed from the VTOC and the catalog when the maximum number of generations for a GDG base is reached in a job. Tivoli OPC does not try to override these rules. You cannot restore GDGs that have rolled off the group as a result of new datasets created in the failing job. Catalog management can only reset the catalog; it cannot recover data that has been deleted.
Recovering from Workstation Failures

Tivoli OPC can transfer work from one workstation to another in the event of a failure or a workstation becoming unavailable. If you specify that operations are automatically rerouted to alternate workstations, ensure that your systems are symmetrical as far as dataset 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 catalog management processing:

- **ERROR** All operations that have catalog management defined as immediate will be changed to have deferred catalog management. This means that you must initiate or discard the action from the Modify Current Plan dialog.
- **RESTART** Catalog management action is discarded by Tivoli OPC.
- **LEAVE** Catalog management action is determined by the succeeding status of the operation.

How Does Catalog Management Work?

When any job starts, a Tivoli OPC tracker that has catalog management specified collects the catalog status of any datasets that:

- Are defined with DD statements in the JCL and
- Have these normal disposition combinations:
  - (NEW,CATLG)
  - (NEW,KEEP)
  - (OLD,UNCATLG)
  - (SHR,UNCATLG)
  - (MOD,CATLG)
  - (MOD,UNCATLG)
- Or have these abnormal dispositions:
  - (,,CATLG)
  - (,,UNCATLG)

Only this information is tracked because these are the cases where the status of the dataset will change in the catalog. Tivoli OPC ignores SYSIN, SYSOUT, and dynamically allocated datasets (for example, IDCAMS allocations).

It sends the dataset information to the controller with the job-start event record. The information for each dataset is:

- Disposition
- ddbname
- Step name
- Dataset names
- Volume serial numbers

---

1 In SMS environments only.
The controller stores the information in the job repository if the job name matches one in the current plan, and catalog management is defined for the operation.

If an operation at a computer workstation ends in error, Tivoli OPC checks if catalog management is defined for the operation. If catalog management is defined for an operation that has ended in error, Tivoli OPC does not allow any user to change the status of the operation until the outstanding actions are either completed or revoked (discarded).

If automatic job recovery is used, catalog management and automatic recovery cooperate to ensure that catalog management actions are either completed or revoked before any automatic recovery action is started for the operation. See Chapter 19, “Automatic Recovery of Tivoli OPC-Controlled Jobs and Started Tasks” on page 243 for information on automatic job recovery.

When catalog management actions are needed, the controller sends instructions for taking these actions to the tracker. The Tivoli OPC tracker recognizes catalog management instructions passed from the controller. The tracker takes suitable actions, such as uncataloging and scratching a dataset. As the tracker processes the catalog management instructions for a particular operation, it reports the status to the controller. This ensures that catalog management actions are not missed if the tracker is shut down. The tracker must have RACF authority to take the actions defined for a dataset. If a security violation occurs, the catalog management status for the operation is set to error, and the dataset cleanup must be performed manually.

Tivoli OPC maintains special status codes for catalog management actions. See “Catalog Management Status Codes” on page 348. You can include these codes (field name CM status) in your ended-in-error and ready list layouts.

If the Tivoli OPC controller on which the catalog management function is active is shutdown and restarted without catalog management, any catalog management actions that were in progress before the shutdown will continue, but no new catalog management actions will be initiated.

**Step-Level Catalog Management**

See Chapter 17, “Restarting an MVS Job at a Step” on page 229 for a general description of step restart. This section describes the extra catalog management requirements and facilities for step restart.

If a Tivoli OPC tracker specifies a value of JOBLOGRETRIEVAL other than NONE on the JOBOPTS initialization statement, it retrieves the job log (either immediate or delayed), compresses it, and sends it to the controller, which (assuming the STORELOG keyword does not suppress storage) stores it in the job repository dataset. See “How Tivoli OPC Gets the Job Log” on page 230 for a description of job log retrieval.

If a job ends in error and deferred catalog management actions have been defined for the operation, or when you request step restart for a job that specifies either immediate or deferred catalog management actions, you can select the step from which to restart the job if job log information is available.
When you request step restart, Tivoli OPC builds the JCL required for the restart, as described in “How Tivoli OPC Tailors the Job” on page 230. When Tivoli OPC has built the JCL for restart, it validates the JCL to identify any potential problems, as described in “How Tivoli OPC Validates the Job” on page 231. If catalog management is requested, messages are inserted to identify the datasets that will be deleted, uncataloged or recataloged by the catalog management function before the job is restarted.

When the step restart is confirmed by the dialog user, Tivoli OPC:

1. Stores the JCL in the job repository dataset.
2. Takes any specified catalog management actions.
3. Submits the job.

Tivoli OPC takes its dataset information from the catalog and from the job log from the last run. Recataloging datasets is impossible if the relevant log is no longer available. Consider this example:

```plaintext
/S1 EXEC PGM=P1
/S2 EXEC PGM=P2
/S3 EXEC PGM=P3
/DD1 DD DSN=TEST.FILE.ONE,DISP=(NEW,CATLG,DELETE),
   VOL=SER=TSOL/zerodot5,SPACE=(TRK,(1,1)),UNIT=339/zerodot,
   DCB=(BLKSIZE=312/zerodot,LRECL=8/zerodot,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:

1. The job fails at step S6.
2. From the error list, you restart the job from step S5.
3. Tivoli OPC does not take any catalog management 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. Tivoli OPC now deletes the dataset that was created in the first run in step S3. This does not happen for Release 2.1 or earlier releases of Tivoli OPC.
7. Tivoli OPC does not catalog the dataset that was uncataloged in step S4. This is because Tivoli OPC checks before each restart whether the datasets in the JCL are in the catalog, and it also has the job log information from the last run, but in this case the information about TEST.FILE.TWO is not found in either the catalog or in the job log from the previous run of the job, so the dataset cannot be cataloged.
Chapter 19. Automatic Recovery of Tivoli OPC-Controlled Jobs and Started Tasks

Tivoli OPC supports the automatic recovery of failed jobs or started tasks if the job or started task fails. If you are running MVS/ESA SP Version 4 Release 1 (or later) with the MVS/ESA cross-system coupling facility (XCF), you can specify that Tivoli OPC 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 Tivoli OPC dialogs.

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 OPC 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 that recovery will be attempted for and how the recovery will be achieved. If the error type is not one that you have covered in your specification, the failed operation will stay on the Operations Ended-in-Error list.

Tivoli OPC retrieves the JCL for automatic recovery from the JCL repository (JS) dataset. This means that automatic recovery can take place only for jobs or started tasks submitted by Tivoli OPC.

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 OPC, such as JCLI, CCUN, JCL, CAN, PCAN, OFxx, or OSxx.
  - An error code set by the job-completion-checker function of Tivoli OPC.

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.
Automatic Recovery

Tivoli OPC 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.
- The error code matches the error code from the Tivoli OPC job and started-task
  tracking function.
- The return code matches the step return codes or abend codes from the Tivoli
  OPC 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 dataset recovery before restarting your
  main application. See “Adding Predecessor Recovery Occurrences to the
  Current Plan” on page 260.
- 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 OPC selects the failed step is described in
“Deciding Which Step of an Operation Has Failed” on page 259. For example, the
error selection criteria if error E might match more than one failed step. In this
Automatic Recovery

If a failing job operation has catalog management action defined and it contains Tivoli OPC automatic recovery statements, catalog management action is performed before automatic recovery. However, note that if you have defined RESSTEP as an automatic recovery statement in the JCL, Tivoli OPC performs the catalog management action for all steps from the RESSTEP value up to and including the last executed step, even if that step was executed after the failing step. If the error occurs in a step before the RESSTEP statement, the catalog management action is performed for the entire job.

The example in Figure 112 explains how catalog management and automatic recovery can work together to ensure that a job is cleaned up and restarted.

```plaintext
///c5197
///c5197%OPC RECOVER ERRSTEP=STEPFAIL,JOBCODE=JCL,DELSTEP=STEPFAIL
///c5197%OPC SCAN
//STEP1 EXEC PGM=IEFBR14
//DD1 DD DSN=EID.EID4R2.CATTEST,DISP=(,CATLG),
// 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 112. Catalog Management Cooperating with Automatic Job Recovery

In Figure 112, 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 catalog management, the job will immediately fail in STEP1 with a JCL error—duplicate dataset. When catalog management is active for the operation, the automatic recovery action is postponed until the catalog management action is completed or discarded. If immediate catalog management actions are defined, the dataset is cleaned up, and the job resubmitted by automatic recovery without delay. When actions are deferred, catalog management must be initiated manually for the operation from the MCP dialog.

See Chapter 18, “Catalog Management” on page 235 for details of the catalog management function.
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 Tivoli OPC dialogs. See “Specifying Workstation Destinations” on page 33. Operations that have not yet started on an inactive workstation can be rerouted to an alternate workstation. Refer to Controlling and Monitoring the Workload for more information about redirecting work to alternate workstations.

The status Tivoli OPC 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 Tivoli OPC RECOVER statement
  - //**OPC Identifies a Tivoli OPC RECOVER continuation statement.

The automatic recovery function also inserts informational statements in the JCL:

- //** OPC Identifies a Tivoli OPC message statement
- //**OPC Identifies a Tivoli OPC comment statement.

See “Tivoli OPC Message and Comment Statements” on page 263 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=PCCHK.
- 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 parameter or subparameter, 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.

Syntax

The syntax of the RECOVER statement follows. If you are unfamiliar with syntax diagrams, see "How to Read the Syntax Diagrams" on page xxviii.
Automatic Recovery

Statement Parameters

All RECOVER statement parameters are optional. There are three RECOVER statement categories: selection, JCL rebuild, and recovery action. Tivoli OPC supports parameters that refer to job steps only for jobs on MVS systems.

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 OPC supports this parameter only for jobs on MVS 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 OPC supports this parameter only for jobs on MVS 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.

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 OPC supports this parameter only for jobs on MVS 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 OPC supports this parameter only for jobs on MVS systems. If the operation specifies a catalog management action, Tivoli OPC will perform the catalog actions for all steps from the RESSTEP value up to and including the last step executed. If the failure occurred in a step before the RESSTEP, the catalog management action is performed for the entire job.

**Note:** RESSTEP on the RECOVER statement is a different, and simpler, function than the step-restart function in the MCP dialog.
MCP step restart retrieves the job log and rebuilds the JCL using that. RESSTEP does not require the job log, unless catalog management 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 whether the occurrence should be restarted.

**RESJOB** Specifies the name of the job or started task from which the occurrence should be rerun.

**ADDAPPL** Specifies an application or a list of applications that should be added as occurrences in the Tivoli OPC current plan.

**RELSUCC** Specifies the application ID of a successor occurrence, or a list of IDs.

**ALTWS** Specifies the name of an alternate workstation that the operation should be run on.

**ALTJOB** Specifies an alternate job or started-task name to use when the job is restarted. This is used in a MAS complex to allow the restart of a job that has not yet ended as far as JES is concerned.

### 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.proctstepname**.

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 OPC 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 264.) The values are those given as the error code on the HANDLING OPERATIONS ENDED IN ERROR panel in the Modify Current Plan dialog.
Automatic Recovery

The JOBCODE keyword values are:

- **Sxxx** Specifies a system abend code.
- **Uxxx** Specifies a user abend code.
- **xxxx** Specifies a case code or an error code set by Tivoli OPC, either directly or 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 OPC when the job or started task terminates. 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 225.

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.
- **Uxxx** Specifies a user abend code.
- **xxxx** Specifies a case code.
Automatic Recovery

\( n \) 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 225.

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. 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 OPC 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 may 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.

Examples:

- **TIME=0700-1600** No recovery actions will occur between 4 p.m. and 7 a.m.
- **TIME=2200-0800** Tivoli OPC will do automatic recovery only between 22.00 and 8.00.
- **TIME=0000-2400** Recovery can be automatic at any time.
- **TIME=0000-0000** Recovery will not be started unless there is manual intervention.

The recovery actions for a job or started task that remains on the ended-in-error list can be manually started up later. (See “Recovery Actions
Automatic Recovery

from the MCP Dialog" on page 262.) 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 OPC uses these parameters to build the JCL stream that will be used to restart the operation.

DELSSTEP

DELSSTEP 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:

   • IF-THEN statements may 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 will be invalid if all steps are deleted between an IF-THEN statement and a corresponding ELSE statement. The result will also be invalid if an IF-THEN statement remaining in the JCL references the name of a step to be 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 Tivoli OPC procedure library with ddname EQQPRLIB.

Note: This is not a JES procedure library but an internal Tivoli OPC procedure library.

The added JCL will be 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 keyword or as the first step that COND=ONLY is not added to. If there is already a RESTART value, it is replaced. If the restart step name is within a JCL procedure, specify stepname.procedurestepname.

* (asterisk) Specifies that the job or started task should be restarted at the first step (possibly a step of a cataloged procedure).

% (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 259 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
Automatic Recovery

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 OPC 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 will be performed, however.

Default: RESTART=Y.

**RESJOB**

The RESJOB parameter deals with 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 on the error handling list with an extended status code indicating an automatic recovery error.

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 (that is, predecessors to the failing operation or operations on any other path from the rerun operation than the failure operation) 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 will be independent of the failed occurrence.
- If RESTART=Y is specified, the recovery applications will be added to the Tivoli OPC 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 260.

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:

```bash
//**OPC RECOVER JOB CODE=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 264.) 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 executed, according to the clock on the MVS 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 will be started at the specified time.

2. An occurrence that has been added to the current plan by automatic job recovery will not become the predecessor to an occurrence that is added later by daily planning, even if normal dependency criteria are met. See page 195 for more information.

**RELSUCC**

This parameter specifies which external successors to the failing operation should be allowed to run even though this, 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
Automatic Recovery

statement causing the release of that day’s D, the occurrence of D the next
day will also wait 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 that the operation should be run
on. 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.

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.

Default: The job or started-task name is not altered.

Recovery Coding Examples

This section contains five examples of JCL for automatic job or started-task
recovery.

Example 1

```plaintext
//A103D01 JOB .....
//SALARY PROC GENER=+1
//S190 EXEC PGM=PAX
//PDS DD DSN=WATSON,DISP=SHR
//PRELPDS DD DSN=T191(&GENER),DISP=(NEW,CATLG,DELETE)
//CARDS DD DSN=T83/zerodot(&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
   .
   .
Procedure library member SALRECOV:
//**OPC RECOVER ERRSTEP=SAX92... (for steps after SAX91)
//SAX91 EXEC SALARY,GENER=0
```

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 264.) They will be replaced by the member SALRECOV from the
Tivoli OPC 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

```
//TI94237A 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=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 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 OPC 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 OPC 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 OPC 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 will get 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 MCP Dialog” on page 262.
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=&XX,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 OPC procedure library (which may be a copy of the RECOVER statements for step STEP02 and a space allocation step) should be included, and the failing job (Tivoli OPC 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 should be added, and the failing job (Tivoli OPC operation) should be restarted at step STEP02. The failing job will be a successor operation to AP301ARA and will be started when AP301ARA is completed.

The fourth RECOVER statement specifies that, for STEP02, 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 will remain in ended-in-error status.
Example 4

In this example, 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

This is the same as example 2, except that this time it is set up for testing the recovery procedure. If the job fails, it will have ended-in-error status independent of the error type. The JCL can be inspected and the recovery actions can be started manually via the MCP dialog.

Deciding Which Step of an Operation Has Failed

More than one step may 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 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.
Automatic Recovery

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 OPC assumes that the step that was executing at the time of failure was the step that would have followed the last step that Tivoli OPC 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 will remain 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 dataset 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 OPC must work out which operation in the successor occurrence depends on which operation in the predecessor occurrence. The operation Tivoli OPC 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 dialogs or the PIF.
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 will 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 OPC ends abnormally, the corresponding operation appears on the Tivoli OPC 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 Tivoli OPC dialog.

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 Modify Current Plan dialog. This means that the operation that is selected as the restart point will get status R, and succeeding operations will get 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.
Automatic Recovery

If an error is detected while processing a RECOVER statement, the extended status field will be 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 Tivoli OPC 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 OPC 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 by Tivoli OPC to be responsible for the JCL including its RECOVER statements. Tivoli OPC checks that this user has the level of authority required to perform the updates requested by the RECOVER statements.

Tivoli OPC 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 dataset, 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 OPC so that authorization checking can be performed.
- When the JCL is updated via a Tivoli OPC dialog, the ID of the last user who updated the JCL is stored by Tivoli OPC. This is then used for authorization checking.

Recovery Actions from the MCP Dialog

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 Modify Current Plan dialog. Selecting an operation with the row command ARC will start the scan.

If you invoke recovery from the MCP dialog, 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.

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
OPC 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 “Tivoli OPC 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 OPC to
take a fresh copy from the library.

---

### Tivoli OPC 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 OPC uses
message statements to insert an error description in the JCL file saved by Tivoli
OPC on the JS dataset. 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. Thus, the `RECOVER` statement will only be treated as such
the first time. You could, however, add the `RECOVER` statement again using the
`ADDPROC` parameter.

**Note:** An automatic recovery statement is commented by Tivoli OPC 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 OPC restart purposes
- On the Tivoli OPC message log
- In the Tivoli OPC JS dataset (in the JCL record of the failed job or started task)
in the form of Tivoli OPC message statements
Successful Recovery Start

The successful start of recovery actions is recorded in all three places, as described previously. Tivoli OPC 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:

```plaintext
//**OPC SCAN
//SAMPLEA JOB (885002,NOBO),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.THERE,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. Tivoli OPC message statements are added to the JCL close to the RECOVER statement.

If an error cannot be associated with a particular JCL record, Tivoli OPC 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 OPC 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 OPC tries to match the error code from the failing operation with a value specified in the JOBCODE and STEPCODE parameter. If no direct match is found and a case code list exists, Tivoli OPC assumes that the codes defined in the JOBCODE 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 OPC 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 OPC subsystem is started and stopped.

Automatic recovery can be deactivated and activated using the SERVICE FUNCTION option from the Tivoli OPC main menu. When automatic recovery is deactivated, the ended-in-error jobs or started tasks remain on the ended-in-error list, independent of RECOVER statements. Requests from the MCP dialog HANDLING OPERATIONS ENDED IN ERROR panel are, however, honored, regardless of whether automatic recovery is activated or deactivated. See “Recovery Actions from the MCP Dialog” on page 262.

### Restartability Considerations

When jobs or started tasks are restarted on workstations other than the workstation that they were initially defined on, you should consider:

- Dataset availability. All the datasets 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).
Automatic Recovery

- 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) datasets. 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:

<table>
<thead>
<tr>
<th>Example 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>//A105C01 JOB ....\n//STEP01 EXEC PGM=IEFBR14\n//A105CTLG DD DSN=A105.INVOICE.BASE(+1),\n// DISP=(NEW,CATLG,DELETE),\n// UNIT=DISK,\n// SPACE=(CYL,(5,1))\n///c5197 DCB PARMS ARE AVAILABLE IN THE MODEL DATA SET.</td>
</tr>
</tbody>
</table>

The following example refers to the generation data group with the former generation 0 as generation -1 and the new generation as generation 0:

<table>
<thead>
<tr>
<th>Example 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>//A105P01 JOB ....\n//STEP01 EXEC PGM=A105PGMP\n//A105PIN DD DSN=A105.INVOICE.BASE(-1), INPUT\n// DISP=OLD\n//A105POUT DD DSN=A105.INVOICE.BASE(0), OUTPUT\n// DISP=OLD</td>
</tr>
</tbody>
</table>

- Do not use backward references. Backward references to datasets 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 \textit{failure}, for example, COND=(0,NE).
- Always code a step name for every step (for example, STEP\textit{nn}, where \textit{nn} is the step sequence number).
Chapter 20. Event-Triggered Tracking

Tivoli OPC event-triggered tracking (ETT) gives you a method of controlling and tracking the unplannable workload in your production environment. ETT adds occurrences to the current plan based on a triggering event. You can use ETT to track work that has been submitted outside Tivoli OPC control, or simply to respond to an on-demand request for processing.

You will probably use the ETT function to implement many automation tasks in your production environment. Tivoli OPC can be used to monitor and highlight unexpected processing and respond accordingly, such as, when unsolicited files are received across a network.

When you have most of your batch processing submitted and tracked by Tivoli OPC, the work submitted outside Tivoli OPC can seem invisible. Using ETT, you can achieve maximum visibility for all processing and give your operators a single point of control.

This chapter describes these topics:

- Triggering event types
- How to activate ETT
- Defining your ETT criteria
- Tracking non-Tivoli OPC jobs
- The occurrence added by ETT
- Using ETT to automate tasks

Triggering Event Types

Tivoli OPC 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 OPC 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 OPC 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 dataset is closed if your installation implements the Tivoli OPC dataset triggering support. The resource may 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 in the Tivoli OPC dialogs, an R-event is not created. Tivoli OPC does not recognize this as an ETT-triggering action. For more information about creating and using resources, see Chapter 3, “Creating Special Resources” on page 41.

Activating ETT

To activate ETT automatically when Tivoli OPC is started, specify ETT(YES) on the JTOPTS statement (see Customization and Tuning).

To activate or deactivate ETT while Tivoli OPC is active, use the SERVICE FUNCTIONS dialog. See Chapter 12, “Using Tivoli OPC Service Functions” on page 205.

Defining Your ETT Criteria

Follow these steps to specify ETT criteria:

1. Specify option 1 (DATABASE) on the Tivoli OPC main menu.
2. Specify option 7 (ETT) on the MAINTAINING OPC/ESA DATA BASES menu.
3. Specify option 2 (MODIFY) on the MAINTAINING EVENT TRIGGERED TRACKING menu.

Figure 113. EQQJMTCL—Modifying ETT Tracking Criteria
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 OPC control with a job name beginning with DAP enters the system, Tivoli OPC adds an occurrence of SAMPLEA to the current plan.

**ID of associated application**
Specify the ID of the application to be triggered.

**ET** The type of event (J or R). See “Triggering Event Types” on page 267.

**JR** This specifies job-name replace tracking. Specify Y if Tivoli OPC will track the job that is being used as the trigger. If you specify N, Tivoli OPC 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 OPC 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 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 dialog 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 careful if you issue the SRSTAT command (or equivalent) with CREATE(NO) AVAIL(YES). No resource is created in the plan, and the status changes from no status to YES and immediately back to
no status. If the SRSTAT command is repeated, this causes another availability switch to YES, and Tivoli OPC will add another occurrence, even if the ETT criteria has AS set to Y. For the AS flag to be useful, therefore, you must specify CREATE(Y) so that the resource is saved in the current plan.

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 Modify Current Plan dialog.

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 OPC cannot allocate an input arrival time before the end of the current plan, the occurrence will not be added. When you use ETT, 877 occurrences of a single application description can be added to the plan. If you need to add more occurrences, replan the current plan regularly to remove completed occurrences.

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.

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. See page 195 for more information.

If the ETT criteria specifies job-name replace, the job name of the first operation is changed by Tivoli OPC 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 OPC ignores the TRACK keyword of the JTOPTS statement for job-name 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.
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 OPC 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 OPC 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 by Tivoli OPC 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 by Tivoli OPC 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 OPC high date. Such a date satisfies Tivoli OPC requirements for non-cyclic period definitions but ensures that applications that refer to the period definition are not accidentally included in the LTP.

If an ETT-added occurrence contains a run cycle that uses the period ETTRCY1, Tivoli OPC uses the input arrival time defined on that run cycle to establish successor dependencies instead of using the occurrence input arrival time, which by default is the triggering-event time.

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.
Chapter 21. Job Tailoring

This chapter describes the Tivoli OPC job tailoring functions, 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. Tivoli OPC job tailoring provides:

- Automatic variable substitution
- Dynamic inclusion or exclusion of inline job statements
- Dynamic inclusion of job statements from the EQQJBLIB dataset or from an exit

For jobs to be submitted on an MVS system, these job statements will be MVS JCL, but Tivoli OPC JCL tailoring directives can be included in jobs to be submitted on other supported operating systems such as AIX/6000. The Tivoli OPC directives have the same format for all supported operating systems.

An alternative method of substituting variables is to use the Tivoli OPC program interface (PIF). The PIF lets user-programs issue requests to the Tivoli OPC subsystem, thereby automating actions that you would otherwise perform with the Tivoli OPC dialogs. 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 OPC supports automatic substitution of variables during job setup and at job submit. Tivoli OPC also supplies several standard variables, which you can use in your job. You can create your own variables using the Tivoli OPC JCL Variable Tables dialog—these variables are stored in variable tables in the Tivoli OPC 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 OPC lets you create variables in job statements, in comment statements, and in any in-stream data within the job. You cannot use Tivoli OPC 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 dialog, you specify whether Tivoli OPC 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 JCL preparer
- By a user-written exit
- Automatically by Tivoli OPC (from information in the Tivoli OPC database)

A variable that must be supplied by the operator is called a promptable variable.
Job Tailoring

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 303). 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 b (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 Tivoli OPC 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 OPC searches for values to variables in this sequence:

1. In tables specified for the operation:
   a. In the variable table identified in the TABLE directive in the job. See “TABLE Directive” on page 302.
   b. In the variable tables identified in the SEARCH directive in the job. See “SEARCH Directive” on page 297.

2. In tables specified for the application:
   a. Specified using the Modify Current Plan (MCP) dialog.
   b. Specified using the Long-Term Plan dialog.
   c. In the variable table associated with the run cycle, if any, for the application occurrence. See “Creating Run Cycles with Rules” on page 80.
   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 62.

3. In the global variable table. See “Global Variable Table” on page 292.

**Note:** 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 “Including and Excluding JCL” on page 295), even if that same variable name is found later. So avoid using several variable tables for the same job, especially if they contain variables with the same name.

Figure 114 on page 275 shows how user-defined variables are matched to individual units of work. User-defined variables are specified in the JCL Variable Tables dialog (A) and are stored in the Tivoli OPC database.

In the Period dialog (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 dialog (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.

JCL variable table names for individual occurrences can be modified or added using the Long-Term Plan dialog (D) or the Modify Current Plan dialog (E).
A JCL variable table name (F) can be referenced by statements within the job by using table directives (see “Including and Excluding JCL” on page 295).

Figure 114. JCL Variable Handling
Invoking and Avoiding Variable Substitution

To invoke variable substitution, do one of these:

- 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.

- Set the VARSUB keyword of the OPCOPTS statement to SCAN (this is the default) and specify the Tivoli OPC directive ///c5197%OPC SCAN in your job. Substitution in the job starts where the SCAN directive is found. See “SCAN Directive” on page 296 for more information.

In the JCL Variable Tables dialog, 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 OPC 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 dialog—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.

Coding Variables in JCL

A Tivoli OPC variable, either user-defined or Tivoli OPC-supplied, 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 OPC 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>
</tbody>
</table>

See Chapter 1 of the Customization and Tuning manual for details.
To maintain compatibility with variable substitution within MVS JCL procedures, Tivoli OPC 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 5 on page 276. An ampersand or percent variable can be assigned a value that is itself a variable.

For example, consider the Tivoli OPC variable LIBRARY in Figure 115.

```
//STEPLIB DD DSN=MY.&LIBRARY.(IEFBR14),DISP=SHR
```

*Figure 115. JCL before Substitution*

If LIBRARY is given value LINKLIB, the JCL line becomes:

```
//STEPLIB DD DSN=MY.LINKLIB(IEFBR14),DISP=SHR
```

*Figure 116. JCL after Substitution*

In Figure 115, the correct substitution would have taken place, even if the completing period had been missing, as in Figure 117.

```
//STEPLIB DD DSN=MY.&LIBRARY(IEFBR14),DISP=SHR
```

*Figure 117. JCL before Substitution*

This is because Tivoli OPC assumes that the variable LIBRARY ends when it detects the left parenthesis ‘(’.

**Note:** The completing period is discarded when a variable is substituted. This is not the case with other termination symbols; they are left in place.

**Ampersand, Percent, and Question Mark Variables**

The three variable types—ampersand (&), percent sign (%), and question mark (?)—cause Tivoli OPC to perform variable substitution in different ways. You can use a variable multiple times with a different prefix symbol within the same job if you want.

**Ampersand Variables**

These variables are substituted from left to right within the line. Ampersand variables correspond to the standard variables in MVS 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 278. A compound variable is also formed if an &-variable immediately follows a ?-variable.

Any string that begins with && is not substituted by Tivoli OPC. This is because the double ampersand within JCL is usually used to denote a temporary dataset. Any such strings are unaffected by Tivoli OPC variable substitution.

If the JCL member contains symbolic parameters in the format &SYMBOLIC and you do not want Tivoli OPC to attempt resolution, define the variable in a Tivoli OPC 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 285 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 Tivoli OPC 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 OPC 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 line of JCL in Figure 118 from a Tivoli OPC-defined job:

```
//STEPLIB DD DSN=MY.&DATA%SET,DISP=OLD
```

*Figure 118. JCL Variable Example*

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 OPC 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 in Tivoli OPC. 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 OPC, a substitution exit, or manually, using the Tivoli OPC dialogs.

Compound variables can be made up of a sequence of many %-variables.
Consider Figure 119. Assume that \texttt{VAR3} has value SIX and \texttt{VAR2SIX} has value JUNE. On the first pass over this line of JCL, the variable \\
\texttt{VAR1\%VAR2\%VAR3...DATA} becomes \texttt{VAR1\%VAR2SIX...DATA}. On the second pass, \\
the variable \texttt{VAR1\%VAR2SIX.} becomes \texttt{VAR1JUNE...DATA}. The value assigned to \\
\texttt{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 dataset qualifier.

In Figure 120, you need only one parenthesis to complete the compound variable. This is because the parenthesis is not discarded at substitution.

If you specify a compound variable \texttt{\%VAR1\%VAR2...DATA}, where \texttt{VAR1} is a promptable variable and \texttt{VAR2} is a (nonpromptable) setup variable, the substitution fails, because Tivoli OPC 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 OPC 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:

\texttt{?VAR1.}

will cause the value of \texttt{VAR1} to be placed on the line that the variable appears on, starting at the column number defined in the Tivoli OPC database.

\texttt{?nnVAR1.}

will cause the value of \texttt{VAR1} to be placed on the line that the variable appears on, starting at the column number specified by \texttt{nn}. Any column value specified for this variable in the Tivoli OPC 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:

```
//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:

```
//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 287). 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 OPC 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:

```
//SYSIN DD *
.....1.....2.....3.....4.....5.....6.....7..
?20VAR1.?21VAR2.
/*
```

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 OPC 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 OPC has changed the space occupied by the variable to spaces. The other data in the line does not move.

**Note:** Tivoli OPC-supplied variables do not have an implied position. When Tivoli OPC-supplied 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.
Tivoli OPC-Supplied Variables

Tivoli OPC supplies variables that you can use in your business. These variables are reserved by Tivoli OPC, in the sense that Tivoli OPC will never try to read variable definitions for these variables from a variable table. The Tivoli OPC 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 Tivoli OPC variables, described in Table 6, are related to information about the occurrence. Occurrence-related variables can be used as setup or submit 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 DDMYY format</td>
</tr>
<tr>
<td>ODMY2</td>
<td>8</td>
<td>Occurrence input arrival date in DD/MM/YY format</td>
</tr>
<tr>
<td>OFREEDAY</td>
<td>1</td>
<td>Denotes whether the occurrence input arrival date is a free day (F) or a workday (W)</td>
</tr>
<tr>
<td>OHH</td>
<td>2</td>
<td>Occurrence input arrival hour in HH format</td>
</tr>
<tr>
<td>OHHMM</td>
<td>4</td>
<td>Occurrence input arrival hour and minute in HHMM format</td>
</tr>
<tr>
<td>OMM</td>
<td>2</td>
<td>Occurrence input arrival month in MM format</td>
</tr>
<tr>
<td>OMMYY</td>
<td>4</td>
<td>Occurrence input arrival month and year in MMYY format</td>
</tr>
<tr>
<td>OWW</td>
<td>2</td>
<td>Occurrence input arrival week of the year in WW format</td>
</tr>
<tr>
<td>OWWD</td>
<td>3</td>
<td>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</td>
</tr>
<tr>
<td>OWWLAST</td>
<td>1</td>
<td>A value, Y (yes) or N (no), that indicates whether the occurrence input arrival date is in the last week of the month</td>
</tr>
<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>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>OWWMONTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>Saturday 1st</td>
<td>1</td>
</tr>
<tr>
<td>Monday 3rd</td>
<td>2</td>
</tr>
<tr>
<td>Monday 31</td>
<td>6</td>
</tr>
</tbody>
</table>
Table 6 (Page 2 of 2). Occurrence-Related Tivoli OPC-Supplied Variables

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Length (in bytes)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>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 Tivoli OPC variables are related to information about the operation that is to be submitted. Operation-related variables can be used only as submit variables. Table 7 describes these variables.

Table 7. Operation-Related Tivoli OPC-Supplied 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>OLDAY</td>
<td>1</td>
<td>Operation latest start day (1–7); 1 represents Monday, 7 represents Sunday</td>
</tr>
<tr>
<td>OLDD</td>
<td>2</td>
<td>Operation latest start day (day in the month)</td>
</tr>
<tr>
<td>OLHH</td>
<td>2</td>
<td>Operation latest start hour</td>
</tr>
<tr>
<td>OLHHMM</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>OLMM</td>
<td>2</td>
<td>Operation latest start time in minutes in MM 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 YYMMD format</td>
</tr>
<tr>
<td>OLYYDDD</td>
<td>5</td>
<td>Operation latest start in Julian date format (YYDDD)</td>
</tr>
<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>
</tbody>
</table>

Date-related Supplied Variables
The third group of Tivoli OPC 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 8 on page 283 describes these variables.
Table 8. Date-Related Tivoli OPC-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>CHHMMSSX</td>
<td>8</td>
<td>Current hour, minute, second, and hundredths of seconds in HHMSSXX 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 YYDDD 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 Tivoli OPC-supplied variables. You define the format you require for these variables using the SETFORM directive described in “SETFORM Directive” on page 298. For example, if you want to substitute the occurrence input arrival date with the format MM:DD:YY, you define the dynamic variable OCDATE:

```
///%OPC SETFORM OCDATE=(MM:DD:YY)
```

Table 9 describes these variables.

Table 9 (Page 1 of 2). Dynamic-Format Date-Related Tivoli OPC-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>
</tbody>
</table>
Job Tailoring

You can use the variables in Table 6 on page 281, Table 7 on page 282, Table 8 on page 283, and Table 9 on page 283 in the same way as any other variables. Note that dynamic-format variables are treated as Tivoli OPC-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

Using an arithmetic expression on Tivoli OPC-supplied date-related variables, you can create temporary variables.

For example, you may 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

//+%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

<table>
<thead>
<tr>
<th>Variable name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<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>OCFRSTW</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>
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 300.

User-Defined Variables and Variable Tables

Besides the Tivoli OPC-supplied variables, you can define variables in a variable table by using the JCL Variable dialog.

1. Select option 9 (JCLVAR) from the MAINTAINING OPC/ESA DATABASES panel, or enter the fast path 1.9 from the OPC main menu. You see the MAINTAINING OPC/ESA JCL VARIABLE TABLES panel, shown in Figure 121.

2. Select option 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 OPC system. See Figure 122.

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 123 on page 286 for an example of variable creation.
Figure 123. EQQJVVC — Creating a JCL Variable Table

The SETUP option specifies when the variable is substituted. See “When Variables Are Substituted” on page 287 for more information.

You can specify that an exit, rather than Tivoli OPC, should substitute the variable. For more details about this exit, refer to Customization and Tuning.

When you select a variable from panel EQQJVVC, you can make detailed modifications to it using this panel:

Figure 124. EQQJVVM — 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.
When you have selected and defined a variable, you can specify:

- Dependencies on the variable (see “Making One Variable Dependent on Another”)
- Validation criteria for the variable (see “Variable Validation” on page 290) by entering the DEP or VER primary command.

**When Variables Are Substituted**

Figure 125 shows how to specify the SETUP option on the panel in Figure 124 on page 286.

<table>
<thead>
<tr>
<th>SETUP</th>
<th>Tivoli OPC substitutes the variable...</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) dataset, 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 job preparer. Tivoli OPC first substitutes SETUP=Y variables. Then it scans the job for promptable variables and displays them. The job preparer enters values for each variable. If there is no setup phase, the job fails with error code OJCV.</td>
</tr>
</tbody>
</table>

*Figure 125. Specifying the SETUP Option*

The job preparer initiates setup by setting the operation to next logical status using the workstation Ready List dialog. Variables can also be substituted by a user-supplied exit rather than by Tivoli OPC.

**Making One Variable Dependent on Another**

You can have variables that depend on other variables. For example, the message class and print destination may both depend on the MVS 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):

1. Enter the DEP command from the MODIFYING A JCL VARIABLE panel. You see the JCL VARIABLE DEPENDENCY VALUE LIST panel, shown in Figure 126 on page 288.
Job Tailoring

Figure 126. EQQJVDVL—JCL Variable Dependency Value List

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 126 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.

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:

```bash
//%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
```

```bash
//%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
```
Restrictions
You cannot:

- 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 \texttt{VALUE REQUIRED=Y} is specified for that 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 an OPC-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” 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 dataset qualifiers that change (for example, from CICS32\_LOADLIB to CICS33\_LOADLIB. Making the dataset qualifier a variable lets you have just one set of jobs, which is important if you make other changes unconnected to the change 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 dialog). But you can use dependent variables even in a fully automatic system, if the independent variable is an OPC-supplied variable.

This example shows how to change variables automatically, depending on the input arrival day of the occurrence. The supplied variable is \texttt{ODAY}, which has values from 1 (Monday) to 7.
This is how the dependent variable (HLQ1) is defined:

<table>
<thead>
<tr>
<th>Command</th>
<th>MODIFYING A JCL VARIABLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Variable table</td>
<td>SPECIALONMONDAY</td>
</tr>
<tr>
<td>Variable name</td>
<td>HLQ1</td>
</tr>
<tr>
<td>DESCRIPTION</td>
<td>high-level-qualifier</td>
</tr>
<tr>
<td>DEFAULT VALUE</td>
<td>OTHER.DAYS</td>
</tr>
<tr>
<td>UPPER CASE</td>
<td>Y</td>
</tr>
<tr>
<td>SETUP</td>
<td>Y</td>
</tr>
<tr>
<td>SUBSTITUTION EXIT</td>
<td>A MVS load module name</td>
</tr>
<tr>
<td>VALUE REQUIRED</td>
<td>Y</td>
</tr>
</tbody>
</table>

Figure 127. Specifying the Variable HLQ1

The default value, OTHER.DAYS, is used when there is no match on the independent variable values.

<table>
<thead>
<tr>
<th>Command</th>
<th>JCL VARIABLE DEPENDENCY VALUE LIST</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent variable</td>
<td>HLQ1</td>
</tr>
<tr>
<td>Variable table</td>
<td>SPECIALONMONDAY</td>
</tr>
<tr>
<td>Default value</td>
<td>OTHER.DAYS</td>
</tr>
<tr>
<td>INDEPENDENT VARIABLE</td>
<td>ODAY</td>
</tr>
<tr>
<td>VALUE OF INDEPENDENT</td>
<td>1</td>
</tr>
<tr>
<td>VALUE OF DEPENDENT</td>
<td>DAY1</td>
</tr>
</tbody>
</table>

Figure 128. Specifying a Special Value for Monday (ODAY=1)

Tivoli OPC 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 OPC 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 129 on page 291). This panel is displayed when you enter the VER command from the MODIFYING A JCL VARIABLE panel (see Figure 124 on page 286).
EQQJVVEP --------------- SPECIFYING VERIFICATION CRITERIA -----------------

Command ==> 

Enter data below, or enter the DEL command to delete the verification.

Variable name : FRED varfred
Variable table : SIMON test jcl variables

VERIFICATION TYPE ===> RANGE__ ALPHA/NUM/ENUM/HEX/BIT/PICT/NAME/DSNAME/RANGE/LIST

LENGTH COMPARISON ===> __ EQ/LT/GT/LE/GE/NE/NL/NG
LENGTH VALUE ===> __ 1 to 44

For type PICT only - use characters C A N 9 X

PICT VALIDATION ===> ____________________________________________

LIST/RANGE NUMERIC ===> Y Y or N (if Y, specify the values below)
LIST/RANGE VALUES ===> 23,26__________________________________________

Figure 129. EQQJVVEP—Specifying Verification Criteria

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 dataset member name.

**DSNAME** The input must be a valid dataset 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
~ or NE   Not equal to
Job Tailoring

¬< or NL Not less than
¬> or NG 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 6-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), Tivoli OPC 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 Y0RK).
2. Tivoli OPC 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 OPC 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 OPC 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 OPC searches for variables only if the //**:\c5197%OPC SCAN directive has been encountered in the job.

Tivoli OPC 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 MVS JCL Procedures” on page 294.

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.
- 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:

```
Example of note 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=> //**:\c5197> 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. See Chapter 1 of the Customization and Tuning manual for details.

Note: Tivoli OPC scans the job only up to the first detected error.
Job Tailoring

Suppressing Variable Substitution

You can specify that variable substitution should be suppressed in jobs that would normally be eligible for variable substitution, using the Tivoli OPC statement

```
//+%OPC BEGIN ACTION=NOSCAN
```

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 job in Figure 130, Tivoli OPC 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
```

Figure 130. Suppressing JCL Variable Substitution, Using ACTION=NOSCAN

A `//+%OPC BEGIN ACTION=NOSCAN` statement must have an associated `//+%OPC END ACTION=NOSCAN` statement, or an error will result. See “Including and Excluding JCL” on page 295 for a complete explanation of Tivoli OPC directives.

Variable Substitution in MVS JCL Procedures

Tivoli OPC variable substitution does not take place within JCL procedures. If you have MVS JCL variables within such procedures, their substitution is independent of any substitution made by Tivoli OPC. Even if the MVS JCL variable name is the same as a Tivoli OPC 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 MVS, you can define the Tivoli OPC-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 Tivoli OPC-tailoring. If a variable is given a default value of `~` (logical not), the variable is deleted from the line.

This means that MVS substitutes the variables that have a `b`, or not-required, value. Tivoli OPC substitutes the variables that have nonblank values, as well as variables that require values.

Because procedures are called after Tivoli OPC substitution has taken place, a situation in which a `procedure` variable is assigned an `Tivoli OPC` variable value can occur. Consider the JCL in Figure 131 and the procedure in Figure 132 on page 295.

```
//EXEC PROC=MYPROC
//DD2 DD DSN=MY..DS2,DISP=SHR
```

Figure 131. Substituting a Variable in a Procedure: Job JCL

If DS2 is defined as a Tivoli OPC variable, it is substituted by Tivoli OPC in the usual way.
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 OPC substituted value of DS2:

```plaintext
//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 MVS 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 OPC value, you can give the variable a default value of blank and set the VALUE REQUIRED attribute to N in the Tivoli OPC database using the JCL Variable Definition dialog. This causes Tivoli OPC 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 OPC 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 OPC can tell if there should be trailing blanks. Use the LENGTH VALUE field on the SPECIFYING VERIFICATION CRITERIA panel (Figure 129 on page 291). If you do not specify a length, trailing blanks are not included in the substitution.

### Including and Excluding JCL

Tivoli OPC can dynamically include or exclude job statements during job setup and job submit. Tivoli OPC 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 OPC uses special comment statements, called directives, to manage the inclusion and exclusion of lines and to control aspects of variable substitution. The directives are:

- SCAN
- SEARCH
- SETFORM
- SETVAR
- TABLE
- BEGIN and END
FETC

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.
- Directive parameters can be coded in any order.
- Directive parameters can occur more than once in the same directive.
- Directive parameters are separated by commas with no embedded blanks between parameters on the same line.
- If more than one parameter value is specified, parentheses are required. For example, this is correct:

  NAME=TABLE1

  But this is incorrect:

  NAME=TABLE1, TABLE2

  It should be defined:

  NAME=(TABLE1, TABLE2)

- A directive specification cannot exceed 71 characters. It can be continued on a new line if the directive is split by a comma after a complete parameter or subparameter.
- Positions 72 to 80 are ignored.
- Each continuation line must begin with /*%OPC in columns 1 to 7 followed by at least one space.
- 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 OPC job substitution routines treat any other directives that it finds as “unknown,” and will take no action.

A description of the Tivoli OPC directives follows. If you are unfamiliar with syntax diagrams, see “How to Read the Syntax Diagrams” on page xxviii.

**SCAN Directive**

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 OPC that variable substitution should start from this line.

**Syntax**

```
SCAN directive

///*%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

```plaintext
//OPSTATUS JOB (ACCOUNT),'Set completed',CLASS=A
//STEP1 EXEC PGM=&MODULE.
//*%OPC SCAN
//STEPLIB DD DSN=OPC.LOAD.&LIBRARY.,DISP=SHR
//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY,DISP=SHR
//EQQMLOG DD SYSOUT=A
//SYSIN DD /c5197
/*
```

SEARCH Directive

This directive defines the variable tables that are searched by Tivoli OPC when attempting to assign a variable a value.

Syntax

```
SEARCH directive
```

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 OPC 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 OPC variables. The following example shows how you can change the search order of the tables using the SEARCH directive.
Job Tailoring

Example 1

```c5197%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

```c5197%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)

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

This directive defines the format of dynamic-format Tivoli OPC-supplied variables. After Tivoli OPC 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

```
SETFORM directive
```/SM590000/SM590000──///c5197%OPC SETFORM─ ──

dynamic-variable-name= (format expression)
```/SM590000/SM590000

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 Tivoli OPC-supplied dynamic variables (see Table 9 on page 283).

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 DDDDDD 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 OPC 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 left unsubstituted. 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 left unsubstituted. MM is substituted by the month part of the date.

You can use more than one delimiter between keywords.

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 OCDATA=(YY/MM/DD)
```

The resulting &OCDATA variable would be: 97/12/365

The examples in Table 10 on page 300 use the same occurrence input arrival date.
### Job Tailoring

#### Table 10. Dynamic-Format Substitution Results

<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)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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 unsubstituted.</td>
</tr>
</tbody>
</table>

### SETVAR Directive

This directive creates a temporary variable using an arithmetic expression together with Tivoli OPC-supplied date variables. After Tivoli OPC 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
SETVAR directive
/SM590000/SM590000──///c5197%OPC SETVAR──────────────────────────────────────────────────/SM590000/SM630000

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Tname=</strong></td>
<td>The name of the temporary variable, beginning with the letter T.</td>
</tr>
<tr>
<td>date variable</td>
<td>One of these Tivoli OPC-defined formats:</td>
</tr>
<tr>
<td>Date formats</td>
<td>See Table 11 on page 302.</td>
</tr>
<tr>
<td>Day-in-year formats</td>
<td>See Table 12 on page 302.</td>
</tr>
<tr>
<td>Day-in-month formats</td>
<td>ODD, OLDD, CDD</td>
</tr>
<tr>
<td>Day-in-week formats</td>
<td>OWWD, CWWD</td>
</tr>
<tr>
<td>Day-of-week formats</td>
<td>ODAY, OLDAY, CDAY</td>
</tr>
<tr>
<td>Week formats</td>
<td>OWW, OLWK, CWW</td>
</tr>
<tr>
<td>Month formats</td>
<td>See Table 13 on page 302.</td>
</tr>
<tr>
<td>Year formats</td>
<td>OYY, OYYYY, CYY, CYYYY</td>
</tr>
<tr>
<td>nnn</td>
<td>A number in the range 0 to 999.</td>
</tr>
</tbody>
</table>
TT This is the type and it can have the following values:

- WD—work days, as defined for the calendar used by the occurrence.
- CD—calendar days.
- WK—weeks. Weeks are converted to days before the arithmetic is performed.
- MO—months. Performing arithmetic on the month portion affects only the month, and possibly the year. Beware of generating invalid dates—adding one month to 970130 gives 970230.
- YR—years.

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 variables (those beginning with the letter C) in the expression for the temporary variable.

Usage Notes
These examples show how this variable is used:

---

**Example 1**

```
//**OPC SCAN
//**OPC SETVAR TVAR=(ODDD+4CD)
```

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=368

---

**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 your expression includes WD, the calculated date must fall within the range of three years earlier and six years later than the current year. If the current year is 1997, the resulting date of the temporary variable must be later than 93/12/31 and earlier than 04/01/01.

Table 11. 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>OYMD</td>
<td>OYMD1</td>
<td>OYMD2</td>
<td>OYMD3</td>
</tr>
<tr>
<td>OLYMD</td>
<td>CDDMMYY</td>
<td>CYMD</td>
<td>CYYMMD</td>
<td>OCDATE</td>
<td>OCFRSTW</td>
</tr>
<tr>
<td>OCFRSTWY</td>
<td>OCFRSTC</td>
<td>OCLASTW</td>
<td>OCLASTW</td>
<td>OCLASTC</td>
<td>CDATE</td>
</tr>
<tr>
<td>OPIADATE</td>
<td>OPLSDATE</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 12. 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 13. Month 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>OMM</td>
<td>OMMYY</td>
<td>OYM</td>
<td>OYYMM</td>
</tr>
<tr>
<td>CYYYM</td>
<td>CYYYYMM</td>
<td>CMM</td>
<td>CMMYY</td>
</tr>
</tbody>
</table>

**TABLE Directive**

This directive defines a variable table that will be searched by Tivoli OPC before the variable tables in any existing concatenation when resolving JCL variables.

**Syntax**

```
TABLE directive
// */% 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 Tivoli OPC JCL Variable Tables dialog (see “User-Defined Variables and Variable Tables” on page 285). The `table name` specified can be the name of a table or an & or % variable. This variable follows the standard Tivoli OPC format and can be promptable or nonpromptable.

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 DATASET2 are defined in both TABLE1 and TABLE2.

Example

```plaintext
//**%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 DATASET2, however, because it was not found before the TABLE directive, is assigned a value from TABLE1.

### BEGIN and END Directives

These directives, used in pairs, denote the following, depending on the value of the ACTION keyword:

- The start and end of Tivoli OPC 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 directive

```plaintext
//**%OPC BEGIN
```

#### END directive

```plaintext
//**%OPC END ACTION=
```
Job Tailoring

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 307.

Usage Notes

In a job, every BEGIN directive must have a matching END directive specifying the same ACTION. For example, the directive:

```
///%OPC BEGIN ACTION=EXCLUDE,PHASE=SUBMIT
```

requires the following matching END directive:

```
///%OPC END ACTION=EXCLUDE
```

If Tivoli OPC 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
///c5197%OPC SCAN
///c5197%OPC BEGIN PHASE=SETUP,ACTION=INCLUDE
///DDNAME1 DD DSN=&HIONE..&DATASET1.,DISP=SHR
///DDNAME2 DD DSN=&HIONE..&DATASET2.,DISP=SHR
///c5197%OPC END ACTION=INCLUDE
```

Example 1 is valid. Lines 2 and 3 will be included in the job for the operation.

**Example 2**

```plaintext
///c5197%OPC SCAN
///c5197%OPC BEGIN PHASE=SUBMIT,ACTION=EXCLUDE
///EXEC PGM=MYPROG
///c5197%OPC BEGIN PHASE=SETUP ACTION=INCLUDE
///DDNAME1 DD DSN=&HIONE..&DATASET1.,DISP=SHR
///SYSOUT DD SYSOUT=A
///c5197%OPC END ACTION=EXCLUDE
///DDNAME2 DD DSN=&HIONE..&DATASET2.,DISP=SHR
///c5197%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.

**Example 3**

```plaintext
///c5197%OPC SCAN
///c5197%OPC BEGIN PHASE=SUBMIT,ACTION=EXCLUDE
///EXEC PGM=MYPROG
///c5197%OPC BEGIN ACTION=NOSCAN
///DDNAME1 DD DSN=&HIONE..&DATASET1.,DISP=SHR
///SYSOUT DD SYSOUT=A
///c5197%OPC END ACTION=NOSCAN
///DDNAME2 DD DSN=&HIONE..&DATASET2.,DISP=SHR
///c5197%OPC END ACTION=EXCLUDE
```

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 8 will be substituted, but the variable HIONE on line 5 will not be substituted because it is within a NOSCAN domain.
Job Tailoring

FETCH Directive

This directive lets you include lines, fetched from a partitioned dataset member or supplied by an exit, in your job.

Syntax

```
FETCH directive

MEMBER=(member name)
EXIT=(exit name)
PHASE=SETUP|SUBMIT
COMP=((comparison expression), (comparison expression), ...)
```

Parameters

**MEMBER=(member name)**

Specifies the member name of a partitioned dataset 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 307.

Usage Notes

The FETCH directive is used to include lines from other partitioned datasets 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 OPC 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)
```
The COMP Keyword on BEGIN and FETCH Directives

A comparison expression lets you specify conditions when BEGIN and FETCH directives will be honored.

Syntax

```
COMP keyword

COMP=(expression1 .EQ. expression2)
```

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.
Job Tailoring

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.

**Note:** The % symbol does not signify a % Tivoli OPC 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 OPC. Neither expression1 nor expression2 can have embedded blanks.

Consider the following examples:

---

**Example 1**

```plaintext
///%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**

```plaintext
///%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**

```plaintext
///%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 expression 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**
```
///%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 Tivoli OPC-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

Tivoli OPC imposes some restrictions on the values that variables can take and positions in the JCL where you can use variable substitution.

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 may occur if the newly created line is longer than the maximum length allowed. For example, if you define a variable &DATASET, which has a length of 8 characters including the &, but give it a value 'MY.DATASET.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 OPC 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 OPC will, however, truncate comments on lines if necessary without giving an error.

Tivoli OPC 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 OPC 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 OPCSMSG in positions 73-79, because Tivoli OPC uses this to position displayed lines at error messages.

Strings You Cannot Use Variables to Represent

Tivoli OPC variables can represent the values of most keywords, but you cannot use variables to represent certain keywords in both standard MVS JCL and in Tivoli OPC directives.

- In the case of MVS 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 OPC directives, the restricted keywords are:

- The string 

- The following command names:
  
- The keyword to the left of each equals sign:
  
- Certain values to the right of the equals sign:

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 OPC 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 OPC 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 (nonpromptable) setup variable, the substitution fails, because Tivoli OPC 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.

Using a Default Calendar Name

Tivoli OPC 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 dialog
3. The calendar called DEFAULT
This chapter details the possible integration and exploitation of the Workload Manager (WLM) component of OS/390. Today, installations process different types of work with different completion and resource requirements. Every installation wants to make the best use of its resources, to maintain the highest possible throughput, and to achieve the best possible system response. Workload management makes this possible.

With workload management, 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 how much resource, for example, CPU and storage, should be given to it to meet the goal. Workload Manager constantly monitors the system and adapts processing to meet the goals.

Tivoli OPC is a job-scheduling product that inputs large quantities of work into OS/390 systems. The work is input when a number of predetermined conditions have been met, but Tivoli OPC does not know the state of the system (in terms of load and capacity) when the work is submitted. One of Tivoli OPC’s main objectives is to run your workload in the most efficient way possible, so that the work completes within the time required by yourself. There are a number of factors however that can occur and prevent Tivoli OPC-controlled work from meeting its deadline time. To avoid any delays, Tivoli OPC exploits the services of WLM, which controls the amount of systems resource available to each work unit, as follows:

- Tivoli OPC detects that a critical job is running late.
- Tivoli OPC calls a documented WLM interface to move the job to a higher-performance service class.
- The job receives more system resources and completes in a quicker time.
- The delay in the batch workload is therefore reduced.

### Environment

To benefit from the WLM function, your system must be 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 OPC 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
Selecting WLM Assist Policies

When you have decided that a job should be considered as critical, Tivoli OPC allows you to choose between different assist 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 dialog or the Modify Current Plan dialog). If you do not specify a policy for a particular critical job, Tivoli OPC applies either the common policy specified in the WLM statement of OPCOPTS or the default common policy (if one has not been specified). Valid assist 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 OPC. This policy is activated by specifying the letter E in the Assist Policy field on the Application Description panel.

Note: The resources are used also when not necessary, but for a short interval.

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 OPC.

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 OPC.

Note: The resources are used also when not strictly necessary, but it can prevent delayed termination of jobs.

C  Conditional. The assist policy is a smart compromise between the Deadline and the Latest Start Time policies as follows:

1. Tivoli OPC 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 OPC 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 assist policies are:
<table>
<thead>
<tr>
<th>Assist 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 |
Appendix A. TSO Commands

This appendix describes the Tivoli OPC TSO commands, BACKUP, JSUACT, OPINFO, OPSTAT, SRSTAT and WSSTAT. It also shows how you can use them in batch mode, using the event-generating batch program, EQQEVPGM. See “EQQEVPGM—Issue Tivoli OPC Commands in Batch” on page 372 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 Tivoli OPC 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 Installation Guide for more information.

Note: When EQQEVPGM is used to issue Tivoli OPC 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 OPC subroutine instead. When you issue the commands from TSO or as input to the EQQEVPGM program, a TSO environment must be established each time and some of the resources remain allocated until the task ends which may lead to a storage shortage if the commands are issued many times. The subroutines are documented in Customization and Tuning.

Important Considerations for Using Tivoli OPC TSO Commands

Consider these points when using 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 OPC address space. For example, the workstation name that you specify for OPSTAT is not verified against the actual workstations that exist in a particular Tivoli OPC current plan. In addition, a single event record can be generated to be used in two or more Tivoli OPC 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.

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 OPC (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 dataset.
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 CP or JS backups by defining the job as an operation in the current plan.

The JS and CP files are managed by the Tivoli OPC 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 Tivoli OPC tracker. The BACKUP command can be issued from any MVS system that runs a tracker from Tivoli OPC 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 Tivoli OPC 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 CP resource. You might do this to:

- Request a backup at a predefined time for disaster recovery purposes.
- Ensure regular backups are taken when Tivoli OPC 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 is incremented every time a new job-tracking record is written. The counter is reset to zero after every CP backup.

Backup of the JS file is performed automatically by Tivoli OPC 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 CP resource is locked to prevent dialog users and other Tivoli OPC 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 dataset to the TSO user's address space, 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 Tivoli OPC’s 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 MVS 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 Tivoli OPC 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 an MVS IPL, no authority verification can be performed.

### Syntax

```
BACKUP command

BACKUP RESDS (CP | JS) TRACE(trace level)

SUBSYS(MSTR | subsystem name | OPCA)
```

### Parameters

**RESDS (CP | JS)**

The RESDS keyword specifies which dataset 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 dataset will be copied to the alternate JS file.

The RESDS keyword must be specified.

**SUBSYS (MSTR | subsystem name | OPCA)**

The name of the Tivoli OPC tracker subsystem that the BACKUP 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 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 Tivoli OPC tracker subsystems on the MVS system that the BACKUP command was directed to.
Attention: This will cause multiple backups to occur if more than one Tivoli OPC tracker subsystem is active on the system.

Subsystem name is the name of the Tivoli OPC subsystem that the BACKUP command will be directed to. If the Tivoli OPC tracker and controller run in separate address spaces in your Tivoli OPC 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>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example 1 - TSO command</strong></td>
</tr>
<tr>
<td>ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE</td>
</tr>
<tr>
<td>BACKUP RESDS(CP) SUBSYS(OPCA)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Example 2 - Batch job</strong></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>//EQQMLIB DD DSN=OPC.MESSAGE.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>
<tr>
<td>/*</td>
</tr>
</tbody>
</table>

In both of these examples, the current plan dataset will be copied to the old current plan dataset, on a subsystem called OPCA.

JSUACT
The JSUACT command is used to activate or inactivate the job submission function.

Usage Notes
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 space, 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 OPC Controller log to indicate whether or not job submission has been activated.

Use of the JSUACT command can be restricted using OPC's 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 MVS 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 OPC 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 an MVS IPL, no authorization verification can be performed.

**Syntax**

```
JSUACT command

+-----------------------------+
| JSUACT ACT (yes | no)       |
| SUBSYS MSTR | subsystem name | OPCA |
| TRACE (trace level | 0)          |
+-----------------------------+
```

**Parameters**

**ACT (Y | N)**

If you want to activate the job submission function specify Y, otherwise N.

**SUBSYS (MSTR | subsystem name | OPCA)**

The name of the OPC controller 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 OPC subsystems on the MVS 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.
Examples

These two examples demonstrate how you can use the JSUACT command in TSO, or in a batch job (using the batch program EQQEVPGM).

<table>
<thead>
<tr>
<th>JSUACT</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example 1 - TSO command</strong></td>
</tr>
<tr>
<td>ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE</td>
</tr>
<tr>
<td>JSUACT ACT(YES) SUBSYS(OPCB)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th><strong>Example 2 - Batch job</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>//JSUACT JOB (ACCOUNT),'Deactivate',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>//EQQMLIB DD DSN=OPC_MESSAGE.LIBRARY,DISP=SHR</td>
</tr>
<tr>
<td>//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY,DISP=SHR</td>
</tr>
<tr>
<td>//EQQMLOG DD SYSOUT=A</td>
</tr>
<tr>
<td>//SYSIN DD /c5197</td>
</tr>
<tr>
<td>JSUACT ACT(N) SUBSYS(OPCB)</td>
</tr>
<tr>
<td>/*</td>
</tr>
</tbody>
</table>

OPINFO

Use the OPINFO command to set the user field of an operation in the current plan. The user field 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 OPC.

Usage Notes

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 dataset to the TSO user's address space, 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 OPC 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 OPC 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 field 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 OPC finds more than one operation that matches your parameters, Tivoli OPC chooses the best operation. If OPINFOSCOPE is IP, Tivoli OPC 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 OPC 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 dialog. If the latest start time is equal, the operation with the highest priority is updated, and so on.

If OPINFOSCOPE is ALL, Tivoli OPC 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 OPC also searches operations with status C and W in the CP. 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 MVS 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 an MVS IPL, no authority verification can be performed.

Syntax
Parameters

**ADID (application description identifier)**

The application identifier of the operation whose user field 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 field is to be updated.

**FORM (printer form name)**

For a printer workstation, specifies the printer FORM name of the operation whose user field is to be updated.

**IA (yymmddhhmm)**

The input arrival date and time of the occurrence containing the operation whose user field is to be updated. You must specify this in the format yymmddhhmm.

**Note:** Tivoli OPC interprets the yy part 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>

**JOBNAME (job name)**

The job name associated with the operation whose user field 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 field is to be updated.
SUBSYS (MSTR | subsystem name | OPCA)
Name of the Tivoli OPC 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 Tivoli OPC tracker subsystems on the MVS system where the OPINFO command is issued.

Note: If the Tivoli OPC trackers and controller run on different subsystems in your Tivoli OPC 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 field. The USERDATA field cannot contain any blanks.

WSNAME (workstation name)
The name of the workstation for the operation whose user field 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).

<table>
<thead>
<tr>
<th>OPINFO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example 1 - TSO command</td>
</tr>
<tr>
<td>ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE</td>
</tr>
<tr>
<td>OPINFO W(BDEC) J(DNCD3000) A(ACLMSDLY) U(USERDATA=EHERE)</td>
</tr>
<tr>
<td>Example 2 - Batch job</td>
</tr>
<tr>
<td>//OPINFOUS JOB (ACCOUNT),'Set completed',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>//EQQMLIB DD DSN=OPC.MESSAGE.LIBRARY,DISP=SHR</td>
</tr>
<tr>
<td>//EQQMLOG DD SYSOUT=A</td>
</tr>
<tr>
<td>//SYSIN DD *</td>
</tr>
<tr>
<td>OPINFO W(BDEC) J(DNCD3000) A(ACLMSDLY) U(PROBREC01234567)</td>
</tr>
<tr>
<td>/*</td>
</tr>
</tbody>
</table>

In both of these examples, an operation for application ACLMSDLY at workstation BDEC will have the data fed back to it.
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) or complete (C) status are ignored. Jobs and started tasks that are running are always allowed to finish—refer to the Ready List dialog description in Controlling and Monitoring the Workload for rules governing the changing of operation status.

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 Tivoli OPC's automatic event reporting (AER) facility. AER can help you coordinate many tasks that are not normally seen by Tivoli OPC. 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 Tivoli OPC tracker.

Usage Notes

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 dataset to the TSO user's address space, 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 EQQUUSIN subroutine for implementing automatic event reporting for general workstations.

With the OPSTAT command, you must specify the WNAME parameter to identify the workstation at which the operation is changing status. Tivoli OPC 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 OPC finds more than one operation which matches the criteria you specified, Tivoli OPC determines the most appropriate operation to update. Tivoli OPC 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 WSNAME parameter and Tivoli OPC 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 dialog, 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 Tivoli OPC’s fixed resource code RL and the subresource RL.WSNAME. 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 MVS 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 an MVS IPL, no authority verification can be performed.

Syntax

```plaintext
OPSTAT command

- ADID(appl description ID)
- CLASS(printer sysout class)
- DURATION.hhmm
- ERRORCODE(operation error code)
- EVDATE.yymmdd
- EVTME.hhmm
- FORM(printer form name)
- IA.yymmddhhmm
- JOBNAME.job name
- NUMJOB.job number
- OPNUM.operation number
- TOKEN.token value
- STATUS.CEITXS
- SUBSYS.OPCA
- SUBSYS.MSTR
- SUBSYS.subsystem name
- WSNAME.workstation name
- TRACE.trace level
```
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** (*ymmd*)
The date of this operation status event. You must specify the date in the format *ymmd*.

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 to Tivoli OPC 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 OPC 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** (*ymmdhhmm*)
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 *ymmdhhmm*.

**Note:** Tivoli OPC interprets the yy part 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>

**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 99999. Tivoli OPC builds a job number in the format USRnnnnn, padding the number with zeros on the left if you specify fewer than 5 digits.

**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 Tivoli OPC controller message log.

Specify the token as a hexadecimal value in the format TOKEN(X‘00ABCDEF’).

STATUS *(E | I | Q | T | S | X | C)*

The new operation status:

- **S** The operation has started at the workstation
- **Q** The started operation will have extended status Q set
- **T** The started operation will have extended status S set
- **C** The operation has completed successfully at the workstation
- **E** The operation has ended in error at the workstation
- **I** The operation has been interrupted at the workstation
- **X** Reset the current status of this operation to the previous logical status.

**Note:** You cannot change an operation with status W to status C with the OPSTAT command. This is because predecessor jobs may not be completed. Changes to operation status using OPSTAT follow the same rules as status changes in the Ready List. For more information, refer to *Controlling and Monitoring the Workload*.

SUBSYS *(MSTR | subsystem name | OPCA)*

Name of the Tivoli OPC 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 Tivoli OPC tracker subsystems on the MVS system where the OPSTAT command is issued.

**Note:** If the Tivoli OPC trackers and controller run on different subsystems in your Tivoli OPC 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).

### OPSTAT

**Example 1 - TSO command**

```
ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE

OPSTAT W(BDEC) ST(C) J(DNCD3000) A(ACLMSDLY)
```

**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 /c5197

OPSTAT W(BDEC) ST(C) J(DNCD3000) A(ACLMSDLY)
```

In both of these examples, an operation for application ACLMSDLY at workstation BDEC is reported as completed.

### SRSTAT

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 Notes**

You can invoke SRSTAT as a TSO command or by using a batch job which executes program EQQEVPGM. If you invoke SRSTAT as a TSO command, you must allocate the EQQMLIB dataset to the TSO user's address space, 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 Tivoli OPC's 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 MVS 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 an MVS IPL, no authority verification can be performed.

See Chapter 3, “Creating Special Resources” on page 41 for information about special resources, how to connect them to workstations, and how to specify intervals.

Syntax

```
SRSTAT command

>>SRSTAT 'resource name'

SUBSYS (OPCA | MSTR | subsystem name)

AVAIL (RESET | NO | YES | KEEP)

DEVIATION (amount | RESET)

QUANTITY (amount | RESET)

CREATE (YES | NO)

TRACE (trace level)
```

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.

**SUBSYS** (**subsystem name** | MSTR | OPCA)

The name of the Tivoli OPC 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 Tivoli OPC tracker subsystems on the MVS system where the SRSTAT command is issued.

**Note:** If the Tivoli OPC trackers and controller in your Tivoli OPC configuration run on different subsystems, specify the name of the tracker subsystem in 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 OPC 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 dialog), 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.

**QUANTITY (RESET | amount | KEEP)**

If you want 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 dialog), 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.

**DEVIATION (RESET | amount | KEEP)**

If you want 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.

**CREATE (NO | YES)**

NO indicates that the resource should not be added to the current plan of the receiving Tivoli OPC 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 OPC adds the resource to the current plan of the receiving Tivoli OPC subsystem if the resource is not in the database. It uses these values:

- **Text**: Blank.
- **Specres group ID**: Blank.
- **Hiperbatch**: No.
- **Used for**: Control.
On error
Blank. If an error occurs, Tivoli OPC uses the value specified in the operation details or, if that is also blank, the value of the ONERROR keyword of RESOPTS.

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).

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 ddbname 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).

---

**SRSTAT**

**Example 1 - TSO command**

ALLOC F(EQQMLIB) 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
//STEP1 LIB 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(EQQMLIB) 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).
The WSSTAT command lets you change the status of a workstation in the current plan. The status information is communicated to the Tivoli OPC 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 Notes**

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 dataset to the TSO user's address space, 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 Tivoli OPC's fixed resource code RL and subsresource 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 MVS 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 an MVS IPL, no authority verification can be performed.

**Note:** If the status of a workstation has been set to offline manually, using the Tivoli OPC dialogs, then you are not allowed to reset it to active using the WSSTAT command.

**Syntax**

```
WSSTAT command

WSNAME --(workstation name)

STATUS --(A)

SUBSYS --(subsystem name)

OPCA

trace level

STARTOPR --(R)

REROUTE --(R)

ALTWS(alt workname)
```
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.

**REROUTE** *(R | L)*

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.

The parameter is optional.

**STARTOPR** *(R | E | L)*

When the workstation status is set to offline or failed, you can specify what Tivoli OPC should do with operations that are currently in started status at the workstation, where:

- **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 will be used as default.

The parameter is optional.

**Note:** If you select STARTOPR(E), a started job continues to run—Tivoli OPC never cancels jobs that have started.

**STATUS** *(A | O | F)*

The status you want to report for the workstation

- **A** Active
- **O** Offline
- **F** Failed.

You must specify this parameter.

**SUBSYS** *(subsystem name | MSTR | OPCA)*

The name of the Tivoli OPC 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 Tivoli OPC tracker subsystems on the MVS system where the WSSTAT command is issued.

**Note:** If the Tivoli OPC trackers and controller in your configuration run on different subsystems, specify the name of the tracker subsystem in this parameter.
**WSNAME** *(workstation name)*  
The name of the workstation to be updated.  
You must specify 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.

The WSSTAT parameters are checked for validity and consistency. The validity checks are carried out in the Tivoli OPC tracker where the command is executed. The validity check processes parameter names, length and type of parameter values.

If the input is valid, a *workstation status event* is generated and communicated to the Tivoli OPC 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).

**Example 1 - TSO command**

```sql
ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE
WSSTAT SUBSYS(OPCB) WSNAME(AS4H) STATUS(0) START(R)
```

**Example 2 - Batch job**

```sql
//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. MVS Operator Commands Supported by Tivoli OPC

You can start, stop, cancel, or modify Tivoli OPC using the following MVS 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.

Starting Tivoli OPC

To start Tivoli OPC, enter this MVS operator command:

```
START Tivoli OPC
S procname (procname = Tivoli OPC JCL procedure name)
```

If a Tivoli OPC 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 Tivoli OPC message log (ddname EQQMLOG) because the message log is already being used by the active started task. If Tivoli OPC 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 the Tivoli OPC subsystem. JES starts this job in the same manner as any ordinary job.

**Note:** Because Tivoli OPC uses JES exits, among other things, to track the progress of MVS jobs, it does not start before JES is active.

Stopping Tivoli OPC

To stop Tivoli OPC, enter the following MVS operator command:

```
STOP Tivoli OPC
P procname (procname = Tivoli OPC JCL procedure name)
```

If you are stopping a controller, the controller creates a backup copy of the current plan dataset (if required) and ends all active functions.
When Tivoli OPC ends, it writes this message to the message log:

```
STOP message log

EQQZ086I NO ACTIVE Tivoli OPC SUBTASKS.
Tivoli OPC IS ENDING
```

### Canceling Tivoli OPC

If Tivoli OPC is still active 5 minutes after you enter the STOP operator command, you must cancel Tivoli OPC.

You may also need to cancel Tivoli OPC 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:

1. **CANCEL Tivoli OPC**
   ```
   C procname,DUMP  (procname = Tivoli OPC JCL procedure name)
   ```

   This causes Tivoli OPC to end with a dump on the SYSMDUMP file (if the ddname is in the started-task JCL). The second way is to enter:

2. **CANCEL Tivoli OPC**
   ```
   C procname  (procname = Tivoli OPC JCL procedure name)
   ```

   This causes Tivoli OPC to end without a dump.

   If the STOP command is ineffective and you have no earlier documentation of the problem, cancel Tivoli OPC with a dump so that the error can be located.

### Modifying Tivoli OPC

Using the MODIFY command, you can supply information to Tivoli OPC after it has started. The syntax of the MODIFY command is:

```
MODIFY Tivoli OPC

F procname,modifyoption
```

where:

- **procname**  Is the Tivoli OPC JCL procedure name
- **modifyoption**  Is one of the following:
  - **S=taskname**  Start the specified Tivoli OPC subtask.
\textbf{P=taskname}

Stop the specified Tivoli OPC subtask.

\textit{taskname} can be one of the following:

- **APPC**: APPC subtask.
- **AR**: Automatic recovery subtask.
- **A4**: APPC tracker router subtask.
- **DC**: Catalog management 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 joblog task.
- **GEN**: General service 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 Tivoli OPC functions require an active NMM task to execute successfully.
- **RODM**: RODM subtask.
- **SUB**: Submit subtask.
- **TA**: TCP/IP router subtask.
- **VTAM**: Network communication function (NCF) subtask.
- **WSA**: Workstation analyzer subtask.

Only the tasks in the Tivoli OPC subtask table can be activated by a \texttt{MODIFY} command. The subtask table is built when Tivoli OPC 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.

\textbf{CPQSTA=ON}

Activates the \texttt{STATMSG(CPLOCK)} message.

\textbf{CPQSTA=OFF}

Deactivates the \texttt{STATMSG(CPLOCK)} message.

\textbf{DSPSTA}

Displays, in message EQQZ095, 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 \textit{Tivoli OPC Messages and Codes}.

\textbf{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.

HB(, TRK)
Sends a heartbeat message on the MVS system log for the OPC subsystem or for all trackers connected to that subsystem if it is an OPC controller. If GEM/OPC instrumentation is active, a heartbeat event is sent to the Tivoli GEM console and an icon is shown for the subsystem and for each active tracker agent connected to that subsystem.

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
Deactivate the single JCL trace.

NEWDSLST
Order a Tivoli OPC tracker to rebuild the dataset triggering filter table EQQDSLST. The new table is read from member EQQDSLST of the dataset referenced to by the EQQJCLIB ddname in the started task JCL for the tracker. The new table replaces the table in ECSA.

NEWNOERR
Order a Tivoli OPC tracker to rebuild the NOERROR table, in the case NOERROR statements have been modified in the parameter library member that contains the JTOPTS statement.

NOERRMEM
Order a Tivoli OPC tracker 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 Tivoli OPC tracker 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: Tivoli OPC opens the EQQPARM library when Tivoli OPC 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 datasets that define the EQQPARM library should be allocated without any secondary extents.
QUELEN=nnnn
Sets the new value of the QUELEN keyword of the JTOPTS statement. Allowed values are 0 to 9999, but a minimum value of 5 is forced.

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 the OPC subsystem. The status can be one of the following:

FULLY_OPERATIONAL
Everything is active and is working properly.

PARTIALLY_OPERATIONAL
The OPC subsystem has limited functionality. For example, if a controller ER is stopped, the controller can still schedule jobs but cannot receive their statuses.

NOTOPERATIONAL
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 the Tivoli OPC dataset 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 dataset 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).

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 Tivoli OPC tracker, as specified in the ROUTOPTS keyword in the initialization statements.

STATUS, SUBTASK
Lists all Tivoli OPC subtasks with their statuses. The status can be ACTIVE or INACTIVE.

STATUS, TRK={trkname | trktype}
Returns the status of a Tivoli OPC tracker agent defined for that controller. The status can be ACTIVE or INACTIVE, indicating whether an active session exists for that tracker agent.
**trkname** The tracker agent destination name defined in the Tivoli OPC PARM member, or the TCP/IP or SNA address of the tracker agent.

**trktype** The type of tracker agent. It can be **TCP**, **SNA**, **APPC** or **ALL**. If you specify one of these values, you will get the status for each defined tracker agent of that type. For example, if you specify TCP, you will get the status of all TCP/IP tracker agents you have defined for that controller; if you specify ALL, you will get the status of all trackers defined to that controller.

**TAKEOVER**

Order 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 OPC address spaces where OPHOST(STANDBY) is specified on the OPCOPTS initialization statement.

**Note:** Takeover can occur automatically if you have specified the TAKEOVER keyword on the XCOPTS initialization statement of a standby system. Refer to *Customization and Tuning* for more information.

**VSTRC=START**

Start a trace on the message log of all VSAM I/O requests. In a busy Tivoli OPC system, you will need a large message-log dataset, and the trace will affect the performance of the Tivoli OPC system.

**VSTRC=STOP**

Stop 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 gives this message on SYSLOG:

```
IEE324I MODIFY REJECTED - TASK BUSY
```

---

**Modifying Tivoli OPC Data Store**

Using the MODIFY command, you can supply information to the Tivoli OPC Data Store after it has started. The syntax of the MODIFY command is:

```
MODIFY Tivoli OPC
F procname,modifyoption
```

where:

**modifyoption**

Is 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.
ARDYTNU Display CINTERVAL value.
ARDYNNS Display MAXSYSL value.
ARDYPM Display all initialization parameters values.

ARMDWR Modify number of active writers.
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/deactivate Communication task traces.
ARDGWR=on/off Activate/deactivate Writer task traces.
ARDGRD=on/off Activate/deactivate Reader task traces.
ARDGJQ=on/off Activate/deactivate JES Queue task traces.
ARDGDB=on/off Activate/deactivate Data Base task traces.
Appendix C. Status, Error, and Reason Codes

This appendix lists status, error, catalog management, and operation reason codes.

Tivoli OPC assigns a status code to every occurrence and every operation in the current plan. In addition, an error code is assigned for any operation that ends in error. When the catalog management function is used, Tivoli OPC maintains status information that reports the progress of the catalog management action.

The codes assigned by Tivoli OPC are not just for documentation purposes. They report the real status of the operation and are used by a number of Tivoli OPC functions to make important decisions, which can ultimately determine an operation outcome.

### Occurrence Status Codes

The occurrence status codes are:

- **C**: Complete
- **D**: Deleted
- **E**: An operation in the occurrence has ended-in-error
- **P**: A pending predecessor exists for the occurrence
- **S**: Started
- **U**: Undecided (the status is not known)
- **W**: No operations in the occurrence have started

### Operation Status and Extended Status Codes

When Tivoli OPC displays the status of an operation, it uses the format xy, where x is the status code and y, if present, is the extended status code.

#### Operation Status Codes

The operation status codes are:

- **A**: Arriving—the operation is ready for processing; no predecessors were defined
- **R**: Ready for processing; all predecessors are complete
- **S**: Started
- **C**: Complete
- **D**: Deleted
- **I**: The operation is interrupted
- *****: Ready—at least one predecessor is defined on a nonreporting workstation; all predecessors are complete
- **E**: The operation has ended-in-error
- **W**: The operation is waiting for a predecessor to complete
- **U**: Undecided—the operation status is not known
Extended Status Codes

Together with the normal status codes, Tivoli OPC maintains extended status codes that provide additional information about the status of operations. The extended status code is not always present.

The following extended status codes are valid, depending on the type and status of the operation:

- Valid for all operations that have a status of arriving (A) or ready (* or R):
  - X Waiting for resource.
  - H A dialog user has used the HOLD command on the operation.
  - N A dialog user has used the NOP command on the operation.

- Valid for all operations that have a status of arriving (A), ready (* or R), started (S), or error (E):
  - M The status of the operation has been manually set by a dialog user from the ready list.
  - Valid only for computer workstation operations that have a status of arrived (A) or ready (* or R):
  - T Waiting until a particular time.
  - L The operation is a late time-dependent operation with the suppress-if-late attribute.
  - R The operation has ended in error but was automatically reset (the completion code is defined in the installation options to be automatically reset).
  - E An error occurred during job submission or release.
  - D Closedown in progress.

Blank Tivoli OPC is in the process of submitting this job. Tivoli OPC is waiting for the availability of a parallel server or a critical resource, or the operation is not to be submitted automatically.

- Valid only for computer workstation operations that have a status of started (S):
  - Q The job has been added to the JES job queue.
  - S The job or started task is executing.
  - M The status of the job or started task has been manually set to S.
  - U Submit in progress.

Blank The job has been successfully submitted but has not yet been reported as added to the JES job queue.

- Valid only for computer workstation operations that have a status of ready (R) or error (E):
  - A The job is waiting for a deferred catalog management action to be initiated or discarded by a dialog user.
  - C The job is waiting for the catalog management action to be completed.

The extended status codes described in this paragraph are available only through the user exit EQQUX007. The extended statuses for the operations in ready (* or
R) or arriving (A) status can discern situations that the panel extended status simply flagged with a blank.

- Valid only for operations with a current status of arriving (A) or ready (* or R) and that are on general non-reporting, computer or general, with WTO-enabled workstations (where the meaning of the extended status is applicable for workstation type):

  I  Idle workstation: the workstation is closed.
  P  Control on parallel servers is activated and all parallel servers are in use.
  1 Control on workstation resource R1 is activated and there is not
  2 Control on workstation resource R2 is activated and there is not enough
      availability of R2.
  J  Automatic job submission is not set.
  K  Job submission deactivated in JTOPTS statement.

- Valid for all operations:
  Z  Failure in updating operation information.
  B  Extended status does not have meaning.

## Error Codes

Tivoli OPC 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.

- **JCCE**  An error during JCC (job completion checker) processing prevented the JCC from determining an error code for the operation.

- **JCL**  A JCL error was recognized after the job or started task began to execute, or a JCL error was recognized after syntax checking in the internal reader.

- **JCLI**  A JCL error occurred immediately; that is, the error was detected before the job or started task began. This code is also possible when both the job-start event (type 2) and the job-end event (type 3J) are missing.

- **MCP**  The operation was manually set to error in the MCP dialog.

- **OFxx**  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.

- **OJCV**  An error occurred during JCL-variable substitution when the job or started task was submitted, or Tivoli OPC detected an error in the RECOVER statement during automatic recovery. Browse the JCL for the operation or the EQQMLOG dataset to find more information about the failure.
Codes

OSEQ  A job or started task began to execute before all its predecessors had completed. This can occur only if the job was not submitted by Tivoli OPC and if either HOLDJOB(NO) or HOLDJOB(USER) is specified for the Tivoli OPC event writer options. Refer to Customization and Tuning.

OSUB  A failure occurred when Tivoli OPC attempted to submit a job or start a started task. In the case of a started task, it could be that the started task is a subsystem that is not started by JES, or the Tivoli OPC subsystem EQQSTC ddname is not allocated to a JES-defined procedure library. The operation should be marked as ended-in-error.

OSUF  A failure occurred when Tivoli OPC 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.

OSUP  A time operation is late, and the SUPPRESSACTION parameter of the JTOPTS initialization statement specified that the operation should be marked as ended-in-error.

OSxx  The system on which the operation is defined has failed. The WSFAILURE parameter on the JTOPTS initialization statement specifies that started operations should be marked as ended-in-error. xx is the status and extended status of the failing operation. Operations that were running (status SS) have a step-code error of OSYS.

PCAN  A print operation was canceled by the operator.

nnnn  Step return code.

Sxxx  System abend code.

Uxxx  User abend code in hexadecimal notation. For example user abend 2750 is represented in Tivoli OPC as UABE.

xxxx  User-defined error code.

Catalog Management Status Codes

When the catalog management function is used, Tivoli OPC maintains status information to report on the progress of the catalog management action. The following CM status codes are possible:

Z  Cancelled—catalog management status is not available. Examine the Tivoli OPC message log for CM errors; one possibility is that the dataset collection at the tracker has failed.

E  Catalog management action has failed.

I  Catalog management has been initiated but has not yet started.

S  The catalog management action has started.

M  The catalog management action is in progress but is delayed waiting for a response from DFHSM.

R  A catalog management action has been restarted. This status is possible if either the controller or tracker stopped while a CM action was in progress.

C  Actions completed normally.
D  The catalog management action has been discarded by a dialog user or the pre-catalog management exit, EQQUX008.

X  Excluded—Tivoli OPC has determined that there are no datasets defined by DD statements in the JCL that require catalog management actions. This status has the same meaning as complete.

---

### Job Log Retrieval Status Codes

When the job log retrieval function is used, Tivoli OPC maintains status information to report on the retrieval of the log. The following status codes are possible:

- **C**  Completed—the controller has received the log.
- **E**  Error. There was an error retrieving the log.
- **I**  Initiated. The controller has sent a retrieval request to the tracker, but the tracker has not yet processed the request.
- **S**  Started. The controller has sent a retrieval request to the tracker, and the tracker has started to retrieve the log.
- **blank**  The controller has not sent any retrieval request to the tracker.

---

### Operation Reason Codes

If your ready list layout includes the RSNC field, you can see these operation reason codes. Note that the codes are listed in hierarchical order. For example, if job submission failed, and job submission is deactivated, code **D** is obtained—not code **F**.

- **D**  Job submission deactivated
- **C**  Workstation is closed
- **P**  All parallel servers in use
- **A**  Automatic reset error condition
- **F**  Job submission failed
- **J**  No automatic job submission
- **L**  Job is late
- **T**  Start time not reached
- **1**  Not enough free WS resource 1
- **2**  Not enough free WS resource 2
- **H**  Closedown in progress
- **S**  Waiting for special resource
Appendix D. Batch Programs

This appendix describes the JCL and parameters that the batch programs need, although you do not need this if you use the dialogs to generate and submit them.

For a description of how to create JCL skeletons for the Tivoli OPC batch jobs, refer to the section “Using the EQQJOBS installation aid” in Installation Guide.

Most Tivoli OPC 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 Tivoli OPC batch load modules, must reside in an authorized library. These batch programs are not invoked with EQQBATCH: EQQYLTOP, EQQPURGE, and EQQEVPGM.

DD Statements Referenced by Tivoli OPC Batch Jobs

Tivoli OPC batch jobs use three categories of DD statements:

- DD statements that are required by EQQBATCH
- DD statements that refer to Tivoli OPC databases
- DD statements that refer to work datasets

For DCB and SPACE requirements of datasets 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 in Tivoli OPC:

- EQQDMSG Output; contains diagnostic messages.
- EQQDUMP Output; contains diagnostic trace information.
- EQQMLIB Input; messages, translatable terms, and headers.
- EQQMLOG Output; messages.
- EQQPARM Input; Tivoli OPC initialization parameters.

This DD statement in the JCL references the member in the parameter library that contains the BATCHOPT statement for the Tivoli OPC subsystem. The BATCHOPT statement defines run-time options for the Tivoli OPC 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.
Batch Programs

STEPLIB Input; Tivoli OPC step library. (This statement is not required if the Tivoli OPC modules are part of the MVS link library.)

SYSIN Input; inline input parameters to the batch programs.

SYSMDUMP Output; the dump dataset.

DD Statements Describing Tivoli OPC Data

Tivoli OPC data is kept in VSAM as well as non-VSAM datasets:

- **EQQADDS** VSAM; application descriptions and JCL variable tables
- **EQQAD2DS** VSAM; application descriptions (used by the batch loader)
- **EQQCKPT** Sequential; checkpoint dataset
- **EQQCP1DS** VSAM; primary-current-plan dataset
- **EQQCP2DS** VSAM; alternate-current-plan dataset
- **EQQCXDS** VSAM; current plan extension, containing special resource information
- **EQQJTxx** Sequential; job-tracking-log dataset (maximum 99)
- **EQQJTARC** Sequential; job-tracking-archive dataset
- **EQQLTBKP** VSAM; long-term plan backup
- **EQQLTDS** VSAM; LTP dataset
- **EQQNCPDS** VSAM; new-current-plan dataset
- **EQQNCXDS** VSAM; new current plan extension, containing special resource information
- **EQQOIDS** VSAM; operator instruction database
- **EQQRDDS** VSAM; resource definition database
- **EQQWSDS** VSAM; workstation description and calendar database.

These datasets are common to most batch programs and to the Tivoli OPC subsystem. Ensure that your batch jobs use the same datasets as the Tivoli OPC subsystem (except when you creating a separate database using the batch loader program, as described in “Batch-Loader Output—Tivoli OPC System or VSAM Dataset?” on page 132).

The batch jobs also allocate work datasets.

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 Tivoli OPC 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 OPC 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>

**Example**

```
//SYSIN DD *
981201000430
/*
```

A batch program for creating a new LTP with the SYSIN DD statement used in the example will produce a LTP for the period 1998/12/01 to 2000/04/30.

All batch programs that use the LTP must have a start day later than 1978/01/01 and an end date before 2066/12/31, because of the way Tivoli OPC handles calendars internally.

The following table summarizes the batch program functions.

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>EQQADCP</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>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>EQQDTTOP</td>
<td>Produce a trial plan.</td>
</tr>
</tbody>
</table>
**Table 15 (Page 2 of 2). Tivoli OPC Batch Programs**

<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>EQQJVPRT</td>
<td>Print JCL variables.</td>
</tr>
<tr>
<td>EQLTCRE</td>
<td>Create a new LTP.</td>
</tr>
<tr>
<td>EQLTMOA</td>
<td>Modify the LTP for all applications or extend the LTP.</td>
</tr>
<tr>
<td>EQLTMOO</td>
<td>Modify the LTP for one application.</td>
</tr>
<tr>
<td>EQLTPRT</td>
<td>Print the LTP.</td>
</tr>
<tr>
<td>EQLTTRY</td>
<td>Make a trial LTP.</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>EQQRCSIM</td>
<td>Simulate return codes with step-level restart.</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 RUN DATES
//++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
//ADCOP EXEC PGM=EQQBATCH,PARM='EQQADCOP',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)
//LREPORT DD SYSOUT=/c5197,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT**
//SYSOUT DD SYSOUT**
//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT**
//EQQDMSG DD SYSOUT**
//LTOCIN DD DCB=(RECFM=FB,LRECL=672,BLKSIZE=6048),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//EQQADDS 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
```
The EQQADDEP batch program provides a printout of a cross-reference of external dependencies on the application description database.

**SYSIN Requirements**
None.

**JCL Example**

```jcl
//**************************************************************
/* APPLICATION CROSS REFERENCE OF EXTERNAL DEPENDENCIES        
//**************************************************************
//ADDEP EXEC PGM=EQQBATCH,PARM='EQQADDEP',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)
//ADREPORT DD SYSOUT=*,
  // DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLLOG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//ADDPIN DD DCB=(LRECL=65,BLKSIZE=3250,RECFM=FB),
  // SPACE=(CYL,(3,1)),DISP=(,DELETE),UNIT=SYSDA
//ADDPOUT DD DCB=(LRECL=65,BLKSIZE=3250,RECFM=FB),
  // SPACE=(CYL,(3,1)),DISP=(,DELETE),UNIT=SYSDA
//ADDPWK01 DD SPACE=(CYL,(1,3)),UNIT=SYSDA
//ADDPWK02 DD SPACE=(CYL,(1,3)),UNIT=SYSDA
//ADDPWK03 DD SPACE=(CYL,(1,3)),UNIT=SYSDA
//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**

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
//******************************************************************************
//* APPLICATION DESCRIPTION MASS UPDATE
//******************************************************************************
//ADUMU EXEC PGM=EQQBATCH,PARM='EQQADMUP',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)
//MUREPORT DD SYSOUT=*
//DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQLOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQMSG DD SYSOUT=*
//ADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//WSDS 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 *

CPU1 Y

XRAYNER U Y ( SAMPLE
OG - CPU1

Y

D

OG - CPU1

SAMPLE

*/
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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789012345678901234567890</td>
<td>LINE 1 (REQUIRED)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ABCDEFGHIJ</td>
<td>XXXXXXXXXX</td>
<td>LINE 2-n (OPTIONAL)</td>
<td></td>
</tr>
</tbody>
</table>

where:
- A = 1 APPLICATIONS, 2 DEPENDENCIES, 3 OPERATIONS, 4 APPL vs OWNER
- B = 1 ALL APPLICATIONS, 2 SELECTED ITEM, 3 FROM FILE
- CDEFGH = (YYMMDD)
- IJ = Number of lines following
- XXXXXXXXXX = Application ID (number specified by IJ)

Notes

If you specify a dataset name on the select panel, the ADUSERDS DD is set to the dataset 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 dataset
JCL Example

```sql
/* ***********************************************************************
/* APPLICATION PRINT PROGRAM
/* ***********************************************************************
/* ADPRT EXEC PGM=EQQBATCH,PARM='EQQADPRT',REGION=2048K
/* STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.AFLIB
/* 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)
/* ADREPORT2 DD DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050),
/*   SPACE=(CYL,(3,1)),DISP=(NEW,DELETE),UNIT=SYSDA
/* EQQMLOG DD SYSOUT=*
/* SYSOUT DD SYSOUT=*
/* SYSPRINT DD SYSOUT=*
/* SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
/* EQUDDUMP DD SYSOUT=*
/* EQQMLOG DD SYSOUT=*
/* ADPRIN DD DCB=(LRECL=200,BLKSIZE=3200,RECFM=FB),
/*   SPACE=(CYL,(3,1)),DISP=(NEW,DELETE),UNIT=SYSDA
/* ADPROUT DD DCB=(LRECL=200,BLKSIZE=3200,RECFM=FB),
/*   SPACE=(CYL,(3,1)),DISP=(NEW,DELETE),UNIT=SYSDA
/* ADPRWK01 DD SPACE=(CYL,(3,1)),UNIT=SYSDA
/* ADPRWK02 DD SPACE=(CYL,(3,1)),UNIT=SYSDA
/* ADPRWK03 DD SPACE=(CYL,(3,1)),UNIT=SYSDA
/* EQQADD DD DSN=EID.EIDAR3.AD,DISP=SHR
/* EQQADD2 DD DSN=EID.EIDAR3.AD,DISP=SHR
/* ADUSERDS DD DSN=NULLFILE,DISP=SHR
/* SYSSN DD *
/* SAMPLE
/* =PARAMETERS(ABCDEFGHIJ)
/* A = 1 APPLICATIONS, A = 2 DEPENDENCIES, A = 3 OPERATIONS
/* A = 4 APPLIC VS OWNER
/* B = 1 ALL APPLICATIONS, B = 2 SELECTED ITEM, B = 3 FROM FILE
/* CDEFGH = (YYMMDD)
/* IJ = NUMBER OF ITEMS
```
EQQADXRF—Print Applications: Cross-Reference Job Names

The EQQADXRF batch program provides a printout of cross-references of job names on the application description database.

SYSIN Requirements
None.

JCL Example

```plaintext
//******************** ******************************************************
/*/ APPLICATION CROSS REFERENCE PROGRAM
//******************** ******************************************************
//ADXRF EXEC PGM=EQQBATCH,PARM='EQQADXRF',REGION=2048K
//STPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//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=*
//SYSPUNCH DD SYSOUT=*
//SYSDUMP DD SYSOUT=*
//SYSDUMP DD SYSOUT=*
//ADWKIN DD DCB=(LRECL=78,BLKSIZE=3900,RECFM=FB),
//    SPACE=(CYL,(3,1)),DISP=(DELETE),UNIT=SYSDA
//ADWKOUT DD DCB=(LRECL=78,BLKSIZE=3900,RECFM=FB),
//    SPACE=(CYL,(3,1)),DISP=(DELETE),UNIT=SYSDA
//ADWKWK01 DD SPACE=(CYL,(3,1)),UNIT=SYSDA
//ADWKWK02 DD SPACE=(CYL,(3,1)),UNIT=SYSDA
//ADWKWK03 DD SPACE=(CYL,(3,1)),UNIT=SYSDA
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//SYSIN DD *
```
EQQAXR00—Print Applications: Cross-Reference Different Items

The EQQAXR00 batch 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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345789012345678901234567890</td>
<td>LINE 1 (REQUIRED)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AABBCCDDEE</td>
<td>where AA, BB, CC, DD, EE are the codes for the sort order (maximum of 5):</td>
<td></td>
<td></td>
</tr>
<tr>
<td>01 = APPLICATION ID</td>
<td>09 = PRIORITY</td>
<td></td>
<td></td>
</tr>
<tr>
<td>02 = APPLICATION TEXT</td>
<td>10 = SYSOUT CLASS</td>
<td></td>
<td></td>
</tr>
<tr>
<td>03 = OP ID</td>
<td>11 = FORM NUMBER</td>
<td></td>
<td></td>
</tr>
<tr>
<td>04 = OP TEXT</td>
<td>12 = SPECIAL RESOURCES</td>
<td></td>
<td></td>
</tr>
<tr>
<td>05 = PERIOD NAME</td>
<td>13 = WS RESOURCE R1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>06 = RUN CYCLE DESCRIPTION</td>
<td>14 = WS RESOURCE R2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>07 = JOB NAME</td>
<td>15 = CALENDAR ID</td>
<td></td>
<td></td>
</tr>
<tr>
<td>08 = JOB CLASS</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

JCL Example

```plaintext
/* APPLICATION CROSS REFERENCE OF SELECTED FIELDS */
/* ADXREF EXEC PGM=EQQBATCH,PARM='EQQAXR00',REGION=2048K */
/* STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.AFLIB */
/* EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS */
/* EQQPARM DD DISP=SHR,DSN=OID.IDA.PARM(IDA) */
/* ADREPORT DD SYSOUT=**, DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050) */
/* EQQMLIB DD SYSOUT=** */
/* SYSPRINT DD SYSOUT=** */
/* SYSOUT DD SYSOUT=** */
/* SYSMOUT DD SYSPRINT=**, DSN=ID.IDA.IDA.SYSDUMP */
/* EQQDUMP DD SYSOUT=** */
/* EQQDMSG DD SYSOUT=** */
/* ADWKIN DD DBC=(RECFM=FBA,LRECL=44000,RECFM=FBA), SPACE=(CYL,(3,1)), DISP=(NEW,DELETE),UNIT=SYSDA */
/* ADWKOUT DD DBC=(RECFM=FBA,LRECL=44000,RECFM=FBA), SPACE=(CYL,(3,1)), DISP=(NEW,DELETE),UNIT=SYSDA */
/* ADWKWK01 DD SPACE=(CYL,(3,1)),UNIT=SYSDA */
/* ADWKWK02 DD SPACE=(CYL,(3,1)),UNIT=SYSDA */
/* ADWKWK03 DD SPACE=(CYL,(3,1)),UNIT=SYSDA */
/* EQQADDS DD DSN=ID.IDA.IDA.3.AD,DISP=SHR */
/* SYIN DD * */
071031215 */
```
**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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>YYYYMMDD</td>
<td>LINE 1 (REQUIRED)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

- **YYYYMMDD** = End date for the report (YY = YEAR, MM= MONTH, DD= DAY)

**JCL Example**

```plaintext
viar/* PRINT ALL CALENDARS */r
/* ****************************************** */r
vr//PRTCAL EXEC PGM=EQQBATCH,PARM='EQQCLPRC',REGION=2048K */r
vr//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB */r
vr//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS */r
vr//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA) */r
vr//CLREPORT DD SYSOUT=*, */r
vr//   DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050) */r
vr//EQQMLOG DD SYSOUT=*, */r
vr//SYSPRINT DD SYSOUT=*, */r
vr//SYSOUT DD SYSOUT=*, */r
vr//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB */r
vr//EQQDUMP DD SYSOUT=*, */r
vr//EQQMSG DD SYSOUT=*, */r
vr//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR */r
vr//SYSIN DD *, */r
vr951231 */r
vr/*/ = ENDDATE OF CALENDARS PRINT (YYYYMMDD)
```
Batch Programs

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>CCCCCCCC</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

YYMMDD = End date for the report (YY = YEAR, MM = MONTH, DD = DAY)

CCCCCCCCCCCCCCC = Calendar for the calculation of the origin date

JCL Example

```plaintext
//PRTPER EXEC PGM=EQQBATCH,PARM='EQQCLPRP',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)
//CLREPORT DD SYSOUT=*
//      DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SOUT DD SYSOUT=*
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDSG DD SYSOUT=*
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//SYSEX DD *
951231DEFAUL
//* = ENDDATE OF PERIODS PRINT (YYMMDD)
//* = CALENDAR NAME
```
EQQDNTOP—Create or Extend the Current Plan

The EQQDNTOP batch program creates or extends the current planning period.

**Note:** Tivoli OPC job tracking uses two datasets to log each event that has been processed. If you need to restart Tivoli OPC from a checkpoint, Tivoli OPC uses the older of the two current plan datasets. Events logged on the current log dataset are then reprocessed. When this reprocessing finishes, a current plan backup copy is created so that the two current plan datasets are again equal. Tivoli OPC 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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>123456789012345678901234567890</td>
<td>(format 1)</td>
<td>(format 2)</td>
</tr>
<tr>
<td>YYYYMMDDHHMMhhmmWabcdefgh</td>
<td>YYYYMMDDHHMMyymmddhhmmbcdefgh</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

- **YYYYMMDDHHMM** = 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)
- **yymmddhhmm** = Plan end date and time (format 2)
- **abcdefgh** = Report flags. Set to 1 or /zerodot. (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.
- 95120112001000000 To extend to 1 December 1995 at 12.00 and have the workstation summary report only.
JCL Example

//***************************************************************
// DAILY PLANNING - PLAN NEXT PERIOD
//***************************************************************
// DONTOP  EXEC PGM=EQQBATCH,PARM='EQQNQTOP',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)
// EQQDIN DD DSN=&&A,DISP=(NEW,PASS),
// DCB=(RECFM=FB,LRECL=273,BLKSIZE=273/zerodot),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
// 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=
// SYSPRINT DD SYSOUT=
// EQQDUMP DD SYSOUT=
// EQQMSG DD SYSOUT=
// SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
// EQQSDS DD DISP=SHR,DSN=EID.EIDA3.WS
// EQQADDS DD DISP=SHR,DSN=EID.EIDA3.AD
// EQQRDDS DD DISP=SHR,DSN=EID.EIDA3.RD
// EQQLTDS DD DISP=SHR,DSN=EID.EIDA3.LT
// EQQLTBKPD DD DISP=SHR,DSN=EID.EIDA3.LB
// EQQCP1DS DD DISP=SHR,DSN=EID.EIDA3.CP1
// EQQCP2DS DD DISP=SHR,DSN=EID.EIDA3.CP2
// EQQCPDS DD DISP=SHR,DSN=EID.EIDA3.NCP
// EQQCXDS DD DISP=SHR,DSN=EID.EIDA3.NCX
// EQQcxDS DD DISP=SHR,DSN=EID.EIDA3.CX
// EQQTARCD DD DISP=SHR,DSN=EID.EIDA.JTARC
// EQQCKPT DD DISP=SHR,DSN=EID.EIDA.CKPT
// SYSin DD *
// 02400 1111111
// = PLSTRTIMEPLENDTIME REPORTS
// PLSTRTIME(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
// 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
Batch Programs

//******************************************************************************
//-- STEP2 DAILY PLAN - SORT OF REPORT RECORDS
//--******************************************************************************
/*
SORT EXEC PGMS=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=273,BKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA 
//SORTWK01 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA 
//SORTWK02 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA 
//SORTWK03 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA 
//SYSIN DD *
SORT FIELDS=(1,13,CH,A) 
/*
//******************************************************************************
//-- STEP3 DAILY PLAN - PLAN NEXT PRINT THE REPORTS
//--******************************************************************************
/*
DPREPORT EXEC PGMS=EQQBJOBK,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=FB,LRECL=121,BKSIZE=6050) 
//EQQMLOG DD SYSOUT=*
//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB 
//EQQDUMP DD SYSOUT=*
//EQQMSG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//EQQPOUT DD DSN=*.SORT.SORTOUT,DISP=(OLD,DELETE)
//SYSIN DD * 
/*
**Batch Programs**

**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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>123456789012345678901234567890</td>
<td>YYMMDHHMM</td>
<td>yymmdhhmm</td>
<td></td>
</tr>
</tbody>
</table>

where:

- **YYMMDHHHMM** = Plan start date and time
- **yymmdhhmm** = Plan end date and time

**JCL Example**

```plaintext
//************************************************************
//* DAILY PLANNING - PRINT CURRENT PERIOD RESULTS
//************************************************************
/DOTOP EXEC PGM=EQQBATCH,PARM='EQQDOTOP',REGION=4096K
/STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
/EQQLIB DD DISP=SHR,DSN=OPCTEST.EA131.EQQMSG
/EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
/SYSPRINT DD SYSOUT=
/EQQMLOG DD SYSOUT=
/SYSOUT DD SYSOUT=
/EQQMLOG DD SYSOUT=
/EQQDUMP DD SYSOUT=
/SYSDUMP DD DSN=EID.EIDA.SYSDUMPB,DISP=MOD
/EQQDIN DD DSN=&A,DISP=(NEW,PASS),
   DCB=(RECFM=FB,LRECL=273,BLKSIZE=273),
   SPACE=(CYL,(3,1)),UNIT=SYSDA
/EQQPIDS DD DSN=EID.EIDAR3.CP1,DISP=SHR
/EQQCP2DS DD DSN=EID.EIDAR3.CP2,DISP=SHR
/EQQCKPT DD DSN=EID.EIDA.CKPT,DISP=SHR
/SYSDIN DD *
95061012009506111600
/* = PLSTRTTIMEPLENDEMTIME
/* PLSTRTTIME(YYMMDHHM) = PLAN START DATE,TIME IF REFRESH
/* PLENDEMTIME (YYMMDHHM) = PLAN END DATE,TIME
```

366  Tivoli OPC Planning and Scheduling the Workload
/*
//***********************************************
//* STEP2 DAILY PLAN - SORT OF REPORT RECORDS
//***********************************************
//*
//SORT EXEC PGM=SORT,REGION=4096K,TIME=1440,COND=(8,LT)
//SYSIN DD SYSOUT**
//SYSOUT DD SYSOUT**
//SORTIN DD DSN=&&A,DISP=(OLD,DELETE)
//SORTOUT DD disp=(NEW,PASS),
// DCB=(RECFM=FB,LRECL=273,BLKSIZ=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SORTWK01 DD disp=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BLKSIZ=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SORTWK02 DD disp=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BLKSIZ=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SORTWK03 DD disp=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BLKSIZ=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SYSIN DD *
// SORT FIELDS=(1,13,CH,A)
/*

//***********************************************
//* 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
//EQQLIB DD disp=SHR,DSN=OPCTEST.ESA131.EQQMSG
//EQQPARM DD disp=SHR,DSN=EID.EIDA.PARM(EIDA)
//SYSIN DD *
// DPREPORT EXEC PGM=EQQBATCH,PARM='EQQDPRPT',COND=(8,LT),
// REGION=4096K
//STEPLIB DD disp=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQLIB DD disp=SHR,DSN=OPCTEST.ESA131.EQQMSG
//EQQPARM DD disp=SHR,DSN=EID.EIDA.PARM(EIDA)
//SYSIN DD *
// DPREPORT EXEC PGM=EQQBATCH,PARM='EQQDPRPT',COND=(8,LT),
// REGION=4096K
//STEPLIB DD disp=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQLIB DD disp=SHR,DSN=OPCTEST.ESA131.EQQMSG
//EQQPARM DD disp=SHR,DSN=EID.EIDA.PARM(EIDA)
//SYSIN DD *
**Batch Programs**

**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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1234567890</td>
<td>1234567890</td>
<td>1234567890</td>
</tr>
<tr>
<td>abcdefgh</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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**

```plaintext
//***********************************************
//* 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.SEQQMSG0
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//EQQDIN DD DSN=&&A,DISP=(NEW,PASS),
//       DCB=(RECFM=FB,LRECL=273,BLKSIZE=273),
//       SPACE=(CYL,(3,1)),UNIT=SYSDA
//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=* 
//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQWSDS DD DISP=SHR,DSN=EID.EIDAR3.WS
//EQQADDS DD DISP=SHR,DSN=EID.EIDAR3.AD
//EQQRDOS DD DISP=SHR,DSN=EID.EIDAR3.RD
//EQQLTDS DD DISP=SHR,DSN=EID.EIDAR3.LT
//EQLTBKP DD DISP=SHR,DSN=EID.EIDAR3.LB
```
Batch Programs

//EQQCP1DS DD DISP=SHR,DSN=EID.EIDAR3.CP1
//EQQCP2DS DD DISP=SHR,DSN=EID.EIDAR3.CP2
//EQQNCPS DD DISP=SHR,DSN=EID.EIDAR3.NCP
//EQQNCXDS DD DISP=SHR,DSN=EID.EIDAR3.NCX
//EQQXDS DD DISP=SHR,DSN=EID.EIDAR3.CX
//EQQJTARC DD DISP=SHR,DSN=EID.EIDA.JTARC
//EQQCKPT DD DISP=SHR,DSN=EID.EIDA.CKPT
//SYSIN DD /c5197
11111111

//STEP2 DAILY PLAN - SORT OF REPORT RECORDS
//*******************************************************************************
/*/   
//SORT EXEC PGM=SORT,REGION=4096K,TIME=1440,COND=(8,LT)
//SYSPRINT DD SYSPRINT DD SYSOUT**
//SORTOUT DD SYSPRINT DD SYSPRINT DD SYSOUT**
//SORTIN DD DSN=#A,DISP=(OLD,DELETE)
//SORTOUT DD DISP=(NEW,PASS),
// DCB=(RECFM=FB,LRECL=273,BLKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SORTWK/01 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BLKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SORTWK/02 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BLKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SORTWK/03 DD DISP=(NEW,DELETE),
// DCB=(RECFM=FB,LRECL=273,BLKSIZE=2730),
// SPACE=(CYL,(3,1)),UNIT=SYSDA
//SYSIN DD /*
// SORT FIELDS=(1,13,CH,A)
/*
//*******************************************************************************

//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
//EQMLIB DD DISP=SHR,DSN=OPCDEV.ESA131_SEQMSG0
//EQQ Parm DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//SYSPRINT DD SYSPRINT DD SYSOUT**,
// DCB=(RECFM=FBA,LRECL=121,BLKSIZ=6050)
//EQQMLOG DD SYSPRINT DD SYSOUT**
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSPRINT DD SYSOUT**
//EQQ MSG DD SYSPRINT DD SYSOUT**
//SYSOUT DD SYSOUT**
//EQQPOUT DD DSN=*.SORT.SORTOUT,DISP=(OLD,DELETE)
//SYSIN DD /*
**Batch Programs**

**EQQDTTOP—Produce a Trial Plan**

The EQQDTTOP batch program provides a trial plan for the current planning period. An LTP 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>1234567890</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **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)
- **abcdefgh** = Report flags. Set to 1 or /zerodot. (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**

```plaintext
//********************************************
//* DAILY PLANNING - PLAN A TRIAL PERIOD
//********************************************
//** EQQDTTOP EXEC PGM=EQQBATCH,PARM='EQQDTTOP',REGION=4096K
//** STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//** EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//** EQQPARAM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//** EQQDIN DD DSN=&&A,DISP=(NEW,PASS),
//** DCB=(RECFM=FB,LRECL=273,BLKSIZE=2730),
//** SPACE=(CYL,(3,1)),UNIT=SYSDA
//** 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:
```
Batch Programs

```plaintext
LRECL=32000 RECFM=VBS

ổiesionallycarss-

DESCRIPTIONS

SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB

EQQSDDS 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

EQQNCDS DD DISP=SHR,DSN=EID.EIDAR3.NCP

EQQJARC DD DISP=SHR,DSN=EID.EIDA.JTARC

EQQCKPT DD DISP=SHR,DSN=EID.EIDA.CKPT

SYSIN DD *

PLSTRTTIME TPLENDTIME REPORTS

PLSTRTTIME(YYMMDDHHMM) = PLAN START DATE, TIME IF REFRESH

PLENDTIME (YYMMDDHHMM) = PLAN END TIME

(HHMM ) = PLAN EXTENSION IN HOURS AND MINUTES

COUNTING ALL DAYS

(HHMMW ) = 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 SYSSOUT=

SYSSOUT DD SYSSOUT=

SORTIN DD DSN=&&A, DISP=(OLD, DELETE)

SORTOUT DD DISP=(NEW, PASS),

DCB=(RECFM=FB, LRECL=273, BLKSIZE=2730),

SPACE=(CYL,(3,1)), UNIT=SYSDA

SORTWK01 DD DISP=(NEW, DELETE),

DCB=(RECFM=FB, LRECL=273, BLKSIZE=2730),

SPACE=(CYL,(3,1)), UNIT=SYSDA
```
Batch Programs

//SORTWK02 DD DISP=(NEW,DELETE),
//   DCB=(RECFM=FB,LRECL=273,BLKSIZEx2730),
//   SPACE=(CYL,(3,1)),UNIT=SYSDA
//SORTWK03 DD DISP=(NEW,DELETE),
//   DCB=(RECFM=FB,LRECL=273,BLKSIZEx2730),
//   SPACE=(CYL,(3,1)),UNIT=SYSDA
//SYSIN DD *
//   SORT FIELDS=(1,13,CH,A)
/*
******************************************************************************
/*  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,BLKSIZEx6050)
//EQQMLOG DD SYSOUT=*
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//EQQOUT DD DSN=* SORT.SORTOUT,DISP=(OLD,DELETE)
//SYSIN DD *
/*

EQQEVPGM—Issue Tivoli OPC Commands in Batch

Use EQQEVPGM to issue Tivoli OPC TSO commands 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 317.

JCL Example

//STEP1 EXEC PGM=EQQEVPGM
//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=*.
//   SORT SORSTAT,DISP=(OLD,DELETE)
//SYSIN 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

```jcl
//**************************************************************
//* PRINT JCL VARIABLE TABLES
//**************************************************************
//JVTPRT 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)
//JVREPORT DD SYSOUT=*
//   DBC=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//EQQPARM DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//JVPRIN DD DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6220),
//   SPACE=(440,(4000,4000)),UNIT=SYSDA
//JVPROUT DD DCB=(RECFM=VBS,LRECL=32756,BLKSIZE=6220),
//   SPACE=(440,(4000,4000)),UNIT=SYSDA
//JVPRWK01 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//JVPRWK02 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//JVPRWK03 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//EQQADDs DD DSN=EID.EIDAR3.AD,DISP=SHR
//SYSIN DD *
PAY
//** = JCL VARIABLE TABLE NAME
```
**Batch Programs**

**EQQLTCRE—Create a New LTP**

The EQQLTCRE batch program creates a completely new LTP. The input parameters to the batch program give the start date and end date to the LTP 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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>123456789012345678901234567890</td>
<td>YYYMMDDyyyyydd</td>
<td></td>
</tr>
</tbody>
</table>

where:

- `YYMMDD = STARTDATE (YY = YEAR, MM = MONTH, DD = DAY)`
- `yyyyydd = ENDDATE (yy = YEAR, mm = MONTH, dd = DAY)`

**JCL Example**

```plaintext
//******************************
//* 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.SEQQMSG
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//LTREPORT DD SYSOUT=*,
//   DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQLOG DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMP
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//LTPRIN DD DCB=(RECFM=FB,LRECL=65,BLKSIZE=4550),
//   SPACE=(4550,(300,300)),UNIT=SYSDA
//LTPROUT DD DCB=(RECFM=FB,LRECL=65,BLKSIZE=4550),
//   SPACE=(4550,(300,300)),UNIT=SYSDA
//LTICIN 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
```
//LTPRWK/zerodot1 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK/zerodot2 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK/zerodot3 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK/zerodot1 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK/zerodot2 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK/zerodot3 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//EQQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR
//EQQLTBKP DD DSN=EID.EIDAR3.LB,DISP=SHR
//EQQLDDS DD DSN=EID.EIDAR3.LD,DISP=SHR,
// AMP=('BUFNI=1/zerodot,BUFND=1/zerodot')
//SYSIN DD *
950511950626
/*
//* = STARTDATE (YYMMDD)
//* ENDDATE (YYMMDD)
// *

**EQQLTMOA—Modify the LTP for All Applications or Extend the LTP**

The EQQLTMOA batch program can modify the LTP for all applications or extend the LTP. (If the start date of the LTP does not move forward each time you extend it, you might have an uncompleted occurrence at the start of the LTP.)

**SYSIN Requirements**

SYSIN data is required only for extending the LTP. 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.

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>123456789012345678901234567890</td>
<td>YMMDD</td>
<td>New LTP end date</td>
</tr>
<tr>
<td></td>
<td>DDDD</td>
<td>Extension in days, counting all days</td>
<td></td>
</tr>
</tbody>
</table>
**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.SEOQMSGO
//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 SYSOUT=*
//EQQDUMP DD SYSOUT=*
//EQQDMSG DD SYSOUT=*
//LTOLIN DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
//   SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOLOUT DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
//   SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRIN DD DCB=(RECFM=FB,LRECL=65,BLKSIZE=4550),
//   SPACE=(4550,(300,300)),UNIT=SYSDA
//LTPROUT DD DCB=(RECFM=FB,LRECL=65,BLKSIZE=4550),
//   SPACE=(4550,(300,300)),UNIT=SYSDA
//LTOCIN DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
//   SPACE=(4410,(300,300)),UNIT=SYSDA
//LTOCOUT DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
//   SPACE=(4410,(300,300)),UNIT=SYSDA
//LTOLWK01 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOLWK02 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOLWK03 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK01 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK02 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK03 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK01 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK02 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK03 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//EQQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR,
//   AMP=('BUFNI=10,BUFND=10')
//EQQLTBKP DD DSN=EID.EIDAR3.LB,DISP=SHR,
//   AMP=('BUFNI=10,BUFND=10')
//SYSIN DD *
//  0002
//  *** YYMMD = YYMMDD WHERE
//  *** YYMMD = YYMMDD = EXTEND DATE OR BLANK
//  *** DDDD = PLAN EXTENSION IN DAYS
//  *** DDDD = COUNTING ALL DAYS
//  *** OR BLANK
//  */```
EQQLTMOO—Modify the LTP for One Application

The EQQLTMOO batch program modifies the LTP 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

```java
/** Long Term Planning - Modify One Occurrence **
LTMODONE 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=*
EQQMSG DD SYSOUT=*
LTIOCIN DD DCB=(RECFM=FB,LRECL=735,BLKSIZE=4410),
   SPACE=((4410,2400),(2400,2400)),UNIT=SYSDA
EQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
EQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR
SYSIN DD *
APP4
// = APPLICATIONID
// APPLICATION ID = SELECTED APPLICATION
```
Batch Programs

**EQQLTPRT—Print the LTP**

The EQQLTPRT batch program prints information on the LTP. The input parameters to the batch program support these print functions:

- Print the LTP for all applications.
- Print the LTP 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:

```
COLUMNS 1 2 3
12345678901234567890123456789012345678
YYMMDDHHMMyyymmddhhmmRSxxxxxxxxxxxxxxxx
where:
YYMMDD = STARTDATE  HHMM = STARTTIME
yyymmdd = ENDDATE    hhmm = ENDTIME
R = REPORTTYPE (F=FULL,D=DEPENDENCIES)
S = SORT ORDER (R = INPUT ARRIVAL DATE, O = OWNER ID AND INPUT
           ARRIVAL DATE, A = OWNER ID APPLICATION ID)
xxxxxxxxxxxxxxxx = APPLICATION ID, or 'ALL'
```

**JCL Example**

```
// ********************************* - PRINT THE LONG TERM PLAN
// _________________________________
//LTREPORT EXEC PGM=EQQBATCH,PARM='EQQLTPRT',REGION=4/zerodot96K
//                                                                 ...
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCTEST.ESA131.SEQQMSG/zerodot
//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=*
//EQQMSG DD SYSOUT=*
```
EQQLTTRY—Make a Trial LTP

The EQQLTTRY batch program simulates an LTP 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>YYMMDD</td>
<td></td>
<td>yyymmdd</td>
<td></td>
</tr>
<tr>
<td>YMMDD</td>
<td>dddd</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

- YYMMDD = plan start date, or blank
- yyymmdd = plan end date (format 1)
- dddd = extension length, in days (format 2)

You can specify the following for a trial LTP:

- **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 LTP 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
//******************************************
//LTPTRYS EXEC PGM=EQQBATCH,PARM='EQQLTTRY',REGION=4096K
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCTEST.EAQ131.SEQQMSG0
//EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//LTRREPORT DD SYSOUT=*,
//     DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQLOG DD SYSOUT=* 
//SYSOUT DD SYSOUT=* 
//SYSPRINT DD SYSOUT=* 
//SYSMDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
//EQQDUMP DD SYSOUT=* 
//EQQDMSG DD SYSOUT=* 
```
Appendix D. Batch Programs

//LTOLIN DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
    SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOLOUT DD DCB=(RECFM=VB,LRECL=1000,BLKSIZE=6220),
    SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRIN DD DCB=(RECFM=FB,LRECL=65,BLKSIZE=4550),
    SPACE=(4550,(300,300)),UNIT=SYSDA
//LTPROUT DD DCB=(RECFM=FB,LRECL=65,BLKSIZE=4550),
    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
//LTOLWK1 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOLWK2 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOLWK3 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK1 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK2 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTPRWK3 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK1 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK2 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//LTOCWK3 DD SPACE=(CYL,(1,5)),UNIT=SYSDA
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
//EQQWSDS DD DSN=EID.EIDAR3.WS,DISP=SHR
//EQQLTDS DD DSN=EID.EIDAR3.LT,DISP=SHR
//EQQLTBKP DD DSN=EID.EIDAR3.LB,DISP=SHR,
    AMP=('BUFNI=1,BUFND=1')
//SYSIN DD *
// YYMMDD YMMDDDDDD WHERE
// YYMMDD = NEW START DATE OR BLANK
// YMMDD = NEW END DATE OR BLANK
// DDDD = PLAN EXTENSION IN DAYS
// COUNTING ALL DAYS
// OR BLANK
/*
**Batch Programs**

**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:

<table>
<thead>
<tr>
<th>COLUMNS</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12345678901234567890</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>format 1 - print in application order</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7yyymdd</td>
<td>format 4 - purge instructions</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

where:

- for format 4, instructions are purged if the valid-to date is less than yymmdd.

**JCL Example**

```plaintext
//**************************************************************
//* 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=*
//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=SYSDA
//OIWKOUT DD DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB),
//       SPACE=(CYL,(2,4)),UNIT=SYSDA
//OIWKWK01 DD SPACE=(CYL,(1,4)),UNIT=SYSDA
//OIWKWK02 DD SPACE=(CYL,(1,4)),UNIT=SYSDA
//OIWKWK03 DD SPACE=(CYL,(1,4)),UNIT=SYSDA
//EQQOIDS DD DSN=EID.EIDAR3.OI,DISP=SHR
//SYSIN DD *
5
/*
```

Tivoli OPC Planning and Scheduling the Workload
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 BATHCOPT 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:

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>8Y</td>
<td>(INSERT NEW, REPLACE OLD, OPERATOR INSTRUCTIONS)</td>
</tr>
<tr>
<td>8N</td>
<td>(INSERT NEW ONLY)</td>
</tr>
</tbody>
</table>

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 Dataset” on page 384.

JCL Example

```plaintext
/* OPERATOR INSTRUCTIONS - BATCH INPUT FROM A SEQ. DATA SET
/* Note:
/* If output is to a stopped opc subsystem, change ^QOIDS 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
EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
OIREPORT DD SYSDUT**, DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
EQQMLOG DD SYSDUT**, SYSPRINT DD SYSDUT**, SYSOUT DD SYSDUT**, SYSDUMP DD SYSDUT**, EQQDUMP DD SYSDUT**, EQQDMSG DD SYSDUT**, OIWKIN DD DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB), SPACE=(CYL,(2,4)),UNIT=SYSDA
OIIWOUT DD DCB=(LRECL=32756,BLKSIZE=32760,RECFM=VB), SPACE=(CYL,(2,4)),UNIT=SYSDA
```

Appendix D. Batch Programs 383
Batch Programs

//OIWKWK01 DD SPACE=(CYL,(1,4)),UNIT=SYSDA
//OIWKWK02 DD SPACE=(CYL,(1,4)),UNIT=SYSDA
//OIWKWK03 DD SPACE=(CYL,(1,4)),UNIT=SYSDA
//EQQADDS DD DSN=EID.EIDAR3.AD,DISP=SHR
/*************************************************************************/
/* IF THE CONTROLLER IS STOPPED WHEN RUNNING THIS JOB, */
/* CHANGE //EQQOIDS TO //EQQOIDS, */
/*EQQOIDS DD DSN=EID.EIDAR3.OI,DISP=SHR */
/*************************************************************************/
//EQQOPIN DD DSN=XRAYNER.OPERATOR.MESSAGES,DISP=SHR
//SYSIN DD *
8Y
/*

Layout of the Operator Instruction Dataset

Specify your operator instructions in the sequential dataset defined by the
EQQOPIN DD statement. The dataset 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:

---

**OI header**

OPC KEY=aaaaaaaaaaaaaaaaaaaaannntttttttttttttt,VALFROM=eeeeeeeeee

---

The characters `OPC` must be in columns 1 to 3. The variables have the following
meaning:

- `aaaaaaaaaaaaaaaaaaa` 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 00059308311200,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 dataset on the SPECIFYING SEQUENTIAL FILE NAME panel. You can display this panel by selecting option 1.5.5.2 from the Tivoli OPC main menu.
Batch Programs

EQQPURGE—Purge DLF Object

The EQQPURGE batch program receives a dataset 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

```
//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 OPC 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 OPC LOAD LIBRARY NAME
//JCLIN DD DISP=SHR,DSN=INPUT JCL DATA SET NAME
//JCLOUT DD SYSLIU=(A,INTRDR)
//SYSPRINT DD SYSOUT=YES
```
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 dataset name could not be located in catalog
**Explanation:** The catalog locate for this dataset name was unsuccessful.
**Action:** EQQPURGE terminates; no object is purged.
**User response:** Datasets that are to be handled by Tivoli OPC Hiperbatch 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 dataset name passed has an invalid length.
**Explanation:** The dataset 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 OPC 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 dataset is neither substituted nor written to the internal reader.
**User response:** Verify that EQQPROC has been started by Tivoli OPC.

**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 dataset 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 dataset.
**Action:** EQQPURGE terminates; no JCL is processed.
**User response:** Verify that the JCLOUT statement is present and correctly specified in EQQPROC.
**EQQRCSIM—Simulate Return Codes**

Use EQQRCSIM to simulate return codes with step-level restart.

When return code simulation is requested, Tivoli OPC adds a step that refers to the cataloged procedure EQQRCSIM to the JCL. EQQRCSIM invokes a return code simulation program EQQSWAUP that must be stored in an APF authorized library.

An example of a return code simulation cataloged procedure is:

```
//RCSIM EXEC PGM=EQQSWAUP
//STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//SYSUDUMP DD SYSOUT=/c5197
```

**SYSIN Requirements**

Normally, the SYSIN DD data for EQQRCSIM is supplied from the Tivoli OPC dialog. However, if edited manually, the layout of the SYSIN DD statement is:

```
COLUMNS 1 2 3
123456789012345678901234567890
NNNNNNNP PPPPPPPPPP CCCC LINE 1 (REQUIRED)
nnnnnnnppppppppcccc LINES 2-256 (OPTIONAL)
where:
NNNNNNNN = Name of first restart step
PPPPPPPPP = Procedure name of first restart step, or blank
CCCC = Return code input to first step, or blank
nnnnnnnn = Name of simulated return code step
pppppppp = Procedure name of simulated return code step, or blank
cccc = Simulated return code, or blank
```

If the SYSIN file is incorrect, the added step abends with one of these abend codes:

<table>
<thead>
<tr>
<th>Abend code</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>U0001</td>
<td>Unable to open the SYSIN file.</td>
</tr>
<tr>
<td>U0002</td>
<td>The SYSIN file defines more than 256 simulated return codes.</td>
</tr>
<tr>
<td>U0003</td>
<td>The SYSIN file refers to a step that is not defined by the restarted job.</td>
</tr>
</tbody>
</table>
JCL Example

```c
//***************************************************************
//* SIMULATE RETURN CODES
//***************************************************************
//RCSTEP EXEC EQQRCSIM
//SYSIN DD *
PAYTAX1A 0002
*/
```

**EQQWSWRT—Print All Workstation Descriptions**

The EQQWSWRT batch program provides a printout of all workstation descriptions from the workstation description database.

**SYSIN Requirements**

None.

**JCL Example**

```c
//***************************************************************
//* PRINT WORK STATION DESCRIPTION
//***************************************************************
//PRWORK EXEC PGM=EQQBATCH,PARM='EQQWSWRT',REGION=2048K
//STELIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
//EQQMLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
//EQQPPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
//WSREPORT DD SYSOUT=*,
//DCB=(RECFM=FBA,LRECL=121,BLKSIZE=6050)
//EQQMLOG DD SYSOUT=
//SYSPRINT DD SYSOUT=
//SYSDUMP DD SYSOUT=
//EQQDSM DD SYSOUT=
//EQQWSDS DD DSN=EID.EIDAR3.DE,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 127.
Appendix E. Tivoli OPC Report Examples

This appendix provides examples of the following reports:

- Calendars
- Periods
- Workstation descriptions
- Application descriptions
- JCL variable tables
- Mass update
- Operator instructions
- LTP
- Daily planning

A description of the batch programs and examples of the JCL are provided in the Appendix D, “Batch Programs” on page 351.

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

**Figure 133. Calendars—General Information**

```
CALENDAR ID : DEFAULT
CALENDAR DESCRIPTION : Default calendar
WORK DAY END TIME : 01.00
START DATE OF PRINTOUT: 95/02/14
END DATE OF PRINTOUT: 95/12/31
LATEST USER UPDATE : 95/02/13
UPDATED BY : KATARIN
DEFAULT STATUS : MONDAY TUESDAY WEDNESDAY THURSDAY FRIDAY SATURDAY SUNDAY
            WORK WORK WORK WORK WORK FREE FREE
```

**Figure 134. Calendars—Description of Specific Dates**

```
DATE       STATUS COMMENTS
--------- ------ ------------------------------
95/04/14   FREE  Good Friday
95/12/24   FREE  Christmas Eve
95/12/25   FREE  Christmas Day
95/12/26   FREE  Boxing Day
96/01/01   FREE  New Year's Day
```

**Figure 135. Calendars—Status of Days**

```
STATUS OF DAYS WITHIN THE PRINTOUT PERIOD
===========================================

MONTH      MONDAY      TUESDAY      WEDNESDAY    THURSDAY    FRIDAY      SATURDAY     SUNDAY
--------  -----------  -----------  -----------  -----------  -----------  -----------  -----------
02  95/02/14 W   95/02/15 W   95/02/16 W   95/02/17 W   95/02/18 F   95/02/19 F
02  95/02/20 W   95/02/21 W   95/02/22 W   95/02/23 W   95/02/24 W   95/02/25 F   95/02/26 F
02/03  95/02/27 W   95/02/28 W   95/03/01 W   95/03/02 W   95/03/03 W   95/03/04 F   95/03/05 F
03  95/03/06 W   95/03/07 W   95/03/08 W   95/03/09 W   95/03/10 W   95/03/11 F   95/03/12 F
03  95/03/13 W   95/03/14 W   95/03/15 W   95/03/16 W   95/03/17 W   95/03/18 F   95/03/19 F
03  95/03/20 W   95/03/21 W   95/03/22 W   95/03/23 W   95/03/24 W   95/03/25 F   95/03/26 F
03/04  95/03/27 W   95/03/28 W   95/03/29 W   95/03/30 W   95/03/31 W   95/04/01 F   95/04/02 F
04  95/04/03 W   95/04/04 W   95/04/05 W   95/04/06 W   95/04/07 W   95/04/08 F   95/04/09 F
04  95/04/10 W   95/04/11 W   95/04/12 W   95/04/13 W   95/04/14 F   95/04/15 F   95/04/16 F
04  95/04/17 W   95/04/18 W   95/04/19 W   95/04/20 W   95/04/21 W   95/04/22 F   95/04/23 F
```

END OF CALENDAR PRINTOUT

© 2023 IBM Corporation. All rights reserved.
Period Reports

GENERAL INFORMATION

START DATE OF PRINTOUT: 95/02/14
END DATE OF PRINTOUT: 96/12/31
CALENDAR ID: DEFAULT
CALENDAR DESCRIPTION: Default calendar

Figure 136. Description of Period Characteristics—General Information

DESCRIPTION OF PERIOD CHARACTERISTICS

<table>
<thead>
<tr>
<th>PERIOD</th>
<th>DESCRIPTION</th>
<th>TYPE</th>
<th>INTERVAL ORIGIN</th>
<th>INTERVAL END</th>
<th>INTERVAL ORIGIN</th>
<th>INTERVAL END</th>
</tr>
</thead>
<tbody>
<tr>
<td>FIRSTTHU</td>
<td>First Thursday in each month</td>
<td>NONCYCLIC</td>
<td>95/01/05 005 THU</td>
<td>95/02/02 033 THU</td>
<td>95/03/02 061 THU</td>
<td>95/04/06 096 THU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>95/05/04 124 THU</td>
<td>95/06/01 152 THU</td>
<td>95/07/06 187 THU</td>
<td>95/08/03 215 THU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>95/09/07 250 THU</td>
<td>95/10/05 278 THU</td>
<td>95/11/02 306 THU</td>
<td>95/12/07 341 THU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>96/01/04 004 THU</td>
<td>96/02/01 032 THU</td>
<td>96/03/07 067 THU</td>
<td>96/04/07 098 SUN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>96/05/02 123 THU</td>
<td>96/06/06 158 THU</td>
<td>96/07/04 186 THU</td>
<td>96/08/01 214 THU</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>96/09/05 249 THU</td>
<td>96/10/03 277 THU</td>
<td>96/11/07 312 THU</td>
<td>96/12/05 340 THU</td>
</tr>
<tr>
<td>FSTTHU7</td>
<td>First Thursday in July</td>
<td>NONCYCLIC</td>
<td>95/07/06 187 THU</td>
<td>96/07/04 186 THU</td>
<td>97/07/03 184 THU</td>
<td></td>
</tr>
</tbody>
</table>

Figure 137. Description of Period Characteristics
Workstation Description Reports

Figure 138. Workstation Description Report

<table>
<thead>
<tr>
<th>Date</th>
<th>Comments</th>
<th>Work Station Closed</th>
<th>Last Updated</th>
</tr>
</thead>
<tbody>
<tr>
<td>95/09/16</td>
<td>Night shift operation only</td>
<td>08.00 - 24.00</td>
<td>ANDERSM 91/04/21 15.21</td>
</tr>
<tr>
<td>95/09/17</td>
<td>Night shift operation only</td>
<td>08.00 - 24.00</td>
<td>ANDERSM 91/04/21 15.21</td>
</tr>
<tr>
<td>95/10/15</td>
<td>Hardware upgrade shutdown</td>
<td>00.00 - 24.00</td>
<td>ANDERSM 91/04/21 15.21</td>
</tr>
</tbody>
</table>

Figure 139. All Workstations Closed Report
### Application Description Reports

**REPORT TYPE:** CROSS-REFERENCE OF JOB NAMES 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 WTO1_030</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>PAYDAILY</td>
<td>71/12/31</td>
<td>CPU1_020 SETP_010 WTO1_005</td>
</tr>
<tr>
<td>PAYMONTH</td>
<td>PAYMONTH</td>
<td>71/12/31</td>
<td>CPU1_040</td>
</tr>
<tr>
<td>PAYMSLIP</td>
<td>PAYMSLIP</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 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 WTO1_030</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>71/12/31 PAYDAILY</td>
<td>CPU1_020 SETP_010 WTO1_005</td>
</tr>
<tr>
<td>PAYM1</td>
<td>71/12/31 PAYM1</td>
<td>CPU1_040</td>
</tr>
<tr>
<td></td>
<td>PAYM2</td>
<td>CPU1_040</td>
</tr>
<tr>
<td>PAYQUERY</td>
<td>71/12/31 PAYQUERY</td>
<td>CPU1_050</td>
</tr>
<tr>
<td>PAYRECOV</td>
<td>71/12/31 PAYRECOV</td>
<td>CPU1_015</td>
</tr>
<tr>
<td>PAYTAXYR</td>
<td>71/12/31 PAYTAXYR</td>
<td>CPU1_015</td>
</tr>
<tr>
<td>PAYW</td>
<td>71/12/31 PAYW</td>
<td>CPU1_020</td>
</tr>
<tr>
<td></td>
<td>PAYWSLIP CPU1_030</td>
<td>CPU1_095 PRT1_090</td>
</tr>
</tbody>
</table>

Figure 140. Cross-References of Job Names and Active Applications
### PRINTOUT OF APPLICATION DESCRIPTIONS

**REPORT TYPE:** CROSS-REFERENCE OF APPLICATIONS AND EXTERNAL DEPENDENCIES

<table>
<thead>
<tr>
<th>APPL ID</th>
<th>S</th>
<th>OP ID</th>
<th>JOBNAME</th>
<th>PREDECESSORS</th>
<th>SUCCESSORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYBACKP</td>
<td>A</td>
<td>CPU1_015</td>
<td>PAYBACKP</td>
<td>PAYDAILY CPU1_020 PAYDAILY</td>
<td>CPU1_015 PAYBACKP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PAYM1 CPU1_050 PAYMSLIP</td>
<td>CPU1_040 PAYTRANS</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PAY2 CPU1_040 PAYTRANS</td>
<td>CPU1_015 PAYTAXYR</td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>A</td>
<td>CPU1_020</td>
<td>PAYDAILY</td>
<td>CPU1_020 PAYDAILY</td>
<td>CPU1_015 PAYBACKP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A PAYM1 CPU1_040 PAYMONTH</td>
<td>CPU1_020 PAYWEEK</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A PAYW</td>
<td></td>
</tr>
<tr>
<td>PAYM1</td>
<td>A</td>
<td>CPU1_040</td>
<td>PAYMONTH</td>
<td>CPU1_020 PAYDAILY</td>
<td>CPU1_040 PAYTRANS</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU1_050</td>
<td>PAYMSLIP</td>
<td>A PAYM2 CPU1_040 PAYMONTH</td>
<td>CPU1_015 PAYBACKP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>CPU1_040</td>
<td>PAYTRANS</td>
<td>A PAYBACKP CPU1_015 PAYTAXYR</td>
<td>CPU1_015 PAYTAXYR</td>
</tr>
<tr>
<td>PAYM2</td>
<td>A</td>
<td>CPU1_040</td>
<td>PAYTRANS</td>
<td>PAYM1 CPU1_040 PAYMONTH</td>
<td>CPU1_015 PAYBACKP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>PAYW CPU1_020 PAYWEEK</td>
<td>CPU1_015 PAYTAXYR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A PAYBACKP CPU1_015 PAYBACKP</td>
<td></td>
</tr>
<tr>
<td>PAYTAXYR</td>
<td>A</td>
<td>CPU1_015</td>
<td>PAYTAXYR</td>
<td>PAYM2 CPU1_040 PAYTRANS</td>
<td>CPU1_015 PAYBACKP</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A PAYBACKP CPU1_015 PAYBACKP</td>
<td></td>
</tr>
<tr>
<td>PAYW</td>
<td>A</td>
<td>CPU1_020</td>
<td>PAYWEEK</td>
<td>PAYDAILY CPU1_020 PAYDAILY</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A PAYM2 CPU1_015 PAYBACKP</td>
<td></td>
</tr>
</tbody>
</table>

>>>>>> END OF APPLICATION DESCRIPTION PRINTOUT <<<<<<<

**Figure 141. Cross-Reference of Applications and External Dependencies**
### Application Descriptions—Common Data

**Priority (PRI):** 1=low, 8=high, 9=urgent

**Run cycle type (RC TYPE):** normal (times and days when run) or negative (times and days when not to run).

---

<table>
<thead>
<tr>
<th>APPL TEXT/OWNER ID</th>
<th>CALENDAR ID/OWNER DESCRIPTION</th>
<th>TYPE Status</th>
<th>PRI RUNC OPER GROUP ID</th>
<th>GROUP DEFINITION</th>
<th>LATEST UPDATE USER</th>
</tr>
</thead>
<tbody>
<tr>
<td>daily payroll jobs</td>
<td>PAYDAILY Sample payroll application</td>
<td>APL 95/01/29 71/12/31</td>
<td>ACTIVE 5 1 3</td>
<td>94/06/08</td>
<td>XRAYNER</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>VALID PERIOD RC</th>
<th>INPUT RC VALID FREE DAY RUN CYCLE DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>95/01/29 DAILY</td>
<td>12.00 16.00 95/01/29 71/12/30 CANCEL</td>
</tr>
</tbody>
</table>
### OPERATION DATA

<table>
<thead>
<tr>
<th>OPERATION NAME</th>
<th>NO.</th>
<th>TEXT</th>
<th>NUMBER</th>
<th>DUR PS</th>
<th>R1</th>
<th>R2</th>
<th>RES</th>
<th>SPC</th>
<th>HRC</th>
<th>FAC</th>
<th>LIM</th>
<th>JOBNAME</th>
<th>FORM</th>
<th>A</th>
<th>H</th>
<th>T</th>
<th>S</th>
<th>E</th>
<th>D</th>
<th>R</th>
<th>C</th>
<th>ARRIVAL</th>
<th>DEADLINE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTO_005 PAYX</td>
<td>00.01</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>PAYDAILY</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>PAYDAILY</td>
<td>X</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>WTO_005</td>
<td>PAYDAILY</td>
</tr>
<tr>
<td>SETP_010 Job</td>
<td>00.03</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>PAYDAILY</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>PAYDAILY</td>
<td>X</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>SETP_010</td>
<td>PAYDAILY</td>
</tr>
<tr>
<td>CPU_020 Runs</td>
<td>00.05</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>PAYDAILY</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>PAYDAILY</td>
<td>X</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>N</td>
<td>N</td>
<td>CPU_020</td>
<td>PAYDAILY</td>
</tr>
</tbody>
</table>

**Figure 143. Application Descriptions—Operation Data**

**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** Catalog management, Y N or D=deferred.

### INTERNAL OPERATION LOGIC

<table>
<thead>
<tr>
<th>LTP PRINT</th>
<th>REASON FOR DEPENDENCY</th>
<th>TRSP INT &amp; EXT</th>
<th>TIME</th>
<th>PREDECESSOR</th>
<th>OPERATION</th>
<th>INTERNAL SUCCESSOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>WTO_005</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>WTO_005</td>
<td>SETP_010 CPU_020</td>
</tr>
<tr>
<td>SETP_010</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SETP_010</td>
<td>CPU_020</td>
</tr>
<tr>
<td>CPU_020</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>CPU_020</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 144. 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>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 0 0 2</td>
<td>0.05</td>
<td>APP1</td>
<td>95/03/03</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>job2</td>
<td>1 0 0 0</td>
<td>0.05</td>
<td>APP2</td>
<td>95/03/03</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>job3</td>
<td>1 0 0 2</td>
<td>0.05</td>
<td>APP3</td>
<td>95/03/03</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>job4 in app1</td>
<td>1 0 0 2</td>
<td>0.05</td>
<td>APP4</td>
<td>95/05/05</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>job5 in app1</td>
<td>1 0 0 2</td>
<td>0.05</td>
<td>APP5</td>
<td>95/05/05</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>job6 in app1</td>
<td>1 0 0 2</td>
<td>0.05</td>
<td>APP6</td>
<td>95/05/06</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Daily payroll backup</td>
<td>1 0 0 2</td>
<td>0.05</td>
<td>BACKUP</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Monthly payroll job</td>
<td>1 0 0 1</td>
<td>0.05</td>
<td>PAYM1</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Print monthly pay slips</td>
<td>1 0 0 0</td>
<td>0.05</td>
<td>PAYM2</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Create bank giro tape</td>
<td>1 0 0 1</td>
<td>0.10</td>
<td>PAYM3</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Run as required</td>
<td>1 0 0 1</td>
<td>0.05</td>
<td>PAYQUERY</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Recover payroll database</td>
<td>1 0 0 2</td>
<td>0.05</td>
<td>PAYRECOV</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>Third Thursday in July</td>
<td>1 0 0 1</td>
<td>0.05</td>
<td>PAYTAXYR</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>pay07, pay10, and pay16</td>
<td>1 0 0 1</td>
<td>0.05</td>
<td>PAYW</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>pay14, pay15</td>
<td>1 0 0 0</td>
<td>0.05</td>
<td>PAYW</td>
<td>95/01/28</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
<tr>
<td>CPU1_050</td>
<td>pay16</td>
<td>1 0 0 1</td>
<td>0.20</td>
<td>TEST3</td>
<td>95/05/01</td>
<td>71/12/31</td>
<td>ACTIVE</td>
<td></td>
</tr>
</tbody>
</table>

---

**Figure 145. 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.
# Application Descriptions—Cross-Reference

Run the EQQAXR00 program to produce the cross-reference list, selecting up to five fields for the sort sequence. Using the AD database dialog, select 6 (XRF OF ITEMS) from the PRINTING APPLICATIONS panel.
<table>
<thead>
<tr>
<th>JCL VARIABLE TABLE ID : PAY</th>
<th>OWNERID : SAMPLE</th>
</tr>
</thead>
<tbody>
<tr>
<td>VARIABLE NAME</td>
<td></td>
</tr>
<tr>
<td>DEPT</td>
<td></td>
</tr>
<tr>
<td>Description :</td>
<td></td>
</tr>
<tr>
<td>Default Value : EXTRA</td>
<td></td>
</tr>
<tr>
<td>Upper Case : N</td>
<td></td>
</tr>
<tr>
<td>Setup : N</td>
<td></td>
</tr>
<tr>
<td>Substitution</td>
<td></td>
</tr>
<tr>
<td>Exit :</td>
<td></td>
</tr>
<tr>
<td>Value Required : N</td>
<td></td>
</tr>
<tr>
<td>Default Pos : 00</td>
<td></td>
</tr>
<tr>
<td>Dialog Text :</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VALIDATION CRITERIA</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Verification Type :</td>
<td></td>
</tr>
<tr>
<td>Variable Length : 00</td>
<td></td>
</tr>
<tr>
<td>Numeric :</td>
<td></td>
</tr>
<tr>
<td>Comparison Oper :</td>
<td></td>
</tr>
<tr>
<td>Validation Pattern :</td>
<td></td>
</tr>
<tr>
<td>Valid Ranges/Values :</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>VARIABLE DEPENDENCIES</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Independent Sets CURRENT VARIABLE VALUE TO NONE</td>
<td></td>
</tr>
</tbody>
</table>

Figure 147. 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

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: Reroutable
   C - 27: Catalog Management

D. Internal Predecessor Fields
   D - 1: Predecessor Op Number
   D - 2: Predecessor WS Name
   D - 3: Transport Time

E. External Predecessor Fields
   E - 1: Predecessor Application ID
   E - 2: Predecessor Op Number
   E - 3: Predecessor WS Name
   E - 4: Transport Time
   E - 5: External Dependency Text

Figure 148. Mass Update Report—Application Descriptions

Update Field: C - 27

Update Performed for Owner ID: SAMPLE, Workstation Name: CPU1

-figure-149-

Figure 149. Mass Update Report—Catalog Management Flag Updated
REPORTS FOR SUBSYS EIDA

REPORT PRODUCED BY : PRINT OPERATOR INSTRUCTIONS IN AD ORDER
DATE OF REPORT : 95/05/3

---

APPL ID | OP NO | VALID FROM | VALID TO | WSID | LAST UPDATE
--- | --- | --- | --- | --- | ---
PAYBACKP | 015 | 71/12/31 24.00 CPU1 | | | 95/02/01 09.13 XRAYNER

---

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.

---

APPL ID | OP NO | VALID FROM | VALID TO | WSID | LAST UPDATE
--- | --- | --- | --- | --- | ---
PAYDAILY | 010 | 71/12/31 24.00 SETP | | | 95/01/30 18.36 XRAYNER

---

THIS JOB CAN RUN ONLY WHEN THE RESOURCE 'PAYROLL.DATABASE' IS FREED BY CICSA.
TO DO THIS, RUN THE CICS TRANSACTION PAYC.

---

APPL ID | OP NO | VALID FROM | VALID TO | WSID | LAST UPDATE
--- | --- | --- | --- | --- | ---
PAYDAILY | 020 | 71/12/31 24.00 CPU1 | | | 95/01/30 18.43 XRAYNER

---

THE PAYROLL DATABASE IF THERE ARE NO ERRORS.
RECOVERY PROCEDURE

---

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 150. Operator Instructions Report
## LTP Reports

The following data appears on the heading page:

<table>
<thead>
<tr>
<th><strong>Data</strong></th>
<th><strong>Explanation</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Report produced by</td>
<td>LTP 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 LTP was last updated</td>
</tr>
<tr>
<td>The LTP covers period</td>
<td>Start day and end day selected in LTP dialog</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>
<tr>
<td>APPL ID</td>
<td>INPUT</td>
</tr>
<tr>
<td>---------</td>
<td>--------</td>
</tr>
<tr>
<td></td>
<td>OWNER ID/OP ID</td>
</tr>
<tr>
<td>CP</td>
<td>SAMPLE3</td>
</tr>
<tr>
<td>LTP</td>
<td>SAMPLE3</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYBACKP</td>
<td>SAMPLE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>SAMPLE</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 152. LTP Report for Applications, Sorted by Run Date
## LTP Report for Applications, Sorted by Owner

This data appears on an LTP full report:

### Data

<table>
<thead>
<tr>
<th>Field</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Appl ID</td>
<td>Application ID of the occurrence</td>
</tr>
<tr>
<td>Owner ID/op ID</td>
<td>Application owner ID and the operation ID of any operations that have changed operation date on the LTP file</td>
</tr>
<tr>
<td>Input arr</td>
<td>Application input arrival time</td>
</tr>
<tr>
<td>Deadline</td>
<td>Application deadline date and time</td>
</tr>
<tr>
<td>PI</td>
<td>Application priority</td>
</tr>
<tr>
<td>Appl/operation text</td>
<td>Application or operation text</td>
</tr>
<tr>
<td>+ LTP Comment</td>
<td>Tivoli OPC-generated comments</td>
</tr>
<tr>
<td>+ Var table</td>
<td>Variable tables</td>
</tr>
<tr>
<td>Depend. op ID</td>
<td>ID of operation with external dependency</td>
</tr>
<tr>
<td>Depending occurrences</td>
<td>All external predecessors and successors listed (for reports on only one application, external dependencies not listed)</td>
</tr>
<tr>
<td>Type, appl ID</td>
<td>PRED or SUCC Application ID of dependent occurrence</td>
</tr>
<tr>
<td>Op ID, input arrival</td>
<td>ID of dependent operation occurrence input interval.</td>
</tr>
</tbody>
</table>

---

### Long Term Plan

#### Appl ID/Op ID

<table>
<thead>
<tr>
<th>Input Arrival</th>
<th>Deadline</th>
<th>Appl/Operation Text</th>
<th>Depend. Op ID</th>
<th>Depending Occurrences</th>
</tr>
</thead>
<tbody>
<tr>
<td>95/05/04 12.00</td>
<td>95/05/05 06.00</td>
<td>backup payroll database CPU1_015 PRED PAYDAILY CPU1_020 95/05/04 12.00</td>
<td>wait for the daily job and other payroll jobs</td>
<td></td>
</tr>
<tr>
<td>95/05/05 12.00</td>
<td>95/05/05 16.00</td>
<td>daily payroll jobs CPU1_020 SUCC PAYBACK CPU1_015 95/05/04 12.00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>95/05/05 12.00</td>
<td>95/05/06 06.00</td>
<td>backup payroll database CPU1_015 PRED PAYDAILY CPU1_020 95/05/05 12.00</td>
<td>wait for the daily job and other payroll jobs</td>
<td></td>
</tr>
<tr>
<td>95/05/05 12.00</td>
<td>95/05/05 16.00</td>
<td>daily payroll jobs CPU1_020 SUCC PAYBACK CPU1_015 95/05/05 12.00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 153.** LTP Report for Applications, Sorted by Owner
TOTAL DURATION PER WORKSTATION

WS DURATION OPS
--- ----- ----- 
CPU1 6.10 4
SETP 0.03 1
WTO1 0.02 2

Figure 154. LTP 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 ops at the workstation
Number of operations at the workstation

Figure 155. LTP Report—Grand Total Workload for Period
# Daily Planning Reports

## General Information

<table>
<thead>
<tr>
<th>Requested Planning Period</th>
<th>95/05/06 09.06 - 95/05/06 24.00</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plan Covers</td>
<td>95/05/06 09.06 - 95/05/13 24.00</td>
</tr>
<tr>
<td>Type of Planning</td>
<td>PLAN NEXT PERIOD</td>
</tr>
<tr>
<td>Long-Term Plan Used</td>
<td>95/05/04</td>
</tr>
<tr>
<td>Old Current Plan Used</td>
<td>NO</td>
</tr>
<tr>
<td>New Current Plan Created</td>
<td>YES</td>
</tr>
<tr>
<td>Workstation Summary</td>
<td>YES</td>
</tr>
<tr>
<td>Daily Operating Plan</td>
<td>YES</td>
</tr>
<tr>
<td>Workstation Plans</td>
<td>YES</td>
</tr>
<tr>
<td>Input Arrival Lists</td>
<td>YES</td>
</tr>
<tr>
<td>Non-Reporting Workstation Plans Only</td>
<td>YES</td>
</tr>
<tr>
<td>Current Period Results</td>
<td>NO</td>
</tr>
<tr>
<td>Previous Period Results</td>
<td>NO</td>
</tr>
<tr>
<td>Missed Feedback Report</td>
<td>YES</td>
</tr>
<tr>
<td>Planned Resource Utilization</td>
<td>YES</td>
</tr>
<tr>
<td>Actual Resource Utilization</td>
<td>YES</td>
</tr>
<tr>
<td>Number of Planned Applications</td>
<td>3</td>
</tr>
<tr>
<td>Number of Planned Operations</td>
<td>6</td>
</tr>
</tbody>
</table>

**Figure 156. Daily Planning Reports—General Information**

The following data appears on the general information page:

<table>
<thead>
<tr>
<th>Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requested planning period</td>
<td>Start and end times as specified in dialog.</td>
</tr>
<tr>
<td>Plan covers</td>
<td>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.</td>
</tr>
<tr>
<td>Type of planning</td>
<td>Name of daily planning function used to produce this set of plans.</td>
</tr>
<tr>
<td>Long-term plan used</td>
<td>Time the LTP was last updated; appears only if the LTP was used as input to this daily planning run.</td>
</tr>
<tr>
<td>Old current plan used</td>
<td>Time the last current plan backup was taken; appears only if the current plan was used in this planning run.</td>
</tr>
<tr>
<td>New current plan created</td>
<td>YES if new current plan created; NO if no plan was created.</td>
</tr>
<tr>
<td>Workstation summary</td>
<td>Report produced, YES or NO.</td>
</tr>
<tr>
<td>Daily operating plan</td>
<td>Report produced, YES or NO.</td>
</tr>
<tr>
<td>Workstation plans</td>
<td>Report produced, YES or NO.</td>
</tr>
<tr>
<td>Input arrival lists</td>
<td>Report produced, YES or NO.</td>
</tr>
</tbody>
</table>
Nonreporting workstation plans only  Report produced, YES or NO.
Current period results  Start and end time for period covered; NO if no report produced.
Previous period results  Start and end time for period covered; NO if no report produced.
Missed feedback report  Report produced, YES or NO.
Planned resource utilization  Report produced, YES or NO.
Actual resource utilization  Report produced, YES or NO.
Number of planned applications  Number of occurrences in daily plan.
Number of planned operations  Number of operations in daily plan.
Number of messages  Number of error, warning, and information messages.

<table>
<thead>
<tr>
<th>APPLICATION ID</th>
<th>APPLICATION OWNER</th>
<th>OP ID</th>
<th>JOBNAME</th>
<th>INPUT ARRIVAL</th>
<th>DEFINED START</th>
<th>DUR</th>
<th>DEADLINE</th>
<th>APPLICATION TEXT</th>
<th>SPEC</th>
<th>RESOURCES</th>
</tr>
</thead>
<tbody>
<tr>
<td>PAYBACKP</td>
<td>5 W</td>
<td>CPU1_015 PAYBACKP</td>
<td>12.00</td>
<td>09 06.00</td>
<td>Backup payroll database</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PAYDAILY</td>
<td>5 W</td>
<td>W10_005 PAYDAILY</td>
<td>12.00</td>
<td>00 01</td>
<td>Daily payroll jobs</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CPU1_020 PAYDAILY</td>
<td>12.04 00.05 16.00</td>
<td>12.04 00.05 16.00</td>
<td>Runs pay04 and pay06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 157. Daily Planning Reports—Daily Operating Plan

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**

- **Application ID**: Application name
- **Application owner**: Owner ID of the application
- **Operation ID**: Operation number and workstation name
- **P**: Application priority
- **S**: Application status
- **Input arrival Start**: Application input time
- **Dur**: Estimated duration of the operation; blank if operation is complete
Report Examples

Defined deadline  Application or operation deadline
Application text  Application descriptive text
Operation text    Operation text
Predecessors     Internal and external predecessors
Application ID Op ID Predecessor application ID: blank if internal predecessor; operation ID: blank if the predecessor occurrence starts after the period covered by this plan
Special resources * indicates that this operation specifies special resources.

Figure 158. Daily Planning Reports—Plan for Workstation

The report in Figure 158 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.

On the PLAN FOR WORKSTATION report, Tivoli OPC 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 OPC 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 159. Daily Planning Reports—Workstation Utilization (Parallel Operations)
Figure 160. Daily Planning Reports—Workstation Utilization (Resource 1)

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.
Figure 161. Daily Planning Reports—Planned Resource Utilization

You see this data on the planned resource utilization report:

**Data Explanation**

- **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.

<table>
<thead>
<tr>
<th>DATE</th>
<th>TIME</th>
<th>INTERVAL START</th>
<th>INTERVAL END</th>
<th>AVAILABLE</th>
<th>QUANTITY</th>
<th>PLANNED</th>
<th>ALLOCATION FAILURES BY</th>
<th>NUMBER OF CONTENTIONS</th>
</tr>
</thead>
<tbody>
<tr>
<td>95/08/07</td>
<td>8.00</td>
<td>95/08/07 10.00</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>95/08/07</td>
<td>10.00</td>
<td>95/08/07 12.30</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>95/08/07</td>
<td>12.30</td>
<td>95/08/07 20.00</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>95/08/07</td>
<td>20.00</td>
<td>95/08/08 08.00</td>
<td>YES</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
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 may not cover a full previous 24 hours when several daily plans have been executed within a planning period.

<table>
<thead>
<tr>
<th>APPL PRIORITY</th>
<th>NO OF COMPLETED APPL</th>
<th>NO OF RERUN APPL</th>
<th>NO OF DELETED APPL</th>
<th>AVERAGE INPUT DELAY</th>
<th>AVERAGE MISSING DEADLINE</th>
<th>AVERAGE EARLY COMPLETED</th>
<th>AVERAGE DEADLINE EARLINESS</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>19.54</td>
<td>1</td>
<td>15.55</td>
<td>1</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 162. Daily Planning Reports—Summary of Completed Applications

The report shown in Figure 162 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.
The report shown in Figure 163 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.

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 may 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.
### Figure 164. Daily Planning Reports—Error Statistics on Completed Applications

The report shown in Figure 164 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.
Figure 165. Daily Planning Reports—Operations in Error

The report shown in Figure 165 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 may 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.

Figure 166. Daily Planning Reports—Missed Feedback Report

The report shown in Figure 166 is printed during a Plan Next Period or Replan Current Period run if OPC has not been able to feedback application durations.

The report lists all operations where the feedback of the actual duration to the application description data set has not been possible.

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 op-ID printed) or operation not found in AD.
Report Examples

ABCD COMPANY PAGE /zerodot/zerodot38
REPORTS FOR SUBSYS EIDA /zerodot3 JUN 95
/zerodot6:/zerodot3

ACTUAL RESOURCE UTILIZATION (95/06/01 20.00 - 95/06/02 20.00)
===================================================================================================================
RESOURCES NAME : PAYROLL.DATABASE
DESCRIPTION: Serializes access to Paymore database
DYNAMICALLY ADDED: NO
INTerval START | INTERVAL END | AVAILABILITY | UTILIZATION | ALLOCATION | CONTENTION | IDLE
DATE TIME | DATE TIME | PLANNED | ACTUAL | PLANNED MAX | ACTUAL MAX | FAILURES | TIME | TIME
===================================================================================================================
95/06/01 20.00 | 95/06/02 08.00 | YES | 100% | 1 | 1 | 0 | 0% | 4%
95/06/02 08.00 | 95/06/02 10.00 | YES | 100% | 0 | 1 | 0 | 0% | 80%
95/06/02 10.00 | 95/06/02 12.30 | YES | 100% | 1 | 1 | 1 | 12% | 17%
95/06/02 12.30 | 95/06/02 20.00 | YES | 100% | 1 | 1 | 0 | 0% | 20%

Figure 167. Daily Planning Reports—Actual Resource Utilization

You see this data on the actual resource utilization report:

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 incremented each time Tivoli OPC tries unsuccessfully to allocate the resource to an operation during this interval—it can be incremented 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.
A

ABARS. See Aggregate Backup and Recovery Support.

active application description. An application description that is complete and ready for use in planning or scheduling.

actual duration. At a workstation, the actual time in hours and minutes it takes to process an operation from start to finish.

adjusted quantity. The current quantity of a special resource, taking the deviation into account.

AD. See application description.

Aggregate Backup and Recovery Support (ABARS). A DFHSM facility that manages backup and recovery of user-defined data set groups (aggregates). Aggregate backup copies and related control information are written as portable data and control files on 3480 or 3420 volumes.

Advanced Program-to-Program Communications (APPC). An implementation of the Systems Network Architecture (SNA), logical unit (LU) 6.2 protocol that allows interconnected systems to communicate and share the processing of programs.

all-days cyclic period. A cyclic period where all days are counted when calculating the interval.

alert. Two Workload Monitor/2 objects, Operations List and Workstations List, can be used to monitor a Tivoli OPC subsystem and notify you if alert conditions are met. The alert can be a sound (Beep), or a message in a window (Message). The Details view of the Plan object must be open to monitor for plan alerts. The List or Icons views of the Operations List object must be open to monitor for operation alerts.

APAR. Authorized program analysis report. A report of a problem that is suspected to be caused by a defect in a current, unaltered release of a program.

API. See application programming interface.

APPC. See Advanced Program-to-Program Communications.

application. A measurable and controllable unit of work that completes a specific user task, such as the running of payroll or financial statements. The smallest entity that an application can be broken down into is an operation. Generally, several related operations make up an application.

application description (AD). A database description of an application.

application group. Type of application description which holds run cycle and calendar information for standard applications or job descriptions which have been defined as a member of the group.

application ID. The name of an application. (For example, PAYROLL or DAILYJOBS.)

application programming interface (API). A formally-defined programming language interface between an IBM system control program or a licensed program and the user of a program.

application transaction program (ATP). A program that uses the Advanced Program-to-Program Communications (APPC) application programming interface (API) to communicate with a partner program at a remote node.

application version. See versions.

ATP. See application transaction program.

authority. The ability to access a protected resource.

authority group. A name used to generate a RACF resource name for authority checking.

automatic events. Events recognized by or triggered by an executing program. Automatic events are usually generated by Tivoli OPC tracking programs but can also be created by a user-defined program.

automatic hold/release. Function used to control jobs that are submitted outside Tivoli OPC. It allows you to define whether such jobs should be automatically released at the appropriate time if placed in HOLD status when submitted.

automatic job and started-task recovery. A Tivoli OPC function that lets you specify, in advance, alternative recovery strategies for operations that end in error.

automatic-reporting workstation. A workstation (for example, a processor or printer) that reports events (the starting and stopping of operations) in real time to Tivoli OPC.
availability. The degree to which a system (and in Tivoli OPC, an application) or resource is ready when needed to process data.

B

batch loader. A Tivoli OPC batch program that you can use to create and update information in the application-description and operator-instruction databases.

buffer. A memory area reserved for performing input/output (I/O) operations.

BMP. Batch message processing.

C

calendar. The data that defines the operation department's work time in terms of work days and free days.

capacity. The actual number of parallel servers and workstation resources available during a specified open interval.

capacity ceiling. The maximum number of operations that a workstation can handle simultaneously.

catalog. A directory of files and libraries, with reference to their locations. A catalog may contain other information such as the types of devices in which the files are stored, passwords, blocking factors.

catalog management. Catalog management is a recovery function of Tivoli OPC, which handles the deleting or uncataloging of datasets created in a job operation that ends in error.

cICS. Customer Information Control System.

closed workstation. A workstation that is unavailable to process work for a specific time, day, or period.

Common Programming Interface (CPI). A consistent set of specifications for languages, commands, and calls to enable applications to be developed across all Systems Application Architecture (SAA) environments.

complete (C). The status of an operation indicating that it has finished processing.

completion code. A Tivoli OPC system code that indicates how the processing of an operation ended at a workstation. See error code.

complex of processors. A JES2 Multi-Access Spool system or a JES3 system with more than one processor.

computer workstation. (1) A workstation that performs MVS processing of jobs and started-task operations, and that usually reports status to Tivoli OPC automatically. (2) A processor used as a workstation. It can refer to single processors or multiprocessor complexes serving a single job queue (for example, JES2 or JES3 systems).

contingency plan. A plan for emergency response, backup procedures, and post-disaster recovery. Synonymous with disaster recovery plan, emergency plan.

controller. The Tivoli OPC component that runs on the controlling system, and that contains the Tivoli OPC tasks that manage the Tivoli OPC plans and databases.

controlling system. The system that the controller runs on.

control on servers. If a workstation is defined with control on servers, OPC/ESA will not start more operations at the workstation than there are available servers.

conversation. In Advanced Program-to-Program Communications (APPC), a connection between two transaction programs over a logical unit-logical unit (LU-LU) session that allows them to communicate with each other while processing a transaction.

conversation verb. In Advanced Program-to-Program Communications (APPC), one of the verbs a transaction program issues to perform transactions with a remote program.

CP. See current plan.

CPI. See Common Programming Interface.

CPI-C. Common Programming Interface for Communications. See also Common Programming Interface.

cross-system coupling facility (XCF). MVS components and licensed programs use the XCF services to provide additional functions in a SYSPLEX.

critical path. The route, within a network, with the least slack time.

current plan (CP). A detailed plan of system activity that covers a period of at least 1 minute, and not more than 21 days. A current plan typically covers 1 or 2 days.

cyclic interval. The number of days in a cyclic period.

cyclic period. A period that represents a constant number of days. There are two types of cyclic periods:
- Work-days-only cyclic period, where only the work days are counted when calculating the number of days in the period.
- All-days cyclic period, where all days are counted.

**D**

daily planning. The process of creating a current plan.

DASD. Direct access storage device.

database. A collection of data that is fundamental to a system. Tivoli OPC uses six databases: calendar, period, workstation description, JCL variable table, application description, and operator instruction.

Data Facility Hierarchical Storage Manager (DFHSM). A licensed MVS program which provides automatic and command functions that manage user storage space and data recovery.

Data Facility Systems Management Subsystem/MVS (DFSMS/MVS). A group of licensed MVS programs which transform system environments from user-managed DASD volumes to administrator-controlled, system-managed data sets.

Data Lookaside Facility (DLF). The MVS/ESA component that manages Hiperbatch objects.

data processing center (DP center). A center or department, including computer systems and associated personnel, that performs input, processing, storage, output, and control functions to accomplish a sequence of operations on data.

Data Store. The Tivoli OPC component managing the job runtime information at the tracked system. It is dedicated to the storing and possible retrieval of sysout datasets belonging to OPC-submitted jobs, to optimize the sysout availability.

DB2. DATABASE 2.

DBCS. Double-byte character set.

ddname. Data definition name.

deadline. See deadline date and deadline time.

deadline date. The latest date by which an occurrence must be complete.

deadline time. The latest time by which an occurrence must be complete.

deadline WTO message. You can specify that Tivoli OPC issue an operator message (EQFW776I) when a started operation has not been marked as completed before the deadline time. In addition to the standard message, the user-defined text that describes the operation is issued as part of the WTO.

default calendar. (1) A calendar that you have defined for Tivoli OPC to use when you do not specify a calendar in an application description. (2) A calendar that Tivoli OPC uses if you have neither specified a calendar in an application description, nor defined your own default calendar.

dependency. A relationship between two operations in which the first operation must successfully finish before the second operation can begin.

descriptive text. User-written text describing the operation. This text is also issued as part of the write-to-operator message if the operation has been started, exceeds its deadline, and has the deadline write-to-operator (WTO) option specified.

Details notebook. See Details view.

Details view. A view of a Workload Monitor/2 object showing details about the object. The Details view of the Plan object shows information about the current plan. The Details view of the Operation object shows information about the selected operation. The Details view of the Workstation object shows information about the selected workstation.

deviation. A temporary variation in the quantity of a special resource.

DFHSM. See Data Facility Hierarchical Storage Manager.

DFSMS/MVS. See Data Facility Storage Management Subsystem.

dialog. The user's online interface with Tivoli OPC.

Disaster Recovery Plan (DRP). A plan for emergency response, backup procedures, and post-disaster recovery. Synonymous with contingency plan, emergency plan.

DLF. See Data Lookaside Facility.

DP center. See data processing center.

DRP. See Disaster Recovery Plan.

duration. The length of time an operation is active at a workstation.
end user. A person who uses the services of the data processing center.

ended-in-error (E). The Tivoli OPC reporting status for an operation that has ended in error at a workstation.

error code. A code set by Tivoli OPC to describe how the processing of an operation ended at a computer workstation.

ETT. See event-triggered tracking.

estimated duration. The estimated length of time an operation will use a workstation. This is initially based on a value that is provided when the operation is defined, but can be adjusted automatically by Tivoli OPC’s feedback mechanism to reflect actual durations.

event. An action that changes an operation’s status and changes the current plan.

event manager. The Tivoli OPC function that processes all tracking events and determines which of these are Tivoli OPC-related.

event reader. A Tivoli OPC task that reads event records from an event dataset.

event tracking. A function of Tivoli OPC that follows events in the operations department in real time and records status changes in the current plan.

event-triggered tracking (ETT). A component of Tivoli OPC that waits for specific events to occur, and then adds a predefined application to the current plan. ETT recognizes two types of events: the reader event, which occurs when a job enters the JES reader, and the resource event, which occurs when the availability status of a special resource is set to “yes”.

event writer. A Tivoli OPC task that writes event records in an event dataset.

exclusive resource. A resource that can be used by only one operation at a time.

expected arrival time. The time when an operation is expected to arrive at a workstation. It can be calculated by daily planning or specified in the long-term plan.

extended status code. Together with the normal status codes, Tivoli OPC maintains extended status codes that provide additional information about the status of operations. The extended status code is not always present.

external dependency. A relationship between two occurrences, in which an operation in the first occurrence (the predecessor) must successfully finish before an operation in the second occurrence (the successor) can begin processing.

feedback limit. A numeric value in the range 100–999 that defines the limits within which actual data that is collected in tracking is fed back and used by Tivoli OPC.

filter criteria. Input values that are used to limit the mass update of applications to only those specified. This term is used in the Tivoli OPC ISPF dialogs.

first critical operation. An operation of an occurrence that has the earliest latest-start-time. The first critical operation of an occurrence determines the critical path.

first operation. (1) An operation in an occurrence that has no internal predecessor. (2) The start node in a network.

fixed resources. A set of resource names used to check the authority of users to access the Tivoli OPC dialogs.

form number. A user-defined code that identifies the type of paper to be used for an operation on a printer workstation. Tivoli OPC can use the form number to identify the different print operations belonging to one job.

free day. Any day that is not a work day.

free-day rule. A rule that determines how Tivoli OPC will treat free days when the application run day falls on a free day.

general workstation. A workstation where activities other than printing and processing are carried out. A general workstation reporting to Tivoli OPC is usually manual, but it can also be automatic. Manual activities can include data entry and job setup.

generic alert. An alert that is broadcast by Tivoli OPC, and collected by NetView, when an operation ends in error. You can specify this as an option when defining application descriptions.

global search character. In Tivoli OPC, a percent sign (%), which represents any single character, or an asterisk (*), which represents any character string of any length.

global variable table. The JCL variable table that Tivoli OPC checks for a variable substitution value if no
value is found in the specific JCL variable table that is associated with the operation.

**Graph view.** (1) A view of the Workload Monitor/2 Workstation object. Shows the total number of operations with different statuses for a single workstation. (2) In the Graphical User Interface for Application Description, a view of the operations that make up an application. It shows the workstation where each operation is run, and dependencies between the operations.

**Graphs view.** A view of the Workload Monitor/2 Workstations List object. Shows the total number of operations with different statuses for each of the workstations that are included in the object.

**group definition.** The application group to which the application description or job description is a member.

**H**

**highest return code.** A numeric value in the range 0–4095. If this return code is exceeded during job processing, the job will be reported as ended-in-error.

**Hiperbatch.** The MVS/ESA facility that stores VSAM and QSAM data in Hiperspace for access by multiple jobs. The facility can significantly reduce the execution time of certain batch streams that access VSAM and QSAM data sets.

**Hot standby.** Using the MVS/ESA cross-system coupling facility (XCF), you can include one or more standby controllers in your configuration. A standby system can take over the functions of a controller if the controller fails or if the MVS/ESA system that it was active on fails.

**I**

**Icons view.** The Workload Monitor/2 objects, Workstations List and Operations List, contain other objects. The Icons view shows an icon for each contained object.

**IMS.** Information Management System.

**incident log.** An optional function available under the job completion checker.

**initiator/terminator.** The job scheduler function that selects jobs and job steps to be executed, allocates input/output devices for them, places them under task control, and at completion of the job, supplies control information for writing job output on a system output unit.

**in-progress operation.** An operation with a status of A, R, *, I, E, or S.

**input arrival time (IAT).** The user-defined date and time when an operation or an application is planned to be ready for processing.

**intermediate start.** The date and time an operation started after processing was interrupted.

**internal date.** Internally, Tivoli OPC uses a two-digit year format when handling dates. In order to handle dates before and after 31 December 1999 correctly, Tivoli OPC uses an origin year of 72 for the internal century window. This means that internally the year 1972 is represented as 00 and 2071 is represented as 99.

**internal dependency.** A relationship between two operations within an occurrence, in which the first operation (the predecessor) must successfully finish before the second operation (the successor) can begin.

**interrupted (I).** A Tivoli OPC reporting status for an operation that indicates that the operation has been interrupted while processing.

**ISPF.** Interactive System Productivity Facility.

**J**

**JCC.** See **job completion checker.**

**JCL.** Job control language. A problem-oriented language designed to express statements in a job that are used to identify the job or describe its requirements to an operating system.

**JCL tailoring.** Tivoli OPC provides automatic JCL tailoring facilities, which enable jobs to be automatically edited using information that is provided at job setup or submit.

**JCL variable table.** A group of related JCL variables. See **variable table.**

**JES.** Job entry subsystem. A system facility for spooling, job queuing, and managing I/O.

**job.** (1) A set of data that completely defines a unit of work for a computer. A job usually includes all necessary computer programs, linkages, files, and instructions to the operating system. (2) In Tivoli OPC, an operation performed at a computer workstation.

**job class.** Any one of a number of job categories that can be defined. By classifying jobs and directing initiators to initiate specific classes of jobs, it is possible to control a mixture of jobs that can be run concurrently.
job-completion checker (JCC). An optional function of Tivoli OPC that allows extended checking of the results from CPU operations.

job description. A single processor (job or started-task) operation and its dependencies.

Job Description dialog. The ISPF dialog used to create job descriptions.

job ID. The JES job ID of the job associated with the operation.

job name. The name of the job associated with an operation. The job name is assigned in the JOB statement of a job. It identifies the job to the system.

job preparation. Job preparation involves modifying jobs in preparation for processing. This can be performed manually, by a job preparer, or automatically by Tivoli OPC JCL tailoring functions.

job setup. The preparation of a set of JCL statements for a job at a job setup workstation. Job setup can be performed manually by an operator, or automatically by Tivoli OPC.

job setup workstation. A general workstation defined with the job setup option. A job setup workstation lets you modify your job or STC JCL before execution.

job submission. A Tivoli OPC process that presents jobs to MVS for running on a Tivoli OPC-defined workstation once the scheduling criteria for the operation is met.

job tracking. A Tivoli OPC process that communicates with operating systems that control computer workstations.

JS. The JCL repository dataset.

layout. In the Graphical User Interface for Application Description, a user-created file that determines which information about each application is displayed when you view a list of application descriptions. An application description contains many details about the application, such as application ID, valid to date, application status, and last user. A layout specifies which details the user wishes to view.

layout ID. A unique name that identifies a specific ready or error list layout.

limit for feedback. See feedback limit.

list, application. In the Graphical User Interface for Application Description, a list of application definitions from which the user can select one to work with. It consists of application definitions selected according to user-specified criteria.

List view. The Workload Monitor/2 objects Workstations List and Operations List contain other objects. The List view shows a list of the contained object and displays data about each contained object.

local. Synonym for channel-attached.

local processor. (1) In a complex of processors under JES3, a processor that executes users' jobs and that can assume global functions if the global processor fails. (2) In Tivoli OPC, a processor in the same installation that communicates with the controlling Tivoli OPC processor through shared DASD or XCF communication links.

logical unit (LU). In Systems Network Architecture (SNA), a port through which an end user accesses the SNA network in order to communicate with another end user and through which the end user accesses the functions provided by system services control points (SSCPs).

LU-LU session type 6.2. See logical unit 6.2.

logical unit 6.2 (LU 6.2). A type of Systems Network Architecture (SNA) logical unit (LU) for communication between peer systems. Synonymous with APPC protocol, see Advanced Program-to-Program Communications (APPC).

long-term plan (LTP). A high-level plan of system activity that covers a period of at least 1 day, and not more than 4 years. It serves as the basis for a service level agreement with your users, and as input to daily planning.

LU. See logical unit.

LTP. See long-term plan.
manipulation button. One of the two mouse buttons. With default mouse settings, the manipulation button is mouse button 2, the button on the right. You press and hold this button to move an object, for example, to drag an object to a printer. Pressing the manipulation button once when the pointer is on an object, opens the object’s pop-menu.

manual reporting. A type of workstation reporting in which events, once they have taken place, are manually reported to Tivoli OPC. This type of reporting requires that some action be taken by a workstation operator. Manual reporting is usually performed from a list of ready operations.

mass updating. A function of the Application Description dialog in which a large update to the application database can be requested.

MCU. Multiple Console Support.

Merged Graph view. A view of the Workload Monitor/2 Workstations List object. Shows the total number of operations with different statuses for all the workstations that are included in the object. The information is shown in a single graph.

modify current plan (MCP). A Tivoli OPC dialog function used to dynamically change the contents of the current plan to respond to changes in the operation environment. Examples of special events that would cause alteration of the current plan are: a rerun, a deadline change, or the arrival of an unplanned application.

most critical application occurrences. Those unfinished applications whose latest start time is less than or equal to the current time.

N

NCF. See Network Communication Function.

NCP. Network Control Program.

NetView operations. Operations that consist of an operator instruction that Tivoli OPC passes to NetView. These operations are run at a general workstation with the WTO option specified.

Network Communication Function (NCF). A VTAM application that submits work to remote systems and passes events back to the Tivoli OPC tracker subsystem on the Tivoli OPC controlling system.

noncyclic period. A period that does not represent a constant number of days or work days. Examples: quarter, academic semester.

nonreporting. A reporting attribute of a workstation, which means that information is not fed back to Tivoli OPC.

O

occurrence. An instance of an application in the long-term plan or current plan.

An application occurrence is one attempt to process that application. Occurrences are distinguished from one another by run date, input arrival time, and application ID. For example, an application that runs four times a day is said to have four occurrences per day.

occurrence group. Consists of one or more application occurrences added to the long-term plan or current plan, where such occurrences are defined as belonging to a particular application group specified in the group definition field of the application description or job description.

offset. Values, in the ranges 1 to 999 and −1 to −999, that indicate which days of a calendar period an application runs on. This is sometimes called displacement.

OI. See operator instruction.

OPC/ESA. Operations Planning and Control/ESA

OPC host. The processor where Tivoli OPC updates the current plan database.

OPC local processor. A processor that connects to the Tivoli OPC host or remote processor through shared event datasets or XCF communication links.

OPC remote processor. A processor connected to the Tivoli OPC host processor via an SNA network. A Tivoli OPC event writer and an event transmitter (Tivoli OPC Network Communication Function) are installed on the remote processor and transmit events to the Tivoli OPC host processor via VTAM.

open interval. The time interval during which a workstation is active and can process work.

operation. A unit of work that is part of an application and that is processed at a workstation.

operation deadline. The latest time when the operation must be complete.

operation latest out. For an operation that has predecessors, the latest out date and time are the latest
start time for the first critical operation in the application occurrence. If the first critical operation has not started by this date and time, then the operation is flagged as late, because it will be impossible for it to start on time based on the sum of the planned durations of all the operations on its critical path.

**operation number.** The number of the operation. This uniquely identifies each operation in an application.

**Operation object.** An object contained in the Workload Monitor/2 Operations List object. It represents one operation in the current plan.

**operation status.** The status of an operation at a workstation.

**operation waiting for arrival.** The status of an operation that cannot begin processing because the necessary input has not arrived at a workstation. This status is applicable only for operations without predecessors.

**Operations List object.** A Workload Monitor/2 object that can be used to display information about operations in the current plan. It contains Operation objects.

**operator instruction (OI).** An instruction that an operator can view when the operator must manually intervene in Tivoli OPC operations.

**origin date.** The date that a period (cyclic or noncyclic) starts on.

**owner ID.** Owner ID is an identifier that represents the application owner.

**parallel operations.** Operations that are not dependent on one another and that can, therefore, run at the same time.

**parallel servers.** These represent the number of operations that can be processed concurrently by that workstation.

**partner transaction program.** An Advanced Program-to-Program Communications (APPC) transaction program located at the remote partner.

**PDF.** Program Development Facility.

**pending application description.** An application description that is incomplete and not ready for use in planning or scheduling. See active application description.

**pending occurrence.** The dummy occurrence created by the daily planning process to honor a dependency that has been resolved in the long-term plan but cannot be resolved in the current plan because the predecessor's input arrival time is not within the current plan end time.

**pending predecessor.** A predecessor dependency to an occurrence which is defined in the long-term plan but not yet included in the current plan. See also pending occurrence.

**period.** A time period defined in the Tivoli OPC calendar.

**personal workstation.** In Tivoli OPC documentation this term is used to refer to a computer that runs IBM Operating System/2.

**PIF.** See program interface (PIF).

**plan.** See current plan.

**Plan object.** A Workload Monitor/2 object that can be used to get information about the status of the current plan. When the Details view of the Plan object is open, the object monitors for current plan alerts if alert conditions have been specified.

**predecessor.** An operation in an internal or external dependency that must finish successfully before its successor operation can begin.

**print workstation.** A workstation that prints output and usually reports status to Tivoli OPC automatically.

**printout routing.** The ddname of the daily planning printout dataset.

**priority.** The priority of an operation is a value from 1 to 9 (where 1=low, 8=high, and 9=urgent). It is one of the factors that determines how Tivoli OPC schedules applications.

**program interface (PIF).** A Tivoli OPC interface that lets user-written programs issue various requests to Tivoli OPC.

**query current plan (QCP) dialog.** An ISPF dialog that displays information taken directly from the current plan. The information includes information on operations, workstations, and application occurrences.

**QSAM.** Queued Sequential Access Method.
R

RACF. Resource Access Control Facility.

read authority. Access authority that lets a user read the contents of a dataset, file, or storage area, but not change it.

ready (R). The status of an operation indicating that predecessor operations are complete and that the operation is ready for processing.

ready list. An ISPF display list of all the operations ready to be processed at a workstation. Ready lists are the means by which workstation operators manually report on the progress of work.

receive. (1) To obtain a message or file from another computer. Contrast with send. (2) In Communications Manager, the command used to transfer a file from a host.

code format. The definition of how data is structured in the records contained within a file. The definition includes record names, field names, and field attributes, such as length and data type.

recovery. See automatic job and started-task recovery.

remote job tracking. The function of tracking jobs on remote processors connected by VTAM links to a Tivoli OPC controlling processor. This function enables a central site to control the submitting, scheduling, and tracking of jobs at remote sites.

remote processor. A processor connected to the Tivoli OPC host processor via a VTAM network.

replan current period. A Tivoli OPC function that recalculates planned start times for all occurrences to reflect the actual situation.

reporting attribute. A code that specifies how a workstation will report events to Tivoli OPC. A workstation can have one of four reporting attributes:

A Automatic
C Completion only
N Nonreporting
S Manual start and completion.

reroutable. Tivoli OPC can reroute operations if the workstation that they are scheduled to run on is inactive. An example of this can be if communication links to the system where the workstation is located fail. This option applies to operations only when they have status R (ready) or W (waiting). When you define an operation, you can specify one of the following reroutable options:

Y The operation is eligible to be rerouted if the workstation becomes inactive.
N The operation will not be rerouted, even though the workstation has an alternate destination.
blank The operation will be rerouted according to the WSFAILURE parameter on the JTOPTS initialization statement. This is the default.

rerun. A Tivoli OPC function that lets an application or part of an application that ended in error be run again.

Resource Object Data Manager. A licensed program that monitors resources and informs subscribing applications of their availability.

restartable. If an operation is defined as restartable, Tivoli OPC can automatically restart that operation if the workstation that it is using becomes inactive. This option applies only to the operation while it has status S (started). The operation will be reset to status R (ready).

return code. An error code that is issued by Tivoli OPC for automatic-reporting workstations.

RODM. See Resource Object Data Manager.

row command. An ISPF dialog command used to manipulate data in a table.

rule. A named definition of a run cycle that determines when an application will run.

run cycle. A specification of when an application is to run. The specification may be in the form of a rule or as a combination of period and offset.

S

SAA. See Systems Application Architecture.

SAF. System Authorization Facility.

schedule. (1) The current or long-term plan. (2) To determine the input arrival date and time of an occurrence or operation.

selection button. One of the two mouse buttons. With default mouse settings, the selection button is mouse button 1, the button on the left. You use this button to select windows, menu choices, pages in a notebook, and buttons. Pressing the selection button twice when the pointer is on an object opens the object to the default view.

send. (1) To send a message or file to another computer. Contrast with receive. (2) In
Communications Manager, the command used to transfer a file to the host.

**server.** The optional Tivoli OPC component that runs on the controlling system and handles requests from remote ISPF dialogs, remote PIF applications, and the Graphical User Interface for Application Description.

**service functions.** Functions of Tivoli OPC that let the user deal with exceptional conditions, such as investigating problems, preparing APAR tapes, and testing Tivoli OPC during implementation.

**service level agreement.** An agreement made between the data processing center and its user groups indicating the service hours and levels, as well as the kind of service the DP center will provide.

**Settings notebook.** See Settings view

**Settings view.** A view of an object that is used to specify properties of the object itself.

**shared DASD.** Direct access storage devices that can be accessed from more than one processor.

**shared resource.** A special resource or workstation resource that can be used simultaneously by more than one operation.

**slack.** Refers to 'spare' time. This extra time can be calculated for the critical path by taking 'Deadline less the Input Arrival less the sum of Operation Durations'.

**SMF.** System Management Facilities. An MVS component that collects and records system and job-related information.

**smoothing factor.** A value in the range 0-100 that controls the extent to which actual durations are fed back into the application description database.

**SMP.** System Modification Program.

**SNA.** See Systems Network Architecture.

**special resource.** A resource that is not associated with a particular workstation, such as a dataset.

**splittable.** Refers to a workstation where operations can be interrupted while being processed.

**standard.** User-specified open intervals for a typical day at a workstation.

**started (S).** A Tivoli OPC reporting status, for an operation or an application, indicating that an operation or an occurrence is started.

**started-task computer workstation.** You can specify that a computer workstation will support started tasks by giving the workstation the STC option. Operations defined to this workstation will be treated as started tasks, not as jobs.

**started-task operations.** Operations that start or stop started tasks. These operations are run at a computer workstation with the STC option specified.

**status.** The current state of an operation or occurrence.

**status code.** Codes that represent the current state of an operation. The status code is often associated with an extended status code.

The status of an operation can be one of the following:

- **A** The operation is waiting for input to arrive.
- **R** The operation is ready for processing (all predecessors have been reported as complete).
- **S** Operation processing has started.
- **C** Operation processing has completed.
- **D** The operation has been deleted from the current plan.
- **I** Operation processing has been interrupted.
- * The operation is ready for processing. There is a predecessor at a nonreporting workstation, but all other predecessors are reported as complete.
- **E** The operation has ended in error.
- **W** The operation is waiting for a predecessor to complete.
- **U** The operation status is not known.

**submit/release dataset.** A dataset shared between the Tivoli OPC host and a local Tivoli OPC processor that is used to send job-stream data and job-release commands from the host to the local processor.

**subresources.** A set of resource names and rules for the construction of resource names. Tivoli OPC uses these names when checking a user's authority to access individual Tivoli OPC data records.

**subsystem.** A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system.

**successor.** An operation in an internal or external dependency that cannot begin until its predecessor completes processing.

**SYSOUT.** A system output stream, also an indicator used in data definition statements to signify that a dataset is to be written on a system output unit.
SYSOUT class. An indicator used in data definition statements to signify that a dataset is to be written on a system output unit. It applies only to print workstations.

SYSPLEX. An MVS/ESA systems complex provides systems management enhancements for coordinating and controlling the data processing facility across multiple systems, while minimizing complexity. Implemented using the 9037 Sysplex Timer and the cross-system coupling facility (XCF) component of MVS/ESA.

Systems Application Architecture (SAA). A formal set of rules that enable applications to be run without modification, in different computer environments.

Systems Network Architecture (SNA). The description of the logical structure, formats, protocols, and operational sequences for transmitting information units through the networks and also operation sequences for controlling the configuration and operations of networks.

T

tail plan. Created during the daily planning process, includes only tail work; that is, work that started during or before the current planning period and that extends beyond its end.


temporary operator instructions. Operator instructions that have a specific time limit during which they are valid. They will be displayed to the workstation operator only during that time period.

time dependent. Tivoli OPC attempts to start operations as soon as possible, when all dependencies have been resolved and processing resources are available. However, you can specify that an operation is time-dependent, so Tivoli OPC will not start it until a specific time.

time zone support. A feature of Tivoli OPC that lets applications be planned and run with respect to the local time of the processor that runs the application. Some networks might have processors in different time zones. The controlling processor will make allowances for differences in time during planning activities to ensure that interacting activities are correctly coordinated.

TP. See application transaction program.

tracker. The Tivoli OPC component that runs on every system in your complex. It acts as the communication link between the MVS system that it runs on and the controller.

tracking event log. A log of job-tracking events and updates to the current schedule.

transport time. The time allotted for transporting materials from the workstation where the preceding operation took place to the workstation where the current operation is to occur. The transport time is used only for planning purposes. Operations will be started irrespective of the transport time specified.

TSO. Time Sharing Option.

turnover. A subfunction of Tivoli OPC that is activated when Tivoli OPC creates an updated version of the current plan.

U

undecided (U). A Tivoli OPC reporting status, for an operation or an application, indicating that the status is not known.

update authority. (1) Access authority to use the ISPF/PDF edit functions of the Tivoli OPC dialog. The authority is given to the user via RACF. (2) Access authority to modify a master file or dataset with the current information.

V

validity period. The time interval defined by an origin date and an end date within which a run cycle or an application description is valid.

variable table. A group of related JCL variables. Tivoli OPC can check these variable tables for substitution values for variables that occur in JCL. This substitution can occur during job setup or at job submit.

versions. Applications with the same ID but different validity dates.

VSAM. Virtual Storage Access Method.

VTAM. Virtual Telecommunications Access Method.

W

waiting (W). A status indicating that an application is waiting for a predecessor operation to complete.

waiting list. A list of jobs that have been submitted but still have uncompleted predecessors. Operations will be included in the waiting list if the JCL is not
submitted by the Tivoli OPC controller and the Tivoli OPC tracker has been started with HOLDJOB(YES).

**work day.** A day on which applications can normally be scheduled to start.

**work-days-only cyclic period.** A cyclic period where only work days are counted when calculating the interval.

**work-day end time.** The time when one Tivoli OPC work day ends and the next day begins. By default, this time is midnight.

For example, if the work-day end time is 02:00, work for Friday can continue until 02:00 on Saturday morning, even if Saturday is a free day. If Saturday and Sunday are free days, no new work will be started until 02:00 on Monday.

**Workload Monitor/2.** A part of Tivoli OPC. It runs on OS/2 Version 2 (or later) and communicates with a Tivoli OPC controller subsystem. It carries data about the subsystem’s current plan from the host to a workstation, and can update operation status.

**workstation.** (1) A unit, place, or group that performs a specific data processing function. (2) A logical place where work occurs in an operations department.

Tivoli OPC requires that you define the following characteristics for each workstation: the type of work it does, the quantity of work it can handle at any particular time, and the times it is active. The activity that occurs at each workstation is called an operation. (3) See also **personal workstation.**

**workstation description database.** A Tivoli OPC database containing descriptions of the Tivoli OPC workstations in the operations department.

**workstation resource.** A physical resource, such as a tape drive, that must be allocated among jobs. When you define a workstation, you can specify the quantity of each of two resources (R1 and R2) that are available to operations. When defining operations to that workstation, you can specify the number of these resources that must be available for the operation to start on that workstation.

**workstation type.** Each workstation can be one of three types: computer, printer, or general.

**write-to-operator workstation.** A general workstation that lets you use Tivoli OPC scheduling facilities to issue a write-to-operator (WTO) message at a specific operator console defined by the workstation destination. NetView can intercept the WTO message and take necessary action.

**WTO message.** Write-to-operator message.

**WTO operations.** Operations that consist of an operator instruction that Tivoli OPC passes to NetView. These operations are run at a general workstation with the WTO option specified.

**X**

**XCF.** MVS/ESA cross-system coupling facility.

**XRF.** Extended recovery facility.
Index

Special Characters

? (question mark) variables 279
* (asterisk) global search character 307
* (asterisk) operation status code 345
& (ampersand) variables 277
% (percent) global search character 307
% (percent) variables 278

Numerics

1 extended status code 346
2 extended status code 346

A

A extended status code 346
A operation status code 345
abend codes
  grouping together 264
access method
  specifying for workstation 39
ACTION keyword 139, 147, 153
activating automatic recovery 265
activating event-triggered tracking (ETT) 207, 268
activating JCL variable substitution 276
activating job submission 205
active
  application status 78
AD (application description) database 128
AD NOT FOUND ON AD FILE message 181
ADDEP 152
ADDEP batch-loader control statement 152
ADGROUPID keyword 137
ADID keyword 137, 155
ADOP batch-loader control statement 145
ADOPPWTO keyword 149
ADRULE batch-loader control statement 141
ADRUN batch-loader control statement 138
ADSR batch-loader control statement 132, 150
ADSTART batch-loader control statement 136
ADTYPE keyword 137
ADVALFROM keyword 137
AEC keyword 147
alert when deadline passes 105
ALERTS initialization statement
All Open Time Intervals panel 35
all workstations closed 394
dall-days cyclic periods 62
allocation status of special resources 50
ALPHA value in variable validation 290
alternate current plan dataset (EQQCP2DS) 199
alternate workstations 24
automatic recovery 246
automatic rerouting to 103
recovery with catalog management 240
ampersand (&) variables 277
AOC/MVS 53
APAR (authorized program analysis report) 207
tape function 207
application description database
  input to current plan 187
Application Description dialog 73
application groups
  guidelines 72
  specifying 75
applications 69—121
  common data 397
  considerations 69
  creating 13
  cross-reference list 400
  cross-references of job names and active applications 395
  ID 71, 77
  in the long-term plan 184
  internal operation logic report 398
  JCL variable tables 274
  mass update report 402
  naming standards 71
  occurrence
    See occurrences
  operation data 398
  operations using particular workstations report 399
predecessor
  See predecessors
  scheduling 78
  specifying 75
  status 78
successor
  See successors
  updating in batch 123
  valid-from date 77
  valid-to date 77
APRIL keyword 143
ARC row command 262
AROPTS initialization statement 206
AUTHUSER keyword 262
ENDTIME keyword 252
PREDWS keyword 260
STARTTIME keyword 252
USERREQ keyword 262
arrival time
  specifying 108

© Copyright IBM Corp. 1991, 1999

431
associating variable tables with applications 274
asterisk (*) global search character 307
asterisk (*) operation status code 345
attributes
  reporting, for workstations 29
audit trail of failed jobs 263
audit trail of recovery actions 263
audit trails
  report 195
AUGUST keyword 143
AUTHDEF initialization statement
  security 262
authority group ID 76
authorization
  and batch programs 352
authorized program analysis report (APAR) tape function 207
AUTHUSER keyword of AROPTS 262
automatic
  job submission 102
  reporting attribute 31
automatic job and started-task recovery 243—266
  activating 206
  deactivating 206
examples 20
automatic recovery
  considerations 223
automation options 101
automation using ETT 272
availability
  controlling with NetView 53
  controlling with RODM 53
  special resources 41, 331
  and LOOKAHEAD 53
workstation 34
Availability of a Workstation panel 34
avoiding loops in variable substitution 311

B
B extended status code 346
BACKUP command
  description 318
BACKUP keyword of JTOPTS 202, 318
backup workstation 24, 246
backward references in JCL 266
batch jobs
  to update the LTP 176
batch loader 73, 127
  control statements 129
    syntax and construction 133
  error messages 159
  reasons for using 123
batch programs
  EQQADMUP 356
  EQQADPRT 357
  EQQADXRF 359
  EQQAXR00 360
  EQQBATCH 351
  EQQCLPRC 361
  EQQCLPRP 362
  EQQDNTOP 363
  EQQDOTO 366
  EQQDRTOP 368
  EQQDTTOP 370
  EQQEVPGM 372
  EQQJVPRT 373
  EQQLTCRE 374
  EQQLTMOA 375
  EQQLTMOO 377
  EQQLTPRT 378
  EQQLTTRY 380
  EQQOIBAT 382
  EQQOIBLK 383
  EQQPURGE 386
  EQQRC SIM 389
  EQQWSPRT 390
  EQQYLTOP 390
BATCHOPT initialization statement
  DYNAMICADD keyword 54
  DYNAMICDEL keyword 54
  PLANHOUR keyword 193
  PREVRES keyword 193
BEGIN JCL directive 303
BIT value in variable validation 290
blank, extended status code 346

C
C extended status code 346
C occurrence status code 345
C operation status code 345
calendar
  creating 10
  defining 59—62
  extra 62
  for variable substitution 312
  general information report 392
  period
    See periods
  period characteristics report 393
  search order 59
  specific dates report 392
  specifying 76
  status of days report 392
  work-day end time 10
CALENDAR keyword 137
CAN error code 347
CANCEL command 25
CANCEL command, MVS 338

commands (continued)
MVS (continued)
MODIFY 338
START 337
STOP 337
TSO 317

comments in LTP reports 181
COMP keyword on JCL directives 307
COMPLETE status 171
completed applications
daily planning report 193, 415
error statistics (daily planning report) 416
summary of, daily planning report 414

completion codes 225
completion-only reporting attribute 31
compound variables 278
computer workstations
alternate 23, 24
job statements 110
job type 23, 24
started task (STC) type 24
using 23

Confirm the Deletion of OI panel 110
considerations for creating plans 170
control option
for workstation fixed resources 38
control statements for batch loader 129
controlled shutdown 36
controller
introduction 1
controlling operations
special resources
and LOOKAHEAD 53
controlling workstations 40
CP
See current plan

Creating a Calendar panel 60
Creating a Calendar Period panel 63
Creating a JCL Variable Table panel 285
Creating a Job panel 118
Creating a Special Resource panel 49
Creating an Application panel 76
Creating an Operator Instruction panel 109
creating applications 13, 69, 75
creating calendars 10, 59
Creating General Information about a Workstation panel 28
creating intervals when resources are available 52
creating job descriptions 117, 118
creating operations 92
creating periods 62
creating plans 19
creating resources dynamically 53
creating rules 78
creating run cycles 78
creating special resources 8, 45
creating temporary variables 300
creating the long-term plan 166
Creating the Long-Term Plan panel 178
creating workstations 5, 23
cross reference
  application descriptions 400
  job names and active applications 395, 396
cross-system coupling facility (XCF) 243
CTIME, JCL variable 283
current plan 165, 187—204
  analyzing problems 196
  automatic backup process 202
  catch up 203
  changing 198
  event-triggered tracking 270
  integrity 199
  NCP takeover 203
  new (NCP) 203
  on-request BACKUP 318
  on-request JSUACT 320
  organization 199
  recreating 191
  reports 193
  trial 192
  turnover 203
Current Plan and Status Inquiry menu 198
current plan extension dataset (EQQCXDS) 199
CWW, JCL variable 283
CWWD, JCL variable 283
cycle specification
  for rules 82
cyclic periods 62
CYMD, JCL variable 283
CYY, JCL variable 283
CYYDDD, JCL variable 283
CYYMM, JCL variable 283
CYYMMD, JCL variable 283
CYYYY, JCL variable 283
CYYYYYY, JCL variable 283

daily planning (continued)
reports
  actual resource utilization 418
  general information report 408
  operating plan 409
  parallel operations, daily planning report 411
  plan for work station 410
  planned resource utilization 413
  resource 1, daily planning report 412
  summary of completed applications report 414
  workstation resource utilization reports 412
  workstation utilization reports 411
data lookaside facility (DLF) 56
database
  application description 128
  operator instruction 128
  workstation description
    See workstation, description database
dataset triggering 56
datasets
  EQQAD2DS 158
  EQQADDS 158
  EQQDUMP 158
  EQQJBLIB 212
  EQQMLIB 158
  EQQMLOG 158
  EQQOID 158
  EQQOIPDS 158
  EQQSTC 212
  EQQWSDS 158
  for long-term plan 176
  JCL repository (EQQJSnDS) 212
  SYSUDUMP 158
date-related supplied variables 282
day
  DAY keyword 143
day specification
    for rules 82
day status 60
dB database
    using 223
DBCS (double-byte character set)
  and job descriptions 70, 118
deactivating automatic recovery 265
deactivating event-triggered tracking (ETT) 268
deactivating job submission 205
deactivating variable substitution 276, 294
deadline
    specifying 80
DEADLINE - START > 24 HRS message 181
DEADLINE WTO option 105
DECEMBER keyword 143
default calendar 10
default time for an operation 33
defining applications 13, 69, 75
defining calendars 10, 59

daily planning
  current plan
    analyzing problems 196
    completed applications report 415
    error statistics on completed applications 416
    missed feedback report 417
    new current plan process 203
    operation in error report 417
/report on completed applications 193
  187
  See also current plan
  analyzing problems 196
  completed applications report 415
  error statistics on completed applications 416
  missed feedback report 417
  new current plan process 203
  operation in error report 417

D
D extended status code 346
D occurrence status code 345
D operation status code 345
daily plan
  report on completed applications 193
  daily planning 187
  See also current plan
  analyzing problems 196
  completed applications report 415
  error statistics on completed applications 416
  missed feedback report 417
  new current plan process 203
  operation in error report 417
Index 435
ERROR CODE = xxxx message 181
error codes 347
catalog management 348
from completion codes 225
job log retrieval 349
set by Tivoli OPC 225
error messages
batch loader 159
generated in long-term plan reports 181
error statistics on completed applications 193, 416
ERROR TRACKING option 101
errors in variable substitution 293
ERRRES keyword of JTOPTS 227
estimated duration
  feedback 105
ETT
  See event-triggered tracking (ETT)
ETT dialog 268
ETT keyword of JTOPTS 207, 268
ETTRCY1 period use by ETT 271
event dataset
time zones 217
event-triggered tracking (ETT) 267—272
  activating 207
  adding an occurrence to the current plan 270
  deactivating 207
  period ETTRCY1 271
events
  how they are generated 215
  triggered by availability changes 56
EVERY
  in rules 82
EVERY keyword 142
EWTROPTS initialization statement
  RETCODE keyword 225, 227
  STEPEVENTS keyword 227
eexample of batch-loader statements 130
examples of job recovery 256
excluding JCL 295
exclusion rule 81
EXECUTE command 210
exits for catalog management 236
extended status codes 346
extending the current plan 190
extending the long-term plan 179
Extending the Long-Term Plan panel 179
external dependencies 72, 94
external dependencies in the LTP 178
external predecessors
  resolving 96

F
failed job recovery 20
failed jobs
  planning 219
failure or success of a job 227
FEbrUARY keyword 143
feedback algorithms
  limit for feedback 107
  smoothing factor 106
feedback options 105
Feedback Options panel 106
feedback report 195
feedback, missed (daily planning report) 417
FETCH JCL directive 306
filter panel 126
fixed resources 37, 41, 99
FORM NUMBER option 104
free days 36, 60
free-day rule
  selecting 86
FREEDAY keyword 143
frequency specification
  for rules 82
FRIDAY keyword 143

G
gDG recovery 266
GENDAYS command 83
general information report
calendars 392
daily planning 408
long-term plan 404
general workstations
  defining 25
  job setup type 25
  preparing JCL 112
WTO type 26
  issuing operations 112
generating events 215
Generating JCL for a Batch Job panel 127
generation data group (GDG)
  catalog management actions 239
  recovery 266
global change to application definitions 125
global JCL variable table 274, 292
global search characters 307
GMT (Greenwich Mean Time) 217
grand total for period, long-term plan report 407
GRAPH command 94
group applications
  specifying 75
group definitions
  guidelines 70, 72
  in the long-term plan 166, 175
  introduction 13
GROUP keyword 137
grouping return codes 264
GTABLE keyword of OPCOPTS 292

H
HEX value in variable validation 290
HIGHEST RETURNCODE option 101, 119
HIGHRC keyword of JTOPTS 147, 227
Hiperbatch 56
histogram showing workstation utilization 181
history function
  using 223
HOLD status 210
HOLD/RELEASE option 104
I

I extended status code 346
I operation status code 345
IADAYS keyword 140
IAT

See input arrival time (IAT)
IATIME keyword 140
imbedded blanks in JCL variables 295
with COMP keyword 308
implementing Tivoli OPC 1
IMS

tracking jobs submitted by 269
in-effect dates 89
in-progress operation 322
inactive workstation
effect on operations 103
including JCL 295, 306
independent variable 287
initialization statements
determining success or failure of a job 227
JTOPTS

HIGHRC keyword 101
JOBSUBMIT keyword 102
NOERROR keyword 101
SUPPRES ACTION keyword 114
SUPPRESSPOLICY keyword 114
WSFAILURE keyword 103
WSOFFLINE keyword 103
SUPPRES ACTION keyword 105
initiators 100
input arrival time (IAT)
importance for rules 78
in long-term plan 166
occurrences 89
operations 108
resolving dependencies 96
instruction database 129
instructions
Edit JCL 110
operator 108
internal dependencies 94
internal operation logic report 398
interrupting workstation activity 32
interval origin 64
intervals
for special resources 52
intervals when workstations are open 35
invoking JCL variable substitution 276
IPL

avoiding lost events 216

J

J event, in event-triggered tracking 267

J extended status code 346
JANUARY keyword 143
JCCE error code 347
JCL

backward references 266
checking 111
error code 347
including and excluding 295
MSGCLASS required for JCC 221
on-request BACKUP of repository file 318
on-request JSUACT of repository file 320
preparing 11
rebuild parameters 252
RECOVER statement 246
repository 12, 111, 185, 212
tailoring 110, 273
variable substitution 273—312
in MVS JCL procedures 294
invoking 276
~ (logical not) 273
promptable variables 273
restrictions 310
supplied variables 281
suppressing 276, 294
syntax 276
validation 290
variables with embedded blanks 295
variable table 64
creating 285
variable table report 401
JCL setup workstations 25
JCL Variable Dependency Value List panel 287
JCLI error code 347
JCLPREPA record
simulating JCL substitution 273
JES initiators 100
controlling with parallel servers 31
job
tailoring 110
job and started-task recovery 243—266
activating 206
deaactivating 206
JOB CLASS option 104
job completion checker
considerations 220
job completion checker (JCC)
setting error codes 226
job computer workstations 23
job control language
See JCL
job description
creating 70, 118
mass updating 124
job description dialog 73, 117
job log
retrieving and storing 230
job log retrieval
  considerations 220
  status codes 349
job name 71
  checking 111
    ETT replace function 268
job options 101, 104
job preparation 11
job priority 209
job recovery
  See also automatic job and started-task recovery
  example 20
job repository 111, 212
job repository dataset 12, 185
job setup workstations 25
  preparing JCL 112
job submission
  activating and deactivating 205
  order 209
job tracking for MVS 215
job variable table
  creating 285
  specifying 81
job-level catalog management 236
job-tracking archive dataset 203
job-tracking log 203
job-tracking-archive dataset (EQQJTARC) 199
job-tracking-log dataset (EQQJTnn) 199
job, WTO, and Print Options panel 101
JOBCHECK keyword 111
JOBCLASS keyword 147
JOBN keyword 147, 155
jobname replace function of ETT 268
JOBOPTS initialization statement
  CATMCLAS keyword 221
  CHSMWAIT keyword 239
JS file backup 318
JS file jsuact 320
JSUACT command
  description 320
JSUACT keyword of JOPTS 320
JOPTS initialization statement (continued)
  SUPPRESSPOLICY keyword 114
  TRACK keyword 270
  WSFailure keyword 103, 246, 348
  WSOFFLINE keyword 103, 246, 347
JULy keyword 143
JUNE keyword 143

K
K extended status code 346
keep-on-error flag
  specifying 98
keeping a sequential dataset with application
descriptions 123
keeping the job log 230
KEEPONERR keyword 151

L
L extended status code 346
LAST keyword 143
late operations 114
latest start time 209
layout of Ol dataset 384
length of the current plan 171
LIMFDBK keyword of JOPTS 147
limit for feedback 107
linking applications together 72
linklist 157
List of Calendar Periods panel 63
List of Generated Dates panel 82
List of JCL Variable Tables panel 285
List of Operator Instructions panel 109
List of Special Resources panel 49
List of Work Station Descriptions panel 28
LIST value in variable validation 290
logging failed jobs 263
logging recovery actions 263
logs
  retrieving and storing 230
long-term plan 165—186
  creating 175
  duration per workstation report 407
  external dependencies 178
  general information report 404
  grand total for period report 407
  histogram showing utilization 181
  input to current plan 187
  refresh function 178, 206
  report 181
Long-Term Plan Occurrences panel 185
long-term planning
  process 177
loop in event-triggered tracking 270
loops in variable substitution 311
losing events at shutdown 216
LTP
  See long-term plan

M
M extended status code 346
maintaining a sequential dataset with application
descriptions 123
Maintaining Job Descriptions menu 117
Maintaining OPC/ESA Databases menu 48
Maintaining Special Resources menu 48
Maintaining the Long-Term Plan menu 175
Maintaining Tivoli OPC JCL Variable Tables menu 285
Making a Trial Long-Term Plan panel 182
making changes conditionally 124
making plans smaller 121
MARCH keyword 143
mass update 123
mass update reports
  application descriptions 402
  update performed for owner ID 402
Mass Updating of Application Description panel 124
MAXJSFILE keyword of JTOPTS 318, 320
MAY keyword 143
MCP error code 347
MEMBER keyword 155
messages
  See error messages
migrated datasets 239
missed feedback report 195, 417
MODIFY command, MVS 338
modify current plan (MCP) dialog 198
  recovery 262
Modifying a JCL Variable panel 286
Modifying a Rule panel 81
Modifying All Workstations Closed panel 37
Modifying an Occurrence panel 185
Modifying Connected Workstations panel 52
Modifying Dependencies panel 186
Modifying ETT Tracking Criteria panel 268
Modifying Intervals for a Special Resource panel 51
modifying occurrences in the LTP 184
Modifying the Current Plan menu 198
MONDAY keyword 143
monitoring resources 46
MONTH keyword 143
MOVED (FREE DAY RULE) message 181
MSGCLASS keyword
  for the job completion checker 221
MSGLEVEL keyword 135
multiple jobs 111
MVS
  CANCEL command 338
  MODIFY command 338

MVS (continued)
  operator commands 337
  START command 337
  STOP command 337

N
NAME keyword 139
NAME value in variable validation 290
naming standards 71
NCP (new current plan) 203
negative occurrences 88
negative offsets 64
negative run cycle 88
NetView
  controlling availability of a resource 53
    using ETT 272
  WTO operations 112
new current plan (NCP) 188, 203
new current plan dataset (EQQNCPDS) 199
new current plan extension dataset (EQQNWXDS) 199
nnnn error code 348
NOERROR keyword of JTOPTS 227
noncyclic periods 62
nonreporting attribute 31
nonreporting workstations
  starting operations 211
NOP command 210
normal mode manager 339
NOSCAN directive 294
NOVEMBER keyword 143
NUM value in variable validation 290

O
OADID, JCL variable 281
OADOWNER, JCL variable 281
OAUGROUP, JCL variable 281
OCALID, JCL variable 281
occurrence group
  in the long-term plan 166, 175
occurrence-related supplied variables 281
occurrences
  adding with event-triggered variables 270
  in long-term plan 166, 175
  negative 88
status codes 345
OCDATE, JCL variable 284
OCFRSTC, JCL variable 284
OCFRSTW, JCL variable 284
OCFRSTWY, JCL variable 284
OCLASTC, JCL variable 284
OCLASTW, JCL variable 284
OCLASTWY, JCL variable 284
OCTIME, JCL variable 284
Index

OCTOBER keyword 143
ODAY, JCL variable 281
ODD, JCL variable 281
ODDD, JCL variable 281
ODESCR keyword 138
ODMY1, JCL variable 281
ODMY2, JCL variable 281
offsets 86
OFREEDAY, JCL variable 281
OFxx error code 347
OHH, JCL variable 281
OHHMM, JCL variable 281
OI (operator instruction) dataset 384
OI database
description 129
OISTART 154
OIT 156
OJCV error code 293, 347
OJOBNAME, JCL variable 282
OLDAY, JCL variable 282
OLDD, JCL variable 282
OLHH, JCL variable 282
OLHHMM, JCL variable 282
OLMD, JCL variable 282
OLMM, JCL variable 282
OLWK, JCL variable 282
OLYMD, JCL variable 282
OLYYDDD, JCL variable 282
OMM, JCL variable 282
OMMYY, JCL variable 282
on-error flag
specifying 98
ONERROR keyword of RESOPTS 50
ONLY
in rules 82
ONLY keyword 142
OOPNO, JCL variable 282
OPCOPTS initialization statement
CATACT keyword 237
GTABLE keyword 292
RECOVERY keyword 206, 265
RODMTASK keyword 44
VARSUB keyword 276
open intervals for workstations 35
Open Time Intervals for One Day panel 34
OPER command 76
operating plan, daily planning report 409
operation data report 398
Operation Details panel 95, 121
operation in error, daily planning report 417
operation number 71
operation reason codes 349
operation-related supplied variables 282
operations
creating 13
creating workstation 92
operations (continued)
defining details 95
dummy 27
ended-in-error 227
error codes
See error codes
extended status codes 346
feedback of information 322
feedback options 105
options 101
parallel servers 100
print 112
resources 210
resources used 97
special resources
and LOOKAHEAD 53
started task 113
status (OPSTAT) command 326
status codes 345
time-dependent 105
user field 322
WTO 112, 211
Operations panel 92, 94
operations using particular workstations, report 399
operator commands, MVS 337
operator instruction (OI) dataset 384
operator instructions
editing JCL 110
loading in batch 127
report example 403
specifying 108
OPIADATE, JCL variable 284
OPIATIME, JCL variable 284
OPINFO command
description 322
security considerations 323
syntax 323
OPINFOSCOPE keyword of JTOPTS 322
OPLSDATE, JCL variable 284
OPLSTIME, JCL variable 284
OPNO keyword 147, 155
OPSTAT command 267
general workstation status 25
introduction 326
updating the current plan 168
OPTIONS batch-loader control statement 134
order of operations 209
origin date 64
origin of periods 62
origin shift
in rules 82
ORIGINSHIFTF keyword 143
OSEQ error code 348
OSEQ error code 348
OSUB error code 348
OSUF error code 348
OSUP error code 348
OSxx error code 348
out-of-effect dates 89
output group
  tracking print operations 112
owner ID
  for applications 72
  mass update report 402
OWNER keyword 135, 138
OWSID, JCL variable 282
OWW, JCL variable 281
OWWD, JCL variable 281
OWWLAST, JCL variable 281
OWWMONTH, JCL variable 281
OYM, JCL variable 282
OYMD, JCL variable 282
OYMD1, JCL variable 282
OYMD2, JCL variable 282
OYMD3, JCL variable 282
OYY, JCL variable 282
OYYDDD, JCL variable 282
OYYMM, JCL variable 282
OYYYY, JCL variable 282

P
P extended status code 346
P occurrence status code 345
panels
  Editing JCL for an MVS Job 233
  EQQACGPP 76
  EQQACRTE 109
  EQQALSML 109
  EQQAMFDP 106
  EQQAMJBP 101
  EQQAMOPL 92
  EQQAMOSL 94
  EQQAMPDL 96
  EQQAMRNL 86
  EQQAMRPL 80
  EQQAMSDP 95, 121
  EQQAMSR 98
  EQQAMTMAP 108
  EQQAMWRP 100
  EQQAODP 110
  EQQAUFIP 126
  EQQAUPDL 124
  EQQAUGGL 125
  EQQDPNL 189
  EQQJCGPP 118
  EQQJMTCL 268
  EQQJMSBP 117
  EQQJVDM 287
  EQQJVMAP 285
  EQQJYTM 285
  EQQJVVCL 285
  panels (continued)
    EQQJVEP 290
    EQQJVVMP 286
    EQQLBATP 176
    EQQLBDM 183
    EQQLCPL 186
    EQQLCHGP 185
    EQQLCREP 178
    EQQLDTEP 179
    EQQLPRAP 180
    EQQLSTAP 182
    EQQLSTOL 185
    EQQLTEP 182
    EQQLTOP 175
    EQQMERSL 229
    EQQMJCLE 233
    EQQMTPP 198
    EQQOBSBP 48
    EQQODCRP 49
    EQQODML 51
    EQQODSL 49
    EQQOQTOP 48
    EQQOVDML 52
    EQQORULEP 81
    EQQORULSL 82
    EQQOSTOP 198
    EQQTCAL 60
    EQQTCRPL 63
    EQQTERL 63
    EQQUTO 205
    EQQWCGP 28
    EQQWMACL 37
    EQQWMAMP 39
    EQQWMATL 35
    EQQWMAVL 34
    EQQWMLSL 28
    EQQWMTOL 34
    EQQWMR 38
    EQQXSUBP 127
  Step Restart Selection List 229
  parallel operations for workstations, daily planning
  report 411
  parallel servers 23, 31
    specifying 100
Paymore example 1
PCAN error code 348
pending application status 78
pending occurrences
  resolving 169
percent (%) global search character 307
percent (%) variables 278
period characteristics report for calendars 393
PERIOD keyword 139
periods
  defining 62—68
  JCL variable table 64
periods (continued)
  offsets 64
  work-days-only 67
phase of substitution 276
PICT value in variable validation 290
PIF (program interface)
  See program interface (PIF)
plan
  current
    See current plan
  example of creating 19
  long-term
    See long-term plan
  reducing the size 121
types 165
plan for workstation, daily planning report 410
PLANHOUR keyword of BATCHOPT 193
planning and control options
  for workstation fixed resources 38
planning system outages 36
PREADID keyword 153
PRED command 94
predecessors
  in the current plan 195
  specifying 94
Predecessors panel 96
PREDWS keyword of AROPTS 260
PREJOBN keyword 149, 153
PREOPNO keyword 149, 153
preparing JCL 11, 112
preparing job statements 25
PREVRES keyword of BATCHOPT 193
PREWSID keyword 149, 153
primary current plan dataset (EQQCP1DS) 199
print of application descriptions 397
print operations 112
  specifying options 104
printer workstations 25
printing the long-term plan 180
Printing the Long-Term Plan - All Applications panel 180
PRIORITY keyword 138
priority of operations 76, 209
problems with the current plan 196
procedure library
  for Tivoli OPC started tasks 24, 110
procedures
  variable substitution 294
producing daily plans 189
program interface (PIF)
  for creating applications 73
  JCL preparation 273
promptable variables 273
PSNUM keyword 149
Q
  Q extended status code 346
  QCP dialog 198
quantity
  of special resources 41, 98, 332
  QUANTITY keyword 151
query current plan (QCP) dialog 198
querying the current plan 198
question mark (?) variables 279
QUEUELEN keyword of JTOPTS 210
R
  R event, in event-triggered tracking 267
  R extended status code 346
  R operation status code 345
  R1 and R2 resources 37, 99
  R1NUM keyword 149
  R2NUM keyword 149
  RACF
    resource profiles 207
  RANGE value in variable validation 290
read list
  resources 210
reason codes 349
reasons for job failure 227
rebuild parameters 252
recataloging datasets 242
RECOVER statement 206, 226, 243
  syntax 246
recovery
  activating 206
  current plan 199
  deactivating 206
  example 20
  job and started task 243
    See also automatic job and started-task recovery
    with catalog management 245
  RECOVERY keyword of OPCOPTS 265
reducing the size of plans 121
refreshing RACF resource profiles 207
refreshing the LTP 178, 206
REPLAN option 191
reporting attributes
  defining 29
reports
  actual resource utilization 418
  application description 397
    cross-reference of applications and external
    dependencies 396
    cross-references of job names and active
    applications 395
  calendars
    general information 392
    period characteristics 393
    specific dates 392
resources (continued)
calendars (continued)
status of days 392
completed applications 415
cross-reference of applications 400
current plan 193
daily planning
  daily operating plan 409
  general information 408
  plan for workstation 410
  summary of completed applications 414
  workstation resource utilization 412
  workstation utilization 411
duration per workstation 407
error statistics on completed applications 416
internal operation logic 398
JCL variable table 401
long-term plan 180
  general information examples 404
  grand total workload for period 407
mass update
  application descriptions 402
  owner ID, updates performed for 402
missed feedback 417
operation data 398
operations in error 417
operations using particular workstations 399
operator instructions 403
planned resource utilization 413
workstation description 394
REROUTABLE keyword 149
REROUTABLE option 103
rerunning a job 229
resetting operations 227
resolving dependencies 96, 195
resolving dependencies in the LTP 178
RESOPTS initialization statement
  DYNAMICADD keyword 54
  ONERROR keyword 50
  RESOURCE keyword 151
Resource Monitor 46
Resource Object Data Manager (RODM) 44
resource profiles
  refreshing RACF 207
resources 41
  allocation status 50
  changing the status 330
  data lookaside facility (DLF) 56
  description database
    input to current plan 187
  fixed R1 and R2 37
  parallel servers 100
  R1 and R2 99
  special 8
    See also special resources
  used by operations 97
Resources for a Workstation panel 38
RESTEP keyword 245
RESTARTABLE keyword 149
RESTARTABLE option 103
restarting a job 229
restarting jobs a second time 242
restrictions on variable substitution 310
RETCODE keyword of EWTROPTS 225, 227
retrieving and storing job logs 230
return codes
  grouping together 264
  HIGHEST RETURNCODE option 101, 119
RODM 44
  special resources 53
RODMOPTS initialization statement 44
RODMTASK keyword of OPCOPTS 44
RULE keyword 140
rules 62
  creating 78
  creating with batch loader 141
rules in run cycles 80
  using rules 80
RUN command 76, 118
run cycle
  creating 78
  examples using offsets 90
  negative 88
  periods 66
  specifying with batch loader 138
  using offsets 86
Run Cycles panel 80
run days 78
Run Days panel 86
run order of operations 209
running Tivoli OPC commands in batch 372

S
S extended status code 346
S occurrence status code 345
S operation status code 345
sample application implementation 1
SATURDAY keyword 143
SAVE command 25
saving job logs 230
SCAN JCL directive 296
schedule 170
SCHEDULED ON FREE DAY message 181
scheduling job closedown 113
scope of OPINFO 322
scope of restart 229, 242
SEARCH JCL directive 297
search order
  calendar 59
  JCL variables 274
searching application descriptions 124
security
  automatic job and started-task recovery 262
  BACKUP command 319
  batch loader 132
  of batch programs 352
  OPINFO command 323
  OSTAT command 327
  refreshing RACF resource profiles 207
  SRSTAT command 330
  WSSTAT command 334
Selecting Long-Term Plan Batch Job panel 176
selecting restart steps 229
selecting work for submission 209
SEPTEMBER keyword 143
sequence of batch-loader control statements 129
servers
  parallel 31, 41
  specifying 100
service functions 205
Service Functions menu 205
SET CLOCK command 217
SETFORM JCL directive 298
Setting Default for Browse panel 183
setting up JCL 11, 112
setup variables 287
SETVAR JCL directive 300
shared and exclusive use of special resources 98
shared spool 26
shift of origin
  in rules 82
shutdown
  avoiding lost events 216
  SHUTDOWNPOLICY keyword of JTOPTS 36
shutting down online systems 113
side information dataset (EQQSID) 199
simple variables 278
simulating JCL substitution 273
smoothing factor 106
SMOOTHING keyword of JTOPTS 147
SMS-managed datasets 239
special resources 41—57
  actual utilization report 418
  allocation status 50
  connecting to workstations 52
  creating 45
  data lookaside facility (DLF) 56
  example 8
  intervals 50
  planned utilization report 413
  reports 56
  shared and exclusive 98
  status (SRSTAT) command 330
  used by operations 97
  using to trigger events 56
  when an operation fails 50
Special Resources panel 98
specific dates report for calendars 392
specifying applications 13, 69, 75
specifying calendars 10, 59
specifying catalog management 102
Specifying Filter Criteria panel 126
specifying JCL variable tables 64, 81, 297, 302
specifying job descriptions 117
specifying operations 92
specifying periods 62
specifying predecessors 94
specifying resource usage 97
specifying rules 78
specifying run cycles 78
specifying special resources 8
specifying the workload manager (WLM) options 102
Specifying Verification Criteria panel 290
specifying when applications should not run 88
specifying workstations 5, 23
splittable workstation 32
spool
  shared 26
SRSTAT command
  for creating resources 45
  in event-triggered tracking 267
  introduction 330
standard days 36
START command 24
START command, MVS 337
start interval 114
start of the LTP 166
start time
  latest 209
start-and-completion reporting attribute 31
started task
  dataset 24
  how Tivoli OPC starts 212
  operations 113
  options 101
  recovery 243
  scheduling closedown 113
  workstations 24
starting operations 209, 210
  job submission 210, 213
  on nonreporting workstations 211
special resources
  and LOOKAHEAD 53
  started tasks 212
  WTO operations 211
STARTTIME keyword of AROPTS 252
status changes in automatic recovery 261
status codes
  catalog management 348
  extended 346
  job log retrieval 349
  occurrence 345
status codes (continued)
operation 210, 345
status of days in calendars 60
status of days report for calendars 392
Status of the Long-Term Plan panel 182

STC
See started task

step restart 229—233
  catalog management 241
  considerations 222

Step Restart Selection List panel 229
step-level catalog management 236
STEPEVENTS keyword of EWTROPTS 227
STELIB 157

STOP command, MVS 337
STORELOG keyword 230

structure of batch-loader control statements 129
SUBFAILACTION keyword of JTOPTS 348
SUBMIT option 102

subtask 215

submitting jobs 205

subroutines
  EQQUSIN 215
  EQQUSINB 215
  EQQUSINO 215
  EQQUSINS 215
  EQQUSINT 215
  EQQUSINW 215

substituting variables in jobs 273—312

SUBSYS keyword 134

subsystem
  cancelling with MVS 338
  modifying with MVS 338
  starting with MVS 337
  stopping with MVS 337

success or failure of a job 227
successor and predecessor workstations
  default in long-term plan 183
  successor dependency resolution in ETT 271
successors
  in the current plan 195
  specifying 94

SUNDAY keyword 143

supplied variables 281
SUPPRESS IF LATE option 105, 114
SUPPRESS ACTION keyword of JTOPTS 348

suppressing variable substitution 276, 294
Sxxx error code 348

syntax diagrams, how to read xxviii

SYS1.PARMLIB 157
SYSCHK dataset 253
SYSSOUT CLASS option 104
SYSSOUT dataset 112
SYSUDUMP dataset 158

T

T extended status code 346
TABLE JCL directive 302

tabular variables 279
tail plan 169
temporary variables 284
TEXT command 92, 105
THURSDAY keyword 143
time
  default for an operation 33
  end of a work day 61
  input arrival 108
  transport time between workstations 33

TIME DEPENDENT option 105
time Specifications panel 108
time stamps 217
time zones 217
time-dependent operations 78, 105, 114
TOD clock, changing 217

TRACK keyword of JTOPTS 270
track log 195
tracking jobs on MVS 215
tracking jobs submitted by CICS or IMS 269
tracking non-Tivoli OPC work using ETT 267
transport time 33, 96
TRANSP keyword 153
trial current plan 192
trial long-term plan 182
triggering event 56, 268
triggering operations 112

TSAVE command 25, 112

TSO commands 317
  BACKUP 318
  JSUACT 320
  OPINFO 322
  OPSTAT 326
  SRSTAT 330
  WSSTAT 334

TUESDAY keyword 143
TYPE keyword 140

U

U extended status code 346
U occurrence status code 345
U operation status code 345

uncataloging datasets
  for job recovery 235

UNDECIDED status 171

update performed for owner ID 402
Updating Data Item panel 125

updating in batch 124

USAGE keyword 151

user field 322
user-defined variables 285
USERREQ keyword of AROPTS 262
using error codes 227
using ETT 272
using feedback options 105
using NetView 112
utilization of workstation resources
  daily planning report 412
utilization of workstations
  daily planning report 411
Uxxx error code 348

V
VALFROM keyword 140
VALFROMD keyword 155
VALFROMT keyword 155
valid-from date 77
validating MVS JCL 111
validation of variable values 290
VALTO keyword 140
variable dependency 287
variable substitution 273—312
  See also JCL, variable substitution
validation 290
variable table
  creating 285
  default values 273
  global 273
  specifying 64, 81, 297, 302
VARSUB keyword of OPCOPTS 276

W
W occurrence status code 345
W operation status code 345
WEDNESDAY keyword 143
WEEK keyword 143
WLM and OPC 313—315
work days 60
Work Station Access Method Information panel 39
work-day end time 10, 61
  in the long-term plan 178
work-days-only cyclic period
  with offsets 67, 87
work-days-only cyclic periods 62
WORKDAY keyword 143
workload manager
workload manager (WLM) function and OPC
  selecting WLM assist policies 313
setting a job as critical 313
workstation
  access method parameters 39
  alternate 246
  See also alternate workstations
availability 34
workstation (continued)
closing 36
computer
  See computer workstations
connecting to special resources 52
controlling 40
defining 23—40
defining reporting attributes 29
description database
  input to current plan 187
destination 33
dummy 27
duration 33
examples 5, 6
fixed resources 37, 41, 99
general
  See general workstations
job setup 112
nonreporting
  See nonreporting workstations
open intervals 35
operations using 399
parallel servers 31
plan for, daily planning report 410
R1 and R2 resources 37
recovery with catalog management 240
report 394
resource utilization report 193
selecting for operation 92
splittable 32
transport time 33
types 23
WTO 112
Workstation Resources and Servers panel 100
write to operator (WTO)
  deadline 105
general workstations 26
  operations 112, 211
WSFAILURE keyword of JTOPTS 246, 348
WSID keyword 147, 155
WSOFFLINE keyword of JTOPTS 246, 347
WSSTAT command
  introduction 334
WTO
  deadline 105
general workstations 26
  operations 112, 211
X
X extended status code 346
xxxx error code 348
Y
YEAR keyword 143

Z
Z extended status code 346
Communicating Your Comments to IBM

Tivoli Operations Planning and Control
Planning and Scheduling the Workload
Version 2 Release 3
Publication No. SH19-4376-02

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM. Whichever method you choose, make sure you send your name, address, and telephone number if you would like a reply.

Feel free to comment on specific errors or omissions, accuracy, organization, subject matter, or completeness of this book. However, the comments you send should pertain to only the information in this manual and the way in which the information is presented. To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

If you prefer to send comments by mail, use the reader's comment form (RCF) at the back of this book. If you wish, you can give the RCF to the local branch office or IBM representative for postage-paid mailing.

If you prefer to send comments by fax, use this number, which is in Italy:
39+06+596+62077

If you prefer to send comments electronically, use this network ID:
ROMERCF at VNET.IBM.COM

Make sure to include the following in your note:
- Title and publication number of this book
- Page number or topic to which your comment applies
Help us help you!

Tivoli Operations Planning and Control
Planning and Scheduling the Workload
Version 2 Release 3
Publication No. SH19-4376-02

We hope you find this publication useful, readable and technically accurate, but only you can tell us! Your comments and suggestions will help us improve our technical publications. Please take a few minutes to let us know what you think by completing this form.

Overall, how satisfied are you with the information in this book? Satisfied  □  Dissatisfied  □

How satisfied are you that the information in this book is: Satisfied  □  Dissatisfied  □
Accurate □  □
Complete □  □
Easy to find □  □
Easy to understand □  □
Well organized □  □
Applicable to your task □  □

Specific Comments or Problems:
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

Please tell us how we can improve this book:
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________
________________________________________________________________________

Thank you for your response. When you send information to IBM, you grant IBM the right to use or distribute the information without incurring any obligation to you. You of course retain the right to use the information in any way you choose.

Name
________________________________________________________________________
Address
________________________________________________________________________
Company or Organization
________________________________________________________________________
Phone No.
________________________________________________________________________
Help us help you!
SH19-4376-02

Tivoli OPC Information Development
Rome Tivoli Laboratory
IBM Italia S.p.A.
Via Sciangai, 53
00144 Rome
Italy

SH19-4376-02