

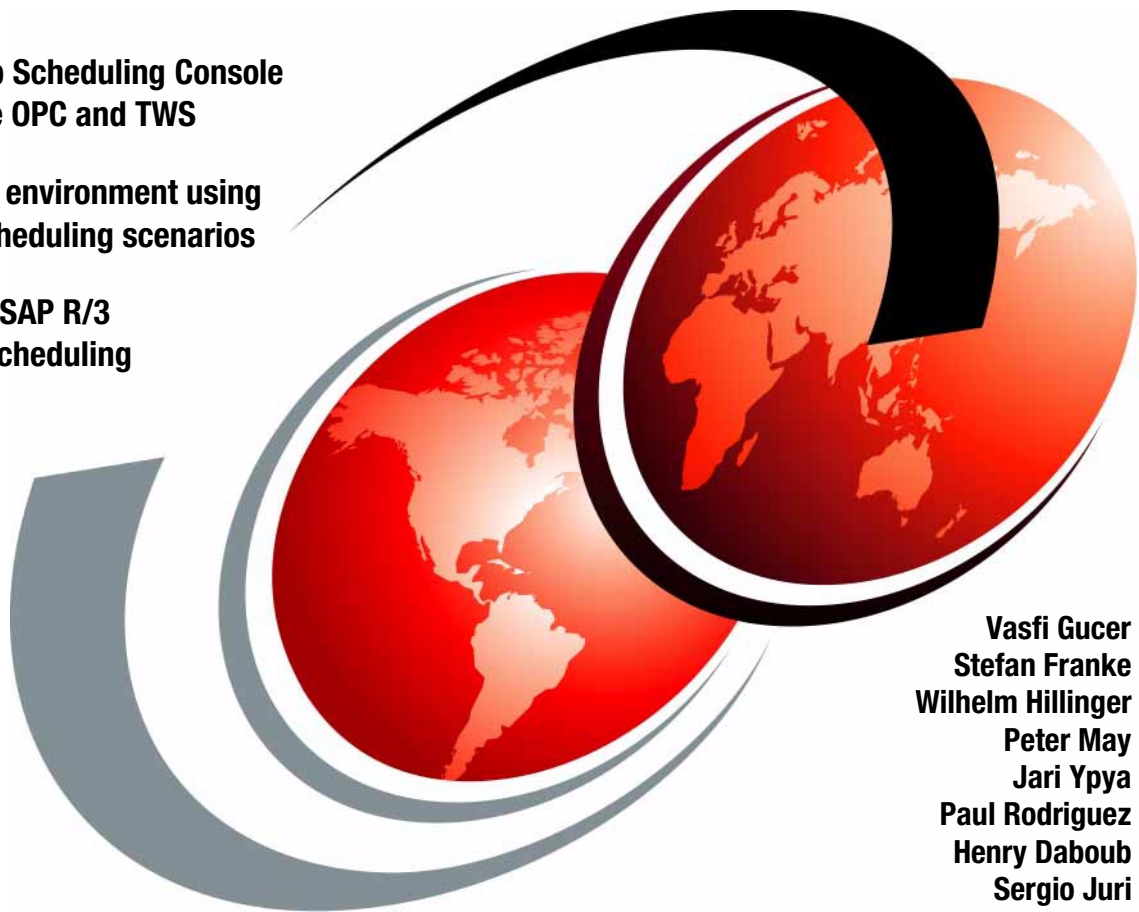
# **End-to-End Scheduling with OPC and TWS**

## **Mainframe and Distributed Environments**

Use the Job Scheduling Console  
to integrate OPC and TWS

Model your environment using  
realistic scheduling scenarios

Implement SAP R/3  
workload scheduling



Vasfi Gucer  
Stefan Franke  
Wilhelm Hillinger  
Peter May  
Jari Ypya  
Paul Rodriguez  
Henry Daboub  
Sergio Juri

[ibm.com/redbooks](http://ibm.com/redbooks)

# **Redbooks**





International Technical Support Organization

**End-to-End Scheduling with OPC and TWS  
Mainframe and Distributed Environments**

August 2000

**Take Note!**

Before using this information and the product it supports, be sure to read the general information in Appendix H, "Special notices" on page 395.

**First Edition (August 2000)**

This edition applies to Tivoli Operations, Planning and Control (OPC) 2.3 and Tivoli Workload Scheduler (TWS) V 7.0.

Comments may be addressed to:  
IBM Corporation, International Technical Support Organization  
Dept. OSJB Building 003 Internal Zip 2834  
11400 Burnet Road  
Austin, Texas 78758-3493

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

**© Copyright International Business Machines Corporation 2000. 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

<b>Figures</b> .....	xi
<b>Preface</b> .....	xvii
The team that wrote this redbook .....	xvii
Comments welcome .....	xx
 <b>Chapter 1. Introduction</b> .....	1
1.1 Job scheduling in the enterprise .....	1
1.2 Overview and basic architecture of OPC .....	2
1.2.1 OPC overview .....	2
1.2.2 OPC concepts and terminology .....	2
1.2.3 OPC architecture .....	4
1.3 Overview and basic architecture of TWS .....	5
1.3.1 TWS overview .....	5
1.3.2 TWS concepts and terminology .....	5
1.3.3 TWS architecture .....	7
1.4 Tivoli Enterprise product structure .....	7
1.4.1 Tivoli Management Framework .....	7
1.4.2 Tivoli Management Agent .....	8
1.4.3 Toolkits .....	9
1.5 Benefits of integrating OPC 2.3 and TWS 7.0 .....	9
1.6 A glimpse into the future .....	10
 <b>Chapter 2. Installing Tivoli OPC V2R3</b> .....	11
2.1 Planning your Tivoli OPC configuration .....	11
2.1.1 Hot standby Controller .....	11
2.2 Data store .....	12
2.3 Before installation .....	12
2.4 JES2 considerations .....	12
2.5 Adding SMF and JES exits for event tracking .....	13
2.6 Defining OPC subsystems .....	14
2.7 Allocating OPC datasets .....	15
2.8 Creating JCL procedures for OPC address spaces .....	15
2.9 Setting up the ISPF environment .....	15
2.9.1 Verifying OPC installation .....	17
2.10 The Job Scheduling Console .....	17
2.11 Supported platforms and minimum system requirements for JSC .....	18
2.12 Installing JSC .....	19
2.12.1 Starting the JSC .....	20
2.13 Creating a Tivoli Administrator .....	21
2.14 Installing and customizing the OPC Connector .....	26

2.14.1	Creating OPC Connector instances	27
2.15	Installing the TCP/IP server	30
2.15.1	Controlling access to OPC using Tivoli Job Scheduling Console	32
2.16	Summary	36
<b>Chapter 3. Tivoli Workload Scheduler 7.0 step-by-step installation</b>		<b>37</b>
3.1	Installation overview	37
3.2	System requirements	38
3.3	Installing TWS Engine 7.0 on Windows NT	38
3.4	Installing TWS Engine 7.0 on UNIX	39
3.5	Installing a dedicated Tivoli Framework to run the TWS or OPC	42
3.5.1	Configuring the TMR Server	55
3.6	Setting the security of TWS engines	70
3.6.1	A note about using the Tivoli Framework security model	72
3.7	Job Scheduling Console installation	72
3.8	Summary	77
<b>Chapter 4. Using the Job Scheduling Console GUI</b>		<b>79</b>
4.1	Starting the Job Scheduling Console	81
4.2	Job Scheduling Console for OPC	85
4.2.1	What you can do with the JSC	86
4.2.2	What you should do through OPC ISPF dialogs	86
4.2.3	Terminology used in the Job Scheduling Console	87
4.2.4	Moving to the Job Scheduling Console	89
4.3	Job Scheduling Console for TWS	110
4.3.1	What can you do with the JSC	113
4.3.2	What you should do using TWS legacy user interfaces	116
4.3.3	Job status mapping	116
4.3.4	Customized lists	117
4.4	Summary	122
<b>Chapter 5. Troubleshooting for Tivoli OPC</b>		<b>123</b>
5.1	Using keywords to describe a problem	123
5.2	Searching the software-support database	124
5.3	Problem-type keywords	124
5.4	Problem analysis procedures	127
5.4.1	Abnormal termination (ABEND or ABENDU) procedure	127
5.4.2	The Diagnostic File (EQQDUMP)	128
5.4.3	System dump dataset	129
5.4.4	LOOP procedure	130
5.4.5	Message (MSG) procedure	130
5.4.6	Performance (PERFM) procedure	132
5.4.7	WAIT procedure	132
5.4.8	Preparing a console dump	133

5.4.9 Information needed for all problems .....	135
5.5 Performing problem determination for tracking events .....	136
5.6 Trace for the TCP/IP Server .....	142
5.7 Trace for the OPC Connector .....	143
5.8 Trace for the Job Scheduling Console .....	145
<b>Chapter 6. Troubleshooting for TWS .....</b>	<b>149</b>
6.1 TWS troubleshooting checklist .....	149
<b>Chapter 7. Tivoli Workload Scheduling Agent for SAP R/3 .....</b>	<b>157</b>
7.1 The anatomy of an Extended Agent .....	157
7.2 What are the benefits of using TWS agent for SAP R/3? .....	161
7.3 SAP R/3 Extended Agent details .....	161
7.3.1 Tivoli Workload Scheduler and R/3 job states .....	162
7.4 Our environment and implementation steps .....	163
7.5 Installing of the TWS Extended Agent for SAP R/3 on AIX .....	164
7.5.1 Software Requirements .....	165
7.5.2 Installation .....	165
7.5.3 r3options file reference .....	167
7.5.4 r3batch.opts file reference .....	168
7.5.5 SAP R/3 configuration .....	169
7.6 Setting up for Tivoli Workload Scheduler 7.0 .....	174
7.6.1 Defining an SAP R/3 Extended Agent workstation .....	174
7.6.2 Defining SAP R/3 Extended Agent jobs .....	177
7.6.3 Defining SAP R/3 Extended Agent job streams .....	189
7.7 Troubleshooting SAP R/3 Extended Agent problems .....	194
7.7.1 Critical files .....	194
7.7.2 Information necessary for troubleshooting .....	195
7.7.3 Common problems and solutions .....	197
7.8 Summary .....	201
<b>Chapter 8. Enterprise scheduling scenarios .....</b>	<b>203</b>
8.1 Detailed description of our environment .....	203
8.1.1 Systems used .....	203
8.1.2 Environments and connections .....	205
8.2 Resource synchronization between OPC and TWS .....	209
8.2.1 OPC resource availability .....	210
8.2.2 Creating the special resource .....	212
8.2.3 Changing OPC Resources status from TWS .....	221
8.2.4 Conclusion .....	236
8.3 Changing TWS Resources from OPC .....	237
8.3.1 Resources in TWS .....	237
8.3.2 Changing TWS resources from OPC .....	238
8.4 Synchronization between OPC and TWS .....	245

8.4.1	Requirements . . . . .	245
8.4.2	Submitting a Job Stream/Job/Command into TWS via CLI . . . .	248
8.4.3	Create jobs for TWS Extended Agent for OS/390 . . . . .	248
8.4.4	Implementation . . . . .	252
8.4.5	Demonstration run . . . . .	260
8.5	User interaction with OPC and TWS . . . . .	265
8.5.1	Network for the scenario . . . . .	265
8.5.2	The solution . . . . .	267
8.6	Summary . . . . .	275
<b>Chapter 9. Tivoli OPC Tracker Agents . . . . .</b>		<b>277</b>
9.1	HW/SW requirements for OPC Tracker Agent for AIX/6000 . . . . .	277
9.1.1	Hardware requirements . . . . .	277
9.1.2	Software requirements . . . . .	277
9.2	Verifying that TCP/IP is operational . . . . .	278
9.3	Tivoli OPC Controller initialization statements . . . . .	279
9.4	Planning your Tracker Agent installation . . . . .	282
9.5	Creating a user group and user IDs . . . . .	283
9.6	Installing and customizing the Tracker Agent . . . . .	283
9.7	Using SMIT to install the required features (AIX only) . . . . .	284
9.8	Setting EQQHOME and EQQINSTANCE variables . . . . .	288
9.8.1	Setting the EQQHOME variable . . . . .	288
9.8.2	Setting the EQQINSTANCE variable . . . . .	288
9.9	Creating links between the directories . . . . .	289
9.10	Updating the configuration parameter file . . . . .	289
9.11	Customizing the directories . . . . .	296
9.12	Customizing the file permissions . . . . .	297
9.13	Restrictions and dependencies on system software . . . . .	298
9.14	Coordinating clock values . . . . .	299
9.15	Running scripts for Tivoli OPC . . . . .	299
9.15.1	Storing scripts . . . . .	299
9.15.2	Writing scripts . . . . .	299
9.15.3	Determining the shell under which scripts run . . . . .	300
9.15.4	Specifying a user ID . . . . .	300
9.15.5	Getting output from scripts . . . . .	300
9.15.6	Testing for errors from commands . . . . .	300
9.15.7	Specifying the path . . . . .	302
9.16	Starting the Tracker Agent . . . . .	302
9.17	Shutting down the Tracker Agent . . . . .	303
9.18	Restarting after an abnormal termination . . . . .	303
9.19	The Tracker Agent utilities . . . . .	304
9.20	Which agent to choose? OPC Tracker Agents or TWS Agents? . . .	305



<b>Chapter 10. Best practices</b>	307
10.1 OPC	307
10.1.1 Moving to OPC	307
10.1.2 OPC roles and responsibilities	308
10.1.3 Naming Standards	309
10.1.4 Resource modelling	311
10.1.5 Input arrival time	312
10.1.6 Deadline time	313
10.1.7 Database and plan relationship	313
10.1.8 Long term plan length	313
10.1.9 Current plan length	314
10.1.10 Calendar maintenance	314
10.1.11 Complex operation dependencies	315
10.1.12 Education	320
10.2 Tivoli Workload Scheduler	321
10.2.1 TWS architecture	321
10.2.2 What version of TWS should be used?	322
10.2.3 Choosing platforms	323
10.2.4 Operating system resources	323
10.2.5 .msg files	325
10.2.6 Ad hoc job/schedule submissions	326
10.2.7 Timeouts	326
10.2.8 Mailman and writer	327
10.2.9 Job Scheduling Console and Remote Console	327
10.2.10 Hardware considerations	328
10.2.11 Monitoring	328
10.2.12 High availability	329
10.2.13 Deployment	329
<b>Chapter 11. TWS Extended Agent for MVS and OS/390</b>	331
11.1 TWS Extended Agent for MVS and OS/390 features	331
11.2 Software prerequisites	331
11.3 Installation on the TWS side	332
11.3.1 Installation on TWS (Unix host)	332
11.3.2 Installation on TWS-(Windows NT host)	333
11.4 Installation on the OS/390 side	333
11.5 Defining an OS/390 Extended Workstation	337
11.6 Creating a method options file	341
11.7 Defining internetwork dependencies for OPC with the new JSC	343
11.7.1 Definitions for OPC Jobstreams launched by TWS	343
11.7.2 Definitions for OPC jobstreams monitored by TWS	345

<b>Appendix A. Tivoli OPC Version 2 Release 3 enhancements</b>	351
A.1 Job Scheduling Console	351
A.2 Catalog Management - Data Availability feature	351
A.3 Improved management of critical jobs	352
A.4 Improved availability with OS/390 Automatic Restart Manager Support	352
A.5 Program Interface (PIF) enhancements	352
A.6 Enhancements for non-OS/390 Tracker Agents	353
A.7 Usability enhancements	353
A.8 New and changed installation exits	354
A.9 New and changed initialization statements	354
<b>Appendix B. TWS Version 7 Release 0 enhancements</b>	357
B.1 Terminology changes	357
B.2 Job Scheduling Console	358
B.3 Overview of the new Job Scheduling Console	359
B.3.1 Database and Plan	360
B.4 Usage notes	361
B.5 Viewing TWS properties	362
B.6 Job Scheduling Console Connectors	362
B.7 Time zones	363
B.7.1 Enabling the time zone feature	363
B.8 Auditing	364
B.8.1 Enabling the audit feature	364
B.8.2 Auditing log format	365
B.9 Audit log header	368
B.10 Sample audit log entries	368
<b>Appendix C. Job Scheduling Console terminology</b>	371
C.1 JSC to OPC and TWS terminology translation	371
C.2 OPC to JSC and TWS terminology translation	375
C.3 TWS to JSC and OPC translation	378
<b>Appendix D. Migrating TWS smoothly from 5.X to 6.1</b>	381
D.1 Why do I need to upgrade TWS?	381
D.2 Migration checklist	381
D.2.1 Preparing for the upgrade	381
D.2.2 Performing the upgrade	384
D.2.3 Post installation tasks	385
D.2.4 Migrating FTAs from 5.x to 6.1	385
D.2.5 Procedure for completely removing TWS from Windows NT	386
<b>Appendix E. Where to find more information</b>	387
E.1 Available redbooks	387
E.2 Forums, user groups and FAQ sites	387

E.3 IBM Learning Center . . . . .	388
<b>Appendix F. Migration to TWS 7.0 . . . . .</b>	<b>389</b>
F.1 Viewing data locally . . . . .	389
F.2 Jnextday and writer changes . . . . .	389
F.3 MIPS and INTEL ABI platforms . . . . .	390
F.4 Behavioral changes . . . . .	390
F.5 Security migration . . . . .	390
<b>Appendix G. TWS log file record types . . . . .</b>	<b>391</b>
<b>Appendix H. Special notices . . . . .</b>	<b>395</b>
<b>Appendix I. Related publications . . . . .</b>	<b>399</b>
I.1 IBM Redbooks . . . . .	399
I.2 IBM Redbooks collections . . . . .	399
I.3 Other resources . . . . .	399
I.4 Referenced Web sites . . . . .	400
<b>How to get IBM Redbooks . . . . .</b>	<b>403</b>
IBM Redbooks fax order form . . . . .	404
<b>Abbreviations and acronyms . . . . .</b>	<b>405</b>
<b>Index . . . . .</b>	<b>407</b>
<b>IBM Redbooks review . . . . .</b>	<b>417</b>

**X** End-to-End Scheduling with OPC and TWS Mainframe and Distributed Environments

---

## Figures

1. The team that wrote this redbook . . . . .	xviii
2. JSC dataflow . . . . .	18
3. JSC Installation Splash . . . . .	20
4. JSC start-up . . . . .	21
5. Create Administrator . . . . .	22
6. Create a new Administrator . . . . .	23
7. Login names . . . . .	24
8. Group names . . . . .	25
9. Tivoli Administrator . . . . .	26
10. Usermap relationship . . . . .	34
11. Logon panel to OPC . . . . .	35
12. Tivoli Desktop . . . . .	36
13. Welcome . . . . .	43
14. Creating the tmesrwd account . . . . .	44
15. Advanced user rights . . . . .	44
16. Logout message . . . . .	44
17. User Information . . . . .	45
18. Installation Password . . . . .	45
19. Remote User File Access . . . . .	46
20. Setup Type . . . . .	46
21. Enter License Key . . . . .	47
22. Enter Database Directory . . . . .	47
23. Database installation window . . . . .	48
24. TME Installation Complete . . . . .	48
25. Select Destination Path . . . . .	49
26. Select Program Folder . . . . .	49
27. Installation Complete . . . . .	50
28. Tivoli Desktop login . . . . .	50
29. Patch Installation . . . . .	51
30. Error message . . . . .	51
31. Browse directory . . . . .	52
32. Install Patch . . . . .	53
33. Patch Install information . . . . .	54
34. Edit Login menu . . . . .	55
35. Set Login Names . . . . .	56
36. Create Policy Region menu . . . . .	56
37. Create Policy Region . . . . .	57
38. Setting Managed Resources . . . . .	57
39. Set Managed Resources . . . . .	58
40. Create Managed Node . . . . .	59

41. Client Install . . . . .	59
42. Add Clients . . . . .	60
43. File Browser . . . . .	60
44. Install Options . . . . .	61
45. Client Install confirmation . . . . .	62
46. Patch installation . . . . .	63
47. Install Patch . . . . .	64
48. Install Product . . . . .	65
49. Media selection . . . . .	66
50. Product selection for install . . . . .	67
51. Install TWS Connector . . . . .	68
52. Multiple TWS Connector instance scenario . . . . .	70
53. Tivoli Job Scheduling Console . . . . .	74
54. JSC Installation . . . . .	74
55. Select language . . . . .	75
56. Choose Install folder . . . . .	75
57. Shortcut location . . . . .	76
58. Installation successful message . . . . .	76
59. JSC start . . . . .	80
60. Starting the JSC . . . . .	81
61. JSC password prompt . . . . .	82
62. JSC release level notice . . . . .	83
63. User preferences file . . . . .	83
64. Open Location . . . . .	84
65. Starting with the JSC . . . . .	84
66. Creating a Job Instance list . . . . .	90
67. Job instance properties . . . . .	91
68. Choosing job instance status . . . . .	92
69. Listing errored jobs . . . . .	93
70. Making a Ready List for a JCL prep workstation . . . . .	94
71. Ready List for a JCL Prep Workstation . . . . .	95
72. Job properties . . . . .	96
73. Changing job status . . . . .	97
74. Creating a new Job Stream . . . . .	98
75. Job stream properties . . . . .	99
76. Defining jobs . . . . .	100
77. Job (operation) details . . . . .	101
78. Job (operation) icon . . . . .	102
79. Defining internal dependencies . . . . .	103
80. Dependency arrow . . . . .	104
81. Scheduling the job stream . . . . .	105
82. Using rules ISPF dialog panel . . . . .	106
83. Using rules JSC dialog . . . . .	107

84. Monthly display of calculated run days . . . . .	108
85. Annual display of calculated run days . . . . .	109
86. Combined display of calculated run days . . . . .	110
87. Legacy GUI . . . . .	111
88. JSC and customized views . . . . .	113
89. Create Plan List . . . . .	118
90. Abended jobs list properties (general page) . . . . .	119
91. Running jobs list properties (general page) . . . . .	120
92. Create New Prompt Status list. . . . .	121
93. Prompt Status List properties. . . . .	122
94. X-Agent network . . . . .	157
95. Extended Agent processing. . . . .	159
96. SAP R/3 environment . . . . .	164
97. Defining a new SAP R/3 Extended Agent Workstation . . . . .	175
98. Fill in the necessary fields in the Properties window . . . . .	176
99. Displaying the newly-defined SAP R/3 Extended Agent Workstation . . . . .	176
100. Defining a new job. . . . .	177
101. Selecting the Task type . . . . .	178
102. Job Definition (general) . . . . .	179
103. Job Definition Task, Select SAP Pick List . . . . .	180
104. Showing all SAP jobs on SAP R/3 application server for user maestro . . . . .	181
105. Propagated Information from SAP Pick List in Task tab . . . . .	181
106. Displaying the new job in Job Definitions List . . . . .	182
107. Defining a new Job . . . . .	183
108. Selecting the task type . . . . .	184
109. Job Definition (general) . . . . .	185
110. Job Definition (task). . . . .	186
111. SAP R/3 Job Definition (general R/3 Job). . . . .	187
112. Defining an SAP R/3 job (steps tab) . . . . .	188
113. Job Definition Window (Job-ID after saving to R/3 database). . . . .	189
114. Defining a Job Stream. . . . .	190
115. Defining an SAP R/3 Job Stream . . . . .	191
116. Adding jobs into an empty job stream . . . . .	191
117. Entering jobs into a job stream . . . . .	192
118. Jobs (represented by icons) within a job stream . . . . .	193
119. Adding dependencies between jobs . . . . .	193
120. Showing job dependencies . . . . .	194
121. Connections needed for JSC operation . . . . .	206
122. Framework setup. . . . .	207
123. TWS network and external connections . . . . .	208
124. Environment for TWS (OPC interaction) . . . . .	209
125. SRSTAT command . . . . .	211
126. Coding the SRSTAT command . . . . .	212

127.Creating a New Logical Resource . . . . .	213
128.Naming the resource . . . . .	214
129.Selecting Job Resources. . . . .	215
130.Adding a Logical Resource Dependency . . . . .	216
131.Defining Job Logical Resource Dependency Settings . . . . .	217
132.Listing scheduled jobs . . . . .	218
133.Job status details. . . . .	219
134.Listing resource status . . . . .	219
135.Selecting Jobs Waiting for Resource . . . . .	220
136.Jobs Waiting on Resource . . . . .	220
137.Changing OPC resources from TWS (Solution 1). . . . .	222
138.Define X/AGENT using the vmsjes method . . . . .	223
139.Define new job in TWS using MVSJES method . . . . .	224
140.Enter job name and parameters . . . . .	224
141.Submit job from TWS . . . . .	225
142.Select LAUNCHYARI job . . . . .	225
143.Changing OPC resources from TWS (Solution 2). . . . .	226
144.Create X/AGENT workstation using MVSOPC method . . . . .	227
145.Create new TWS job with MVSOPC method . . . . .	228
146.Enter OPC Job Stream (Application) name. . . . .	228
147.Submit job from TWS . . . . .	229
148.Select TWS job TWSYARI . . . . .	229
149.Changing OPC resources from TWS (Solution 3). . . . .	230
150.Define X/AGENT using vmsjes method . . . . .	231
151.Create a new job definition . . . . .	233
152.Enter name of the script . . . . .	234
153.Submit job . . . . .	235
154.Enter Job name . . . . .	235
155.Create new resource . . . . .	238
156.Properties of a resource . . . . .	238
157.Interact TWS Resources from OPC . . . . .	239
158.Interaction between OPC and TWS . . . . .	247
159.Create a new job definition in TWS. . . . .	249
160.Task type selection panel . . . . .	249
161.Extended Agent selection list . . . . .	250
162.General Job definition properties for Extended Agent Job, NOTIFY. . . . .	251
163.Syntax for adding OPC application, COLLECT#01. . . . .	251
164.Rule-based run cycle for job stream account#01 . . . . .	252
165.Graphical display of the run cycle used. . . . .	253
166.Content of script submitted by OPC Controller to Tracker Agent . . . . .	253
167.Contents of the /opt/maestro/scripts/submit1 script . . . . .	254
168.Job stream STATISTIC . . . . .	254
169.General job properties of the CALCULATE job. . . . .	255



170.Task job properties of the CALCULATE job . . . . .	256
171.Deadline of CALCULATE job as defined in Job properties job stream . .	257
172.Listing of the /opt/maestro/scripts/threshold script . . . . .	257
173.General properties of the NOTIFY job. . . . .	258
174.Task properties of the NOTIFY job . . . . .	259
175.Resource properties of the AMI_MAX Special Resource . . . . .	259
176.Availability interval properties of the AMI_MAX special resource . . . . .	260
177.Running the ACCOUNT#01 application on OPC . . . . .	260
178.Running the STATISTIC jobstream on TWS (last line) . . . . .	261
179.Status of the STATISTIC jobstream and job notify on TWS . . . . .	261
180.Status of COLLECT#01 OPC jobstream waiting for special resource . .	262
181.Status of the AMI_MAX OPC special resource . . . . .	263
182.OPC special resource made available . . . . .	263
183.OPC jobstream, COLLECT#01, finished successfully . . . . .	264
184.Notify TWS job finished successfully. . . . .	264
185.Network used for this scenario . . . . .	266
186.Selecting internetwork dependency . . . . .	267
187.Creating the internetwork dependency . . . . .	268
188.Selecting the OPC workstation . . . . .	269
189.Linking OPC Predecessor to TWS Job Stream. . . . .	270
190.TWS Job Stream . . . . .	271
191.Arranging Job Stream icons . . . . .	272
192.TWS Job Stream Dependent on an OPC application . . . . .	273
193.Monitoring the OPC Job Stream . . . . .	274
194.Monitoring the TWS Job Stream . . . . .	275
195.Ports . . . . .	283
196.Controller parameter file of the Tracker Agent (Part 1) . . . . .	291
197.Controller parameter file of the Tracker Agent (Part 2) . . . . .	292
198.Filtering a Job stream View in the JSC . . . . .	310
199.Complex dependency . . . . .	315
200.Dependency simplified with non-reporting workstation . . . . .	316
201.Creating a non-reporting workstation . . . . .	317
202.Job waiting for resources . . . . .	319
203.Issuing the EX command . . . . .	320
204.Course description on the Web . . . . .	321
205.Creating Xagent workstation . . . . .	338
206.Workstation definitions . . . . .	339
207.Workstation view . . . . .	341
208.create new job definition . . . . .	343
209.Task type selection . . . . .	344
210.Internetwork definitions . . . . .	344
211.Add Internetwork dependencies . . . . .	346
212.Finding the Network agent . . . . .	346

213.	Define the internetwork dependency . . . . .	347
214.	Creating links to the TWS network . . . . .	348
215.	Internetwork dependency at the Schedule level . . . . .	348
216.	Add rows . . . . .	349
217.	Complete the dependency . . . . .	349
218.	Default Database lists . . . . .	359
219.	Default Plan lists . . . . .	360

---

## Preface

The beginning of the new century sees the data center with a mix of work, hardware, and operating systems previously undreamt of. Today's challenge is to manage disparate systems with minimal effort and maximum reliability. People experienced in scheduling traditional host-based batch work must now manage distributed systems, and those working in the distributed environment must take responsibility for work running on the corporate OS/390 system.

This redbook considers how best to provide end-to-end scheduling using Tivoli Operations Planning and Control, Version 2 Release 3 (OPC), and Tivoli Workload Scheduler, Version 7 Release 0 (TWS).

In this book, we provide the information to install the necessary OPC and TWS software components and configure them to communicate with each other.

In addition to technical information, we will consider various scenarios that may be encountered in the enterprise and suggest practical solutions. We will describe how to manage work and dependencies across both environments using a single point of control.

---

### The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Austin Center.



*Figure 1. The team that wrote this redbook*

**Vasfi Gucer** is a Project Leader at the International Technical Support Organization, Austin Center. He has worked for IBM Turkey for 10 years and has been with the ITSO since January 1999. He has more than six years experience in the areas of systems management and networking hardware and software on distributed platforms. He has worked on various Tivoli customer projects as a systems architect in Turkey and the U.S. Vasfi is also a Certified Tivoli Consultant.

**Stefan Franke** is a Customer Support Engineer of IBM Global Services based in the Central Region Support Center in Mainz, Germany. He has eight years experience in OS/390 System Managements products, specializing in OPC, and one year in TWS. His areas of expertise include installation and tuning, defect and non-defect Problem Determination, and onsite Customer support.

**Wilhelm Hillinger** is an I/T specialist working as a systems engineer in IBM Global Services in Austria. He has 15 years experience with UNIX systems and network management as well as three years experience in Windows NT. He has been working at IBM for two years. His areas of expertise include all Tivoli core applications for the management of distributed environments. He is also in charge of supporting all Tivoli Workload Scheduler (TWS) products and, with the newest version of TWS, their integration into the Tivoli Framework.

**Peter May** is a Technical Trainer based in the UK specializing in Systems Management. He has 30 years experience in computer operations management. Peter is currently working for the Tivoli Academy in Brussels, Belgium, and is responsible for the OPC and Tivoli Service Desk for OS/390 curriculum. He writes courses and teaches them worldwide and is a Member of the Institute of IT Training.

**Jari Ypya** is an IT Specialist in IBM Global Services in Finland. He has seven years experience in the Distributed computing field including UNIX/Linux and Windows systems. He has two years experience with Tivoli Workload Scheduler (TWS) and other Tivoli Framework products. His areas of expertise include TWS and e-Business applications on the AIX and Linux operating systems. Jari also holds an AIX System Administrator Certificate.

**Paul Rodriguez** is a Staff Engineer with the Tivoli Systems Support Center in Austin, Texas. Paul has 14 years of technical experience in hardware and software support. For the last two years, he has been providing technical support for the Tivoli Workload Scheduler (TWS). His area of expertise is in the integration of TWS with SAP, PeopleSoft, MVS (JES, OPC, CA7), ORACLE, and BAAN.

**Henry Daboub** is currently a member of the Tivoli Customer Support Global Response Team. The GRT provides onsite support to Tivoli customers with an emphasis on large environments. Henry has more than 12 years experience in software development and support and has supported IBM workload management products since 1997.

**Sergio Juri** is a member of the Product Strategy team for Tivoli Workload Scheduler. He has been an ITSO Specialist, and had previously authored nine books on various Systems Management areas, like problem and change management and workload scheduling. Sergio has presented at conferences like Guide-Share, Planet Tivoli, AOTC and many other US and international events. Sergio started his career in IBM Argentina, and is now working for Tivoli Systems in the US. His e-mail address is [sergio.juri@tivoli.com](mailto:sergio.juri@tivoli.com).

The team would like to express special thanks to Warren Gill and Cesare Pagano for their major contributions to this book. They did a great job in finding the contacts, reviewing the material, and, most importantly, encouraging us throughout the whole redbook process.

Thanks also to Finn Bastrup Knudsen and Doug Specht from IBM for reviewing the book for us.

We also would like to thank the following people for their invaluable contributions to this project:

Maria Pia Cagnetta, Anna Dawson, Eduardo Esteban, Jamie Meador, Geoff Pusey, Bob Rodriguez, Craig Sullivan, Jose Villa  
Tivoli Systems

Caroline Cooper, Morten Moeller  
International Technical Support Organization, Austin Center

Theo Jenniskens, Wolfgang Heitz, Ingrid Stey  
IBM

---

## Comments welcome

### Your comments are important to us!

We want our Redbooks to be as helpful as possible. Please send us your comments about this or other Redbooks in one of the following ways:

- Fax the evaluation form found in “IBM Redbooks review” on page 417 to the fax number shown on the form.
- Use the online evaluation form found at [ibm.com/redbooks](http://ibm.com/redbooks)
- Send your comments in an Internet note to [redbook@us.ibm.com](mailto:redbook@us.ibm.com)

---

## Chapter 1. Introduction

This redbook provides practical examples for using Tivoli Operations Planning and Control, Version 2 Release 3 (OPC), and Tivoli Workload Scheduler, Version 7 Release 0 (TWS 7.0), to provide end-to-end scheduling in an industry environment.

We will discuss the basic architecture of the OPC and TWS products and show how they work with the Tivoli Framework and the Job Scheduling Console (JSC). The JSC is a common graphical user interface (GUI) introduced in these versions of the products, which gives the user one standard interface with which to manage work on both OPC and TWS.

We will recommend techniques for the installation of OPC, TWS, Tivoli Framework, and JSC and provide troubleshooting tips. We will document how to achieve a successful integration between the Tivoli Framework and JSC.

Job scheduling in the enterprise will be considered; several real-life scenarios will be examined and best practice solutions recommended.

---

### 1.1 Job scheduling in the enterprise

The enterprise needs orderly reliable sequencing and management of process execution throughout the enterprise. Scheduling is the nucleus of the data center, but the IT environment consists of multiple strategic applications, such as SAP/3 and Oracle, payroll, invoicing, e-commerce, and order handling running across different operating systems, such as UNIX, OS/390, and AS/400, and on hardware from vendors, such as HP, Sun, and IBM. There are legacy systems that must be maintained running alongside new Internet-enabled mission-critical systems.

Workloads are increasing, accelerated by e-commerce. Staffing and training requirements increase, and too many platform experts are needed. There are too many consoles and no overall point of control. 24X7 availability is essential and must be maintained through future migrations, mergers, acquisitions, and consolidations. Also, batch processing is becoming even more critical because it provides the infrastructure to other applications.

There can be dependencies between jobs running in different environments. For example, a customer can fill out an order form on their Web browser that will trigger a UNIX job that acknowledges the order, an AS/400 job that orders parts, an OS/390 job that debits the customer's bank account, and a Windows NT job that prints a document and address labels.

The solution is to use OPC and TWS working together across the enterprise. The Job Scheduling Console provides a centralized control point with a single interface to the workload regardless of whether that workload is running under OPC or TWS.

---

## **1.2 Overview and basic architecture of OPC**

OPC has been scheduling and controlling batch workloads in data centers since 1977. The product has been extensively developed and extended to meet the increasing demands of customers worldwide. An overnight workload consisting of 100,000 production jobs is not unusual, and OPC can easily manage this.

### **1.2.1 OPC overview**

OPC databases contain information about the work that is to be run, when it should run, and the resources that are needed and available. This information is used to calculate a forward forecast called the long term plan. Data center staff can check this to confirm that the desired work is being scheduled when required. The long term plan usually covers a time range of four to twelve weeks. A second plan is produced that uses the long term plan and databases as input. The current plan usually covers 24 hours and is a detailed production schedule. OPC uses this to submit jobs to the appropriate processor at the appropriate time. All jobs in the current plan have OPC status codes that indicate the progress of work. When a job's predecessors are complete, OPC considers it ready for submission. It checks that all requested resources are available, and when these conditions are met, it causes the job to be submitted.

OPC consists of a Controller and trackers. The Controller manages the databases and the plans and causes the work to be submitted at the appropriate time. A tracker is required for every operating system on which OPC controlled work runs. The tracker records details of job starts and ends and passes that information to the Controller, which updates the current plan with statuses.

It is very important to understand that OPC contains a logical model of resources. The data center can choose how much information they want to define in OPC databases. The more accurate they make the model, the more accurate OPC plans are and the more efficiently work is processed.

### **1.2.2 OPC concepts and terminology**

OPC uses the following important concepts:



## **Plans**

OPC builds operating plans from your descriptions of the production workload. First, a long-term plan (LTP) is created, which shows (usually, for one or two months) the job streams (applications) that should be run each day and the dependencies between jobstreams. Then, a more detailed current plan is created. The current plan is used by OPC to submit and control jobs (operations). You can simulate the effects of changes to your production workload, calendar, and installation by generating trial plans.

## **Job streams**

A job stream is a description of a unit of production work. It includes a list of the jobs (related tasks) associated with that unit of work. For example, a payroll job stream might include a manual task in which an operator prepares a job, several computer-processing tasks in which programs are run to read a database, update employee records, and write payroll information to an output file, and a print task that prints paychecks.

## **Workstations**

OPC supports a range of work process types, called workstations, that map the processing needs of any task in your production workload. Each workstation supports one type of activity. This gives you the flexibility to schedule, monitor, and control any data center activity, including:

- Job setup (manual and automatic)
- Jobs
- Started tasks
- NetView communication
- Print jobs
- Manual preprocessing or postprocessing activity.

## **Special resources**

You can use OPC special resources to represent any type of limited resource, such as tape drives, communication lines, or a database. A special resource can be used to serialize access to a dataset or to limit the number of file transfers on a network link. The resource does not have to represent a physical object in your configuration, although it often does.

## **Dependencies**

Most data processing activities need to occur in a specific order. Activities performed out of order can create invalid output or corrupt your corporate data. This results in costly reruns, missed deadlines, and dissatisfied customers.

In OPC, you can specify dependencies for jobs when a specific processing order is required.

### **Calendars**

OPC uses calendar information so that jobstreams are not scheduled to run on days when processing resources are not available (for example, Sundays and holidays). This information is stored in an OPC calendar. OPC supports multiple calendars for enterprises where different departments have different work days and free days or when multiple data centers in different states or regions are controlled from a single site.

### **Business processing cycles**

Tivoli OPC uses business processing cycles, or periods, to calculate when your jobstreams should be run. When you create a jobstream, you specify when it should be planned using a run cycle. You can use rule-based run cycles to specify run days using rules, such as *Third Thursday in July* or *Every work day in week 40*, where you select the words from a multiple-choice panel.

## **1.2.3 OPC architecture**

OPC consists of a *Controller* and one or more *trackers*. The Controller runs on an OS/390 system. The Controller manages the OPC databases and the long term and current plans. The Controller schedules work and causes jobs to be submitted to the appropriate system at the appropriate time.

Trackers are installed on every system managed by the Controller. The tracker is the link between the Controller and the managed system. The tracker submits jobs when the Controller instructs it to do so, and it passes job start and job end information back to the Controller.

Trackers are available for OS/390, OS/400, AIX/6000, OS/2, Sun Solaris, SunOS, Digital OpenVMS, Windows NT, SGI IRIX, HP-UX, and OS/390 USS. It is possible to schedule and track jobs on applications, such as SAP R/3.

The main method of accessing the Controller is via ISPF panels, but several other methods are available including Program Interfaces, TSO commands, and a GUI interface.

Version 2.3 introduced a new Java GUI, the Job Scheduling Console (JSC). The JSC provides a common interface to both OPC and TWS.

---

## **1.3 Overview and basic architecture of TWS**

TWS is a development of the Unison Maestro product that has a 15 year track record. During the processing day, TWS' production control programs manage the production environment and automate most operator activities. It prepares jobs for execution, resolves interdependencies, and launches and tracks each job. Because jobs begin as soon as their dependencies are satisfied, idle time is minimized and throughput improves significantly. Jobs never run out of sequence, and, if a job fails, TWS handles the recovery process with little or no operator intervention.

### **1.3.1 TWS overview**

There are two basic aspects to job scheduling in TWS: The database and the plan. The database contains all the definitions for scheduling objects, such as jobs, job streams, resources, and workstations. It also holds statistics of job and job stream execution as well as information on the user ID that created an object and when an object was last modified. The plan contains all job scheduling activity planned for a period of one day. In TWS, the plan is created every 24 hours and consists of all the jobs, job streams, and dependency objects that are scheduled to execute for that day. All job streams for which you have created a run cycle are automatically scheduled and included in the plan. As the day goes by, the jobs and job streams that do not execute successfully can be rolled over into the next day's plan.

### **1.3.2 TWS concepts and terminology**

TWS uses the following important concepts:

#### **Job Streams and Calendars**

Central to TWS' ability to manage batch job execution are the job streams created using the Job Scheduling Console. Each job stream is scheduled to run on a specific set of dates and times and consists of a list of jobs that execute as a unit, such as the weekly backup application, along with times priorities and other dependencies that determine the exact order of execution.

Job Streams are dated using actual dates, days of the week, or calendars. A calendar is a set of specific dates. You can create as many calendars as required to meet your scheduling needs. For example, you can define a calendar, named PAYDAYS, containing a list of pay dates, a calendar, named

MONTHEND, containing a list of each last business day of the month for the current year, and a calendar, named HOLIDAYS, containing a list of your company's holidays. At the start of each processing day, TWS automatically selects all the job streams that run on that day and carries forward incomplete job streams from the previous day.

### **Workstations**

A workstation is usually an individual computer on which jobs and job streams are executed. A workstation definition is required for every computer that executes jobs in the TWS network. Primarily, workstation definitions refer to physical workstations.

However, in the case of extended agents and network agents, the workstations are logical definitions that must be hosted by a physical TWS workstation.

There are several types of workstations in a TWS network:

#### **Job Scheduling Console Client**

Any workstation running the Job Scheduling Console GUI can manage the TWS plan and database objects. The Job Scheduling Console works like a remote console and can be installed on a machine that does not have the TWS engine installed.

#### **Master Domain Manager**

The Domain Manager in the topmost domain of a TWS network. It contains the centralized database files used to document scheduling objects. It creates the production plan at the start of each day and performs all logging and reporting for the network.

#### **Domain Manager**

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

#### **Backup Domain Manager**

A backup domain manager is a fault-tolerant agent capable of assuming the responsibilities of its Domain Manager.

#### **Fault-tolerant Agent**

A workstation capable of resolving local dependencies and launching its jobs in the absence of a Domain Manager.

#### **Standard Agent**

A workstation that launches jobs only under the direction of its Domain Manager.

#### **Extended Agent**

A logical workstation definition that enables you to launch and control jobs on other systems and applications, such as Baan, PeopleSoft, Oracle Applications, SAP R/3, and MVS JES2 and JES3.

#### **Network Agent**

A logical workstation definition for creating dependencies between jobs and job streams in separate TWS networks.

### **1.3.3 TWS architecture**

TWS consists of a Master Domain Manager that contains the centralized database files used to document scheduling objects. It creates the production plan at the start of each day and performs all logging and reporting for the network.

All communications to agents are routed through the Domain Manager, which is the management hub in a domain. The network can be managed by a mix of agents. Fault-tolerant agents are capable of resolving local dependencies and launching their jobs should a network interruption cause a loss of communication with their Domain Managers because each one is given a set of scheduling instructions at the beginning of every processing day.

Version 7.0 introduced a new Java GUI, the Job Scheduling Console (JSC). The JSC provides a common interface to both TWS and OPC

---

## **1.4 Tivoli Enterprise product structure**

The complete Tivoli Enterprise product suite consists of three components: Framework, applications, and toolkits.

### **1.4.1 Tivoli Management Framework**

The Tivoli Management Framework is the foundation for other Tivoli Enterprise and third-party management products. It provides the graphical

desktop, object-oriented databases, and base services used by other products

All Tivoli Enterprise applications share the Tivoli Management Framework, an open object-oriented framework that includes a set of managers, brokers, and agents that conform to the Object Management Group/Common Object Request Broker Architecture (OMG/CORBA) specification. OMG/CORBA technology allows major differences between computer operating systems to be hidden from the Tivoli Enterprise user, and it allows key services to be encapsulated in objects that can be used by multiple management applications.

The Tivoli Management Framework provides platform independence, a unifying architecture for all applications, and the ability for third-party vendors to easily adapt to, or plug into, the framework. It also provides a rich set of Application Program Interfaces (APIs) and services enabling customers to write applications that also plug into, or leverage, the Tivoli Management Framework. Tivoli APIs, which have been adopted by the Desktop Management Task Force (DMTF) and the Open Group (formerly X/Open) as a basis for a systems management framework provide common network and systems management services including scheduling, transaction support, configuration profiles, and a generic object database user facility.

#### 1.4.2 Tivoli Management Agent

Tivoli Enterprise applications, such as TWS, are installed on top of the framework

The Tivoli Management Agent greatly extends the scalability of Tivoli Enterprise and significantly increases the number of resources Tivoli Enterprise can handle. At the same time, it enables those resources to be used more efficiently. Systems running the agent do not maintain a client database; so, the amount of storage the agent requires is less than 2 MB.

The Tivoli Management Agent components include:

- **Endpoint** - An endpoint is any machine managed in the Tivoli Management Region (TMR). Endpoints receive profile distributions, execute tasks, run monitors, and send events.
- **Gateway** - A gateway performs all communications with its assigned endpoints without requiring additional communications with the TMR Server. The gateway invokes endpoint methods on the endpoints or runs gateway methods for the endpoint.

- **Endpoint manager** - An endpoint manager establishes and maintains the relationship between an endpoint and a gateway and is automatically created on the TMR Server when you install the server.

The endpoint and gateway enable Tivoli Enterprise to become a three-tiered management hierarchy. The chief advantage of endpoints is scalability. Each gateway can manage thousands of endpoints, and each Tivoli Management Server can manage up to 200 gateways. As a result, you can gather required management information from thousands of endpoint machines and remotely manage those machines with very little overhead. Also, because an endpoint installation requires only 1 to 2 MB of disk space, endpoints consume few computer resources.

After the initial installation process, all endpoint communications are handled by a gateway instead of the Tivoli Management Server. Shifting a share of the management processes to the gateway reduces computing demand on the TMR Server. Typically, machines installed as endpoints are not used for the daily management operations in a network. Therefore, you cannot create, modify, or delete Tivoli Enterprise objects from an endpoint – The TMR Server performs these functions.

### 1.4.3 Toolkits

Toolkits are supplied to enable customers to extend the functions of applications and develop new applications using standard APIs.

---

## 1.5 Benefits of integrating OPC 2.3 and TWS 7.0

Both OPC and TWS have individual strengths. While an enterprise running OS/390 and distributed systems could schedule and control work using only one of these tools, a complete solution requires OPC and TWS to work together.

OPC's long term plan gives peace of mind by showing the workload forecast for weeks into the future. TWS fault-tolerant agents keep scheduling work even when it loses communication with the Domain Manager. OPC manages huge numbers of jobs through a sysplex of connected OS/390 systems. TWS Extended Agents can control work on applications, such as SAP R/3 and Oracle.

Data centers that need to schedule and control significant amounts of both host OS/390 and distributed work will be most productive when they get their OPC and TWS systems connected and working cooperatively.

This redbook will show you how to achieve this.

---

## 1.6 A glimpse into the future

Tivoli's end-to-end scheduling strategy is to move OPC and TWS products closer to each other until the products are combined in one Tivoli Scheduler product.

The new Job Scheduling Console (JSC), which will be discussed in detail in Chapter 4, "Using the Job Scheduling Console GUI" on page 79, is a very important step in this transformation since it provides the same interface and the same paradigms for both TWS and OPC.

In later releases, the JSC is to be enhanced to incorporate all functions that are available from OPC ISPF Panels and TWS Composer and Conductor interfaces.

In the future, it is expected that the functional differences between the two products will be eliminated by implementing the missing functions in the products, and, eventually, TWS and OPC will be evolved into a single end-to-end scheduling product.



---

## Chapter 2. Installing Tivoli OPC V2R3

This chapter describes the tasks that are necessary to install Tivoli OPC V2R3 successfully. It does not describe every step required to install the base product because the *Tivoli OPC V2R3 Installation Guide*, SH19-4379, gives a full, accurate, and complete description of the installation process, and it is not necessary to duplicate that information here. However, we will give detailed instructions on how to install the new Job Scheduling Console (JSC) and its prerequisites, such as the OPC Connector and TCP/IP Server support.

---

### 2.1 Planning your Tivoli OPC configuration

Before installing the product, you have to choose the configuration and connection type between the controlling and controlled systems of OPC that fits your needs. OPC systems can be connected through any of the following methods:

- Shared DASD
- XCF Communication links
- VTAM links
- APPC applies to OPC tracker agents for AS400. TCP/IP applies to OS/2 and many UNIX environments.

The documentation does not give any recommendations; so, we suggest that the XCF link be used because it gives you the ability to implement a Hot standby Controller and use the new feature called Data store. Moreover, it is the fastest connection. Be aware that XCF local mode is not supported. If you need ideas about possible configurations and connections, you will find many examples on page 167 of the *Tivoli OPC V2R3 Installation Guide*, SH19-4379.

#### 2.1.1 Hot standby Controller

If you connect your Tivoli OPC systems using the MVS 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 the Controller if the Controller fails or if the MVS system on which it was active fails. You can create a standby Controller on one or more Tivoli OPC controlled systems within the XCF group. Each standby system must have access to the same resources as the Controller.

---

## 2.2 Data store

The new Catalog Management Data Availability feature improves OPC performance for job restart and job log retrieval functions. Job runtime information, such as the sysout datasets, is maintained locally on the tracked system. The Controller retrieves this information only when it is 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. This new feature is especially useful when a joblog archiving product is used concurrently with OPC and prevents the use of OPC EXIT11 (joblogretrieval exit), which you have to write yourself.

---

## 2.3 Before installation

The Program Directory provided with your Tivoli OPC distribution tape may include technical information that is more recent than the information provided in this guide. In addition, the Program Directory describes the program temporary fix (PTF) level of the Tivoli OPC licensed program that is shipped to you. It is an important document when you install Tivoli OPC. The Program Directory contains instructions for unloading the Tivoli OPC software and information about additional maintenance for your level of the distribution tape. Before you start installing Tivoli OPC, check the preventive service planning (PSP) bucket for recommendations added by the service organizations after your Program Directory was produced. The PSP includes a recommended service section that includes high impact or pervasive (HIPER) APARs. Ensure that the corresponding PTFs are installed before you start a Tivoli OPC subsystem.

---

## 2.4 JES2 considerations

JES2 type and configuration in your installation has big implications on your OPC configuration. Follow these recommendations:

In a JES2 Multi Access Spool (MAS), you need to be sure that *every* member that participates in the JES2 checkpoint has an OPC tracker, JES, and SMF exits installed, whether or not one member is currently drained. This is because JES2 removes the output processing of the system affinity (sysaff).

In other words, even if you force a job to run on a particular machine with the tracker, the output processing (when the output reaches the Spool) can be done anywhere including a machine that does not have a tracker; therefore,

events will be lost and the current plan will not be updated accordingly. See also DOC APAR PQ07742.

If you are not sure what your MAS looks like, go to SDSF and enter:

```
/SD MEMBER
```

You will receive output similar to that shown in the following screen:

```
$HASP673 MEMBER(1) 010
$HASP673 MEMBER(1) NAME=SC47,STATUS=ACTIVE,IND=NO,
$HASP673 LASTART=(QUICK,(2000.133,
$HASP673 12:34:53)),
$HASP673 TIME=(2000.139,22:14:34.72),
$HASP673 VERSION=OS 2.8.0,SLEVEL=0,
$HASP673 SSNAME=JES2,BOSS=YES
$HASP673 MEMBER(2) 011
$HASP673 MEMBER(2) NAME=SC48,STATUS=ACTIVE,IND=NO,
$HASP673 LASTART=(QUICK,(2000.125,
$HASP673 12:22:25)),
$HASP673 TIME=(2000.139,22:14:34.16),
$HASP673 VERSION=OS 2.8.0,SLEVEL=0,
$HASP673 SSNAME=JES2,BOSS=YES
$HASP673 MEMBER(3) 012
$HASP673 MEMBER(3) NAME=SC49,STATUS=ACTIVE,IND=NO,
$HASP673 LASTART=(QUICK,(2000.138,
$HASP673 22:28:42)),
$HASP673 TIME=(2000.139,22:14:34.44),
$HASP673 VERSION=OS 2.8.0,SLEVEL=0,
$HASP673 SSNAME=JES2,BOSS=YES
```

---

## 2.5 Adding SMF and JES exits for event tracking

*This task must be performed when installing a tracker.* OPC tracking depends on several events created during the life of a job. These events are created by SMF and JES exits. It is very important that these exits be assembled, linked, and installed as documented in the installation manual. *In particular, the EXIT7 (JES2) needs to be reassembled and relinked after JES2 macros have been modified through maintenance.* To simplify the installation of Tivoli OPC event tracking, several sample event-tracking exits can be found in your sample library, SEQQSAMP. To assemble and install exits, you can use the provided sample JCL to install the exits as SMP/E usermods, or, alternatively, you can assemble and link-edit the exits yourself. For JES exits, apply usermods in the CSI where JES is included; this is the best method. It has the advantage that SMP automatically reassembles the exits if maintenance is applied to the JES control blocks upon which Tivoli OPC is dependent.

For JES2 exits, which have been reassembled, at least a JES2 hotstart must be performed in order to fetch the new version.

---

## 2.6 Defining OPC subsystems

You must define the name of every new Tivoli OPC subsystem in the active subsystem-name-table member of SYS1.PARMLIB. It is advisable to install at least two Tivoli OPC controlling systems, one for testing and one for your production environment.

It is recommended that you install the tracker and the Controller in *separate* address spaces on the controlling system. This because you must not shut down both subsystems for maintenance or customizing purposes. To define the subsystems, update the active subsystem name table member (IEFSSNxx) in SYS1.PARMLIB. Include records as in the following example:

```
SUBSYS SUBNAME(subsystem name) INITRTN(module name) INITPARM  
( 'maxecsa,suffix' )
```

`subsystem name` is the name assigned to a Tivoli OPC subsystem. The name must be two to four characters. All the Tivoli OPC subsystem names, as defined in the SYS1.PARMLIB member IEFSSNnn, must be unique within a GRS complex. Also, the Tivoli OPC subsystem names of the OPC Controllers must be unique within your OPCplex/OPC network (both local and remote systems). See also DOC APAR PQ19877 for more details.

`module name` is the name of the subsystem initialization module, **EQQINITD** (for Tivoli OPC 2.3.0).

`maxecsa` defines the maximum amount of extended common service area (ECSA) that is used to queue Tivoli OPC job-tracking events. The value is expressed in kilobytes (1 KB equals 1024 bytes). The default is 4, which means that a maximum of 4 KB (4096 bytes) of ECSA storage is needed to queue Tivoli OPC job-tracking events. When a tracker subsystem is down, the events are buffered in common storage until the tracker is restarted; so, if the size of ECSA is not sufficient, you will lose events and message EQQZ035E is written in the message log. The size of the ECSA depends on your installation and the expected amount of possible buffered events. The MAXECSA table, on page 53 of the Installation Guide, shows you the relationship between the MAXECSA value and the number of events that can be held in common storage.

`suffix` is the module name suffix for the EQQSSCM module that EQQINITD loads into common storage. EQQSSCM is the subsystem communication

module. The suffix must be a single character. Because the name of the module shipped with Tivoli OPC is EQQSSCMD, you should specify a suffix value of *D*.

---

## 2.7 Allocating OPC datasets

Carefully consider where Tivoli OPC datasets are allocated in your production environment. Some datasets can be highly active. Avoid placing these datasets on DASD volumes with high activity because this can result in poor performance due to contention. Also, consider the ability to recover datasets if a DASD volume becomes unusable. If you place all your Tivoli OPC datasets on the same volume, you must recover many datasets before you can continue your Tivoli OPC service. The space to allocate for your datasets depends upon the workload at your installation. It is difficult to give precise figures for the amount of space you will need. The space allocated by the sample JCL should give you enough room to at least get started. Refer to page 67 of the *Tivoli OPC V2R3 Installation Guide*, SH19-4379, to see a detailed description of the required datasets and the possibility of customizing to your installation.

---

## 2.8 Creating JCL procedures for OPC address spaces

The EQQJOBS dialog generates several members in the output library that you specified. Members EQQOPCA, EQQOPCB, and EQQOPCS contain started-task JCL that is tailored with the values you entered in the dialog. The “Hardware and Software Requirements” section of the *Tivoli OPC V2R3 Installation Guide*, SH19-4379, states that the Controller-started task needs 5 MB below and 32 MB above the 16 MB line. This might not be sufficient anymore. The region value is strictly dependent on the OPC customization and workload; so, it may even be necessary, in order for OPC to work correctly, to specify a region value of 64 MB, which means anything available below and 64 MB above the 16 MB line.

---

## 2.9 Setting up the ISPF environment

This table describes the ISPF and Tivoli OPC datasets that you must allocate to the TSO session to execute the Tivoli OPC dialog. We strongly recommend that you place the OPC datasets at the top of your ISPF concatenation because some kind of maintenance modifies the subsystem code and the dialog. If both are not on the same level, it may cause unpredictable results. Take this into consideration when modifying panels and concatenate them

before working with the OPC panel library. Table 1 lists the ISPF and OPC datasets.

Table 1. ISPF and OPC dialog datasets

DDNAME	OPC use	created by / found in
SYSPROC	Clist library	SEQQCLIB
ISPPROF	user session defaults	your existing ISPPROF
ISPLIB	panel library	SEQQPxxx
ISPLMLIB	message library	SEQQMxxx
ISPSLIB	skeleton JCL library	EQQJOBS option 2
ISPTLIB	read tables	SEQQTBL0

**Note**

If you use the ISPF command table, EQQACMDS, invoke Tivoli OPC as a separate ISPF application with the name, EQQA. If you want to use a different ISPF application name, such as EQQB, create a command table with the name EQQBCMDS. If necessary, you can modify or create an ISPF command table using ISPF/PDF Option 3.9. Note that ISPF/PDF Option 3.9 writes the created or modified table to the dataset allocated to the ISPTABL.

If you notice that dialog commands, such as *right* or *left*, are not working anymore, the invoked ISPF application name does not match the used command table.

There is a useful TSO command, called *ISRDDN*, that you can use to check the current dialog allocations. All you have to do is enter the TSO *ISRDDN* command from any OPC panel. It might be useful to press the Help key to get familiar with all the functions of *ISRDDN*. In addition, TSO *ISRDDN* can help you easily find all the libraries in your allocation that contain a specific member.

The next screen shows an example of TSO *ISRDDN* output.

TSO ISRDDN OUTPUT.

Current Data Set Allocations

Command ==>

Scroll ==>PAGE

Volume	Disposition,Act,DDname	Data Set Name List	Actions: B E V F C I Q
O260R2	SHR,KEEP > ADMPCF	GDDM.SADMPCF	
O260R2	SHR,KEEP > ISPLLIB	OPCESA.V2R3M0.SEQQIMD0	
O26SM4	MOD,KEEP > ISPLLOG	SFRA4.SPFTMP.LOG	
O260R1	SHR,KEEP > ISFMLIB	OPCESA.V2R3M0.SEQQMSG0	
O260SM	SHR,KEEP >	VS4.ISPF.MLIB	
O260R2	SHR,KEEP >	ISP.SISPMENU	
O260R2	SHR,KEEP >	ISF.SISFMLIB	
O260C1	SHR,KEEP > ISPPLIB	OPCESA.V2R3M0.SEQQPENU	
O260SM	SHR,KEEP >	OPCESA.V2R3M0.SEQQPNL0	
O260R1	SHR,KEEP >	ISP.SISPPENU	
O260R2	SHR,KEEP >	ISF.SISFPLIB	
O260R2	SHR,KEEP >	CPAC.ISPPLIB	
O260R2	SHR,KEEP >	MK4.USER.PENU	

### 2.9.1 Verifying OPC installation

When you have installed a tracker, Controller, standby Controller, or server, you should start it and perform initial verification procedures. To fully verify Tivoli OPC, start all Tivoli OPC address spaces in your configuration and create database entries, a long-term plan, and a current plan. This is required to verify job submission and connections between systems.

Refer to page 105 of the *Tivoli OPC Installation Guide*, SH19-4379, where a detailed verification procedure will guide you step-by-step.

---

## 2.10 The Job Scheduling Console

Next, we will describe how to install the Job Scheduling Console (JSC), Connector and TCP/IP server for OPC. JSC communicates with the OPC subsystem through the OPC Connector, a protocol converter that translates the instructions entered through the console into OPC commands, which, in turn, communicate with the new TCP/IP server task. The OPC Connector needs a Tivoli Managed Region (TMR) server or Managed node, while you can install the JSC on any workstation that has a TCP/IP connection to the Connector. You need a Connector instance for each Controller. The Job Scheduling Service (JSS) provides a common Framework interface to job scheduling applications. The TCP/IP server support must be installed on the same system on which the OPC Controller resides. For each Controller, you

must have a TCP/IP server installed. Figure 2 shows the dataflow from the JSC to the Controller subsystem with all the necessary components.

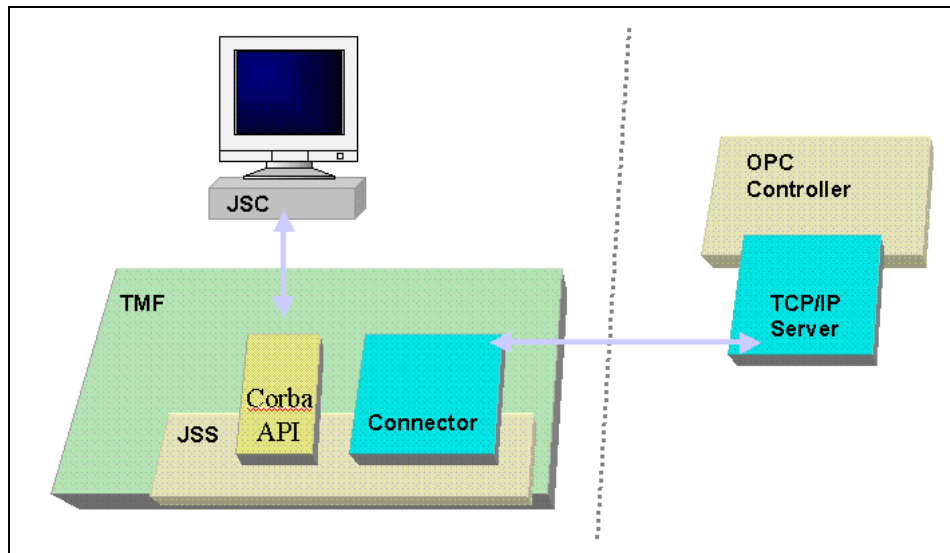


Figure 2. JSC dataflow

## 2.11 Supported platforms and minimum system requirements for JSC

Table 2 identifies the supported platforms and minimum hardware and operating system requirements for the Job Scheduling Console for OPC. For more detailed and up-to-date information, see the *Tivoli OPC V2R3 Job Scheduling Console Release Notes*, GI10-9233.

Table 2. System requirements for jsc

Platform	Operating system
Required: Pentium 233 with 64MB RAM Recommended: Pentium 2 ,266 with 128 MB RAM	Microsoft Windows NT Version 4.0 with SP 3 or later Microsoft Windows 95 Microsoft Windows 98
RS/6000	AIX 4.2 or 4.3
SPARC System	Sun Solaris 2.6 or 2.7

In addition, you need to have the following software installed on your machine or a machine that you can access:



- The OPC Connector on a TMR Server running Tivoli Management Framework Version 3.6.1.

**Note**

Section 3.5, “Installing a dedicated Tivoli Framework to run theTWS or OPC” on page 42, explains the installation of Tivoli Framework.

- TME 10 OPC Version 2.1 or Tivoli OPC Versions 2.2 or 2.3. For OPC V2R3 the TCP/IP Server support is already in the base coding. For V2R1 and V2R2, have a look at APARs PQ21320 and PQ21321.
- MVS Version 5.0 or OS/390 Versions 1.0 or 2.0.
- If you are running MVS Version 5.0 or OS/390 Version 1.0, you need TCP/IP Version 3.2 with C socket API support.
- For AIX systems running the Job Scheduling Console, you must install the Java Development Kit (JDK) 1.1.8. This kit is automatically installed with the Job Scheduling Console on all the other platforms.

You can download the Java Development Kit from the following Web site:

[www.ibm.com/java/jdk/download/index.html](http://www.ibm.com/java/jdk/download/index.html)

---

## 2.12 Installing JSC

Perform the following steps to install the Job Scheduling Console for OPC:

1. Insert the Tivoli Job Scheduling Console CD-ROM into the system CD-ROM drive or mount the CD-ROM from a drive on a remote system. For this example, the CD-ROM drive is drive F:.
2. Perform the following steps to run the installation command:

**On Windows:**

- a. From the Start menu, select the **Run...** option to display the Run dialog.
- b. In the Open field, enter F:\Install.

**On AIX:**

- a. Type the following command:  

```
jre -nojit -cp install.zip install
```
- b. If that does not work, try:  

```
jre -nojit -classpath [path to] classes.zip:install.zip install
```
- c. If that does not work either, on sh-like shells, try:

```
cd [to directory where install.zip is located] CLASSPATH=[path to]
classes.zip:install.zip export CLASSPATH java -nojit install
```

d. Or, for csh-like shells, try:

```
cd [to directory where install.zip is located] setenv CLASSPATH [path to]
classes.zip:install.zip java -nojit install
```

**On Sun Solaris:**

- a. Change to the directory where you downloaded `install.zip` before running the installer.
- b. Enter `sh install.bin`.

The splash window, shown in the Figure 3, is displayed.



Figure 3. JSC Installation Splash

Note that you can also select languages other than English from this window.

### 2.12.1 Starting the JSC

Perform the following steps to start the JSC:

1. Depending on your platform, start the JS Console in the following way:

**On Windows:**

Depending on the shortcut location that you specified during installation, click the JS Console icon or select the corresponding item in the Start menu.

**On Windows 95 and Windows 98:**

You can also start the JSC console from the command line. Just type `runcon` from the `\bin\java` subdirectory of the installation path.

**On AIX:**

Type `./AIXconsole.sh`.

**On Sun Solaris:**

Type `./SUNconsole.sh`.

A *Tivoli Job Scheduling Console* start-up window is displayed as shown in Figure 4.

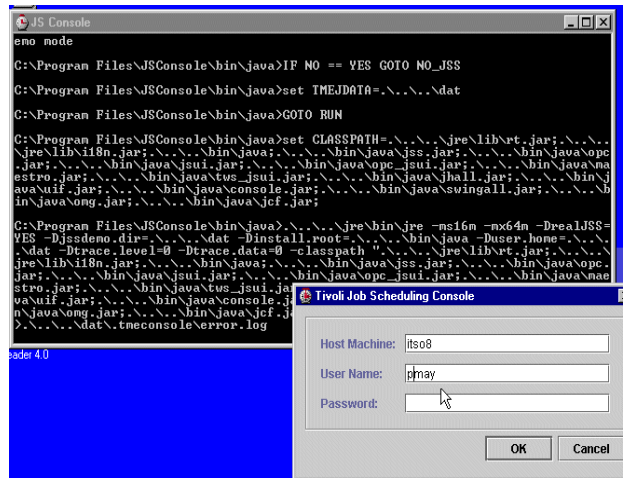


Figure 4. JSC start-up

Enter the following information and click the **OK** button to proceed:

1. *Host Machine* - This is the name of the Tivoli managed node that runs the OPC Connector.
2. *Login As* - This is the user ID of the Tivoli administrator of the host machine running the OPC Controller. (See also mapping file in Section 2.15, "Installing the TCP/IP server" on page 30, for a detailed explanation.)
3. *Password* - This is the password to the host machine running the OPC Connector.

---

## 2.13 Creating a Tivoli Administrator

A Tivoli Administrator is needed to install and manage the OPC Connector on the TMR Server or Managed node and is used as a vehicle for controlling

access through the JSC. We recommend that a Tivoli Administrator be created for every user who wants to use the JSC. Perform the following steps from the Tivoli desktop to create it:

1. Double Click on the **Administrators** icon and select **Create->Administrator** as shown in Figure 5.

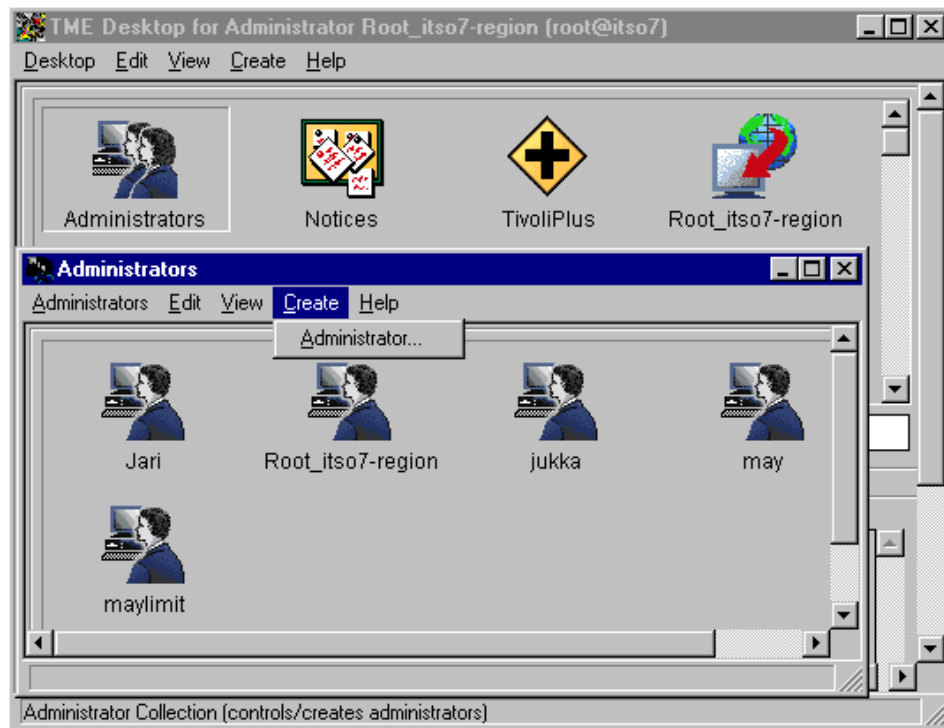


Figure 5. Create Administrator

2. Enter the Tivoli Administrator name you want to create.
3. Press **Set Logins** to specify the login name. This field is important because it will be used to determine the UID with which many operations are performed. *It also represents a UID at the operating system level.* This is shown in Figure 6 on page 23.

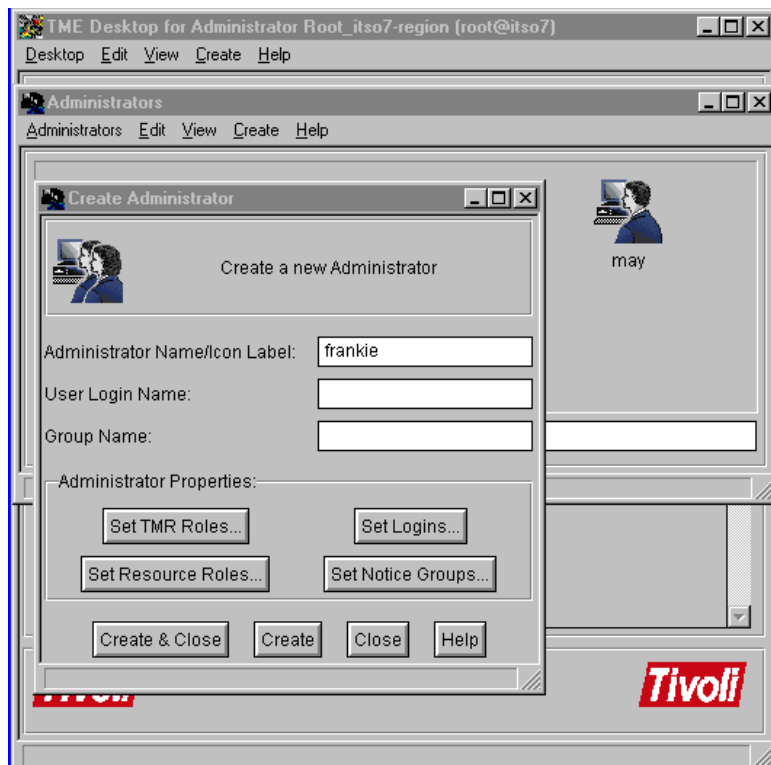


Figure 6. Create a new Administrator

Type in the Login name and press **Enter**. Then, select **Set & Close** as shown in Figure 7 on page 24.

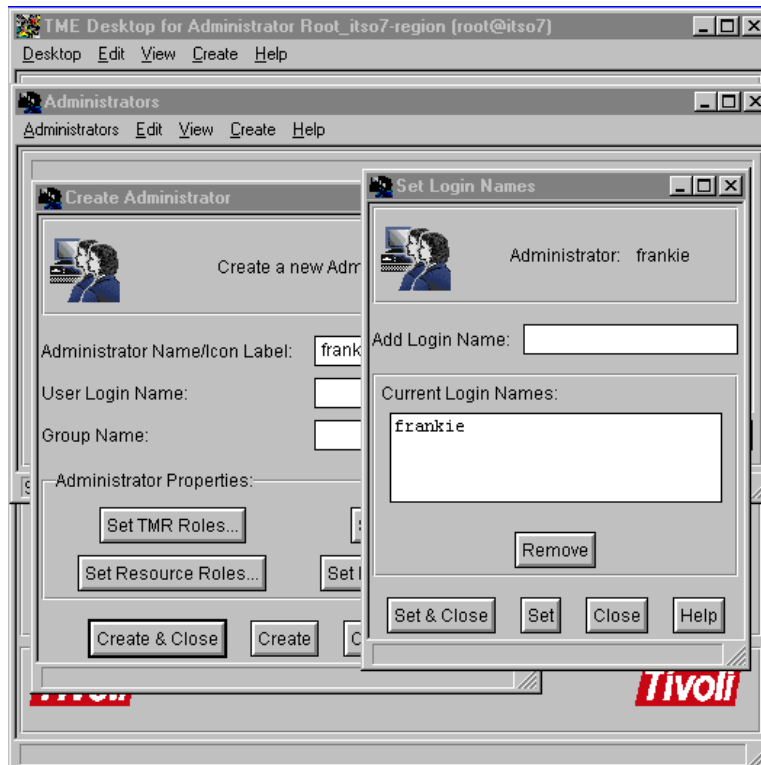


Figure 7. Login names

4. Enter the name of the group. This field is used to determine the GID under which many operations are performed. Then select **Create and Close**. This is shown in Figure 8 on page 25.

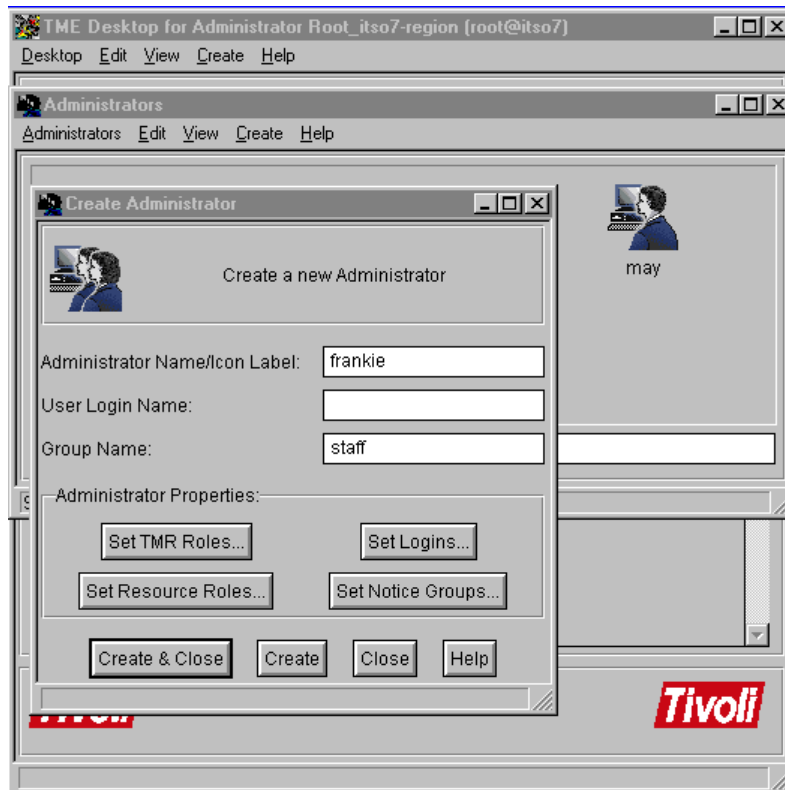


Figure 8. Group names

You will see again the administrator panel with the newly-defined Tivoli Administrator as shown in Figure 9 on page 26.



Figure 9. Tivoli Administrator

## 2.14 Installing and customizing the OPC Connector

The installation of the OPC Connector is divided into three steps:

1. Install the Job Scheduling Services (JSS). Skip this step if you already have the JSS installed on your TMR server and Managed nodes.
2. Install the OPC Connector.
3. Create OPC Connector instances. You must create one instance for each OPC Controller that you want to access from the Job Scheduling Console.

For topics one and two, refer to Chapter 4 of the *Tivoli OPC V2R3 Job Scheduling Console User's Guide*, GC32-0402, for a complete step-by-step installation.



Table 3 lists the System requirements for OPC Connect.

Table 3. System requirements for OPC Connector

Platform	Operating System
Pentium 233 with 64 MB RAM	Microsoft Windows NT 4.0 with SP 3 or later
RS/6000	AIX 4.2 or 4.3
HP9000 700/800 series	UX 10.x and 11
SPARC System	Sun Solaris 2.6 or 2.7

### 2.14.1 Creating OPC Connector instances

You have to create one OPC Connector instance for each OPC Controller that you want to access with the Tivoli Job Scheduling Console. To create an instance, provide the information that enables the OPC Connector to connect to the OPC TCP/IP server.

From the command line, use the following command to create each new OPC Connector instance:

```
wopcconn -create [-h node] -e engine_name -a address -p port
```

where:

- *node* is the name or the ID of the managed node on which you are creating the instance. The name of the TMR server is the default.
- *engine\_name* is the name of the new instance.
- *address* is the IP address of the OS/390 system where the OPC subsystem to which you want to connect is installed.
- *port* is the port number of the OPC TCP/IP server to which the OPC Connector will connect.

You can also run the *wopcconn* utility in interactive mode. To do this, perform the following steps:

1. At the command line, enter *wopcconn* with no arguments.
2. Select choice number **1** in the first menu.

#### **Authorization Roles**

To manage OPC Connector instances from a TMR server or Managed node, you must be a Tivoli Administrator. To control access to OPC, the TCP/IP

server associates each Tivoli administrator to a RACF user. For this reason, a Tivoli Administrator should be defined for every RACF user.

Each Tivoli administrator has one or more roles. To use or manage OPC Connectors, you need the following roles:

- user
  - To use the instances
  - To view instance settings
- admin, senior, or super
  - To perform all actions available to the user role
  - To create and remove instances
  - To change instance settings
  - To start and stop instances

### ***Managing OPC Connector instances***

Use the *wopcconn* utility to create, remove, and manage OPC Connector instances. This program is downloaded when you install the OPC Connector. Table 4 describes how to use the *wopcconn* in the command line to manage OPC Connector instances.

*Table 4. How to manage Connector instances*

<b>If you want to...</b>	<b>Use this syntax:</b>
create an instance	<code>wopcconn -create [-h node] -e engine_name -a address -p port</code>
stop an instance	<code>wopcconn -stop -e engine_name   -o object_id</code>
start an instance	<code>wopcconn -start -e engine_name   -o object_id</code>
restart an instance	<code>wopcconn -restart -e engine_name   -o object_id</code>
remove an instance	<code>wopcconn -remove -e engine_name   -o object_id</code>
view the settings of an instance	<code>wopcconn -view -e engine_name   -o object_id</code>
change the settings of an instance	<code>wopcconn -set -e engine_name   -o object_id [-n new_name] [-a address] [-p port] [-t trace_level] [-l trace_length]</code>

where:

- *node* is the name or the object ID (OID) of the managed node on which you are creating the instance. The TMR server name is the default.
- *engine\_name* is the name of the new or existing instance.
- *object\_id* is the object ID of the instance.
- *new\_name* is the new name for the instance.
- *address* is the IP address of the OS/390 system where the OPC subsystem to which you want to connect is installed.
- *port* is the port number of the OPC TCP/IP server to which the OPC Connector must connect.
- *trace\_level* is the trace detail level from 0 to 5. *trace\_length* is the maximum length of the trace file. You can also use `wopcconn` in interactive mode. To do this, just enter the command, without arguments, in the command line.

### **Example**

We used a Tivoli OPC V2R3 with the IP address *9.39.62.19*. On this machine, a TCP/IP Server connects to Port *3111*. *ISTO7* is the name of the TMR server where we installed the OPC Connector. We called this new Connector instance *OPC*.

With the following command, our instance has been created:

```
wopcconn -create -h isto7 -e opc -a 9.39.62.19 -p 3111
```

By issuing `wopcconn` without any subparameters and reviewing the instance *OPC*, you will get the output shown in the following screen.

```

***** OPC Connector manage program *****
View/Change attributes menu

Name                : OPC
Object id           : 1929225022.1.1289#OPC::Engine#
Managed node       : itso7
Status              : Active

OPC version         : 2.3.0

2. Name              : OPC

3. IP Address or Hostname: 9.39.62.19
4. IP portnumber     : 3111

5. Trace Length      : 524288
6. Trace Level       : 3

0. Exit

```

---

## 2.15 Installing the TCP/IP server

The new TCP/IP server is required to connect to the OPC Connector and access the Controller subsystem via the OPC Program Interface (PIF).

You will find the JCL example for the started task in Samplib member *EQQOPCS* when running OPC V2r3. For prior releases, you must install the enhancement APARs, *PQ21320* or *PQ21321*. After the installation, you can gain full functionality of the Job Scheduling Console as with the V2R3 release.

First, you have to modify the JCL of EQQOPCS in following way:

- Concatenate the C runtime library on the steplib in the server JCL.(CEE.SCEERUN). In our system, the library is linklisted; so, we don't have to use the steplib.
- If you have more than one TCP/IP stack or the name of the procedure that was used to start the TCPIP address space is different from TCPIP, introduce the SYSTCPD DD card pointing to a data-set containing the TCPIPJOBNAME parameter (see DD SYSTCPD in the TCP/IP manuals).

Because our TCP/IP stack on the OS/390 calls TCPIP4, we changed the server JCL in the following way:

```
//OPCSP      EXEC PGM=EQQSERVER,REGION=OM,TIME=1440
//*****
//* THIS IS A SAMPLE STARTED TASK PROCEDURE FOR AN OPC TCP/IP SERVER
//* IT SHOULD BE REVIEWED AND MODIFIED AS REQUIRED
//* TO SUIT THE NEEDS OF THE INSTALLATION.
//** MOD 210601 BY MICHAELA ***
//*****
//STEPLIB    DD DISP=SHR,DSN=OPCESA.V2R3M0.SEQQLMD0
//SYSTCPD    DD DISP=SHR,DSN=TCPIP.IV4.TCPPARMS(TCPDATA)
//EQQMLIB    DD DISP=SHR,DSN=OPCESA.V2R3M0.SEQQMSG0
//EQQMLOG    DD DISP=SHR,DSN=OPC.V2R3M0.MLOGS
//EQQPARM    DD DISP=SHR,DSN=OPC.V2R3M0.PARM(OPCSP)
//SYSMDUMP   DD DISP=MOD,DSN=OPC.V2R3M0.SYSDUMPS
//EQQDUMP    DD DISP=SHR,DSN=OPC.V2R3M0.EQQDUMPS
//*
```

- Customize the Server Parameters file (see the EQQPARM DD statement in Server JCL). For example, you could provide the following parameters in the server parameters file.

#### Note

Be aware that the calendar definition is required but missing in the OPC documentation. The calendar definition should be as follows:

- **SERVOPTS SUBSYS(OPCA)** /\* OPC Controller the server connects \*/
- **USERMAP(MAP1)** /\* Eqqparm member for usermap \*/
- **PROTOCOL(TCPIP)** /\* Communication protocol \*/
- **PORTNUMBER(3111)** /\* Port the server connects \*/
- **CODEPAGE(IBM-037)** /\* Name of the host codepage \*/
- **INIT CALENDAR(DEFAULT)** /\* Name of the OPC calendar \*/

The SUBSYS, PROTOCOL(TCPIP) and CALENDAR are mandatory to use the Tivoli Job Scheduling Console. For information about these parameters, refer to *Tivoli OPC Customization and Tuning Guide*, SH19-4380-02 “Chapter 1: -SERVOPTS and INIT Initialization Statement”. A detailed explanation of Usermap you can find in Chapter 2.14.1. For *TCP/IP 3.3* and later the started task UID that is assigned the Server A/S must have a valid OMVS segment in the RACF user profile. Also be sure that the port you try to use is not reserved by another application. An entry in section *Reserve ports for the following servers* of the related TCP/IP profile can be a useful information for other users.

Start the TCP/IP server and verify the server log for possible error messages.

```

20.56.38 EQQZ015I INIT STATEMENT: SERVOPIS SUBSYS(OPCA)
20.56.38 EQQZ015I INIT STATEMENT:          PROTOCOL(TCPIP)
20.56.38 EQQZ015I INIT STATEMENT:          ARM(NO)
20.56.38 EQQZ015I INIT STATEMENT:          PORTNUMBER(3111)
20.56.38 EQQZ015I INIT STATEMENT:          CODEPAGE(IBM-037)
20.56.38 EQQZ015I INIT STATEMENT:          USERMAP(MAP1)
20.56.38 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
20.56.38 EQQZ015I INIT STATEMENT: INIT      CALENDAR(DEFAULT)
20.56.38 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
20.56.38 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER MAP1
20.56.38 EQQZ015I INIT STATEMENT: USER 'FRANKIE@ITSO7-REGION' RACFUSER(SFRA1)
20.56.38 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
20.56.38 EQQZ015I INIT STATEMENT: USER 'MAY@ITSO7-REGION' RACFUSER(SFRA4)
20.56.38 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
20.56.38 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER MAP1      IS: 0000
20.56.38 EQQZ005I OPC SUBTASK SERVER          IS BEING STARTED
20.56.39 EQQPH00I SERVER TASK HAS STARTED
20.56.39 EQQPH09I THE SERVER IS USING THE TCP/IP PROTOCOL

```

### 2.15.1 Controlling access to OPC using Tivoli Job Scheduling Console

Tivoli Framework performs a security check when a user tries to use the Tivoli OPC Tivoli Job Scheduling Console, checking the user ID and password. The Tivoli Framework associates each user ID and password to an Administrator. OPC resources are currently protected by RACF. The Tivoli Job Scheduling Console user should only have to enter a single user ID/password combination and not provide two levels of security checking (at the Tivoli Framework level and then again at the Tivoli OPC level). The security model is based on having the Tivoli Framework security handle the initial user verification while at the same time obtaining a valid corresponding RACF user ID. This makes it possible for the user to work with the security environment in MVS. MVS security is based on a table mapping the Tivoli Administrator to an RACF user ID. When a Tivoli Framework user tries to initiate an action on MVS, the Tivoli Administrator ID is used as a key to obtain the corresponding RACF user ID. The Server uses the RACF user ID to build the RACF environment to access Tivoli OPC services; so, the Tivoli Administrator must relate, or map, to a corresponding RACF user ID. There are two ways of getting the RACF user ID:

1. The first way is by using the RACF Tivoli-supplied predefined resource class, TMEADMIN. For this, you must have one of the following prerequisites:
  - OS/390 V2 with Security Server Feature
  - OS/390 from V1R3 to V2R5 with Security Server Feature with PTF
  - UW91076 installed

- MVS/ESA V5R2.2 with Resource Access Control Facility (RACF) V2R2 with PTF UW37652

Consult the “Implementing Security in Tivoli OPC” section in the *Tivoli Operations Planning and Control V2R3 Customization and Tuning Guide*, SH19-4380, for the complete setup of the TMEADMIN RACF class.

2. The other way is to use a new OPC Server Initialization Parameter to define a member in the file identified by the EQQPARM DD Statement in the Server startup job.

This member contains all the associations between a TME user with a RACFuser ID. You should set the parameter USERMAP in the SERVOPS Server Initialization Parameter to define the member name:

USERMAP(USERS)

The member, USERS, of the Initialization Parameter data set could contain the following *with the same logic as the TMEADMIN class*:

- USER 'MAY@ITS07-REGION' RACFUSER(SFRA4) RACFGROUP(TIVOLI)
- USER 'WILHELM@ITS07-REGION' RACFUSER(WILH2) RACFGROUP(TIVOLI)
- USER 'FRANKIE@ITS07-REGION' RACFUSER(SFRA2) RACFGROUP(TIVOLI)
- USER 'MICHAELA@ITS07-REGION' RACFUSER(MICHA6) RACFGROUP(TIVOLI)
- USER 'JARI@ITS07-REGION' RACFUSER(YARI) RACFGROUP(TIVOLI)

Table 5 shows the relationship between security products and the chosen security solution.

Table 5. Security solution

Security Product	Level	Solution	Prerequisite
Security Sever (RACF)	OS/390 V2R6 or later	TMEADMIN	None(TMEADMIN class provided insOS/390 base
Other SAF compliant		TMEADMIN	Manually define TMEADMIN class(using EQQ9RFE and EQQ9RF01 samples)
In every case		OPC ID mapping table	

Security Product	Level	Solution	Prerequisite
Security Server (RACF)	OS/390 from V1r3 to V2R5	TMEADMIN	Uw91076 (PTF for RACF V2.3.0)
RACF	MVS V5r2.2	TMEADMIN	UW37652(PTF for RACF V2.2.0)
Other SAF compliant		TMEADMIN	Manually define TMEADMIN class(using EQQ9RFE and EQQ9RF01 samples)
In every case		OPC ID mapping table	

We want to point out that every RACF user who has update authority to this usermap table may get access to the OPC subsystem. For your security, planning the usermap table should be an important resource to protect.

We used the OPC ID mapping table in our environment because it is an easy and comfortable way to show access to the OPC subsystem without the need for deep RACF knowledge.

Figure 10 shows the relationship between the EQQPARM member and the usermap.

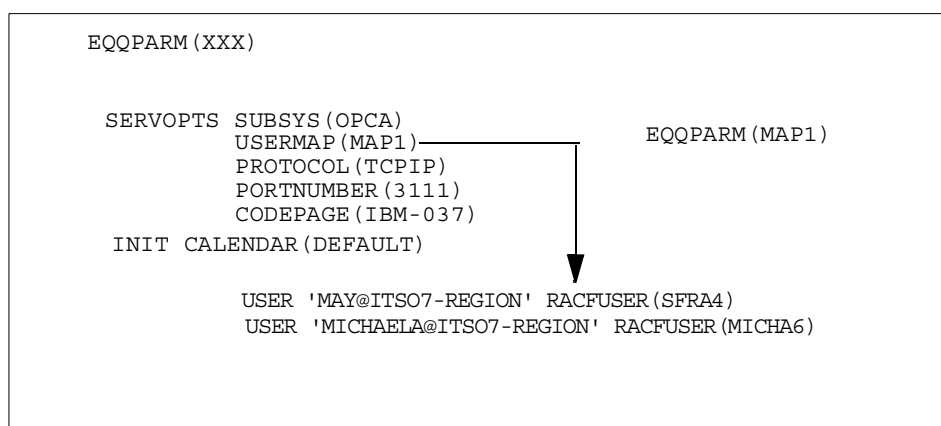


Figure 10. Usermap relationship



We want to demonstrate the path from the JSC logon, through the Tivoli Framework to the TCP/IP server:

1. A user, May, opens a JSC logon panel in order to log on to the OPC Controller as shown in Figure 11.

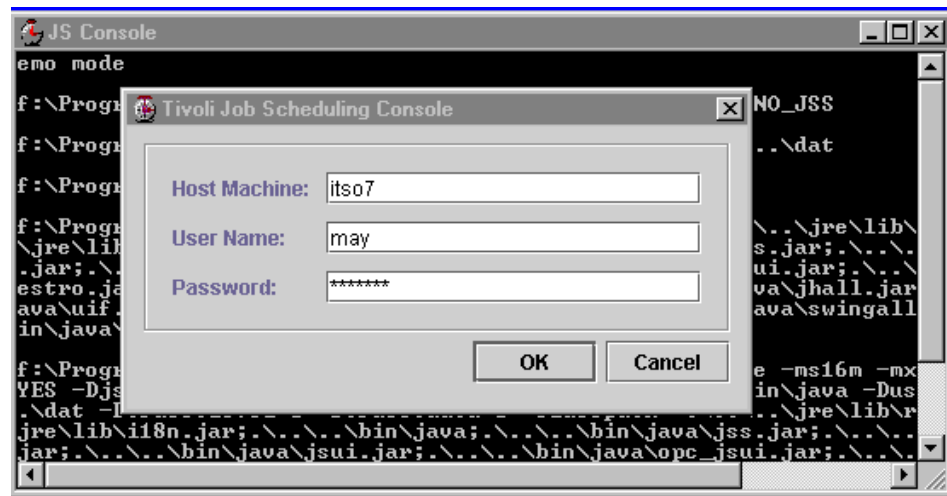


Figure 11. Logon panel to OPC

2. The user, May, is defined as a Tivoli Framework Administrator in the Tivoli region, ITS07.

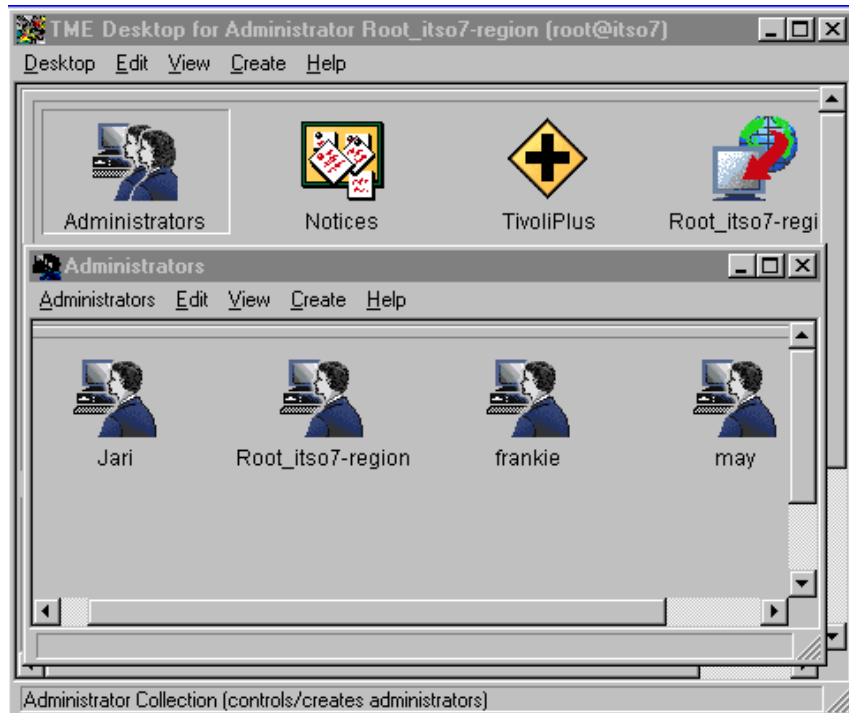


Figure 12. Tivoli Desktop

3. The usermap on the OPC TCP/IP server maps the user Tivoli Framework Administrator, May, to a RACF user ID, SFRA4; therefore, access is granted.

#### Usermap

```
USER 'MAY@ITSO7-REGION' RACFUSER(SFRA4)
```

## 2.16 Summary

In this chapter, we have covered the installation of OPC V2R3 Controller, Job Scheduling Console (JSC), OPC Connector, and TCP/IP Server support. We also gave examples of how to create a Tivoli Administrator, which is required to access to JSC.

You may refer to the *Tivoli OPC V2R3 Installation Guide*, SH19-4379, for more information on installing these modules.

---

## Chapter 3. Tivoli Workload Scheduler 7.0 step-by-step installation

This chapter provides step-by-step installation instructions for Tivoli Workload Scheduler (TWS) and Job Scheduling Console including Tivoli Framework setup.

---

### 3.1 Installation overview

The following are the products used in the testing environment:

- Tivoli Management Framework 3.6.2
- AIX 4.3.3
- Windows NT 4.0 Service Pack 5 and Service Pack 4
- Tivoli Workload Scheduler 7.0
- Job Scheduling Services 1.1
- TWS Connector 1.0
- Job Scheduling Console 1.1

The following is a summary of the TWS environment installation steps:

1. Install TWS Engine 7.0.
2. Install Tivoli Management Framework (TMF). You will need the TMR Server to complete the installation of TWS. This redbook will include instructions required for the minimal installation of Tivoli Framework. More information about TMF installation and setup is covered in the *TME 10 Framework 3.6 Planning and Installation Guide*, SC31-8432.
3. Use the Tivoli Desktop to create a Managed Node on the host on which you will install the TWS Master and, optionally, on all TWS Workstations to which you want direct access through the Job Scheduling Console.
4. Install Job Scheduling Services and TWS Connector on the TMR Server and the TWS Master using Tivoli Desktop and, optionally, on all the TWS FTAs that you want to access directly through the Job Scheduling Console.
5. From the Tivoli Desktop, create a Tivoli Administrator for TWS. In other words, either add the TWS user as a login for the TMF Administrator, or create a new administrator with the TWS user login.
6. Set the security of TWS Workstations.
7. Install Job Scheduling Console on all nodes where you need to control production using a GUI.

---

## 3.2 System requirements

The following are the system requirements for TWS:

### ***On Windows NT:***

- Window NT version 4.0 with Service Pack 4 or higher.
- NTFS partition with approximately 200 MB free disk space is recommended. The amount of space needed may increase if you have many jobs and save log files for a long time.
- TCP/IP network.
- A TWS user account is required for proper installation. You can create the account beforehand or have the *Setup program* create it for you.

### ***On Unix:***

At least 250 MB of free disk space is recommended. The amount of space needed may increase if you have many jobs and save log files for a long time. Master and Domain managers save more information to disks.

### ***On all operating systems:***

If your TWS network includes many TWS Workstations in different locations connected by different bandwidth, TWS Master should be fast enough to produce the new day production. If you choose machines that are too slow, newday production cycle can take a long time, and you definitely do not want to loose any valuable time. You should also choose fast I/O and disks for the Master because TWS keeps lots of data on harddisks. TWS writes data to message files and log files, and it does not actually cache I/O requests because it keeps data as “realtime” as possible. However, there are some optimization solutions, which are explained in Chapter 10, “Best practices” on page 307.

We recommend that you choose a UNIX machine with at least 512 MB of memory and fast SCSI- or SSA-disks as the TWS Master.

Job Scheduling Services requires at least Tivoli Framework Version 3.6.1.

---

## 3.3 Installing TWS Engine 7.0 on Windows NT

You may refer to the installation manual, *Tivoli Workload Scheduler 7.0 Planning and Installation Guide*, GC32-0422, while installing TWS Engine 7.0 on Windows NT.

---

### 3.4 Installing TWS Engine 7.0 on UNIX

There are very clear instructions for installing TWS Engine 7.0 on UNIX in the manual, *Tivoli Workload Scheduler 7.0 Planning and Installation Guide*, GC32-0422. This part includes more information about installing TWS Engine 7.0 on an AIX V4. Installation.

We recommend that you create a separate file system so that you do not fill up the root file system and so that other programs do not fill up the TWS file system.

We used file system size 250 MB (500,000 blocks in 512 byte blocks), which should be enough for TWS Domain Managers and Master. The size of a file system depends on the amount of jobs your TWS Workstation needs to manage and the amount of time you choose to retain log files.

1. Log in as root and create a TWS user and group. We named it *maestro*. You can use the group name, *tivoli*:
  - a. Type `smitty group`.
  - b. Select **Add a Group** and type `tivoli` in the Group Name field. Leave other options as default.
  - c. Type `smitty user`.
  - d. Select **Add a User** and type `maestro` for the User Name, `tivoli` for the Primary Group, and `/opt/maestro` for the HOME directory.
  - e. Change the password for the TWS user:  

```
$ passwd maestro.
```
  - f. Log in to localhost using telnet and change the password. This change must be made because AIX, by default, requires you to change the user password at the first login:  

```
$ telnet localhost
```
2. Log in as root, create the file system, and type `smitty storage`.
  - a. Select **File Systems**
  - b. Select **Add / Change / Show / Delete File Systems**
  - c. Select **Journaled File Systems**
  - d. Select **Add a Journaled File System**
  - e. Select **Add a Standard Journaled File System**
  - f. Select **rootvg** in the Volume Group name field and press enter as seen in the following.

### Add a Standard Journaled File System

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
Volume group name	rootvg	
* SIZE of file system (in 512-byte blocks)	[500000]	#
* MOUNT POINT	[/opt/maestro]	
Mount AUTOMATICALLY at system restart?	yes	+
PERMISSIONS	read/write	+
Mount OPTIONS	[]	+
Start Disk Accounting?	no	+
Fragment Size (bytes)	4096	+
Number of bytes per inode	4096	+
Allocation Group Size (MBytes)	8	+

F1=Help	F2=Refresh	F3=Cancel	F4=List
Esc+5=Reset	F6=Command	F7=Edit	F8=Image
F9=Shell	F10=Exit	Enter=Do.	

### 3. Mount the file system:

```
$ mount /opt/maestro
```

### 4. Mount the TWS CD-ROM:

```
$ mount /cdrom
```

or, in earlier versions of AIX:

```
$ mkdir /cdrom
$ mount -v cdrfs -r /dev/cd0 /cdrom
```

### 5. Change the directory to TWS home and untar the installation package:

```
$ cd /opt/maestro
$ tar -xvf /cdrom/TIVOLI/AIX/MAESTRO.TAR
```

### 6. Execute the customizations script:

```
$ sh customize -new -thiscpu mdm -master -mdm
```

### 7. Edit *.profile* file to include paths *TWSHome* and *TWSHome/bin* (*/opt/maestro/.profile*). Also, add Tivoli environment script to the *.profile* file. For example, type:

```
PATH=/bin:/usr/bin:/usr/local/bin:/opt/maestro:/opt/maestro/bin
export PATH
. /etc/Tivoli/setup_env.sh
```

8. To start the TWS network management process, Netman, automatically as a daemon each time you boot your system, add one of the following to the `/etc/rc` file or the proper file for your system:

- To start Netman only (normally, on TWS FTAs, not TWS Master):

```
if [-x /opt/maestro/StartUp]
then
echo "netman started..."
/bin/su - maestro -c "/opt/maestro/StartUp"
fi
```

- Or, to start the entire TWS process tree (typically, on the TWS Master):

```
if [-x /opt/maestro/bin/comman]
then
echo "Workload Scheduler started..."
/bin/su - maestro -c "/opt/maestro/bin/comman start"
```

9. Configure the TWS Master. Log in to the TWS Master as maestro user. Use the `composer` command to add TWS Workstations and SFinal Job Stream to the database:

```
$ composer
- new
```

The following is an example workstation definition for a Master Domain Manager. For more information on workstation definitions, refer to the *Tivoli Workload Scheduler 7.0 Reference Guide*, GC32-0424.

```
cpuname MDM
os UNIX
node 10.69.14.8
description "Master Domain Manager"
for Maestro
autolink on
resolvedep on
fullstatus on
end
```

10. Create a new symphony file that includes the Master Domain Manager workstation definition. To do this, add the *final* job stream to the production cycle. This job stream contains the *Jnextday* job, which automates the creation of the symphony file.

```
$ composer add Sfinal
```

11. Run the Jnextday job:

```
$ Jnextday
```

12. When the Jnextday job completes, check the status of TWS:

```
$ conman status
```

If TWS started correctly, the status should be *Batchman=Lives*.

Raise the limit to allow jobs to execute. The default job limit after installation is set to zero, which means that no jobs will execute; so, you may want to raise the job limit now:

```
$ conman limit;10
```

You can now begin defining additional scheduling objects in the CLI, including workstations, jobs, and job streams, or you can continue to install the TWS Connector and the Job Scheduling Console.

Note that new objects are not recognized until the Jnextday job runs in the final job stream. By default, the final job stream runs at 5:59am. If you want to incorporate new scheduling objects sooner, you can run Jnextday manually as in the preceding step 11 or use the `conman submit` command. For information about defining your scheduling objects, refer to the *Tivoli Workload Scheduler 7.0 Plus Users Guide*, GC31-8401.

---

### 3.5 Installing a dedicated Tivoli Framework to run the TWS or OPC

The Tivoli Workload Scheduler engine is separated from the Tivoli Management Framework. It is not a Tivoli Management Framework application. However, the TWS Connector and the OPC Connector are Tivoli Management Framework applications. The TWS Connector is required if you want to operate TWS through the Job Scheduling Console (the new graphical user interface). The TWS Connector requires the Job Scheduling Services (JSS) to be installed into your Tivoli Framework environment. If you want to operate OPC through the Job Scheduling Console, you need to install the JSS and the OPC Connector. Both products, OPC and TWS, can use the same graphical interface.

Tivoli Workload Scheduler 7.0 also includes the legacy GUI, but only the JS Console has Time zone support. From the legacy GUI, you cannot modify a workstation or a schedule that had time zone support added from the command line interface or from the JS Console. Moreover, the JS Console is required if you want to operate both Operations Planning and Control (OPC) and Tivoli Workload Scheduler engines from the same user interface.

Before you begin installing the Tivoli Framework, verify the following conditions:



1. Any *oservs* currently running on the systems you are about to install must be shut down (only for previously installed versions of Tivoli Enterprise)
2. The names of the Tivoli Managed Nodes must be included in the */etc/hosts* file, the Windows NT LMHOSTS file, the NIS host map, or the name server.
3. All potential Tivoli users must have at least read access to the Tivoli directories.

If your company has not already installed Tivoli Framework, the following are the detailed instructions to install the TMR Server on Windows NT. For more information and instructions to install on UNIX platforms, you may refer to the manual, *TME 10 Framework 3.6 Planning and Installation Guide*, SC31-8432. If you will install Tivoli Management Framework only for TWS, OPC, and JSC, you may perform the following steps:

1. Insert the Tivoli Framework 3.6 CD-ROM.
2. Select **Start->Run** from the Windows NT Desktop, type *D:\setup.exe*, and press **Enter** where *D:\* is the CD-ROM drive.
3. Click **Next** to start installation as shown in Figure 13.

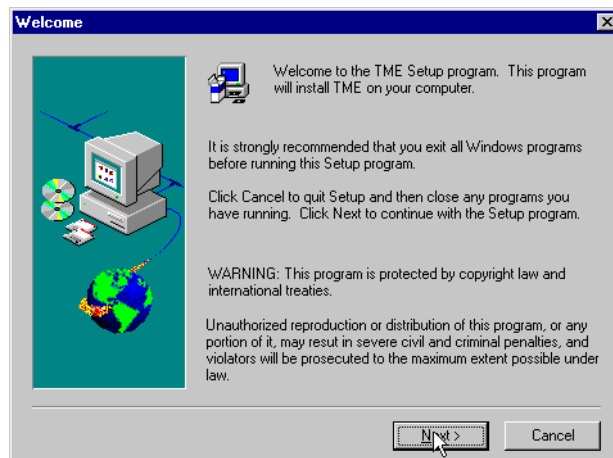


Figure 13. Welcome

4. Press **Yes** to create the *tmesrvd* account locally as shown in Figure 14 on page 44.

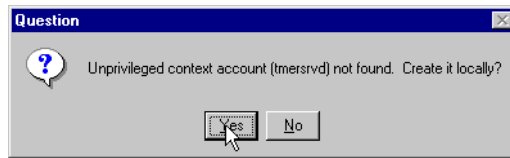


Figure 14. Creating the tmesrsvd account

5. Press **Yes** to add advanced user rights to the TME Administrator as shown in Figure 15.

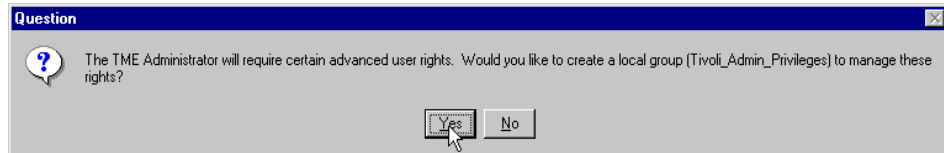


Figure 15. Advanced user rights

6. The window, shown in Figure 16, instructs you to log out and back in. Logoff Windows and log in as Administrator. Execute D:\setup.exe again.

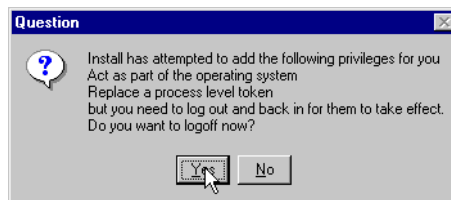


Figure 16. Logout message

7. Type in the user information in the window shown in Figure 17 on page 45.

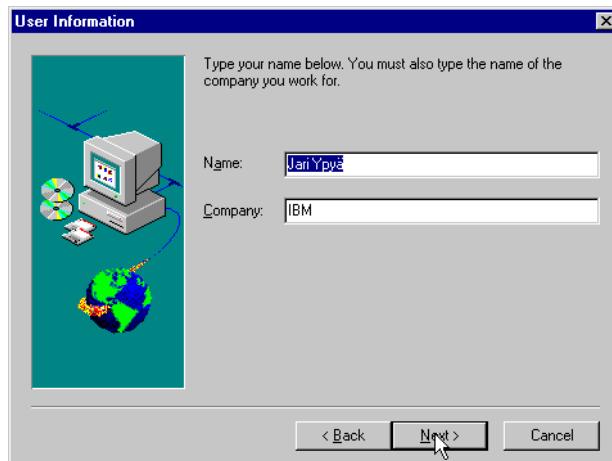


Figure 17. User Information

8. You will be prompted with an installation password screen as shown in Figure 18. We did not use the installation password, and you can leave it empty.

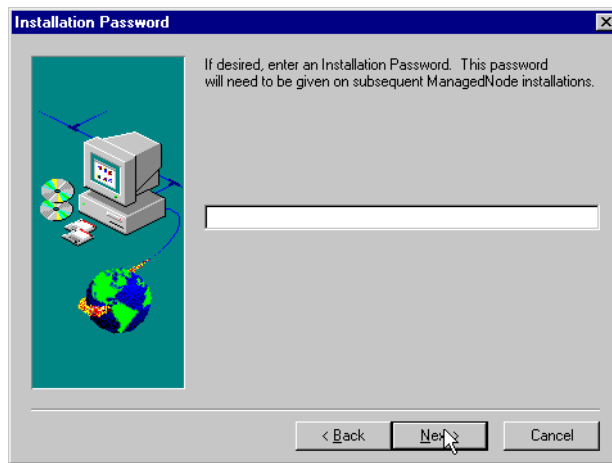


Figure 18. Installation Password

9. You can leave the following Remote User File Access window empty since we do not intend to access remote file systems.

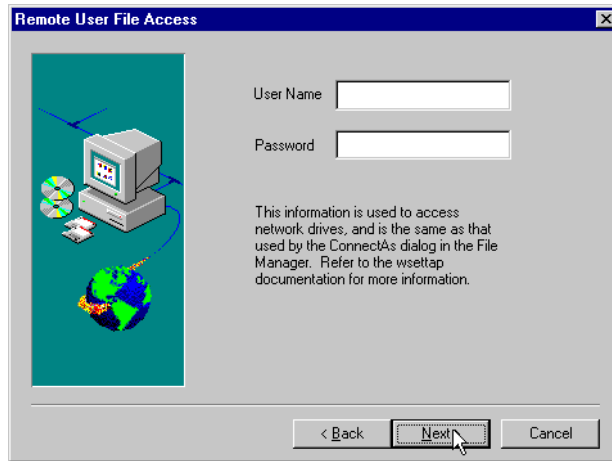


Figure 19. Remote User File Access

10. In the window, shown in Figure 20, select the **Typical** install option. You can use the browse button to select a different installation directory if necessary.

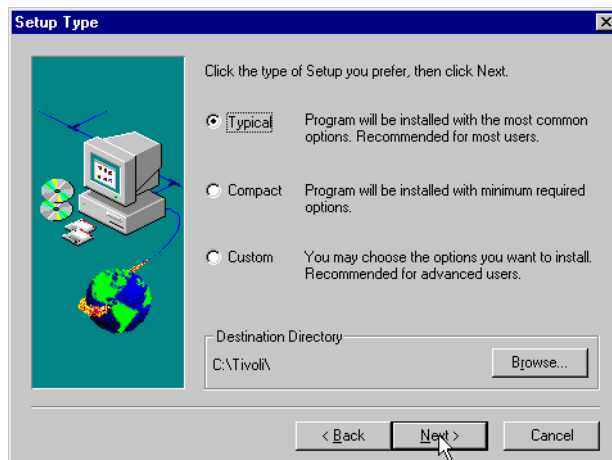


Figure 20. Setup Type

11. In the window, shown in Figure 21 on page 47, enter the TME 10 License Key. If you do not have it already, you may request it. Along with your TME 10 software, you should have received a TME 10 License Key Request Form. Complete this form and return it to your customer support provider.

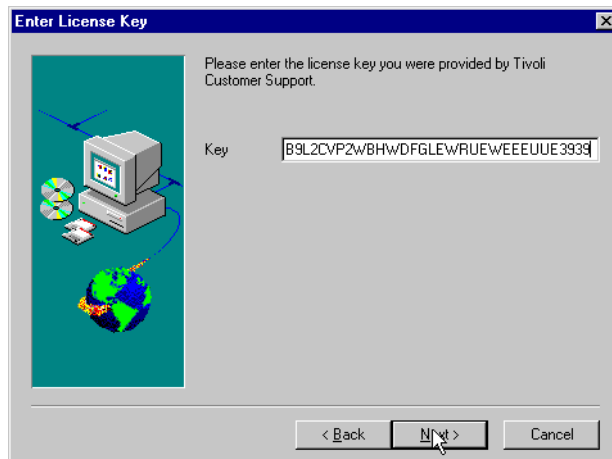


Figure 21. Enter License Key

12. Enter the location of the Tivoli database directory in the window shown in Figure 22.

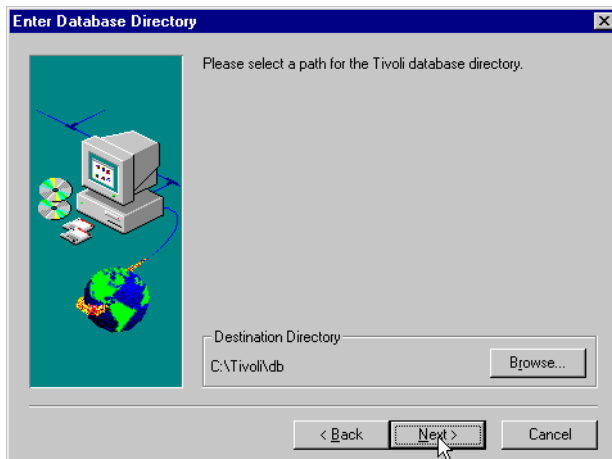


Figure 22. Enter Database Directory

13. You will see an MS-DOS window that will report the database installation status as shown in Figure 23 on page 48. Press any key after the initialization completes.

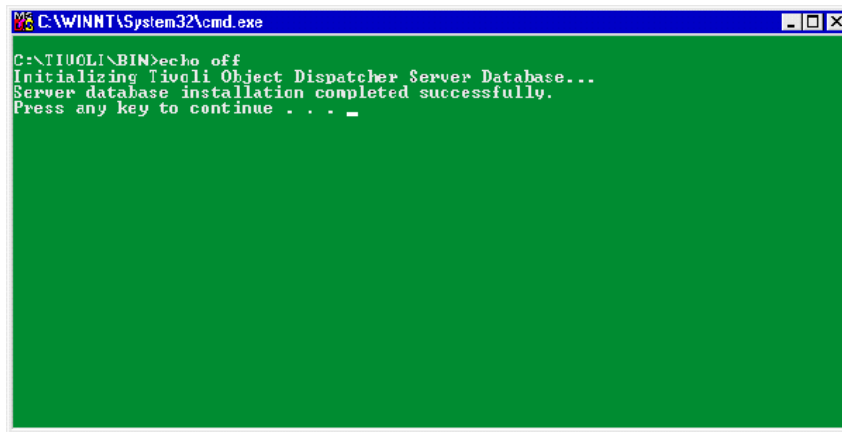


Figure 23. Database installation window

14. Select **Yes** in the window, shown in Figure 24, to restart your computer and then log in as Administrator user.

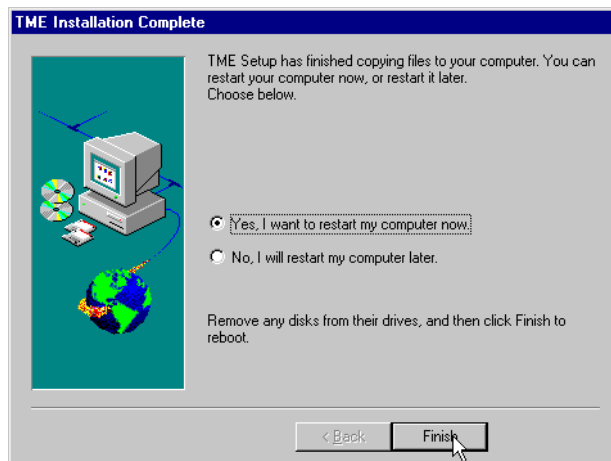


Figure 24. TME Installation Complete

15. Normally, the Tivoli Object Dispatcher service will be started automatically by the system, but if you do not use U.S. keyboard mappings, the service will *not* start automatically. If you are not using U.S. keyboard mappings on Windows NT, the following tasks are required:

The copied DLL is kbd[country-code].dll. In the following example, kbdfi.dll is used for the Finnish keyboard (fi):

- a. Copy `c:\winnt\system32\kdbfi.dll` to `c:\winnt\system32\kdbus.dll`
- b. Start the **Tivoli Object Dispatcher** service from Control Panel/Services.

16. Install the Tivoli Desktop. Execute `D:\pc\desktop\disk1\setup.exe`.

17. Choose the destination directory as shown in Figure 25.

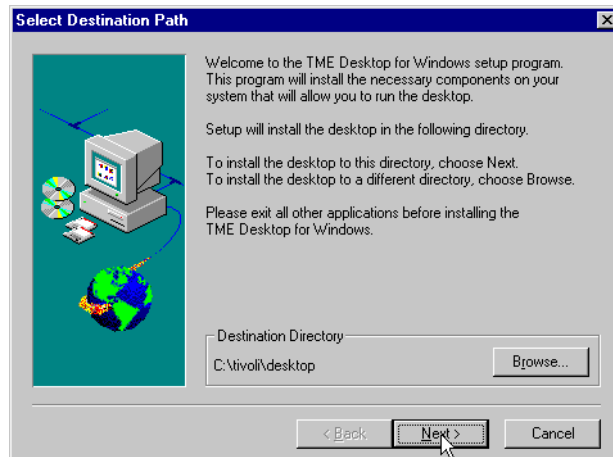


Figure 25. Select Destination Path

18. Next, select the Program Group as shown in Figure 26.

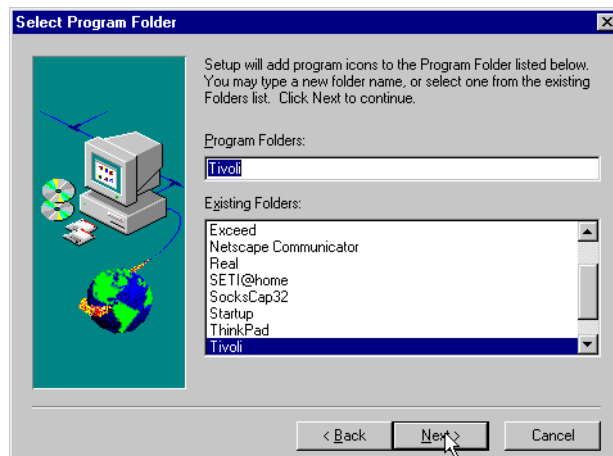


Figure 26. Select Program Folder

19. In the window, shown in Figure 27, you will see the Installation Complete message.

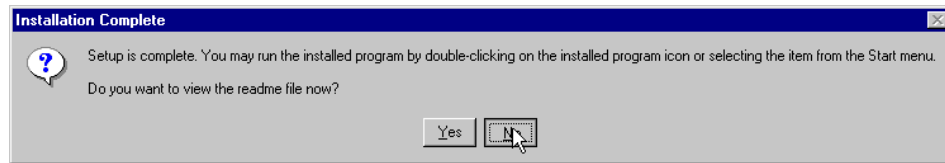


Figure 27. Installation Complete

20. After installing Tivoli Framework 3.6, you need to install 3.6.2 patches.  
Insert the Tivoli Framework 3.6.2 Patch CD-ROM.

21. Start the Tivoli Desktop from Windows by selecting **Start->Tivoli->Tivoli**.

22. Log in to the local machine. Use your Windows IP address or host name as shown in Figure 28.

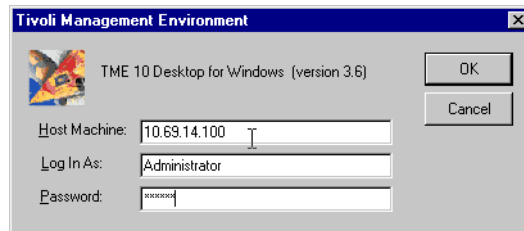


Figure 28. Tivoli Desktop login

23. Select **Desktop->Install->Install Patch** from the menu shown in Figure 29 on page 51.



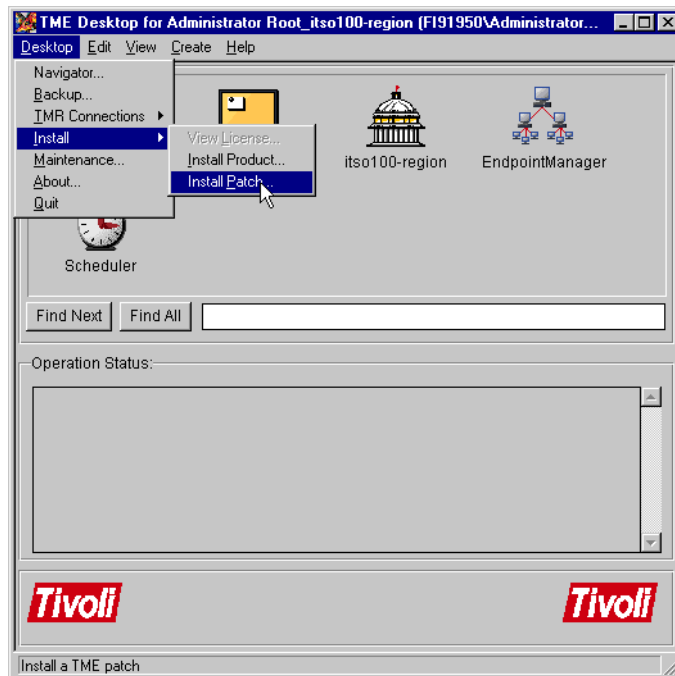


Figure 29. Patch Installation

24. You will probably get the error message, shown in Figure 30, because the program could not find the Installation Images from the default directory. Just click **OK**.

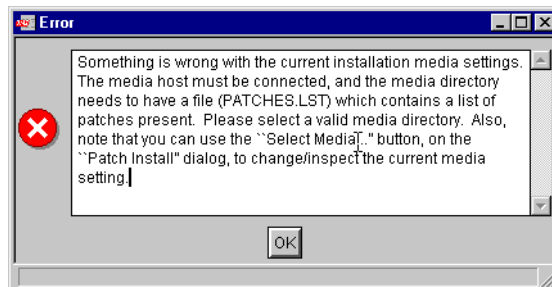


Figure 30. Error message

25. Set the directory point to the CD-ROM location as shown in Figure 31 on page 52.

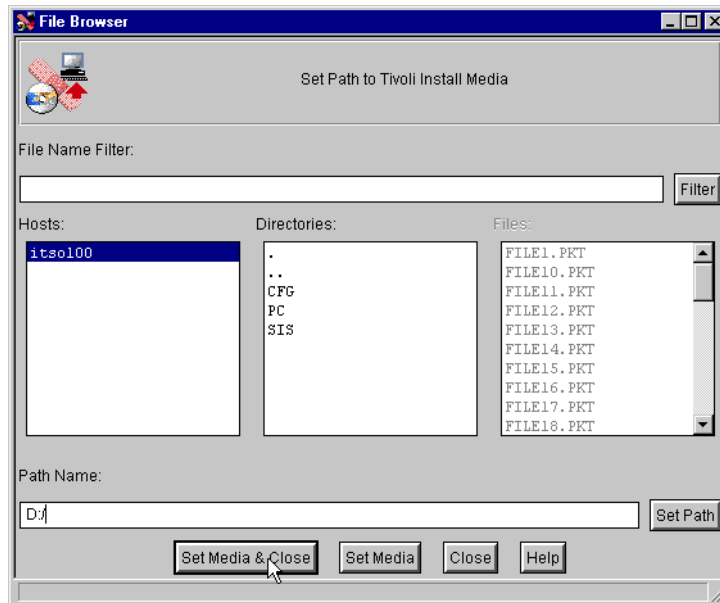


Figure 31. Browse directory

26. Select **Tivoli Management Framework 3.6.2 Maintenance Release** and click **Install & Close** as shown in Figure 32 on page 53.

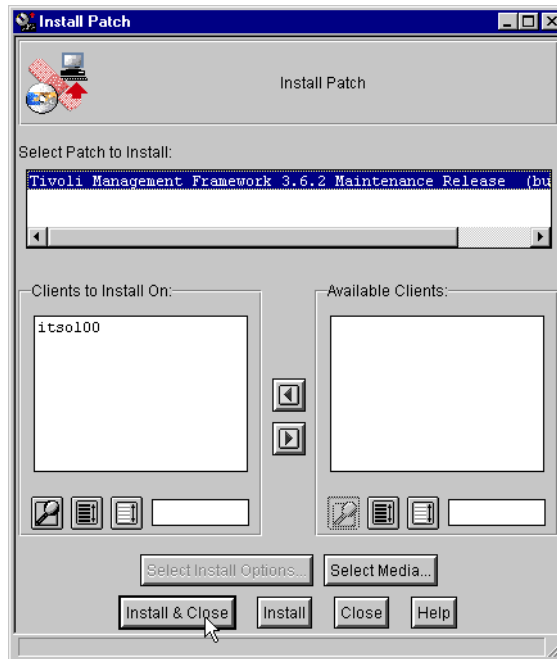


Figure 32. *Install Patch*

27. You will receive a window similar to the one in Figure 33 on page 54. Click **Continue Install**.

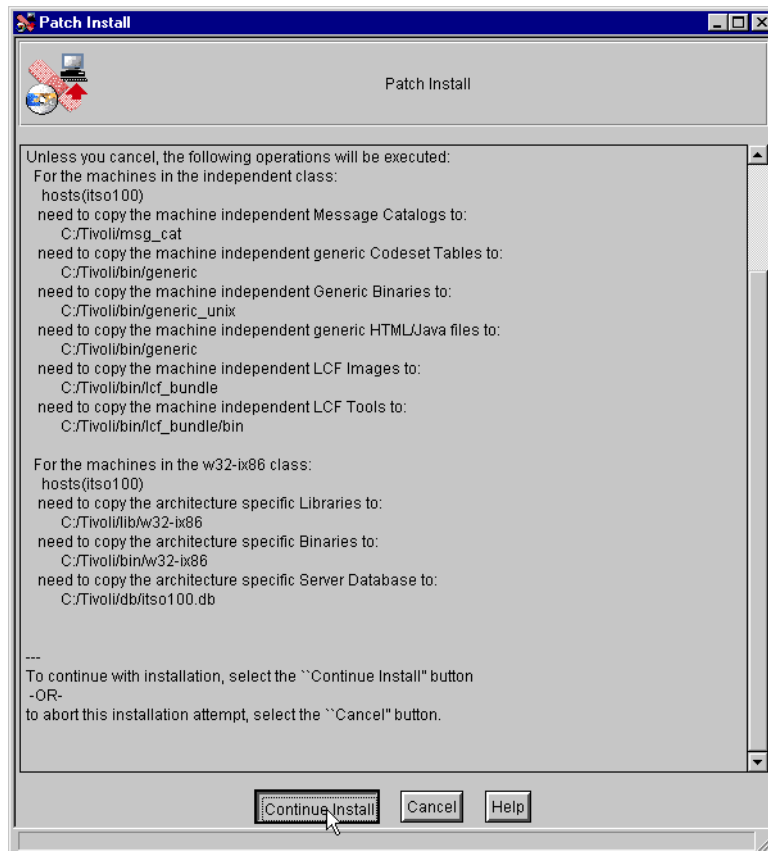


Figure 33. Patch Install information

28. When the installation finishes, quit the Tivoli Desktop.
29. Go to the Control Panel/Services to stop and start the **Tivoli Object Dispatcher**.
30. Make a new Command Line Icon that includes setting up Tivoli environments. This is very useful when you need to execute Tivoli commands from CLI.

Right-mouse click on the desktop and choose **New ->Shortcut**. In the Command line box, type the following:

```
%windir%\system32\cmd.exe /k
%windir%\system32\drivers\etc\Tivoli\setup_env.cmd
```

Your basic TMR Server is now installed.

### 3.5.1 Configuring the TMR Server

Once you install your TMR Server, you have to configure it for the TWS environment. In our environment, we had our TMR Server in a different host than our TWS Master. We performed the following steps in our environment:

1. Add the correct logins to the Tivoli Administrator user (in our case, Root\_itso7-region) by right-clicking and selecting the **Edit Login ...** menu as shown in Figure 34.

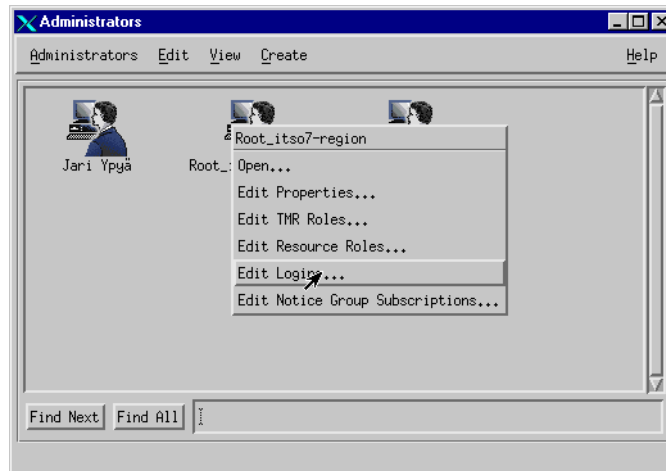


Figure 34. Edit Login menu

2. The user login name must be the same as the TWS user in the host that you are connecting through JSC. In the screen, shown in Figure 35 on page 56, we entered `maestro@itso8` where `itso8` is our TWS Master.

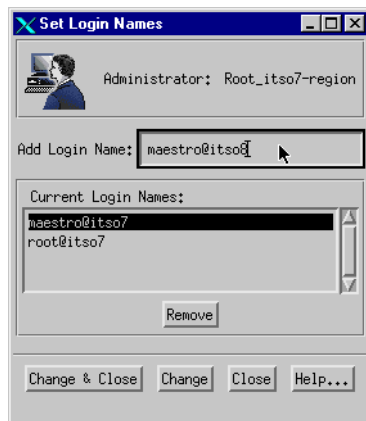


Figure 35. Set Login Names

- Next, we need to create a Policy Region for TWS. Select **Create->Region** as shown in Figure 36.

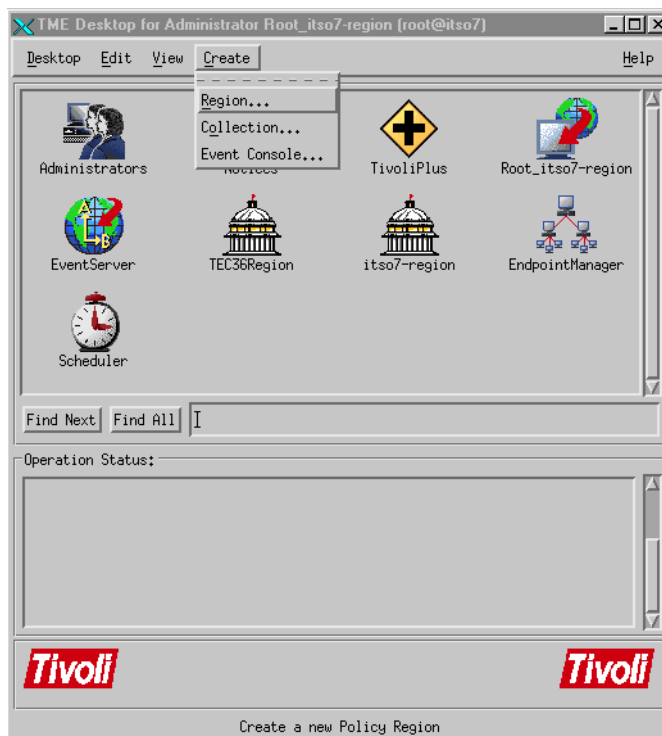


Figure 36. Create Policy Region menu

4. Enter **TWS** as the Policy Region name as shown in Figure 37.



Figure 37. Create Policy Region

5. Next, you have to set the Managed Resources for that Policy Region. Right-click on the **TWS Region** and select **Managed Resources** as shown in Figure 38.

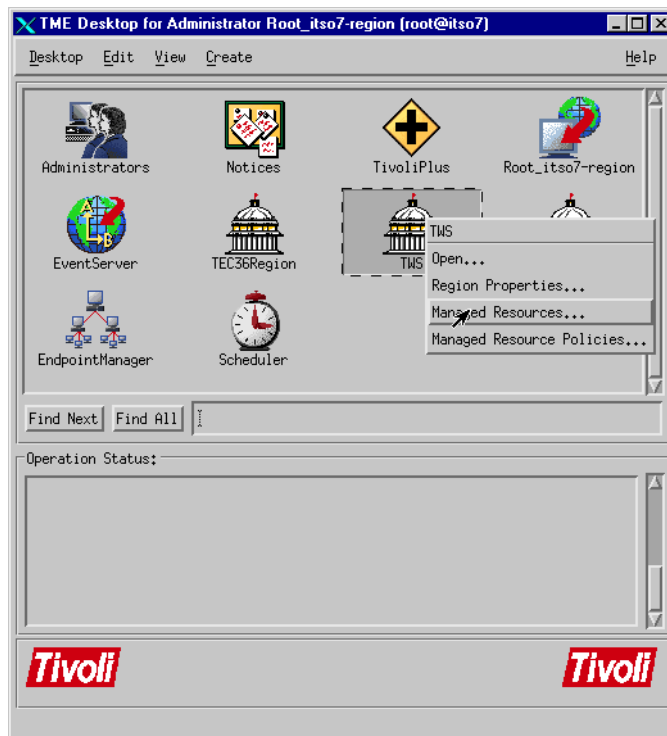


Figure 38. Setting Managed Resources

6. Select, at least, **Managed Node** and **ProfileManager** as shown in Figure 39.

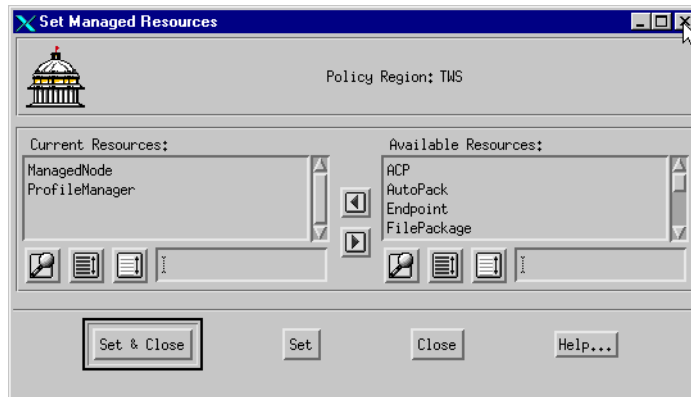


Figure 39. Set Managed Resources

7. Install Managed Node for your TWS Master. In this case, the node is `itso8.dev.tivoli.com`. Log on to TWS Master, create a file system size of 200 MB, and mount it to the directory, `/Tivoli`. Refer to Section 3.4, “Installing TWS Engine 7.0 on UNIX” on page 39, to see how to create a file system.
8. Mount the Tivoli Framework 3.6 CD in the TMR Server.
9. Double click on the **TWS** Policy Region, and then select **Create ManagedNode** as shown in Figure 39 on page 58.



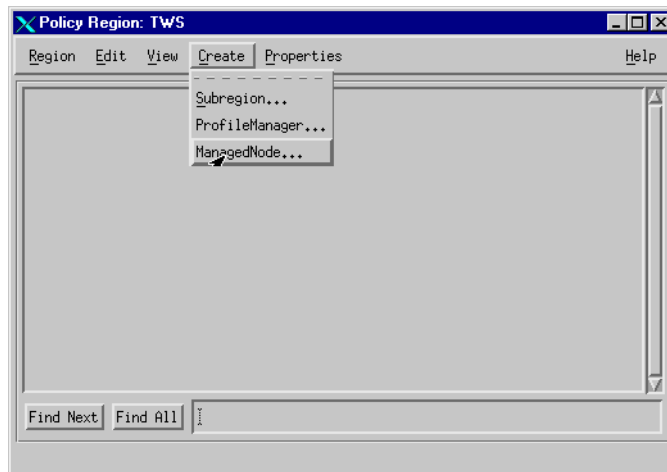


Figure 40. Create Managed Node

10. Insert the root user's password of the node you are installing, and then select **Add Clients** as shown in Figure 41.

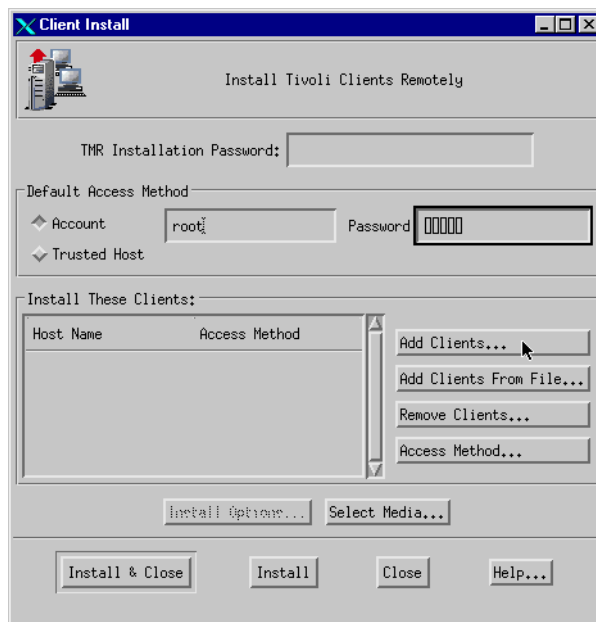


Figure 41. Client Install

11. In the Add Clients window, shown in Figure 42, type `itso8.dev.tivoli.com` for the node name, and then select **Add & Close**.

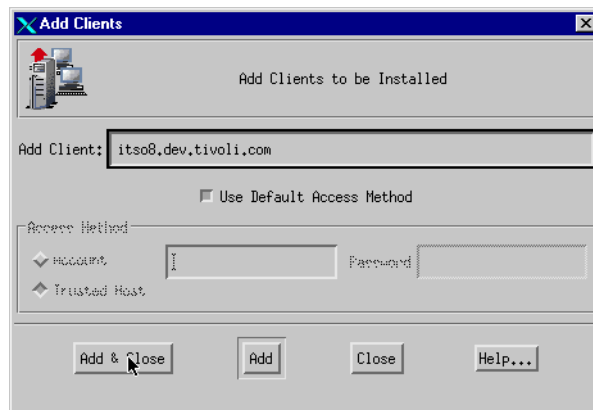


Figure 42. Add Clients

12. Next, you have to select the media from which to install. Click on your TMR Server and the **/cdrom** directory as shown in Figure 43.

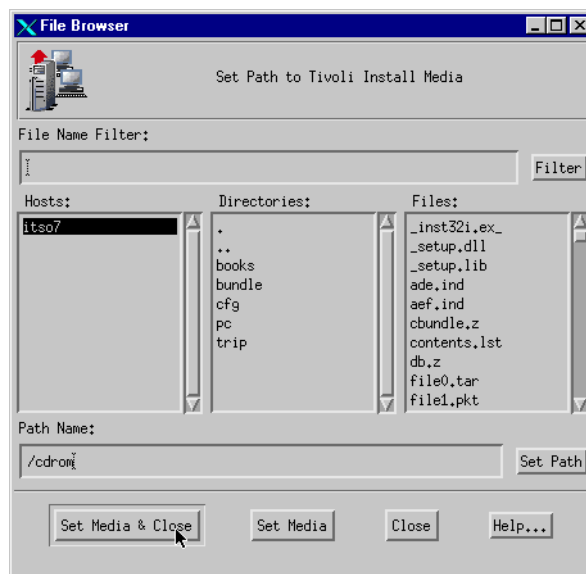


Figure 43. File Browser

13. In the Install Options window, shown in Figure 44 on page 61, you need to specify the correct directories for Tivoli Framework components. You can

see our settings there. If the directories that you specify do not exist, you can activate the **When installing, create “Specified Directories” if missing** checkbox.

You can also check the **Arrange for start of the Tivoli daemon at system (re)boot time** checkbox if you want to start Tivoli Framework automatically after a (re)boot.

Press **Set** and then **Close** after making your selections.

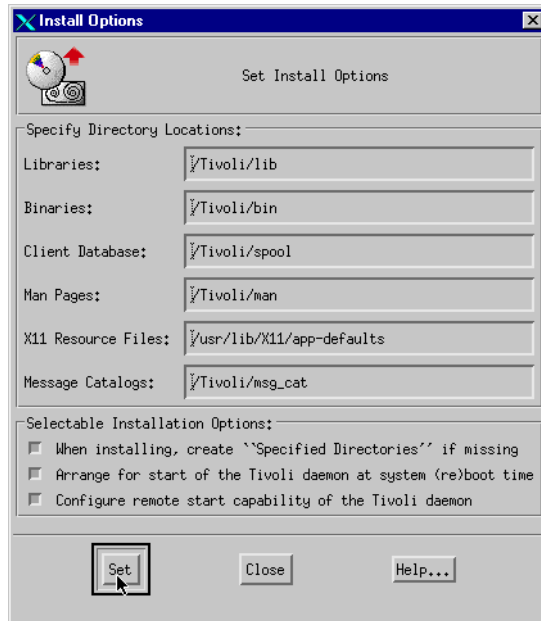


Figure 44. Install Options

14. In the next window, shown in Figure 45 on page 62, click **Continue Install** to begin the actual process.

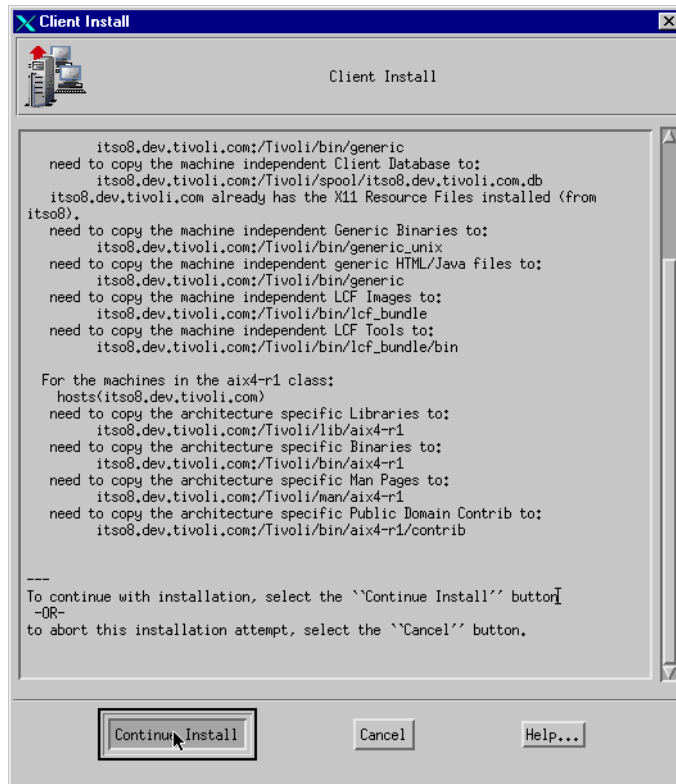


Figure 45. Client Install confirmation

15. You probably want to use Tivoli Framework 3.6.2; so, you have to install patches. Unmount the Tivoli Framework 3.6 CD-ROM and mount the 3.6.2 TMF Patch CD-ROM.

Select **Desktop->Install->Install Patch** to install Tivoli Framework Patch 3.6.2 as shown in Figure 46 on page 63.

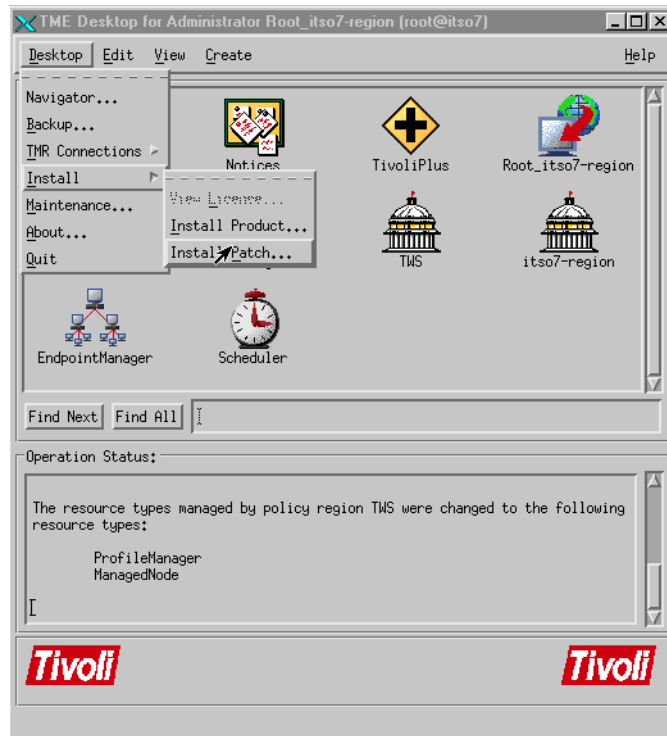


Figure 46. Patch installation

16. In the Install Patch window, shown in Figure 47 on page 64, select **Tivoli Management Framework 3.6.2. Maintenance Release**, and then select the clients to install the patch.

Click **Install & Close** after making your selections.

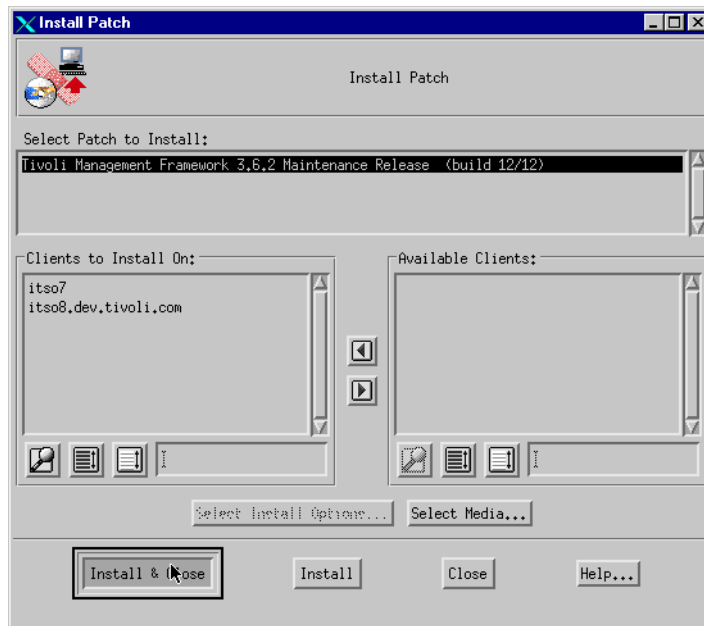


Figure 47. Install Patch

17. Next, you need to install Job Scheduling Services to the TMR Server and the TWS Master. Unmount Tivoli Framework 3.6.2 Patch CD-ROM and mount the TWS CD-ROM.
18. Select **Desktop->Install->Install Product** to install Job Scheduling Services as shown in Figure 48 on page 65.

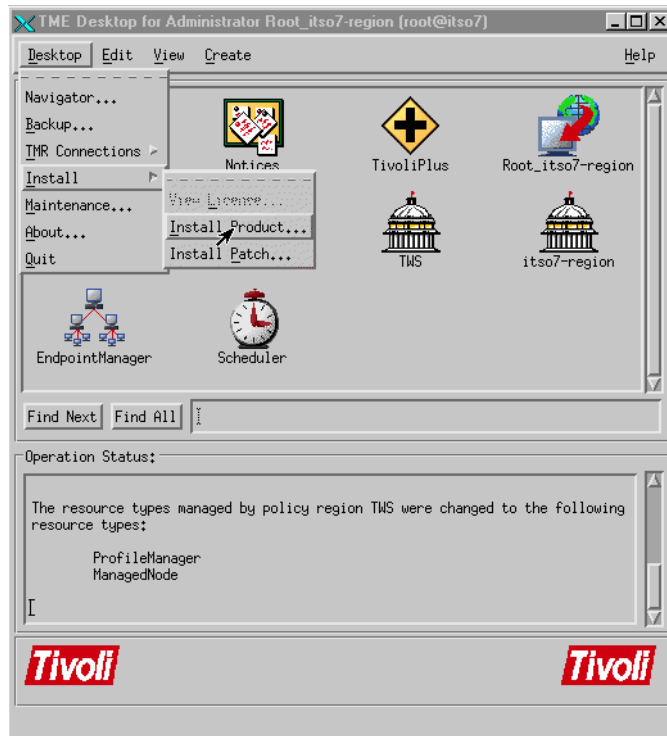


Figure 48. Install Product

19. Select media and change path to /cdrom/TWS/TWSSCon as shown in Figure 49 on page 66.

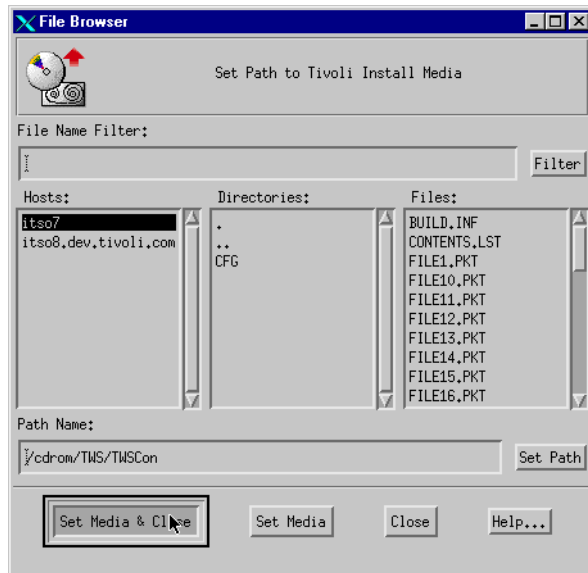


Figure 49. Media selection

20. First, select **Tivoli Job Scheduling Console**. Install to both the TWS Master and TMR Server nodes.

After making your selections, click **Install & Close**, as shown in Figure 50 on page 67. This will install the Tivoli Job Scheduling Console to the TWS Master and the TMR Server.



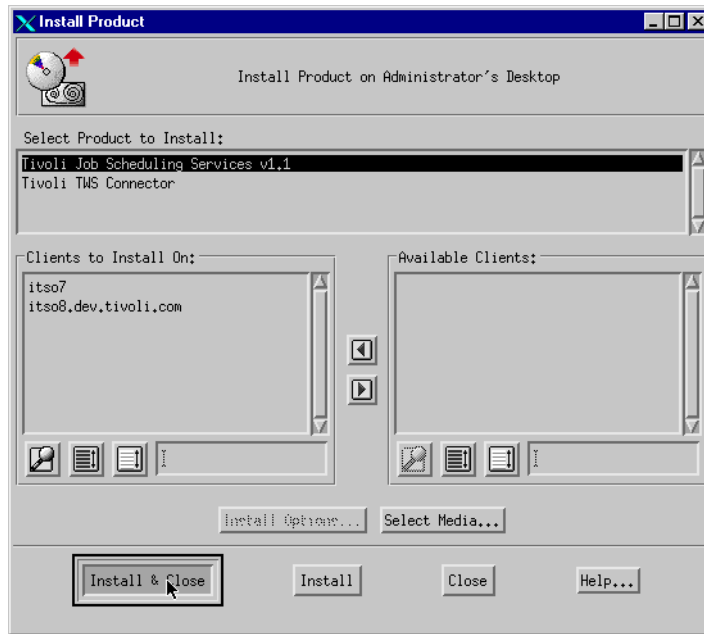


Figure 50. Product selection for install

21. Next, you need to install the TWS Connector to the TMR Server and TWS Master. Select **Desktop->Install->Install Product** from the Tivoli Desktop, and then choose **Tivoli TWS Connector**, as shown in Figure 51 on page 68. You will be prompted with an Install Options panel. *Do not create an instance in this phase* because we install two clients at a time, and instance names have to be unique. You can create instances later from the command line using the `wtwsconn.sh` utility.

Press **Set** and then **Close** to finish the installation of TWS Connector.

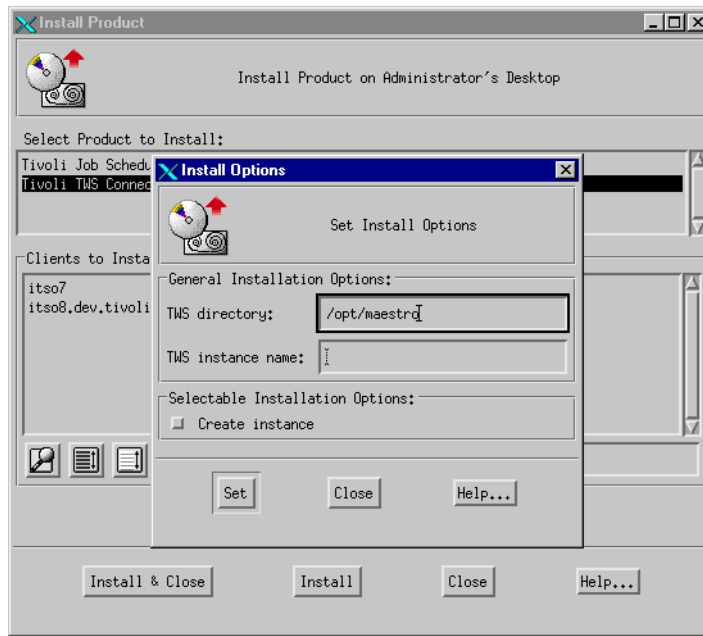


Figure 51. Install TWS Connector

#### Note

You can verify the installation of the TWS Connector using the Tivoli Framework `wlookup` command. The TWS Connector creates the following Framework classes, which are visible with the `wlookup` command.

- MaestroEngine
- MaestroPlan
- MaestroDatabase

For example, you can run `wlookup` as follows:

```
wlookup -R | grep Maestro
MaestroDatabase
MaestroEngine
MaestroPlan
```

22. Next, you need to create a TWS Connector instance for each TWS engine that you want to access with the Job Scheduling Console. To create TWS Connector instances, you must be a Tivoli administrator with admin, senior, or super authorization roles. To create an instance, use the

`wtwsconn` command on the TMR Server or Managed Node where you installed the TWS Connector that you need to access through the Job Scheduling Console.

The `wtwsconn` command is invoked to create, remove, view, stop, and set TWS Connector instances. The following screen shows various options that can be used with the `wtwsconn` command.

```
wtwsconn.sh -create [-h <node>] -n <instance> -t <twmdir>
wtwsconn.sh -remove -n <instance>
wtwsconn.sh -view -n <instance>
wtwsconn.sh -stop -n <instance> | -t <twmdir>
wtwsconn.sh -set -n <instance> -t <twmdir>
```

Log in to the TMR Server as root user and add an instance to the TWS Master using the command shown in the following screen.

```
$. /etc/Tivoli/setup_env.sh
$wtwsconn.sh -create -h itso8 -n master -t /opt/maestro

Scheduler engine created
Created instance: itso8, on node: itso8
MaestroEngine 'maestroHomeDir' attribute set to: /opt/maestro
MaestroPlan 'maestroHomeDir' attribute set to: /opt/maestro
MaestroDatabase 'maestroHomeDir' attribute set to: /opt/maestro
```

Creating multiple TWS Connector instances is required if you want to access JSC services from multiple TWS engines. Figure 52 on page 70 explains a possible scenario where you use a TWS Master and a TWS Backup Master.

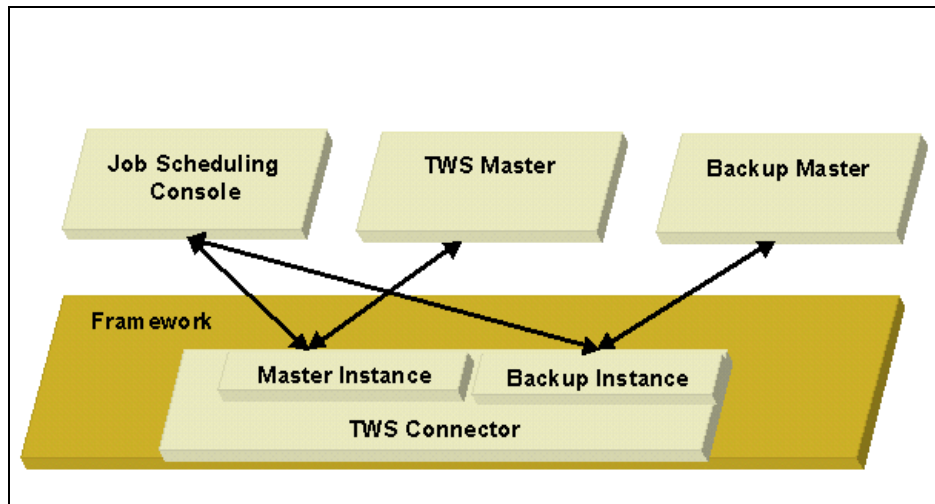


Figure 52. Multiple TWS Connector instance scenario

A TWS Connector starts up automatically when someone issues a command from the JSC. However, you have to stop it manually. On Windows NT, you must stop the TWS Connector before running the `makesec` command. On UNIX, you can stop it either before or after running the `makesec` command.

That is all for Tivoli Framework and JSC installation.

### 3.6 Setting the security of TWS engines

After you install the TWS engine and before you proceed to install the JSC, you must set the TWS security. Each workstation in a TWS network (domain managers, fault-tolerant agents, and standard agents) has its own *Security file*. The Security file contains one or more user definitions. Each user definition identifies a set of users, the objects they are permitted to access, and the types of actions they can perform. TWS programs and commands determine a user's capabilities by comparing the user's name with the user definitions in the Security file. You can maintain a file on each workstation, or you can create a Security file on the master domain manager and copy it to each domain manager, fault-tolerant agent, and standard agent.

The Security file template is installed as `TWShome/Security`, and a compiled operational copy is installed as `TWShome/./unison/Security`.

In order to set the security of the TWS engines, perform the following steps:

1. Log in to TWS Master as maestro user and dump current security to a file:

```
$ dumpsec > mysec
```

2. Modify security file. Add the **TMF Administrator user** into the file. In our case, the user is Root\_itso7-region. Execute `vi mysec` from a command line and add Root\_itso7-region as shown in the following screen.

```
USER MAESTRO
  CPU=@+LOGON=maestro,root,Root_itso7-region
BEGIN
  USEROBJ CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,ALTPASS
  JOB CPU=@ ACCESS=ADD,ADDDEP,ALTPRI,CANCEL,CONFIRM,DELDEP,DELETE,DI
SPLAY,KILL,MODIFY,RELEASE,REPLY,RERUN,SUBMIT,USE
  SCHEDULE CPU=@ ACCESS=ADD,ADDDEP,ALTPRI,CANCEL,DELDEP,DELETE,DI
SPLAY,LIMIT,MODIFY,RELEASE,REPLY,SUBMIT
  RESOURCE CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,RESOURCE,USE
  PROMPT ACCESS=ADD,DELETE,DISPLAY,MODIFY,REPLY,USE
  FILE NAME=@ ACCESS=CLEAN,DELETE,DISPLAY,MODIFY
  CPU CPU=@ ACCESS=ADD,CONSOLE,DELETE,DISPLAY,FENCE,LIMIT,LINK,MODIF
Y,SHUTDOWN,START,STOP,UNLINK
  PARAMETER CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY
  CALENDAR ACCESS=ADD,DELETE,DISPLAY,MODIFY,USE
END
~
~
"mysec" 13 lines, 699 characters
```

3. Stop the instances. Log in TWS Master as maestro user. Instances start automatically when needed by JSC.

```
wmaeutil ALL -stop ""

AWS20710766I Stopping servers for <master> Maestro instance...
AWS20710758I Done stopping the ENGINE server
AWS20710758I Done stopping the DATABASE server
AWS20710758I Done stopping the PLAN server
```

4. Stop the TWS engine with the following command:

```
$ conman stop
```

5. Use the `makesec` command to compile and install the new operational Security file with the following command:

```
$ makesec mysec
```

6. Start the TWS Engine with the following command:

```
$ conman start
```

Use dumpsec and makesec every time you want to modify the Security file. Changes to the Security file take effect when TWS is stopped and restarted.

To have consistent security on all TWS Workstations, you should copy */opt/unison/Security* to other TWS Engines. Remember to stop the Connector instances after updating Security and also stop and start TWS Engines or *you can wait until the next Jnextday to run*.

### **3.6.1 A note about using the Tivoli Framework security model**

With TWS 7.0, TWS security is integrated with Tivoli Framework security. You need to define Tivoli Administrators (in our example, Root\_itso7-region) in the TWS Security file. In the authorization phase, the TWS server maps the Tivoli Framework user to TWS authorized users. This is also valid for OPC 2.3 where the OPC Server maps the authorized identity to a corresponding RACF user ID. Using the Tivoli Framework security has the following benefits:

- It enables schedulers to be accessed by Tivoli Framework users.
- Single sign-on to avoid separate authorization for each scheduler.
- Consistency with the security model used for other Tivoli applications.
- Finer granularity in defining access rights.

When the Job Scheduling Console user signs on to the Tivoli Framework by providing a user ID and password, the checking is done by Tivoli Framework security, and the user ID is associated to an Administrator. No other user ID/password pair will be requested to sign on.

---

## **3.7 Job Scheduling Console installation**

The Job Scheduling Console has the following system requirements:

- CD-ROM drive for installation
- Approximately 50 MB free disk space
- 128 MB RAM memory
- TCP/IP network communications
- Java version 1.1.8 for UNIX machines

The Tivoli Job Scheduling Console is available for the following operating systems:

- Microsoft Windows NT Version 4.0 with Service Pack 4 or later
- Windows 2000

- Windows 98
- AIX 4.2.1 or 4.3
- Sun Solaris 2.6 or 2.7
- HP-UX 10.x or 11.0

Perform the following steps to install the Job Scheduling Console:

1. Log in as root or Administrator.
2. Insert the Tivoli Job Scheduling Console CD-ROM into the system CD-ROM drive, or mount the CD-ROM from a drive on a remote system. For this example, the CD-ROM drive is drive D.
3. Run the installation command:

**On Windows:**

From the **Start** menu, select the **Run...** option to display the Run dialog. Enter `d:\jsgui\windows\install.exe` in the Open field.

**On AIX:**

Set the `DISPLAY` variable if you are logged in from a remote machine as shown in the following screen:

```
$ export DISPLAY=10.68.14.100:0
$ sh install.bin
InstallAnywhere is preparing to install...

Please choose a Java virtual machine to run this program.
(These virtual machines were found on your PATH)
-----
1.      /usr/bin/java
2.      /usr/bin/jre
3.      Exit.

Please enter your selection (number): 1
```

4. In the next window, shown in Figure 53 on page 74, you can select the language of JSC. The default is English. Press **OK** after making your selection.



Figure 53. Tivoli Job Scheduling Console

5. Choose the installation location in the next window.
6. Choose **Customized Install** if you want to choose a language other than English or install selective components as shown in Figure 54. Selecting Full Package is sufficient for English versions. We selected **Customize Install** to show you the different options.

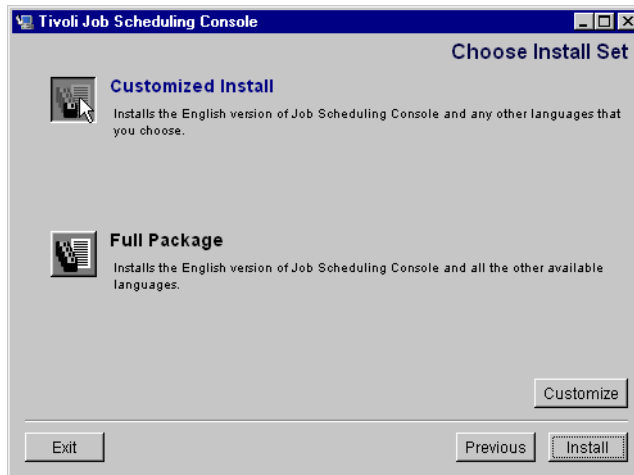


Figure 54. JSC Installation

7. In the next window, shown in Figure 55 on page 75, you can select the languages you want to install.



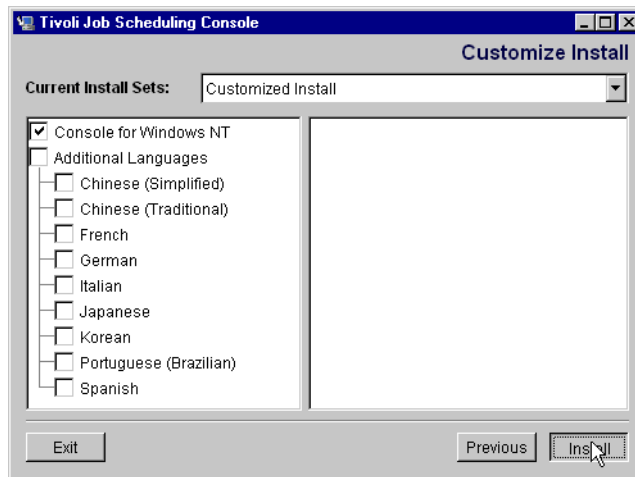


Figure 55. Select language

8. Next, you need to choose the install folder as shown in Figure 56.

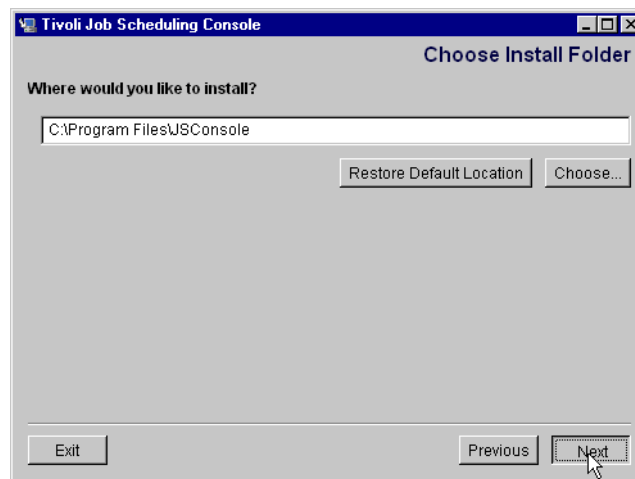


Figure 56. Choose Install folder

9. In the next window, shown in Figure 57 on page 76, choose where to install the shortcut icons.

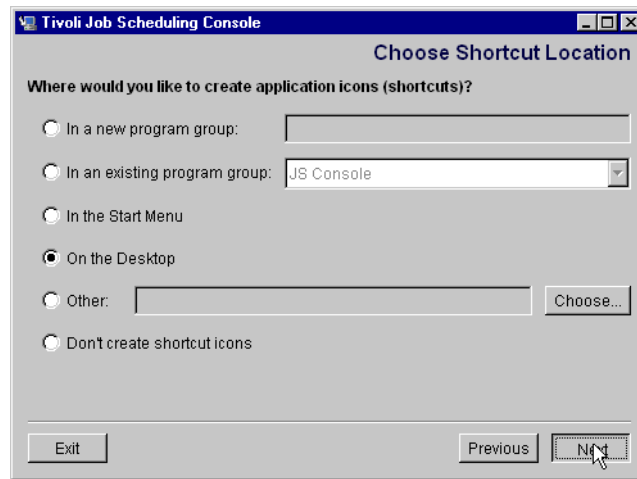


Figure 57. Shortcut location

10. The next window, shown in Figure 58, shows that the installation was successful.



Figure 58. Installation successful message

11. In AIX, you have to modify *AIXconsole.sh* to be able to start JSC.

```
$ cd /opt/maestro/JSConsole/bin/java
$ vi AIXconsole.sh
```

Modify the JAVAPATH to point to the directory in which your Java files reside.

```
JAVAPATH=/usr/jdk_base
```

## 12. Start JSC:

### ***In Windows:***

- DoubleClick **JSC icon**.

### ***In AIX:***

- It is recommended that you create a shell script that starts JSC. Just create a file, for example, one named JSC and add the lines in the following screen.

```
$ vi /usr/bin/jsc
cd /opt/maestro/JSCConsole/bin/java
./AIXconsole.sh
```

- Save the file and change file attributes as follows:

```
$ chmod 755 /usr/bin/jsc
```

- Just enter `jsc` from a window, and the Job Scheduling Console will start.

---

## 3.8 Summary

In this chapter, we covered the installation of our TWS environment. At a minimum, the installation requires the following steps:

1. Install the TWS Master and Workstations.
2. Install the TMR Server (either on a machine separate from the Master or on the same machine).
3. If your TWS Master is on a different machine than the TMR Server, create a Managed Node on the host on which you will install the TWS Master.
4. Install Job Scheduling Services and TWS Connector on the TMR Server and the TWS Master.
5. Create a Tivoli Administrator for TWS.
6. Set the security of TWS Workstations.
7. Install the Job Scheduling Console on all nodes where you need to control production using a GUI.



---

## Chapter 4. Using the Job Scheduling Console GUI

The Job Scheduling Console Graphical User Interface (JSC GUI) is a standard interface to Tivoli Workload Scheduler (TWS) and Operations Planning and Control (OPC). You can manage work running under TWS and OPC from the same GUI. The JSC is the new GUI, which replaces existing GUIs for both TWS and OPC.

Although you can access information about all your TWS- and OPC-controlled work from the same JSC panel menu, the fields on the windows will be different between OPC- and TWS-controlled work. There are product-specific extensions to the JSC to cater to the requirements of each product.

This chapter discusses the use of the JSC GUI with particular emphasis on:

- What you can do with the JSC
- What you should do currently from legacy user interfaces
- Considerations when moving to the JSC GUI from the current TWS GUI
- Considerations when moving to the JSC GUI from the OPC dialogs

### Note

The JSC is in continuous development and new functionality added on each release. Therefore, functions that are now only available from the legacy user interfaces are being migrated to the JSC.

Figure 59 on page 80 shows the JSC start window. The processors, known as engines, being managed at the ITSO are listed in the left pane. They are:

- OPC - OPC-controlled OS/390 work
- itso7 - TWS-controlled AIX work
- itso9 - TWS-controlled AIX work
- master - TWS-controlled AIX work

OPC represents an OPC system running on an OS/390 processor in Mainz, Germany. The others are TWS systems running on AIX processors in Austin, Texas.

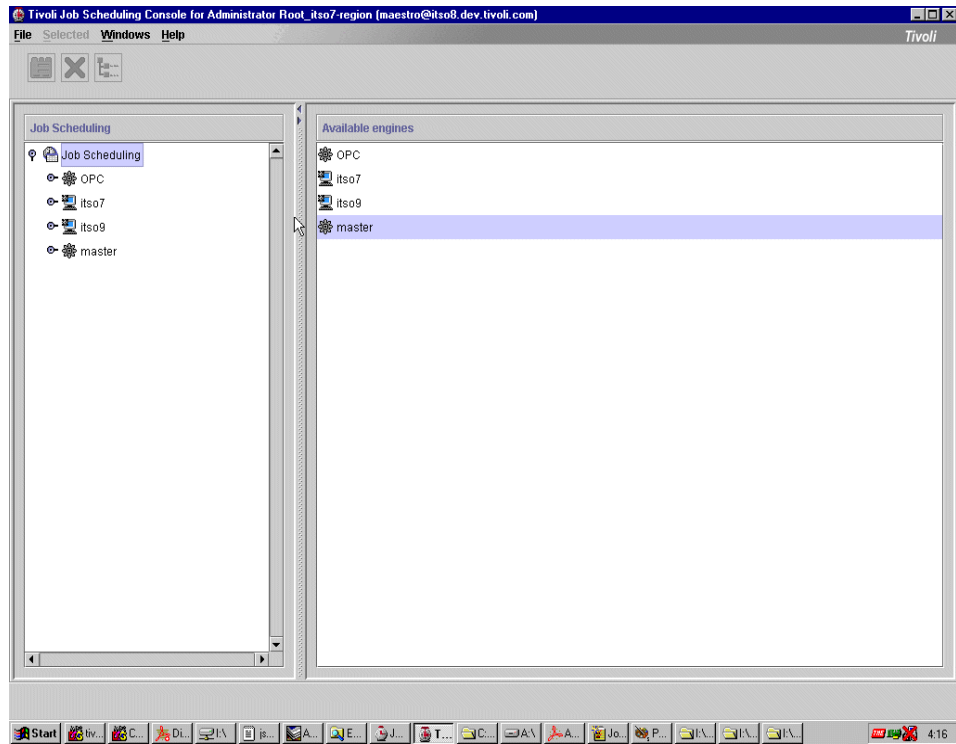


Figure 59. JSC start

Clicking on the **key symbol** to the left of the name expands the menu.

## 4.1 Starting the Job Scheduling Console

Start the JSC according to the method you chose at installation. Figure 60 shows the JSC being selected from a Windows NT start menu.

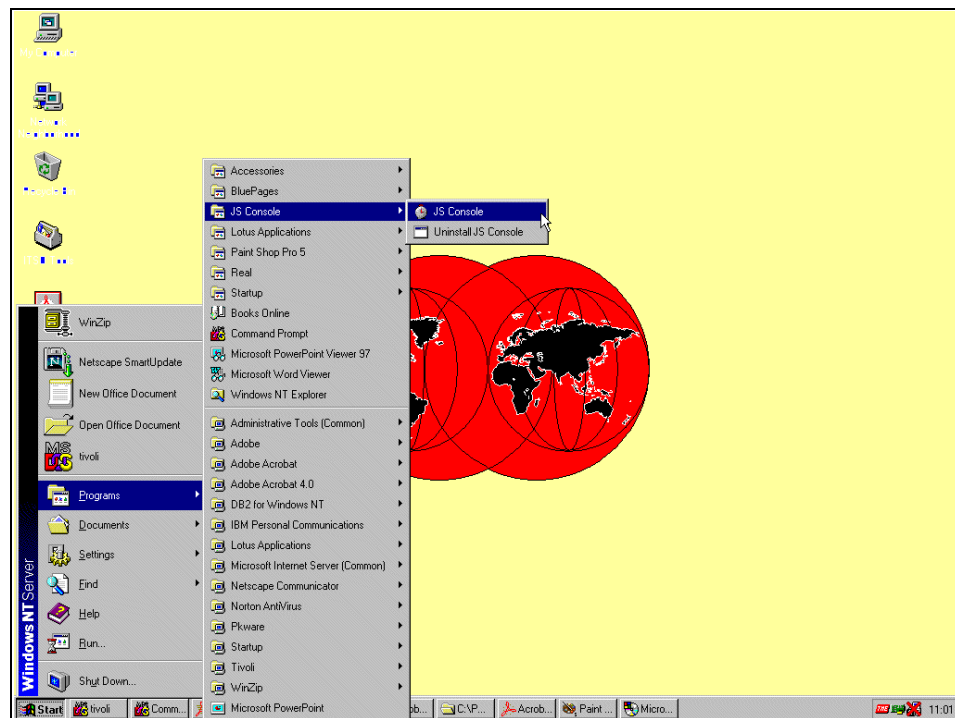


Figure 60. Starting the JSC

As the JSC starts, you are asked to enter the location of the engine on which the Tivoli Framework is running and your password as shown in Figure 61 on page 82.

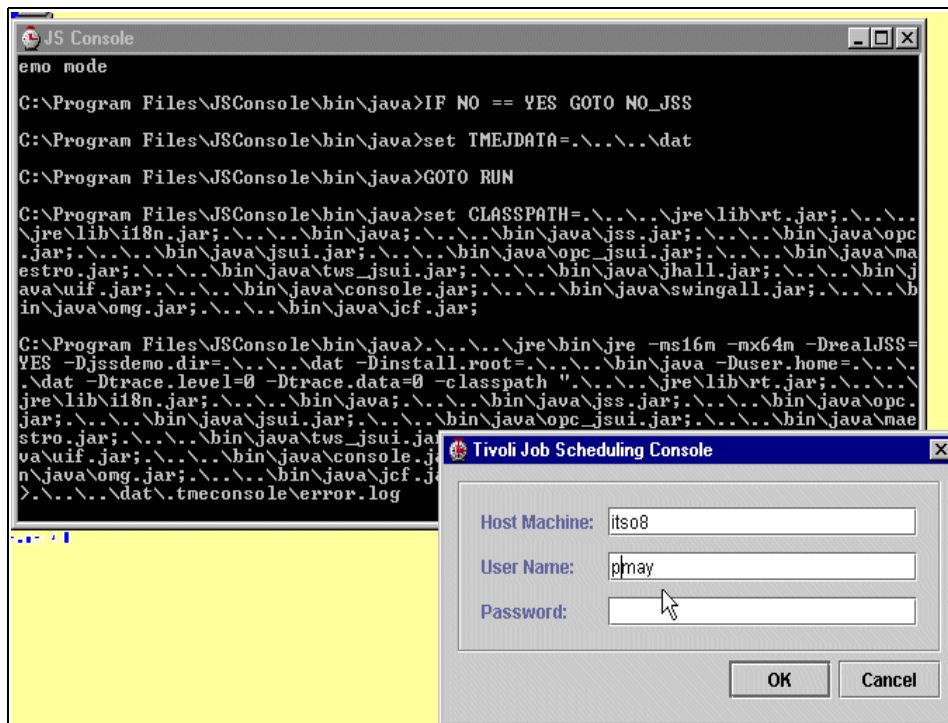


Figure 61. JSC password prompt

As the JSC starts, it briefly displays a window with the JSC level number, as shown on Figure 62 on page 83.

**Note**

Do not close the MS-DOS window. Closing the MS-DOS window terminates the JSC session; reduce the risk of doing this by minimizing the window.





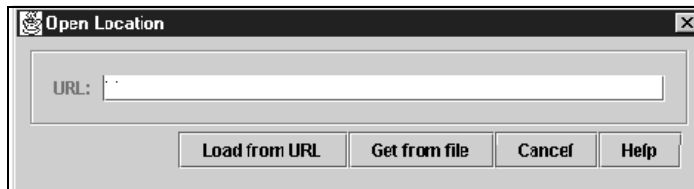


Figure 64. Open Location

This lets you copy a pre-customized user profile, which could contain list views appropriate to the user. It could also be used to turn off the showing of the next window, shown in Figure 65.

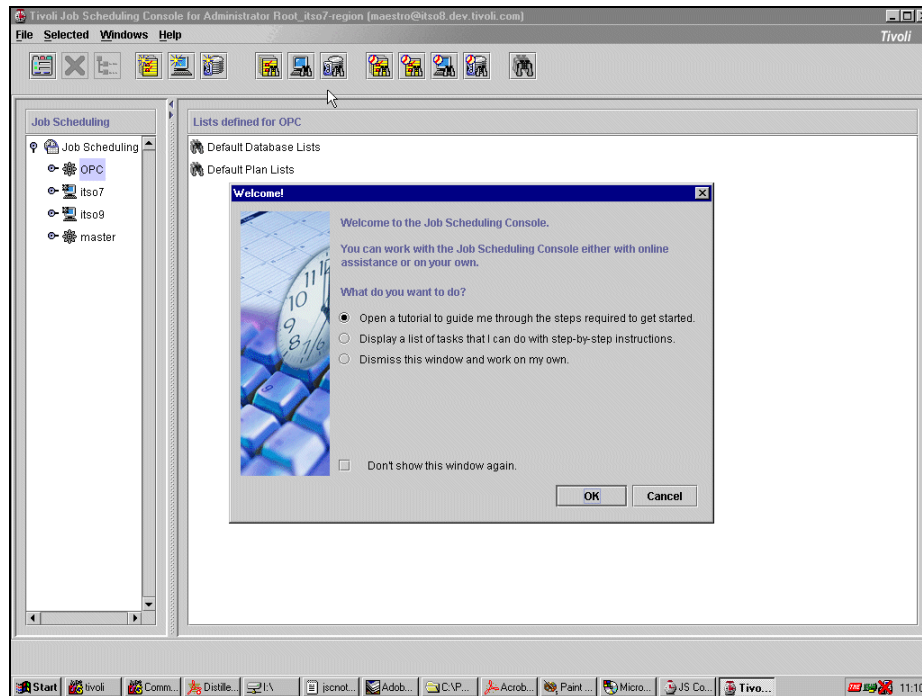


Figure 65. Starting with the JSC

This window gives you the option to read the online tutorial. Unless you turn the future display of this window off by ticking **Don't show this window again**, it will be shown every time you start the JSC. The tutorial encourages you to create a workstation and then a job stream (OPC application).

It is very unlikely that you will want your users to do this in a production database, and you may consider it worthwhile to load a pre-customized standard profile file onto users' machines when you install the JSC. Our recommendation is for you to configure your JSC lists to your data center standard preferences with the intention of using these settings as the default initialization file. Once you have set them up, save them and install the file on users' machines when you install the JSC.

The file name is GlobalPreferences.ser, and it can be found in the JSC directory, JSConsole/dat/.tmeconsole/USER@NODE\_LANGUAGE, in Windows or in \$HOME/.tmeconsole/USER@NODE\_LANGUAGE in UNIX (where USER is the user name used to log into TWS; NODE is the node to log in on, and LANGUAGE is your language settings in TWS). It will be about 20 KB in size, depending on the number of list views you have created. (There is another GlobalPreferences.ser file in the tivoliconsole subdirectory that is about 1 KB in size. It is not the one that contains the changed preferences).

**Note**

JSC client uses Windows regional settings to display dates and times. Change the regional settings from the control panel according your country. After rebooting Windows, dates and times will be shown in the selected format.

---

## 4.2 Job Scheduling Console for OPC

OPC GUIs have been available for many years. There is a GUI for creating application descriptions (The GUI for Application Description) and a GUI for monitoring and managing work in the current plan (Workload Monitor/2). The JSC GUI combines the functions of these GUIs and provides additional functionality. However, since very few people use these GUIs, we assume that you are currently using OPC ISPF dialog panels. Therefore, this section discusses the JSC in relation to the ISPF dialogs rather than comparing the JSC to earlier OPC GUIs. It summarizes what can and cannot be done with the JSC and recommends and shows how to create lists that emulate certain OPC dialog panels. It does not attempt to duplicate the excellent step-by-step guide on how to use the JSC, *Tivoli OPC V2R3 Job Scheduling Console User's Guide*, GC32-0402.

**Note**

The JSC frequently requires a right mouse click on an item to display a menu of available functions

#### **4.2.1 What you can do with the JSC**

You can use the JSC to work in two areas: The database and the current plan.

In the database, you can:

- Create, modify, and browse applications and job descriptions
- Browse a single view of *all* planned occurrences for an application with multiple run cycles
- Create, modify, and browse workstations
- Create, modify, and browse special resources

In the current plan, you can:

- List applications and their operations and
  - Browse and modify them
  - Reset their status
  - EX them
  - Rerun them by resetting their status
  - Delete them
  - Display a timeline
- List workstations and
  - Browse and modify them
  - List operations using them
- List special resources and
  - Browse and modify them
  - List operations using them

#### **4.2.2 What you should do through OPC ISPF dialogs**

Some administration functions not present on the JSC can be performed through legacy user interfaces. The following sections summarize them:

#### 4.2.2.1 Database

Access the calendar, period, operator instruction, Event-triggered Tracking, and JCL variable databases.

#### 4.2.2.2 Long Term Plan

All Long Term Plan related tasks.

#### 4.2.2.3 Plan - JCL Edit.

Editing jobs or correcting JCL on failed jobs. Using a JCL preparation workstation to prep jobs.

#### 4.2.2.4 Plan - New Occurrence

Add a new occurrence (job stream instance) into the plan.

#### 4.2.2.5 Plan - Ended in Error list

A view of operations in error status can be created on the JSC, but a more rich functionality is still available from the ISPF Ended in Error panel. This includes JCL edit, application rerun from a selected operation, step level restart, joblog browse, automatic recovery attempt, remove, delete, and complete from group.

#### 4.2.2.6 Batch jobs

Invoke OPC batch jobs; so, you can extend or modify the Long Term and Current Plans.

#### 4.2.2.7 Service Functions

Access OPC Service Functions ; so, you can stop or start Job Submission, Automatic Recovery, and Event-Triggered Tracking, or do a Long Term Plan refresh.

### 4.2.3 Terminology used in the Job Scheduling Console

The JSC uses a standardized terminology for both OPC and TWS work. Table 6 lists OPC terms, their JSC equivalents, and brief explanations.

Table 6. Terminology uses in the JSC

OPC	JSC	Explanation
Application Description	Job stream	A sequence of jobs including the resources and workstations that support them and scheduling information.

OPC	JSC	Explanation
Application Group	Job stream template	A grouping of job streams that provides scheduling information, such as a calendar, a free-day rule, and run cycles that can be inherited by all the jobstreams that have been created using the template.
Current plan	Plan	A detailed plan of system activity that covers a period of at least one minute and not more than 21 days, but typically one or two days. The plan encompasses all job and job stream instances and the resources and workstations involved in running them.
External Dependency	External job	A job from one job stream that is a predecessor for a job in another job stream.
In-effect date for run cycles	Valid from	The first date that a run cycle applies.
Input arrival time	Earliest start time	The time that the job or job stream is scheduled to start.
Negative run cycle	Exclusionary run cycle	Specifies when a job stream must not be run.
Occurrence	Job stream instance	A job stream that is scheduled for a specific run date in the plan.
OPC Controller	Engine	The OPC component that runs on the controlling system and that contains the OPC tasks that manage the OPC plans and databases.
Operation	Job	A unit of work that is part of a job stream and that is processed at a workstation.
Operation number	Job identifier	The number that identifies a job.

OPC	JSC	Explanation
Operations in the current plan	Job instances	A job scheduled for a specific run date in the plan.
Out-of-effect date for run cycles.	Valid to	The last date that a run cycle applies.
Run cycle with offsets.	Offset-based run cycle	Includes a user-defined period and an offset, such as the 3rd day in a 90-day.
Run cycle with rules.	Rule-based run cycle	Includes a rule, such as the first Friday of March or the second workday of the week.
Special resources	Logical resources	Any type of limited resource, such as tape drives, communication lines, databases, or printers, that is needed to run a job.
Status: Complete	Successful	The job or job stream has been completed.
Status: Delete	Canceled	The job or job stream has been deleted from the plan.
Status: Started	Running	The job has started (jobs only).
Task	Task	A job performed at a computer workstation.

#### 4.2.4 Moving to the Job Scheduling Console

The JSC supplies some default views of the database and plans. OPC users moving to the JSC will not see views matching ones they are familiar with on OPC, such as *workstation ready* and *ended in error* lists. However, it is easy to simulate them by creating JSC plan job lists with appropriate filters. A filter on jobs in Error status will show all errored jobs. A filter on jobs in Arrived or Ready status at a particular workstation will be the equivalent of an OPC ready list.

##### 4.2.4.1 Creating a JSC Error and Ready Lists

We recommend that you create various JSC lists corresponding to your frequently-used OPC panels. Once you have done this, you may save the Global Preferences and propagate it to new users.

Click on the icon that represents your OPC system in the Job Scheduling pane on the left of the JSC window. Then, use the right mouse button and click on **Default Plan Lists**. Select **Create Plan List** on that menu, and **Job Instance** on the next menu as shown in Figure 66.

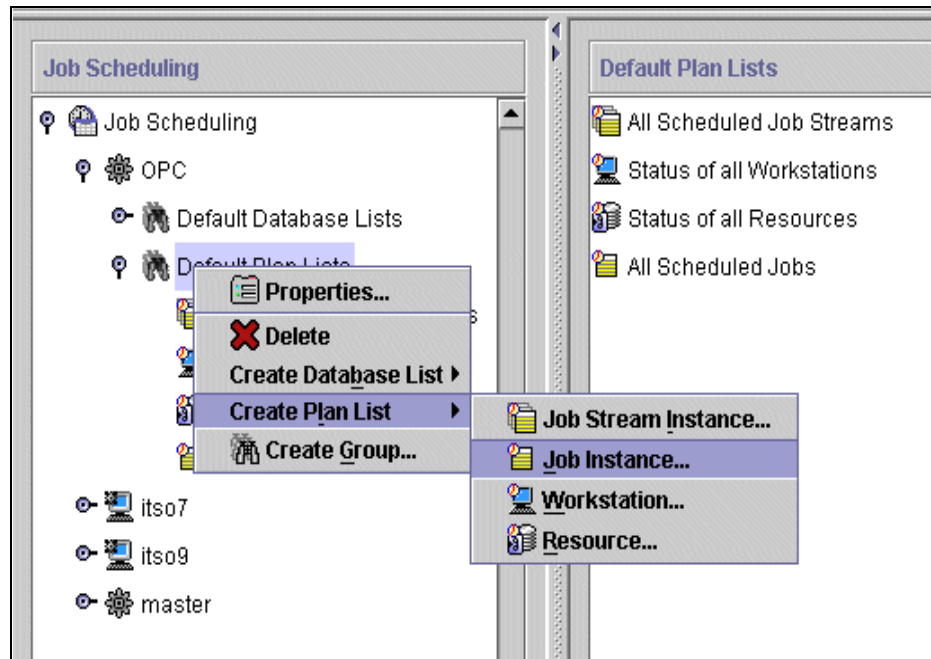


Figure 66. Creating a Job Instance list

A Job Instance List properties window is displayed, as shown in Figure 67 on page 91.



**Properties - Job Instance List**

Name: Job Instance List 6

**Periodic Refresh Options**

☐ Periodic refresh      Period (secs): 60      **Apply defaults**

**Filter Criteria**

Job stream:

Job identifier:

Task name:

Workstation:

Owner:

Authority group:

**Priority**

From: 1 ▼      To: 9 ▼

**Dates**

From Date:   Time:

To Date:   Time:

☒ **Status**  ▼

☐ **Internal Status**  ▼

☐ **Critical job**      **WLM policy**  ▼

**OK**      **Apply**      **Cancel**      **Help**

Figure 67. Job instance properties

Change the name and select the internal status radio button to display OPC statuses. Select error status as shown in Figure 68 on page 92.

**Properties - Job Instance List**

Name: Ended in Error List

**Periodic Refresh Options**

☒ Periodic refresh      Period (secs): 120      **Apply defaults**

**Filter Criteria**

Job stream:

Job identifier:

Task name:

Workstation:

Owner:

Authority group:

**Priority**

From: 1      To: 9

**Dates**

From:  Date:

To:  Date:

**Status**

☐ Status

☒ Internal Status

**Status Selection:**

- ☐ Arriving
- ☐ Ready
- ☐ Started
- ☐ Complete
- ☐ Deleted
- ☐ Interrupted
- ☐ Ready - non reporting workstation
- ☒ Error

Error:

☐ Critical job      WLM policy:

**Buttons:** OK, Apply, Cancel, Help

Figure 68. Choosing job instance status

You can use filter fields to limit the error list display to one workstation, and you can choose to define a periodic refresh rate. Here, we have chosen to refresh every 120 seconds. Save the new list with the **OK** button.

**Note**

Automatic periodic refresh time interval is available for each query filter and may also be defined at the top most level, to be used by all query filters.

Now, the new Ended-in-Error list appears in the Job Scheduling pane, and when you click on it, the right pane displays the job streams containing jobs in error as shown in Figure 69.

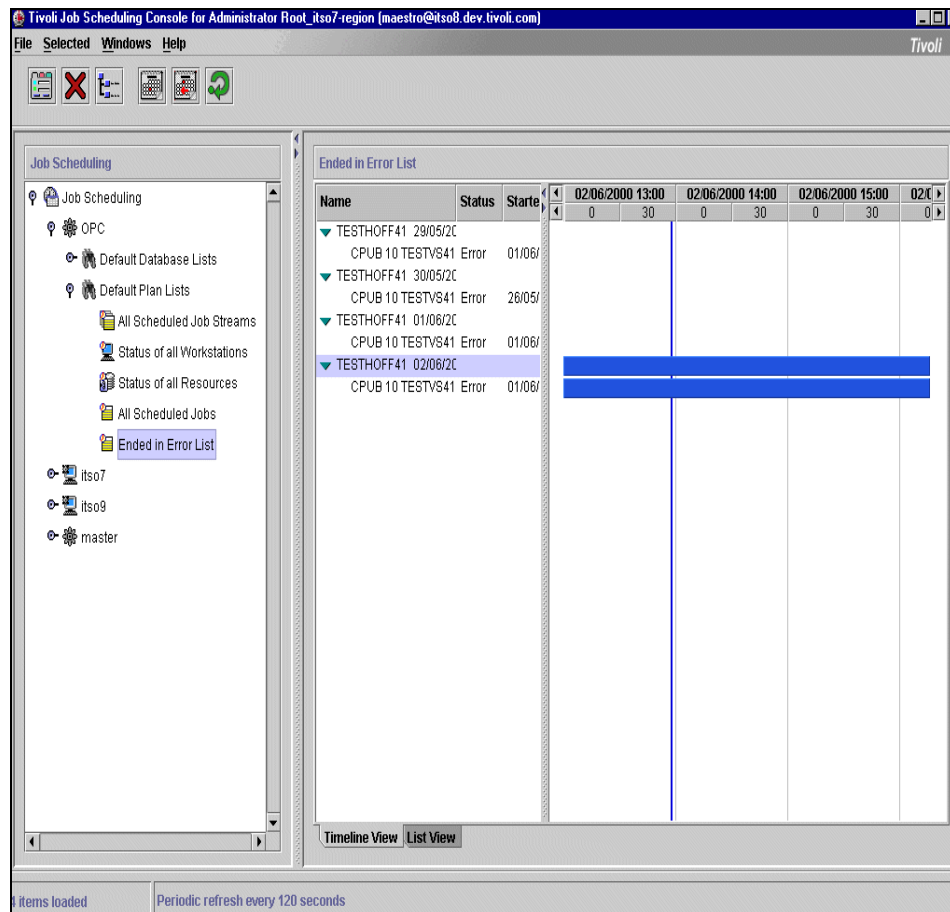


Figure 69. Listing errored jobs

Use the same process to make the Ready Lists. Select the relevant statuses and workstation as shown in the Figure 70 on page 94.

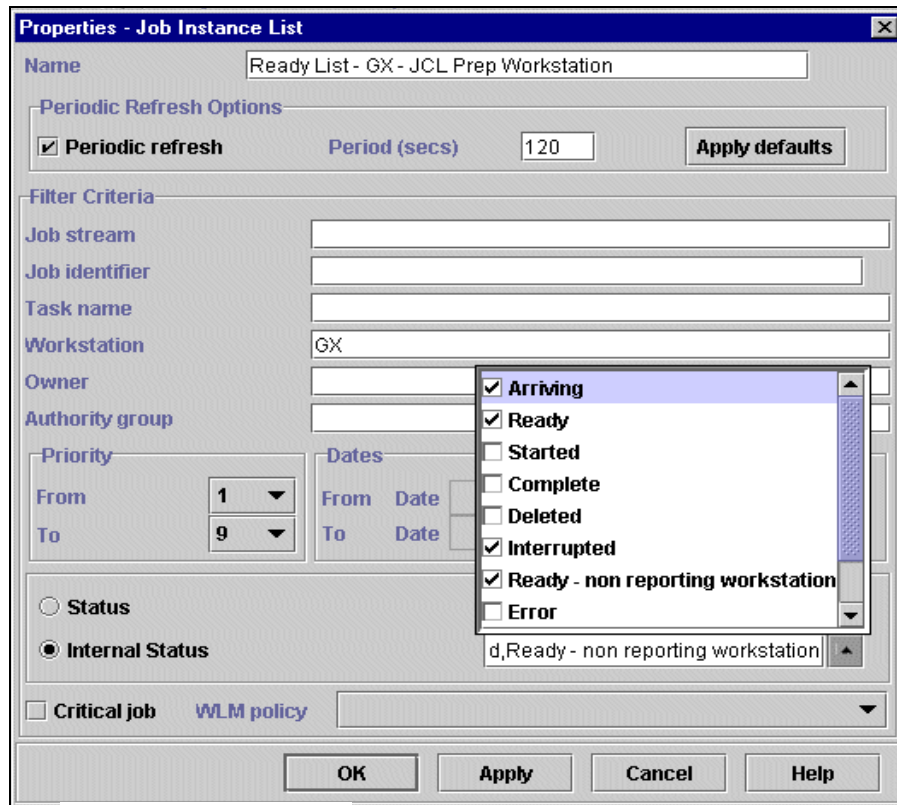


Figure 70. Making a Ready List for a JCL prep workstation

The JCL prep workstation identifier is GX, and the statuses selected are Arriving, Ready, and Interrupted. Once it is saved, when you click on the list in the Job Scheduling pane, you see the same jobs you would see on the OPC Ready List for that workstation as shown in the Figure 71 on page 95.

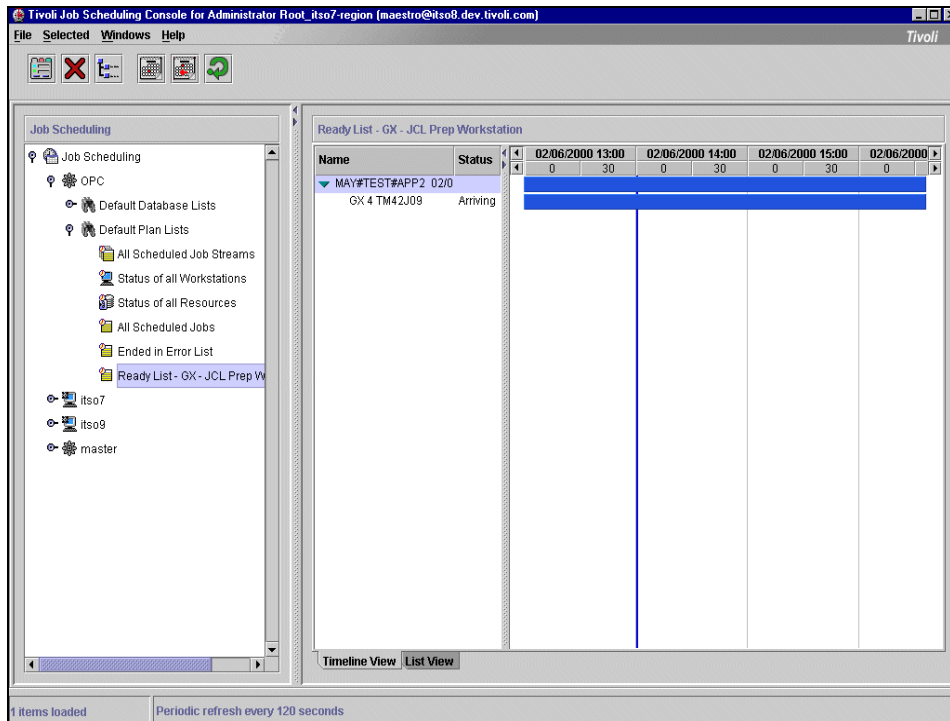


Figure 71. Ready List for a JCL Prep Workstation

To change the status, use the right mouse button and click on the operation. You will see a window like the one shown in Figure 72 on page 96.

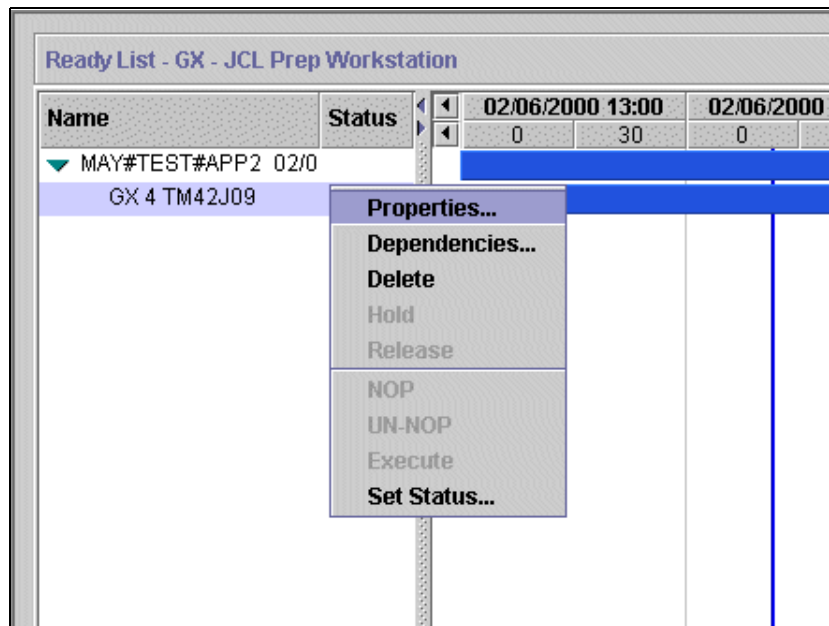


Figure 72. Job properties

Select **Set Status**, and you will see the window shown in Figure 73 on page 97.

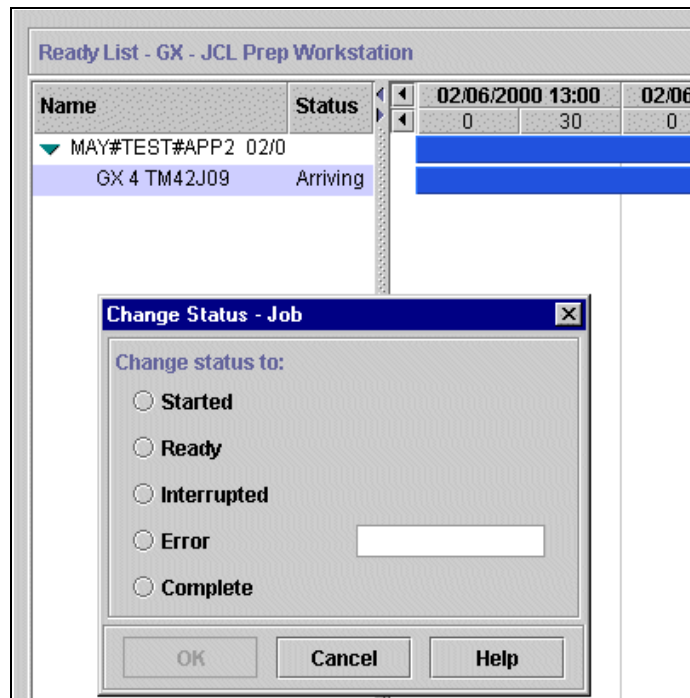


Figure 73. Changing job status

You can change the status to C and, thus, release successor jobs, but you cannot edit JCL from the JSC.

#### 4.2.4.2 Creating a Job Stream (Application)

Using the JSC to create a new Job Stream in the OPC Database should cause no problems for anyone who can create an application using the OPC ISPF dialog panels. The same information is required, but the look and position of the fields are different. The JSC offers benefits of a graphical interface, while the text-based ISPF dialogs may be faster for people who are familiar with them. However, the JSC does have two major facilities that are not available with the ISPF dialogs.

- You can view all planned occurrences of a job stream with multiple run-cycles. Whereas the ISPF dialog command, `GENDAYS`, calculates only the effect of one rules-based run cycle, the JSC shows the combined effect of all run cycles, both rules- and period/offset-based.
- You can create the jobs (operations) and make internal and external dependencies using a graphical view.

To create a job stream, right-click on the icon for the OPC engine in the left pane, and then select **New Job Stream** as shown in Figure 74.

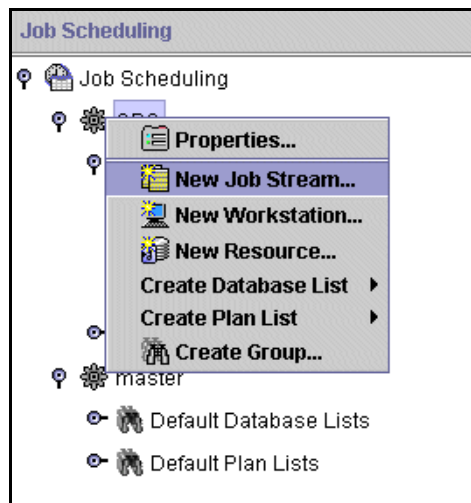


Figure 74. Creating a new Job Stream

Selecting New Job Stream produces two windows as shown in the Figure 75 on page 99.



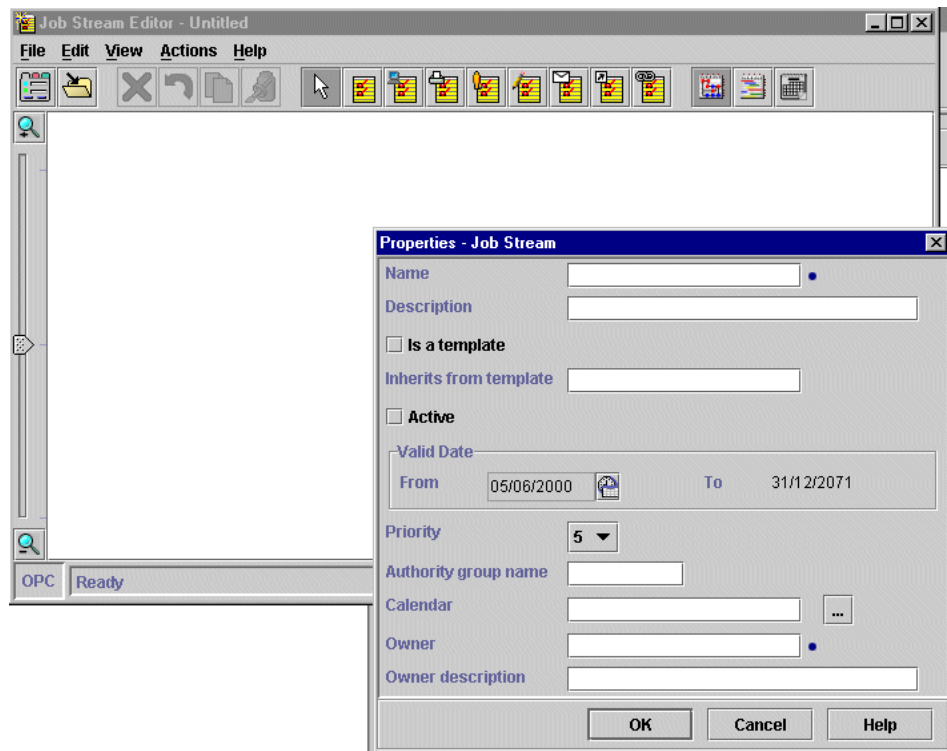


Figure 75. Job stream properties

Figure 75 shows the Properties window, which is the equivalent of the application administration panel. The underlying window is the graphical operation list. The dots to the right of the input fields mean that responses are required. When the Properties window is complete, click **OK**, and the Job Scheduling Editor window, shown in Figure 76 on page 100, will become active.

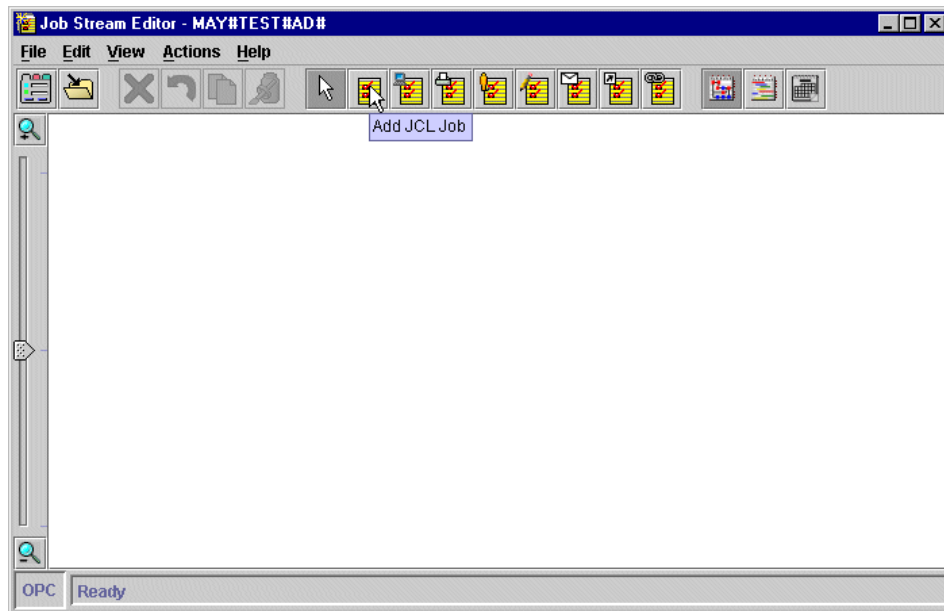


Figure 76. Defining jobs

Icons at the top represent different workstations types. To create a job (operation), click the appropriate icon. When the mouse pointer is moved inside the window, it becomes cross-shaped. Click where you want the job to be displayed. The Job (operation) details screen is shown in Figure 77 on page 101.

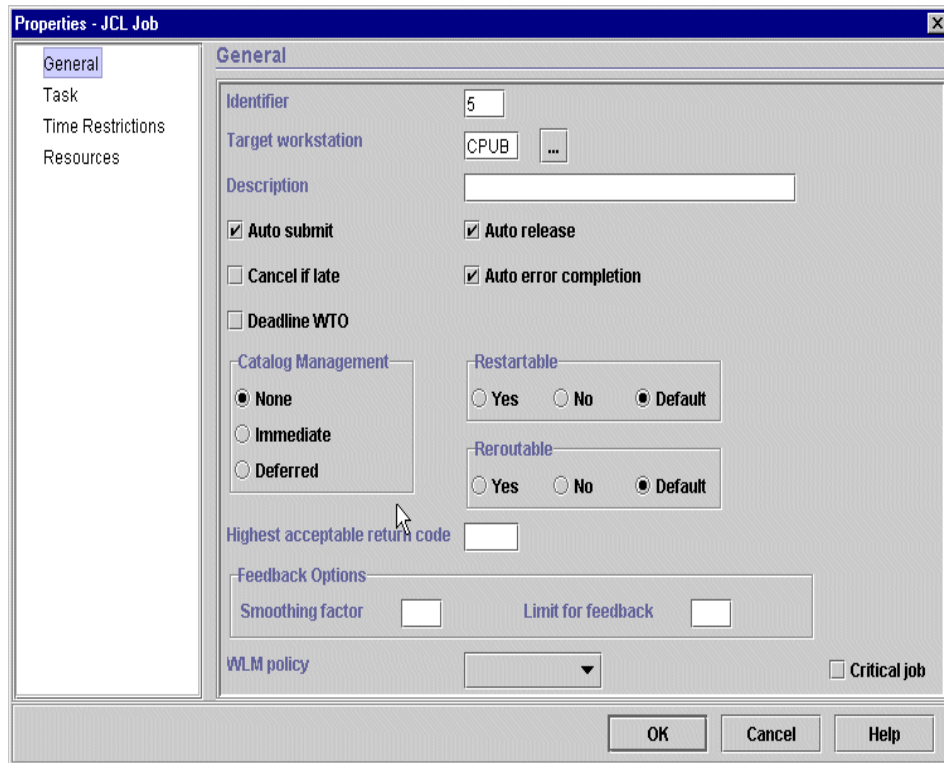


Figure 77. Job (operation) details

Most of the operation details can be entered in the window shown in Figure 77, except for what is perhaps the single most important one. To enter the job name, you must click on **Task** in the menu on the left.

#### Note

If you do not set an operation start time in Time Restrictions, the time is defaulted to 00:00 on day 0 when the job stream is saved. This will give you a warning icon in the time line view indicating time inconsistencies. To prevent this, after saving the job stream, you should immediately update the job stream and tick the **Follow Job Stream Rules** box, which will remove the 00:00 setting.

Once you save this new operation, an icon appears in the frame as shown in Figure 78 on page 102.

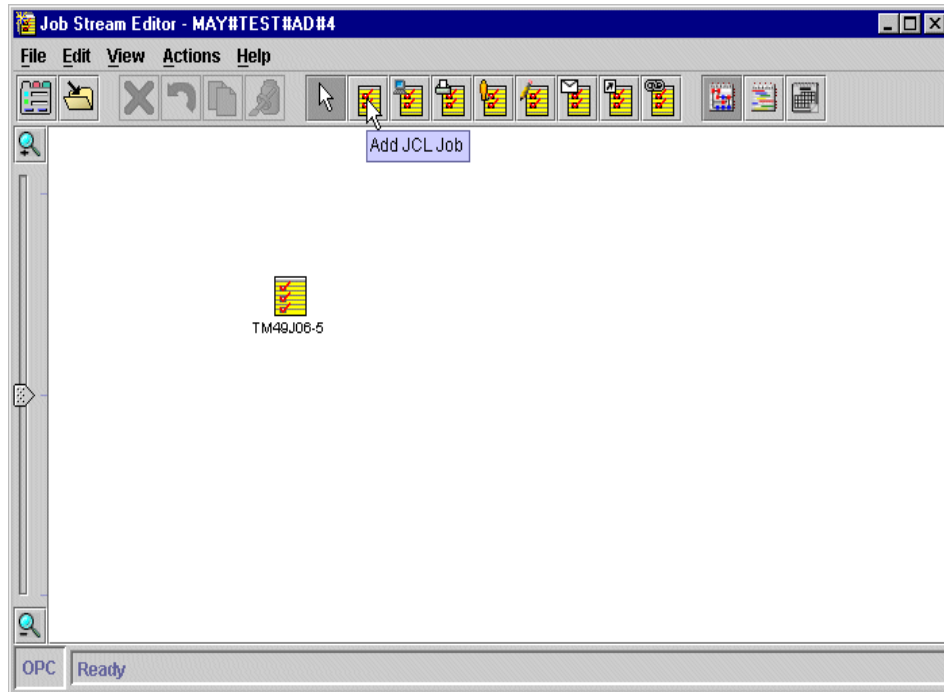


Figure 78. Job (operation) icon

More operations may be created in the same way. When there is more than one job in a job stream (operation in an application), all must be linked together with dependencies. This is easily done by clicking on the appropriate icon on the menu bar as shown in the Figure 79 on page 103.

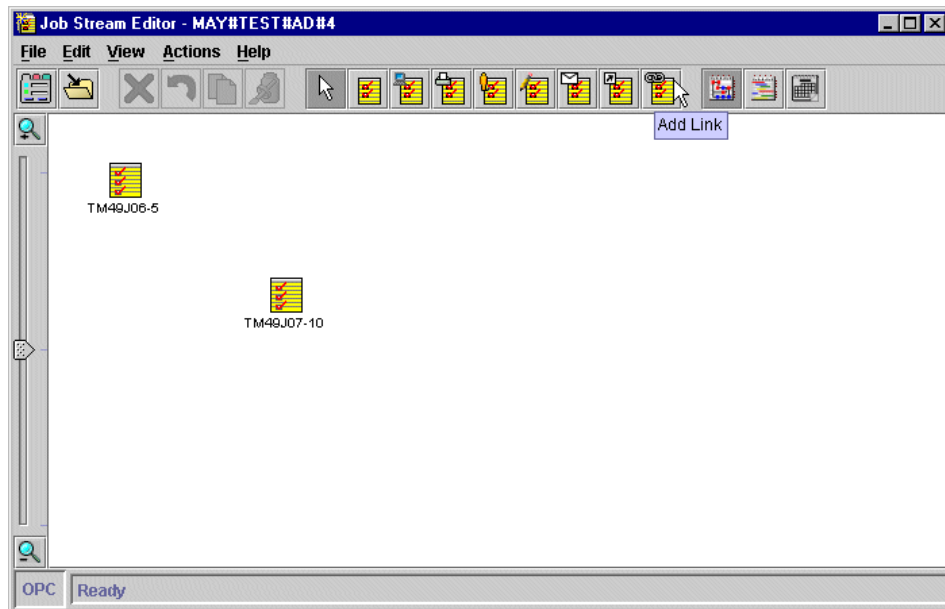


Figure 79. Defining internal dependencies

There is a clear difference when defining dependencies. In OPC, operations point to their predecessors. In the JSC, you must first click on the predecessor and keep the mouse button depressed while you drag an arrow to the successor as shown in Figure 80 on page 104.

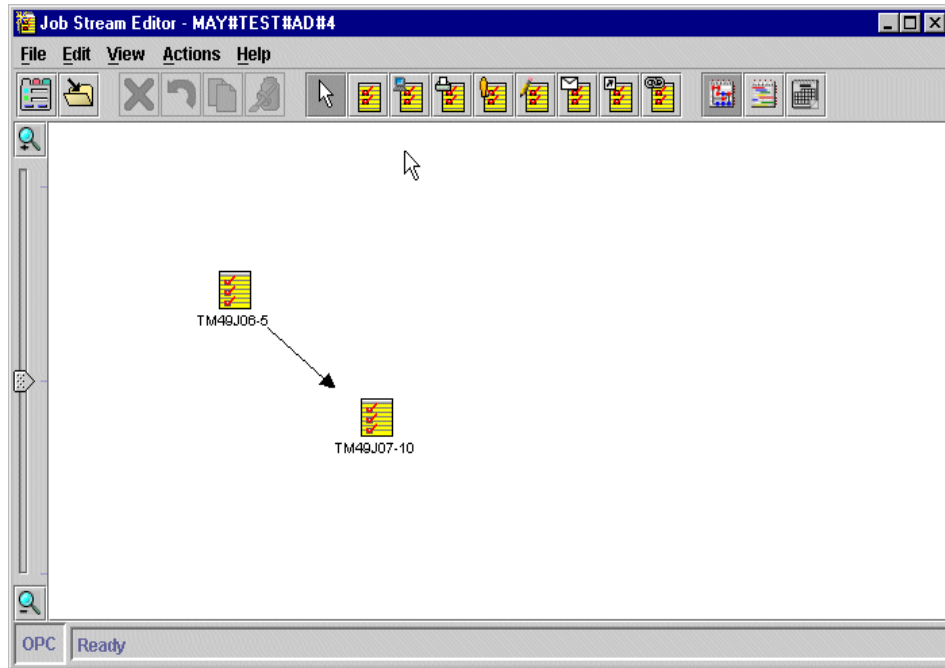


Figure 80. Dependency arrow

The dependency arrow points from predecessor to successor. External jobs can be placed in the window by clicking on the Actions menu, and external dependency links made by clicking on the appropriate icon.

To schedule the job stream, click on the **Rule-based run cycle** icon at the right of the menu. A scheduling window is displayed instead of a job window as shown in Figure 81 on page 105.

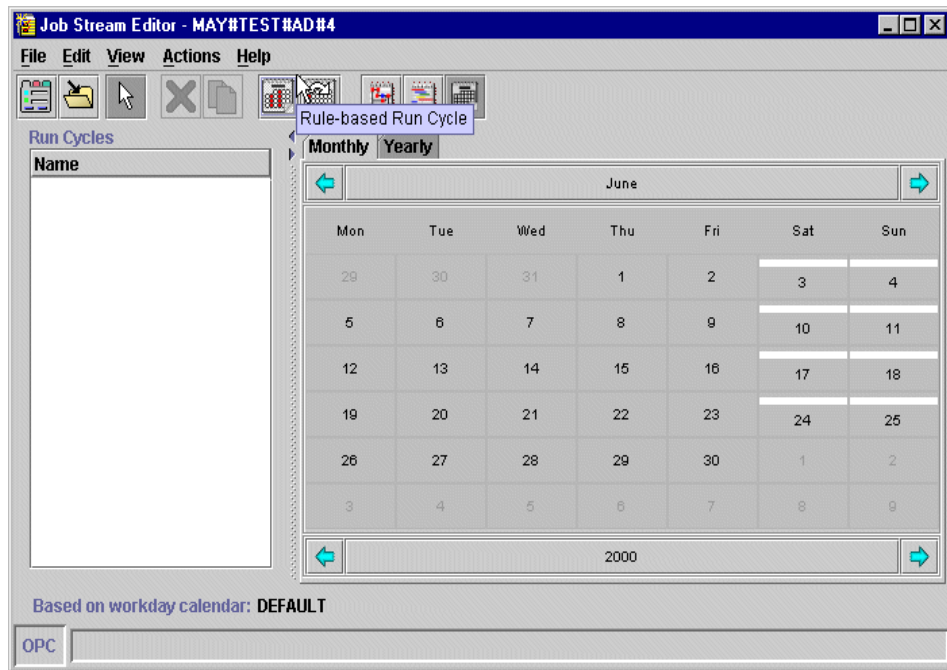


Figure 81. Scheduling the job stream

Use the appropriate icon to select a rules- or period/offset-based run cycle. For offset-based run cycles, you can choose the period you want from a list and specify the offsets. A major advantage of the JSC is that you can see the calculated scheduling days.

For rules scheduling, the only difference from OPC ISPF dialog panels is that the ISPF dialog only requires *Only* or *Every* to be selected as shown in Figure 82 on page 106.

```

----- MODIFYING A RULE -----
Command ==>

Enter the GENDAYS command to display the dates generated by this rule
Enter S and user data in the fields below to define a rule

Application   : MAY#TEST#APP1           Peter Mays first JSC AD
Rule          : SOMETHN2

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

```

Figure 82. Using rules ISPF dialog panel

To schedule the application to run Every Wednesday in the Year, only one selection needs to be made in the Frequency section. The JSC requires that, in addition to Only or Every, a frequency must be selected as shown in Figure 83 on page 107.



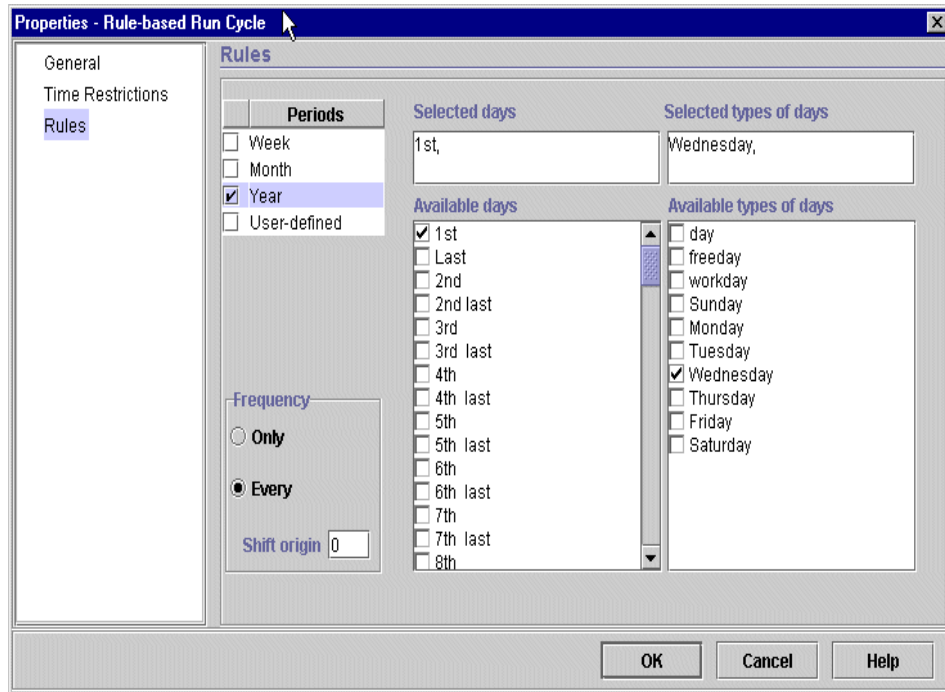


Figure 83. Using rules JSC dialog

The rule cannot be saved and the OK button remains greyed out until at least one selection is ticked in each of the Periods, Available days, and Available types of days. To get the same result as in the ISPF dialog, choose **1st**.

The JSC will display all calculated run days on either a monthly calendar, as shown in Figure 84 on page 108, or an annual calendar.

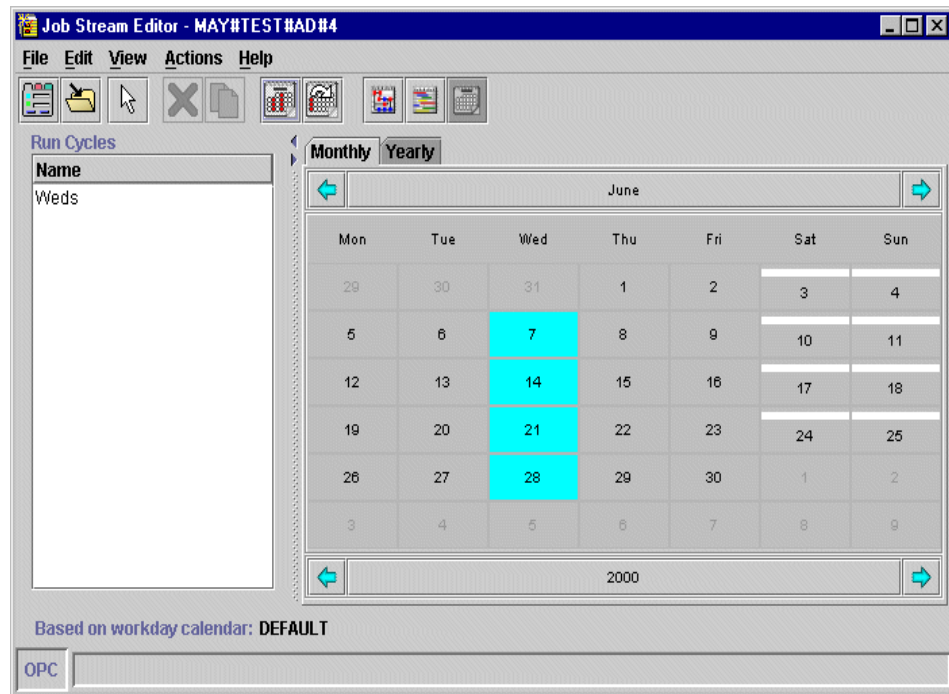
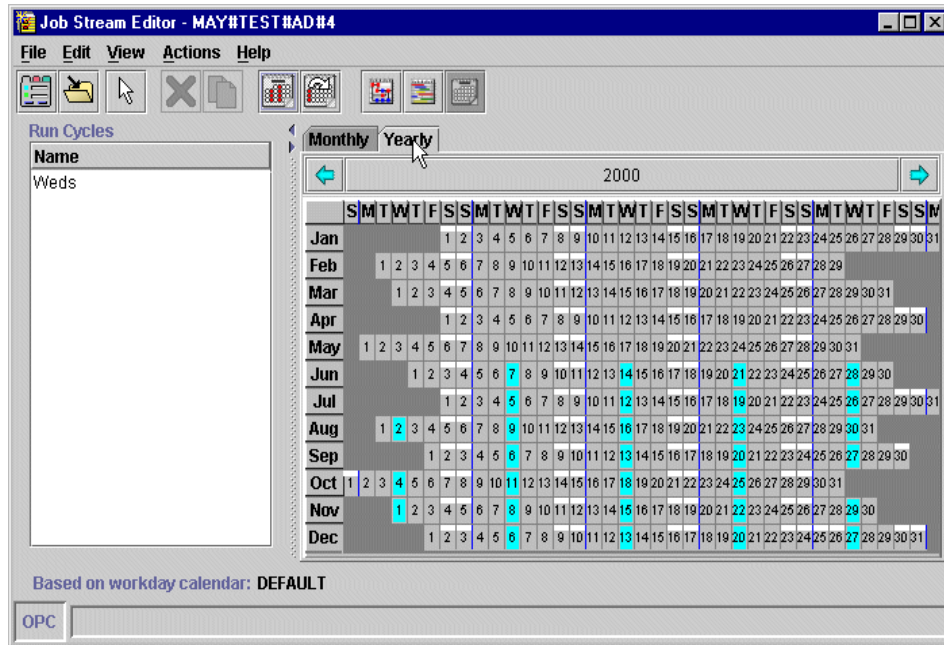


Figure 84. Monthly display of calculated run days

The run days are shown colored pale blue. A yearly calendar can be selected by clicking on the tab at the top and is shown in Figure 85 on page 109.



If there are multiple run cycles, the display will show all of the calculated run days, and, by clicking on the run cycle name on the left, the days it has scheduled are highlighted with a dark blue bar as shown in Figure 86 on page 110.

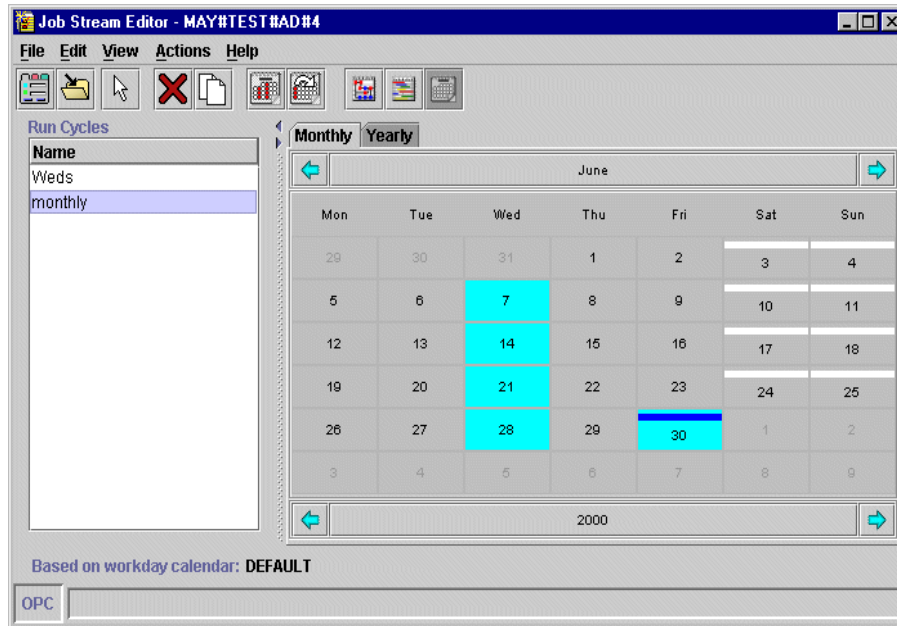


Figure 86. Combined display of calculated run days

This display is a very useful resource for the application builder and the scheduler, but it must always be remembered that this is a calculation of run days. Actual scheduling is done by the Long Term Plan batch job.

### 4.3 Job Scheduling Console for TWS

There are actually three different interfaces available for TWS: The command line interface, the legacy GUI, and the new Job Scheduling Console. Each of these has its own advantages and disadvantages.

The legacy GUI is no longer supported by Tivoli, but it is fully-functional, at least in TWS Version 7.0. Although the legacy GUI is faster than JSC, it lacks some features of JSC, such as time zones and the ability to manage OPC. From the legacy GUI, you cannot modify a workstation or a schedule that had time zone support added from the command line interface or from the JSC.

Figure 87 on page 111 shows various screen captures from the legacy TWS GUI.

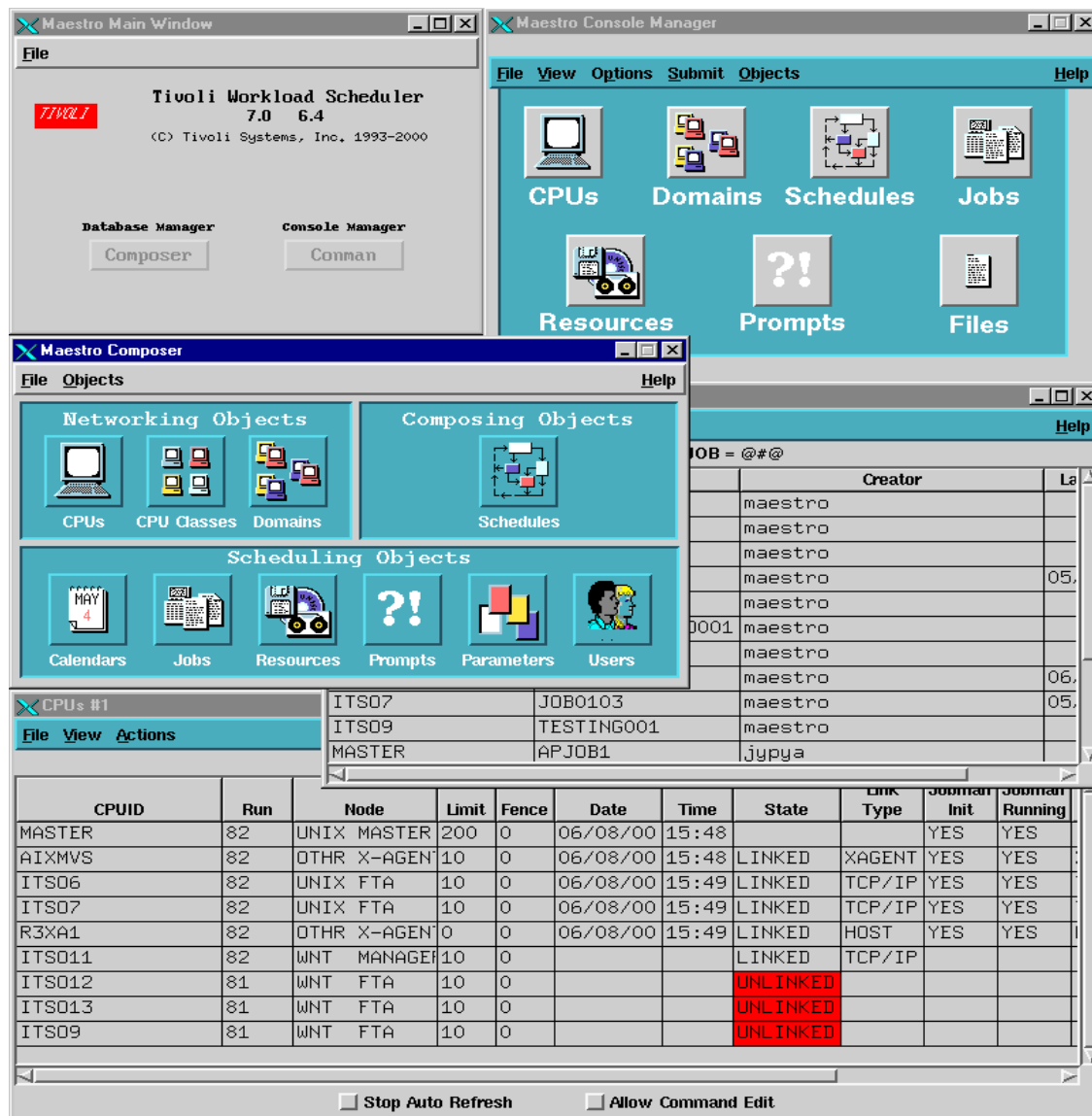


Figure 87. Legacy GUI

The command line interface (CLI) is used for certain advanced features. Some of the capabilities of the CLI are not available in the Job Scheduling Console. The Job Scheduling Console and CLI are independent and can be run simultaneously to manipulate TWS data. Sometimes, the CLI is simply faster, and you do not need any graphical displays to use it. The

disadvantages are complicated commands and output that is not always clear. Experienced users may prefer the CLI for some tasks.

```
$ conman
MAESTRO for UNIX (AIX)/CONMAN 7.0 (1.4) (C) Tivoli Systems Inc. 1998
Installed for group 'DEFAULT'.
Locale LANG set to "en_US"
Schedule (Exp) 06/08/00 (#82) on MASTER. Batchman LIVES. Limit: 200, Fence: 0
, Audit Level: 1
%sc
  CPUID      RUN  NODE      LIMIT FENCE    DATE    TIME    STATE  METHOD  DOMAIN
MASTER      82  *UNIX MASTER  200    0  06/08/00 15:48  I J           MASTERDM
AIXMVS      82  OTHR X-AGENT  10     0  06/08/00 15:48 LXI JX mvsopc MASTERDM
ITSO6       82  UNIX FTA     10     0  06/08/00 15:49 LTI JW           MASTERDM
ITSO7       82  UNIX FTA     10     0  06/08/00 15:49 LTI JW           MASTERDM
R3XA1      82  OTHR X-AGENT  0      0  06/08/00 15:49 LHI JH r3batch MASTERDM
ITSO11      82  WNT  MANAGER  10     0                      LT           NTDOMAIN
%
%sbdb master#"test";alias="testi23"
Submitted MASTER#test to batchman as MASTER#JOBS.TESTI23
%sj master#jobs.@
                                (Est) (Est)
CPU      Schedule  Job      State Pr Start  Elapse  Dependencies
MASTER  #JOBS      ***** ABEND 10 15:57 00:01
                                TESTI23 ABEND 10 15:57 00:01 #J23608
%
```

JSC is the most recent GUI for TWS. It is multi-platform Java-based application integrated with the Tivoli Framework, and, because of Java, you need a fast machine to run JSC or to get good performance. On the other hand, JSC offers a very nice interface for using TWS and OPC.

Figure 88 on page 113 shows several screen captures from the JSC GUI.

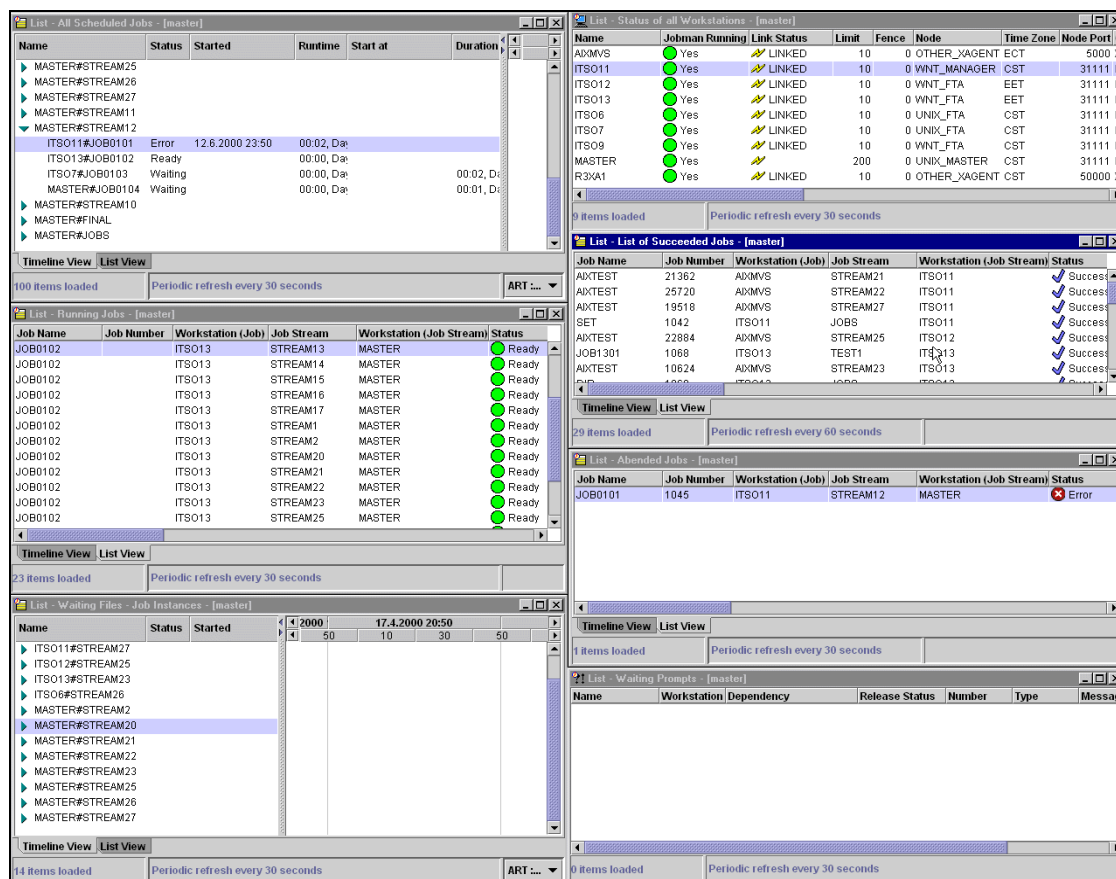


Figure 88. JSC and customized views

### 4.3.1 What can you do with the JSC

From the Job Scheduling Console, you are able to access the scheduling functions for TWS and OPC from a common interface. The left side of the console displays the job scheduling tree. The servers, TWS or OPC, are in this view, and there are groups of default lists for the database and the plan under them.

#### In database views, you can:

- Create, modify, copy, and delete:
  - Database groups
  - List of customized views

- Workstations
- Workstation Classes
- Domains
- Job Streams
- Job Definitions
- Resources
- Prompts
- Parameters
- Users
- Calendars

**In plan views, you can:**

- Create, delete, and modify groups and list of customized views
- Monitor status of:
  - Workstations
  - Job streams
  - Jobs
  - Resources
  - Files
  - Prompts
  - Domains
- Submit:
  - Jobs
  - Job Stream
  - Ad-hoc jobs
- Confirm jobs
- Change jobs to be confirmed
- Release dependencies of job streams
- View jobs output
- Cancel jobs
- Kill running jobs
- Rerun jobs



- Modify job stream's:
  - Priority
  - Limit of job stream
  - Starting time
  - Deadline time
  - Carry forward
  - File dependencies
  - Resource dependencies
  - Prompt dependencies
  - Time zone
- Modify job instance's:
  - Priority
  - Starting time
  - Deadline time
  - Repeat range
  - File dependencies
  - Resource dependencies
  - Prompt dependencies
- Start and stop workstations
- Link and unlink workstations
- Change fence of workstations
- Change limit of workstations
- Change units of resources
- Switch domain managers
- Reply prompts

JS Console and TWS Connectors will now restrict the queries to only what the user is allowed to see via the TWS Security file with the DISPLAY access keyword. With legacy GUI lists, show jobs and show schedules showed all objects (of workstations, schedules, and jobs) regardless of the access keywords.

### 4.3.2 What you should do using TWS legacy user interfaces

The following tasks should be performed the TWS legacy GUI or from the CLI:

- Monitor log files in real-time. You can monitor the main log file in the TWS Master in CLI using the following command:  
`console sess;level=4`  
or, from the UNIX command shell:  
`tail -f $TWSHOME/stdlist/[year].[month].[date]/MAESTRO`
- View history. In legacy GUI and CLI, the command is `Set Symphony`.
- View the job streams coded in scripts using the internal TWS scheduling commands. Sometimes, it is useful to see how a job stream is created with one quick look. Looking at pure code could be more efficient than navigating the GUI interfaces. Use the CLI for this.
- Modify TWS security. The only way to do this is through the command shell with the commands, `dumpsec` and `makesec`.
- Use wildcards to submit jobs and job streams.
- Print reports. This is done through the command shell. There are several commands for generating reports.
- Arrange objects in plan or database lists in ascending or descending order.
- Execute console commands directly.
- Save a job stream that needs resources from other workstations.

### 4.3.3 Job status mapping

Table 7 describes how the Job Scheduling console status correlates to the TWS internal status for jobs.

Table 7. Job status mapping table

JS console status	TWS internal status
WAITING	ADD, PEND, WAIT, WAITD, INTRO, HOLD
READY	READY
RUNNING	EXEC, SUCCP, ABENP
SUCCESSFUL	SUCC
ERROR	ABEND, FAIL

JS console status	TWS internal status
CANCELED	Status of the job when it was canceled. Canceled flag is set.
HELD	Priority = 0, WAITING, READY
UNDECIDED	ERROR, EXTRN
BLOCKED	SUSP

#### 4.3.4 Customized lists

By default, JSC creates lists where you can see all job streams, all jobs, and the status of all prompts, workstations, resources, files, and domains. You would probably want to see more precisely what is happening. Here are some useful examples of customized lists:

- Abended jobs
- Running jobs
- Prompts waiting reply

##### 4.3.4.1 Abended jobs

With this list, you can see only jobs that stopped because of an error in a workstation. Perform the following steps:

1. Select **Create Plan List->Job Instances** from JSC as shown in Figure 89 on page 118.

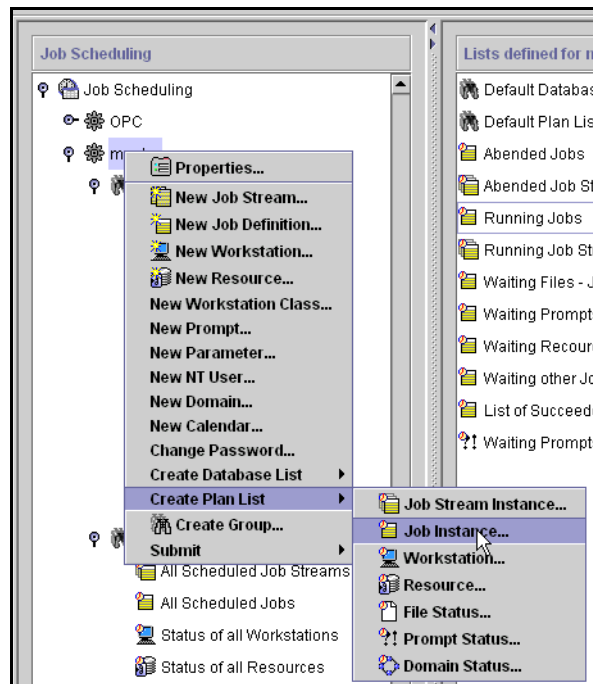


Figure 89. Create Plan List

2. Give the name, Abended jobs, as shown in Figure 90 on page 119.
3. Change the refresh to **Periodic refresh** and seconds to **30 sec**, which is the lowest you can set.
4. Set the status to **Error**.
5. Leave the other selections at their default settings.

**Properties - Job Instance List**

**General**

Name: Abended Jobs

**Periodic Refresh Options**

☒ Periodic refresh    Period (secs): 30    **Apply defaults**

**Filter Criteria**

Job Name: \*

Job Stream: \*

Workstation (Job Stream): \*

Login:

**Status** ☒    Error

**Internal Status** ☐

**Recovery Options**

☐ Stop    ☐ Continue    ☐ Rerun    ☒ None

**Priority**

From: 0    Hold    High    Go

To: 101    Hold    High    Go

**OK**    **Apply**    **Reset**    **Cancel**    **Help**

Figure 90. Abended jobs list properties (general page)

#### 4.3.4.2 Running jobs

With this customized list, you will see only currently running jobs. Perform the following steps:

1. Select **Create Plan List->Job Instances** from JSC as shown in Figure 89 on page 118.
2. Enter the name, *Running jobs*, as shown in Figure 91 on page 120.
3. Set the status to **Ready** and **Running**.
4. Leave the other selections at their default settings.

**Properties - Job Instance List**

**General**

Name: Running Jobs

**Periodic Refresh Options**

☒ Periodic refresh    Period (secs): 30    **Apply defaults**

**Filter Criteria**

Job Name: \*

Job Stream: \*

Workstation (Job Stream): \*

Login:

**Status** ☒    Ready,Running

**Internal Status** ☐

**Recovery Options**

☐ Stop    ☐ Continue    ☐ Rerun    ☒ None

**Priority**

From: 0    **Hold High Go**

To: 101    **Hold High Go**

**OK**    **Apply**    **Reset**    **Cancel**    **Help**

Figure 91. Running jobs list properties (general page)

#### 4.3.4.3 Prompts waiting reply

With this customized list, you will only see prompts that are waiting for a reply.

1. Select **Create Plan List->Prompt Status** from JSC as shown in Figure 92 on page 121.

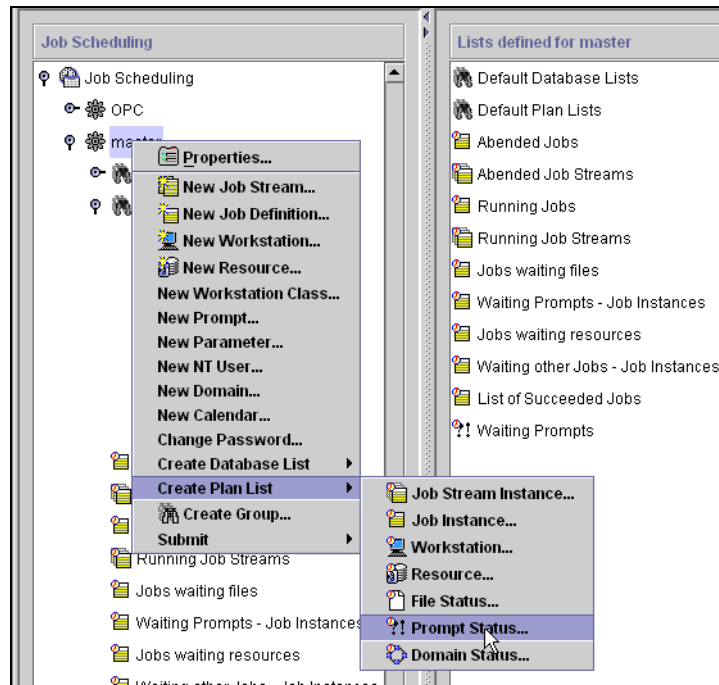


Figure 92. Create New Prompt Status list

2. Enter the name, Prompts waiting reply, as shown in Figure 93 on page 122.
3. Set the periodic refresh to **30** seconds.
4. Set the Prompt Name as an asterisk (\*).
5. Set the status to **Asked**.

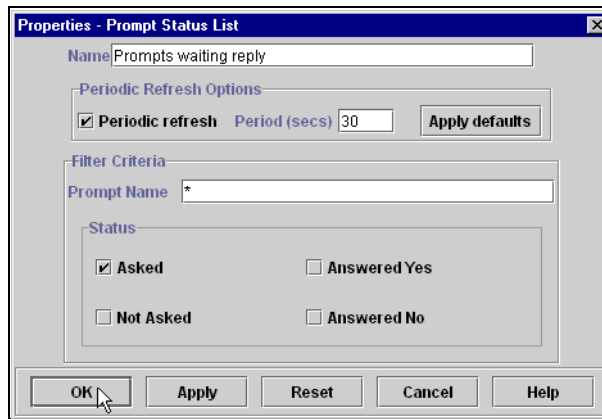


Figure 93. Prompt Status List properties

## 4.4 Summary

In this chapter, we discussed the Job Scheduling Console Graphical User Interface (JSC GUI), which is the standard interface for Tivoli Workload Scheduler (TWS) and Operations Planning and Control (OPC). It is possible to manage work running under TWS and OPC from the same GUI; so, customers who are using OPC and TWS together as an end-to-end scheduling solution will find JSC especially handy since operators need to be educated only on one user interface.

We also covered functions that must be performed exclusively from either the OPC ISPF interface, the TWS legacy GUI or the CLI. It is expected that new releases of the JSC will include some or all of these functions.



---

## Chapter 5. Troubleshooting for Tivoli OPC

This chapter is intended to help you with the most common problems encountered in Tivoli OPC. We will show you how to identify the distinguishing features that will help you obtain a solution. The answer might then be found in manuals, but, often, it is not possible to get a solution or circumvention without involving the Tivoli support structure. However, it can be helpful for the first analysis or for providing the right documentation when you are facing a problem. A good guideline for this chapter is the Tivoli OPC book, *Diagnosis Guide and Reference*, LY19-6406.

To identify an error, you must first gather information related to the problem, such as abend codes and dumps. You can then determine whether the problem is in Tivoli OPC. If the problem is in Tivoli OPC, this chapter helps you classify and describe the problem. The external symptoms of several problems are described to help you identify which problem type to investigate. Each problem type requires a different procedure when you describe the problem. Use these procedures to build a string of keywords and to obtain documentation relevant to the problem. This combination of a keyword string and associated documentation helps you describe the problem accurately to the Tivoli service personnel.

---

### 5.1 Using keywords to describe a problem

A *keyword* is a word or abbreviation that describes a single aspect of a program failure to the IBM Support Center. You use keywords to describe all aspects of a problem, from the Tivoli OPC component ID to the area of failure. You then use the problem analysis procedures to build a keyword string. For example, if your program failure is due to the abnormal termination of a task, the keyword is ABEND. Other keywords are also formed to describe particular aspects of the abnormal termination, such as the name of the module where the abend occurred. These keywords are then combined to form a keyword string.

Let us look at the following example: 5697OPC01 ABEND0C4 EQQYVARG

In this example, 5697-OPC01 is the Tivoli OPC component ID; ABEND is the problem type, and 0C4 is the abend code. EQQYVARG is the module containing the abend.

---

## 5.2 Searching the software-support database

To determine if the problem has been noted before, you can use the keyword string that you create to search the software-support database. If a problem similar to yours is described in the database, a solution is probably available. To widen or narrow the database search, you can vary the keyword string you develop. If you have access to the Tivoli support database, you can use the keyword string to search for solutions of problems similar to yours. Link to the Tivoli support database at the following URL:

[www.tivoli.com/support/](http://www.tivoli.com/support/)

---

## 5.3 Problem-type keywords

The problem-type keywords are used to identify the failure that occurred. Table 8 lists the keywords and the problem types they identify.

*Table 8. Keywords*

Keywords	Meaning
Abend	Abnormal end
Abendu	Abnormal end with user abend code
DOC	Documentation
Loop	loop
Wait	wait
MSG	message
Perf	Performance
INCORROUT	Incorrect output

### **ABEND**

Choose the ABEND keyword when the Tivoli OPC program comes to an abnormal end with a system abend code. You should also use ABEND when any program that services Tivoli OPC (for example, VTAM) terminates it, and one of the following symptoms appears:

- An abend message at an operator console. The abend message contains the abend code and is found in the system console log.
- A dump is created in a dump dataset.

### **ABENDU**

Choose the **ABENDU** keyword when the Tivoli OPC program comes to an abnormal end with a user abend code and the explanation of the abend code states that it is a program error. Also, choose this keyword when a user abend (which is not supposed to signify a program error) occurs when it should not occur, according to the explanation. If a message was issued, use the **MSG** keyword to document it.

### ***DOC***

Choose the **DOC** keyword when one or more of the following symptoms appears:

- There is incomplete or inaccurate information in a Tivoli OPC publication.
- The published description of Tivoli OPC does not agree with its actual operation.

### ***INCORROUT***

Choose the **INCORROUT** keyword when one or more of these symptoms appears:

- You received unexpected output, and the problem does not appear to be a loop.
- The output appears to be incorrect or incomplete.
- The output is formatted incorrectly.
- The output comes from damaged files or from files that are not set up or updated correctly.

### ***LOOP***

Choose the **LOOP** keyword when one or more of the following symptoms exists:

- Part of the program (other than a message) is repeating itself.
- A Tivoli OPC command has not completed after an expected period of time, and the processor usage is at higher-than-normal levels.
- The processor is used at higher-than-normal levels, a workstation operator experiences terminal lockout, or there is a high channel activity to a Tivoli OPC database.

### ***MSG***

Choose the **MSG** keyword to specify a message failure. Use this keyword when a Tivoli OPC problem causes a Tivoli OPC error message. The

message might appear at the system console or in the Tivoli OPC message log, or both. The messages issued by Tivoli OPC appear in the following formats:

- **EQQ** FnnnC
- **EQQ** FFnnC
- **EQQ** nnnnC

The message is followed by the message text. The variable components represent:

- F or FF - This is the Tivoli OPC component that issued the message
- nn, nnn, or nnnn - This is the message number
- C - Severity code of **I** (information), **W** (warning), or **E** (error)

The following is message number examples:

- EQQN008E
- EQQWI10W
- EQQF008I

If the message that is associated with your problem does not have the **EQQ** prefix, your problem is probably not associated with Tivoli OPC, and you should not use the **MSG** keyword.

### ***PERFM***

Choose the ***PERFM*** keyword when one or more of the following symptoms appears:

- Tivoli OPC event processing or commands, including commands entered from a terminal in session with Tivoli OPC, take an excessive amount of time to complete.
- Tivoli OPC performance characteristics do not meet explicitly stated expectations. Describe the actual and expected performances and the explicit source of the performance expectation.

### ***WAIT***

Choose the ***WAIT*** keyword when one or more of the following symptoms appears:

- The Tivoli OPC program, or any program that services this program, has suspended activity while waiting for a condition to be satisfied without issuing a message to indicate why it is waiting.

- The console operator cannot enter commands or otherwise communicate with Tivoli OPC, and Tivoli OPC does not appear to be in a loop.

---

## 5.4 Problem analysis procedures

This section details the procedures that you use to further describe a problem. First, you gather the information for the specific problem type. When you have chosen a problem-type keyword, you need to collect problem documentation and create a keyword string to describe the problem. To do this, gather the information for the specific problem.

- System or user abnormal-termination procedure (ABEND or ABENDU)
- Documentation procedure (DOC)
- Incorrect output procedure (INCORROUT)
- Loop procedure (LOOP)
- Message procedure (MSG)
- Performance procedure (PERFM)
- Wait procedure (WAIT).

### 5.4.1 Abnormal termination (ABEND or ABENDU) procedure

A malfunction in the system can cause an abnormal termination (abend). Abend categories are:

- User abend
- System abend

User abends originate in the application program. Tivoli OPC abend codes are documented in Appendix A, “Abend Codes”, of the *Tivoli Operations Planning and Control V2R3 Diagnosis Guide and Reference*, LY19-6405, and also in *TME 10 OPC Messages and Codes*, SH19-4480. A common user abend in Tivoli OPC is 3999.

#### User Abend 3999

**Explanation:** Tivoli OPC internal validity checking has discovered an error condition (internal Tivoli OPC error). A message that contains the reason for the abend, as well as other debugging information, is written to the Tivoli OPC diagnostic file, EQQDUMP.

**Problem determination:** None.

**System programmer response:** Call your IBM representative.

You may find the occurrence of a user abend in the MLOG indicated by a message, such as *EQQF004E "Data Router task abended while processing the following queue element"* and in the system log:

```
IEA995I SYMPTOM DUMP OUTPUT
  USER COMPLETION CODE=3999
  TIME=15.46.40  SEQ=00456  CPU=0000  ASID=0031
  PSW AT TIME OF ERROR  078D1000   800618CE  ILC 2  INTC 0D
  ACTIVE LOAD MODULE      ADDRESS=00054DF8  OFFSET=0000CAD6
  NAME=EQQBEX
  DATA AT PSW  000618C8 - 00181610  0A0D1812  0A0D47F0
  GPR  0-3  80000000  80000F9F  00000F9F  000844E8
  GPR  4-7  C5D8D8C2  C4C54040  C5E7C9E3  40404040
  GPR  8-11 00000000  00000001  00000F9F  00061728
  GPR 12-15 00000000  001DA4C0  800579D2  00000000
  END OF SYMPTOM DUMP
```

In addition, OPC writes diagnostic informations in its EQQDUMP dataset:

*EQQ0000T MODULE: EQQDXQPR, REASON: INVDEST*

System abends can occur, for example, when a program instruction refers to a storage area that does not exist anymore.

#### 5.4.2 The Diagnostic File (EQQDUMP)

When Tivoli OPC internal validity checking discovers error conditions within the network communication function, debugging information is written to the Tivoli OPC diagnostic file (defined by ddname EQQDUMP). For serious error conditions, Tivoli OPC abends with user code 3999 as well. The diagnostic information consists of message EQQ0000T, which gives the name of the module in error and the reason for the error in two 8-byte character strings. Tivoli OPC also writes a formatted version of the trace table to the diagnostic

file. In most situations, Tivoli OPC will also *snap* the data that it considers to be in error.

### Trace Information

Tivoli OPC maintains an internal trace to make it possible to see the order that its modules have been invoked in prior to an abend. The trace is wraparound with an end mark after the last trace entry added. Each entry consists of two 8-byte character fields: The module name field and the reason field. The end mark consists of a string of 16 asterisks (X'5C'). For most abnormal terminations, a trace table is written in the diagnostic file (EQQDUMP). These trace entries are intended to be used by IBM staff when they are diagnosing Tivoli OPC problems. A trace entry with the reason PROLOG is added upon entry to the module. Similarly, an entry with EPILOG is added at the exit from the module. When trace entries are added for other reasons, the reason is provided in the reason field.

#### 5.4.3 System dump dataset

An abnormal end (abend) of major tasks may affect the entire OPC PLEX and can jeopardize the whole production. The recovery and terminating manager of the operating system produces valuable information for diagnostic purposes. Therefore, it is extremely important to make sure that this information is kept in a dataset, called the system dump dataset, for further analysis.

The sample JCL procedure for an OPC address space includes a SYSMDUMP DD statement, and a dump dataset is allocated by the EQQPCS02 JCL created by EQQJOBS. SYSMDUMP is the dump format preferred by the service organization. Ensure that the dump options for SYSMDUMP include RGN, LSQA, TRT, CSA, and GRSQ on systems where an OPC/ESA address space will execute. To display the current SYSMDUMP options, issue the MVS/ESA command, `DISPLAY DUMP,OPTIONS`. You can use the `CHNGDUMP` command to alter the SYSMDUMP options. Note that this will only change the parameters until the next IPL. Do not forget to insert a SYSMDUMP DD statement into the JCL of your PIF programs. It is very important to use the right disposition of the dump dataset because you have to be sure that the Dump written to the dataset will not be replaced by maintask or recursive abends. Therefore, we recommend that you use `DISP=MOD`. The disadvantage of the disposition is that the dump dataset can be burst when multiple dumps are written; so, make sure that you save the dumps and clear it afterwards.

The following is a SYSMDUMP example:

```
//SYSMDUMP DD DISP=MOD,DSN=OPC.V2R3M0.DMP
```

Please note that //SYSOUT=\* destroys the internal format of the Dump and renders it useless. When you experience an Abend and find no dumps in your dump datasets have a look to your DAE (Dump analysis and elimination) set up. DAE can be used to prevent the creating of certain kind of dumps. See also the *OS/390 V2R9.0 MVS Initialization and Tuning Guide*, SC28-1751.

#### 5.4.4 LOOP procedure

If your problem type is LOOP, you should take the following steps:

- Use the Tivoli OPC message log or system console log to help you identify what happened just before the program loop occurred.
- Obtain a dump using the `MVS DUMP` command. The internal system trace is very helpful for the IBM support representative to analyze a loop. The default trace table is 64 KB for each processor and could not be enough when encountering a wide spread loop. We recommend increasing the trace table to 120 KB before obtaining the dump with the following console command:

```
/Trace ST,120K
```

- To become familiar with obtaining a console dump, see also Section 5.4.8, “Preparing a console dump” on page 133.
- Document instruction addresses from within the loop, if possible.
- Provide a description of the situation leading up to the problem.

#### 5.4.5 Message (MSG) procedure

If your Tivoli OPC problem type is MSG, you should take the following steps:

- Look up the message in Tivoli OPC Messages and Codes for an explanation. This manual includes information on what action Tivoli OPC takes and what action the operator should take in response to a message. If you plan to report the problem, gather the documentation before you take action.
- Copy the message identifier and the message text. The IBM Support Center representative needs the exact message text.
- Supplement the MSG keyword with the message identifier. You use the supplemented keyword in your keyword string when searching the software support database.

With OS/390 V2R5, Tivoli OPC introduced a new task, EQQTTOP, that handles the communication between the TCP/IP server that now runs on



UNIX system services (USS) in full function mode. EQQTTOP uses C coding in order to use the new C socket interface. New messages are implemented, some of them pointing to other S/390 manuals.

Example:

*EQQTT20E THE RECEIVE SOCKET CALL FAILED WITH ERROR CODE 1036*

- **Explanation** - An error was encountered when the TCP/IP communication task attempted to issue a receive socket call to TCP/IP. The ERRNO value is the error code returned by the failing socket call.
- **System action** - Depending on the failing call, either the TCP/IP communication task is terminated or the specific socket connection is closed. Whenever possible, the task is automatically restarted. If the socket connection was closed, it is reestablished.
- **System programmer response** - Check the error code in the TCP/IP API manual and make any possible corrective action. If the error reoccurs, save the message log (EQQMLOG) and contact your IBM representative.

To find the cause, look in the “System Error Codes for socket calls” chapter of the *TCP/IP V3R2 for MVS API Reference*, SC31-7187.

Table 9. Socket error codes

Error number	Message name	Error description
1036	EIBMNOACTIVETCP	TCP/IP is not active

New modify commands in OPC are a comfortable way to get important information very quickly. When you want to find out which OPC task is active or inactive (other than by looking into MLOG for related messages), enter the following command in SDSF:

**Query the subtasks**

*/F procname,status,subtask*

*/\*where procname is the subsystem name of the Controller or tracker \*/*

```

F OPCA,STATUS,SUBTASK
EQQZ207I NORMAL MODE MGR  IS ACTIVE
EQQZ207I JOB SUBMIT TASK  IS ACTIVE
EQQZ207I DATA ROUTER TASK IS ACTIVE
EQQZ207I TCP/IP TASK      IS ACTIVE
EQQZ207I EVENT MANAGER    IS ACTIVE
EQQZ207I GENERAL SERVICE  IS INACTIVE
EQQZ207I JT LOG ARCHIVER   IS ACTIVE
EQQZ207I EXTERNAL ROUTER  IS ACTIVE
EQQZ207I WS ANALYZER      IS ACTIVE

```

The above screen shows that the general service task has an inactive status. To find more details, have a look into MLOG. The modify commands are described in the *TME 10 OPC Quick Reference*, GH19-4374.

#### 5.4.6 Performance (PERFM) procedure

If your problem concerns performance, you should:

- Document the actual performance, the expected performance, and the source of information for the expected performance. If a document is the source, note the order number and page number of the document.
- Document the information about your operating environment, such as the number of active initiators, the number of TSO users, and the number of Tivoli OPC users connected. Any user modifications to the program exits, REXX programs, and command lists can affect performance. You should consider whether the user-installed code, REXX programs, or CLISTs are contributing to the problem.
- Document any modifications to your system. Performance problems can be related to various system limitations. Your market division representative might be able to identify possible causes of a performance problem.

##### Note

Performance considerations are discussed in Chapter 10, “Best practices” on page 307.

#### 5.4.7 WAIT procedure

If your problem type is WAIT, you should take the following steps:

- Research the activity before system activity was suspended, identifying which operation is in the wait state.

- Specify any messages that were sent to the Tivoli OPC message log or to the system console.
- Obtain a dump using the MVS DUMP command. Check if the dump options include RGN and GRSQ.
- A wait state in the system is similar to a hang. However, the processing is suspended. Usually, it is recognized by a poor dialog or no job submission. A probable cause could be that one task holds a resource while other tasks are waiting until the owning task releases the resource. Such resource contentions can happen a lot of the time but are not serious if they are resolved in a short time. If you experience a long wait or hang, you can display an eventual resource contention when entering the following command in SDSF:

```
COMMAND INPUT ==> /D grs,c

ISG343I 23.23.19 GRS STATUS 043
S=SYSTEMS SYSZDRK OPCATURN2
```

SYSNAME	JOENAME	ASID	TCBADDR	EXC/SHR	STATUS
MCEVS4	OPCA	003F	007DE070	EXCLUSIVE	OWN
MCEVS4	SFRA4CC	002C	007DEDC8	EXCLUSIVE	WAIT

As you see, there are two tasks trying to get access (or lock) for one resource exclusive. Exclusive means that no other task can get the lock at the same time. An exclusive lock is usually an update access. The second task has to wait until the first, which is currently the owner, releases it. Message ISG343I returns with two fields, called Major and Minor name. In our example, *SYSZDRK* is the major name, and *OPCATURN2* is the minor name. *SYSZDRK* represents the active OPC current plan while the first four digits of the minor name represents your OPC subsystem name. With this information, you can search for known problems in the Software database. If you find no hint, your IBM support representative may ask you for a *console dump*.

#### 5.4.8 Preparing a console dump

The console dump contains a snapshot of virtual storage areas, such as System dumps. The major difference is that a System dump is created by the operating system when an abnormal end happens. The console dump has to be created by you via OS/390 commands from the system console. The dump options are very important because they influence the different parts of

storage to be dumped. For waits or hangs, the GRSQ option must be turned on.

The following panel shows the display of the current dump options.

```
COMMAND INPUT ==>/d d,o
RESPONSE=MCEVS4
IEE857I 18.22.00 DUMP OPTION 371
SDUMP- ADD OPTIONS (ALLPSA,SQA,LSQA,RGN,LPA,TRT,CSA,SWA,SUMDUMP,
                  ALLNUC,Q=YES,GRSQ),BUFFERS=00000000K,
                  MAXSPACE=00001200M,MSGTIME=99999 MINUTES
```

SDUMP indicates the options for the SYSMDUMP, which is the preferred type of dump. The options shown are sufficient for almost every dumps in OPC. For a detailed explanation, refer to the OS/390 System commands, *options for SDUMP types*. If you miss one of these options, you can change it with the change dump command (CD) command. For GRSQ, as an example:

CD SET,SDUMP=(GRSQ)

You need to be sure that the dump datasets, which have been provided by the OS/390 installation, are free to be taken.

```
COMMAND INPUT ==>/d d,t
RESPONSE=MCEVS4
IEE853I 18.43.40 SYS1.DUMP TITLES 385
SYS1.DUMP DATA SETS AVAILABLE=003 AND FULL=000
CAPTURED DUMPS=0000, SPACE USED=00000000M, SPACE FREE=00001200M
```

The example in the previous screen shows that all three can be used for console dumps. If not, you can clear a certain one. Make sure that nobody needs it anymore.

The `//dd clear,dsn=00` command means that Sys1.Dump00 is eligible

#### 5.4.8.1 Dump the failing system

Now, you are ready to obtain the console dump for further analysis.

```
COMMAND INPUT ==> dump comm=(demo)
19:13:27.27 SFRA4 00000290 DUMP COMM=(DEMO)
19:13:27.30 SFRA4 00000090 *17 IEE094D SPECIFY OPERAND(S) FOR DUMP COMMAND
```

Enter outstanding reply number, 17, as follows:

```
COMMAND INPUT ==>/17,tsoname=(opca)

SFRA4 00000290 R 17,TSOname=(OPCA)
SFRA4 00000090 IEE600I REPLY TO 17 IS;TSOname=(OPCA)
00000090 IEA794I SVC DUMP HAS CAPTURED: 482
00000090 DUMPID=049 REQUESTED BY JOB (*MASTER*)
00000090 DUMP TITLE=DEMO
00000290 IEF196I IGD100I 40AA ALLOCATED TO DDNAME SYS00273
00000290 IEF196I IEF285I SYS1.DUMP01
00000290 IEF196I IEF285I VOL SER NOS= 0260C1.
00000090 IEA611I COMPLETE DUMP ON SYS1.DUMP01
```

Do not be confused about the TSONAME parameter. It specifies the name of an address space to be dumped. Alternatively, you can use ASID(hex) as well. Dump processing finished successfully as indicated by the message, *IEA611I*. Please verify the existence of this message when you provide the dump to your local IBM support.

#### 5.4.9 Information needed for all problems

Even when you are unable to identify a problem type, you should gather the following information for any problem you have. Begin your initial problem analysis by examining the contents of the message log dataset with the following steps:

1. Obtain a copy of the Tivoli OPC message log. This is a sequential dataset defined by the EQQMLOG ddname.
2. Record the Tivoli OPC component ID: 5697-0PC01. The component ID should be the first keyword in the string preceding the problem type and other modifier keywords.
3. Record the maintenance level for all operating environments, particularly those for MVS, JES, ISPF, and RACF.
4. Document any additional program temporary fixes (PTFs) or APARs that have been applied to your level of Tivoli OPC.
5. If the problem is within the network communication function, obtain copies of the Tivoli OPC EQQDUMP file.
6. Obtain copies of the Tivoli OPC diagnostic files defined to the user address space and to the subsystem address space by SYSMDUMP.
7. Obtain a copy of the system log.

8. Reconstruct the sequence of events leading to the problem. Include any commands entered just before the problem occurred.
9. Write down the exact events that lead to the problem:
  - What was the first indication of the problem?
  - What were you trying to do?
  - What should have happened?
  - What did happen?
  - Can you re-create the problem?
10. Specify any unique information about the problem or about your system:
  - a. Indicate any other applications that were running when the problem occurred.
  - b. Describe how Tivoli OPC was started.
  - c. Describe all user modifications to active Tivoli OPC programs.

If more information is needed, an IBM Support Center representative will guide you concerning any additional diagnostic traces that you can run.

---

## 5.5 Performing problem determination for tracking events

Successful tracking of jobs in Tivoli OPC relies on the creation of different events written in the Tracker address space and processed from the Controller. The Controller waits for the complete arrival of these events and updates the current plan accordingly.

The different tracking events are listed in Table 10:

*Table 10. Tracking events*

Event number	Event name	Meaning
<b>1</b>	Reader event	A job has entered the system
<b>2</b>	Start event	a job has started to execute
<b>3S</b>	Step end event	A step has finished execution
<b>3J</b>	Job end event	A job has finished execution

Event number	Event name	Meaning
<b>3P</b>	Job termination event	A job has been added to the output queue
<b>4</b>	Print event	an output group has been printed
<b>5</b>	Purge event	A job output has been purged from the JES spool

The events are prefixed with either **A** (for JES2) or **B** (for JES3). At least the set of type 1, 2, 3J, and 3P events is needed to correctly track the several stages of a job's life. The creation of step-end events (3S) depends on the value you specify in the STEPEVENTS keyword of the EWTROPTS statement. The default is to create a step-end event only for abending steps in a job or started task. The creation of print events depends on the value you specify in the PRINTEVENTS keyword of the EWTROPTS statement. By default, print events are created.

If you find that the current plan status of a job is not reflecting the status in JES, you may have missing events. A good starting point is to run the OPC AUDIT package for the affected occurrence to easily see which events are processed from the Controller and which are missing, or you can browse your event datasets for the jobname and jobnumber to prove which events are not written.

Problem determination depends on which event is missing and whether the events are created on a JES2 or JES3 system. In Table 11 on page 138, the first column refers to the event type that is missing, and the second column tells you what action to perform. The first entry in the table applies when all

event types are missing (when the event dataset does not contain any tracking events).

Table 11. Problem determination of tracking events

Type	Problem determination actions
<b>ALL</b>	<ol style="list-style-type: none"> <li>1. In the EQQMLOG dataset, verify that the event writer has started successfully.</li> <li>2. Verify that the definition of the EQQEVDS ddname in the Tivoli OPC started-task procedure is correct, that is, the events are written to the correct dataset.</li> <li>3. Verify that the required exits have been installed.</li> <li>4. Verify that the IEFSSN nn member of SYS1.PARMLIB has been updated correctly and that an IPL of the MVS system has been performed since the update.</li> </ol>
<b>A1</b>	<p>If both A3P and A5 events are also missing:</p> <ol style="list-style-type: none"> <li>1. Verify that the Tivoli OPC version of the JES2 exit 7 routine has been correctly installed. Use the \$T EXIT(7) JES command.</li> <li>2. Verify that the JES2 initialization dataset contains a LOAD statement and an EXIT7 statement for the Tivoli OPC version of JES2 exit 7 (OPCAXIT7).</li> <li>3. Verify that the exit has been added to a load module library reachable by JES2 and that JES2 has been restarted since this was done. If either A3P or A5 events are present in the event dataset, call an IBM service representative for programming assistance.</li> </ol>
<b>B1</b>	<ol style="list-style-type: none"> <li>1. Verify that the Tivoli OPC version of the JES3 exit IATUX29 routine has been correctly installed.</li> <li>2. Verify that the exit has been added to a load-module library that JES3 can access.</li> <li>3. Verify that JES3 has been restarted.</li> </ol>



Type	Problem determination actions
A2/B2	<ol style="list-style-type: none"> <li>1. Verify that the job for which no type 2 event was created has started to execute. A type 2 event will not be created for a job that is flushed from the system because of JCL errors.</li> <li>2. Verify that the IEFUJI exit has been correctly installed: <ol style="list-style-type: none"> <li>a. Verify that the SMF parameter member SMFPRM nn in the SYS1.PARMLIB dataset specifies that the IEFUJI exit should be called.</li> <li>b. Verify that the IEFUJI exit has not been disabled by an operator command.</li> <li>c. Verify that the correct version of IEFUJI is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, MVS uses the first IEFUJI module found.</li> <li>d. Verify that the library containing this module was updated by the Tivoli OPC version of IEFUJI and that MVS has been IPLd since the change was made.</li> </ol> </li> </ol>

Type	Problem determination actions
A3S/B3S	<p>If type 3J events are also missing:</p> <ol style="list-style-type: none"> <li>1. Verify that the IEFACTRT exit has been correctly installed.</li> <li>2. Verify that the SMF parameter member SMFPRM nn in the SYS1.PARMLIB dataset specifies that the IEFACTRT exit should be called.</li> <li>3. Verify that the IEFACTRT exit has not been disabled by an operator command.</li> <li>4. Verify that the correct version of IEFACTRT is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, MVS uses the first IEFACTRT module found.</li> <li>5. Verify that this library was updated by the Tivoli OPC version of IEFACTRT and that MVS has been IPLd since the change was made.</li> </ol> <p>If type 3J events are not missing, verify, in the EQQMLOG dataset, that the event writer has been requested to generate step-end events.</p> <p>Step-end events are only created if the EWTROPTS statement specifies STEPEVENTS(ALL) or STEPEVENTS(NZERO) or if the job step abended.</p>
A3J/B3J	<p>If type 3S events are also missing, follow the procedures described for type 3S events.</p> <p>If type 3S events are not missing, call an IBM service representative for programming assistance.</p>
A3P	<p>If A1 events are also missing, follow the procedures described for A1 events.</p> <p>If A1 events are not missing, call an IBM service representative for programming assistance.</p>
B3P	<ol style="list-style-type: none"> <li>1. Verify that the Tivoli OPC version of the JES3 exit IATUX19 routine has been correctly installed.</li> <li>2. Verify that the exit has been added to a load-module library that JES3 can access.</li> <li>3. Verify that JES3 has been restarted.</li> </ol>

Type	Problem determination actions
A4/B4	<ol style="list-style-type: none"> <li>1. If you have specified PRINTEVENTS(NO) on the EWTROPTS initialization statement, no type 4 events are created.</li> <li>2. Verify that JES has printed the job for which no type 4 event was created. Type 4 events will not be created for a job that creates only held SYSOUT datasets.</li> <li>3. Verify that the IEFU83 exit has been correctly installed: <ol style="list-style-type: none"> <li>a. Verify that the SMF parameter member SMFPRM nn in the SYS1.PARMLIB dataset specifies that the IEFU83 exit should be called.</li> <li>b. Verify that the IEFU83 exit has not been disabled by an operator command.</li> <li>c. Verify that the correct version of IEFU83 is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, MVS uses the first IEFU83 module found.</li> <li>d. Verify that the library containing this module was updated by the Tivoli OPC version of IEFU83 and that MVS has been IPLd since the change was made.</li> <li>e. For JES2 users (A4 event), ensure that you have not specified TYPE6=NO on the JOBCLASS and STCCLASS statements of the JES2 initialization parameters.</li> </ol> </li> </ol>
A5	<ol style="list-style-type: none"> <li>1. Verify that JES2 has purged the job for which no A5 event was created.</li> <li>2. Ensure that you have not specified TYPE26=NO on the JOBCLASS and STCCLASS statements of the JES2 initialization parameters.</li> <li>3. If A1 events are also missing, follow the procedures described for A1 events.</li> <li>4. If A1 events are not missing, call an IBM service representative for programming assistance.</li> </ol>

Type	Problem determination actions
B5	<ol style="list-style-type: none"> <li>1. Verify that JES3 has purged the job for which no B5 event was created.</li> <li>2. If B4 events are also missing, follow the procedures described for B4 events.</li> <li>3. If B4 events are not missing, call an IBM service representative for programming assistance.</li> </ol>

## 5.6 Trace for the TCP/IP Server

If you encounter problems on the TCP/IP server, you can insert Diagnose Flags for the server within the server parameter file in order to trace the type of data you are looking for. The trace information will be put into the EQQMLOG of the server. To activate the trace, define the diagnose statement in the following way:

```
DIAGNOSE SERVERFLA(.....)
```

The following screen lists the types of data that can be traced. Ask your IBM support center for the right settings related to your problem.

```

SERVERFLA(X'01000000')    *    BUFFERS
SERVERFLA(X'02000000')    *    CONVERSATION
SERVERFLA(X'04000000')    *    TCPIP SOCKET CALLS
SERVERFLA(X'00800000')    *    PIF
SERVERFLA(X'00010000')    *    SECURITY
SERVERFLA(X'03000000')    *    BUFFERS AND CONVERSATION
SERVERFLA(X'05000000')    *    BUFFERS AND TCPIP
SERVERFLA(X'06000000')    *    CONNECTION AND TCPIP
SERVERFLA(X'07000000')    *    CONNECTION BUFFERS AND TCPIP
SERVERFLA(X'FFFFFFF')     *    ALL
SERVERFLA(X'FF7FFFFF')    *    ALL WITHOUT PIF

```

---

## 5.7 Trace for the OPC Connector

The trace levels, listed in Table 12, are available for an OPC Connector instance:

Table 12. Connector trace level

Level	Traced data
0	Errors
1	Called methods Connections IDs
2	Filters PIF requests Numbers of elements returns in queries
3	Flow and connection details Main functions in/out and main steps
4	Services functions in/out Main functions in/out values
5	Frequently called functions in/out

The default length of the trace is 512 KB for each instance. When the length is exceeded, the trace is wrapped around. If an unexpected error occurs, the trace must be copied as soon as possible. You will find the trace in the directory, \$DBDIR/OPC/engine\_name.log.

The trace can either be activated at the line command with `wopcconn` or in interactive mode. To control the current settings, issue the following command:

```
wopcconn -view -e engine_name | -o object_id
```

In order to set a trace level, issue the following command:

```
wopcconn -set -e engine_name | -o object_id [-t trace_level] [-l  
trace_length]
```

Or, use interactive mode, `wopcconn`, as follows:

1. Type `wopcconn` from a 3270 session.

```
***** OPC Connector manage program *****
Main menu

1. Create new OPC Connector
2. Manage an existing OPC Connector

0. Exit
```

2. Select option **2 Manage an existing OPC Connector**.

```
T
***** OPC Connector manage program *****
Select instance menu

1. OPC

0. Exit
```

3. Choose the instance you want to change.

```
***** OPC Connector manage program *****
Manage menu

Name                : OPC
Object id           : 1929225022.1.1771#OPC::Engine#
Managed node       : its07
Status              : Active

OPC version          : 2.3.0

1. Stop    the OPC Connector
2. Start   the OPC Connector
3. Restart the OPC Connector

4. View/Change attributes

5. Remove instance

0. Exit
```

4. Select option **4** to change the attributes of the instance.

```

***** OPC Connector manage program *****
View/Change attributes menu

Name                : OPC
Object id           : 1929225022.1.1771#OPC::Engine#
Managed node       : itso7
Status              : Active

OPC version         : 2.3.0

2. Name              : OPC

3. IP Address or Hostname: 9.39.62.19
4. IP portnumber     : 3111

5. Trace Length      : 524288
6. Trace Level       : 0

0. Exit

```

5. Change the trace level with option **6** to your value.

```

***** OPC Connector manage program *****
View/Change attributes menu

Name                : OPC
Object id           : 1929225022.1.1771#OPC::Engine#
Managed node       : itso7
Status              : Active

OPC version         : 2.3.0

2. Name              : OPC

3. IP Address or Hostname: 9.39.62.19
4. IP portnumber     : 3111

5. Trace Length      : 524288
6. Trace Level       : 1

0. Undo changes
1. Commit changes
The changes works the next time you start the OPC Connector

```

6. Commit your changes and restart the OPC Connector to activate it.

---

## 5.8 Trace for the Job Scheduling Console

In case of errors, check the message identifier to understand the source of the error.

- **GJS0xxx** - JS Console error
- **GJSQxxx** - OPC specific error

Read error details for explanations and suggested behaviors. Consult the trace file. Remember that error tracing is active by default. Also, check the file, `bin\java\error.log`, for untraced errors.

The Trace utility provides a strong mechanism to find and diagnose JS Console problems. A log file is produced with monitoring of all JS Console activities. Tracing can work at different detail levels to filter data of interest. Trace output can be customized.

Open the console file:

- `...\bin\java\console.bat` on Win32
- `.../bin/java/SUNconsole.sh` on Solaris
- `.../bin/java/AIXconsole.sh` on AIX

Find the section where the user can customize variable values. Locate the two variables, *TRACELEVEL* and *TRACEDATA*. They should be set to 0 by default.

```
T
REM ----- Section to be customized -----
REM change the following lines to adjust trace settings
set TRACELEVEL=0
set TRACEDATA=0
REM ----- End of section to be customized -----
```

Change the value of the variable, *TRACELEVEL*, to activate the control flow trace at different levels. Change the value of the variable, *TRACEDATA*, to activate the data flow trace at different levels. Acceptable values range from 0



to 3. TRACELEVEL also admits the value, -1, which completely disables the trace as shown in Table 13.

Table 13. Tracelevel values

Trace level value	Trace type
-1	No trace at all (to be used only in particular cases).
0	Only errors and warnings are recorded.
1	Errors, warnings, and info/debug lines are recorded.
2	Errors, warnings, and methods entry/exit are recorded.
3	Errors, warnings and info/debug lines and method entry/exit are recorded.

Table 14 lists trace data values and trace types.

Table 14. Tracedata values

Tracedata value	Trace type
0	No data is traced.
1	Datastructures from/to the Connector are traced.
2	The internal value of the JS Console beans are recorded.
3	Both data structures and bean internal values are recorded.

**Note**

Tracing can downgrade GUI performances. Use major values of TRACELEVEL and TRACEDATA only when necessary.

Trace files can become huge. Use advanced customization to optimize disk space allocation. Move or delete log files related to previous executions.



---

## Chapter 6. Troubleshooting for TWS

In this chapter, we will give you a checklist for troubleshooting TWS problems.

---

### 6.1 TWS troubleshooting checklist

The following is the TWS troubleshooting checklist:

#### ***FTAs not linking to the Master***

- *If netman is not running on the FTA:*
  - If netman has not been started, start it from the command line with the `StartUp` command. Note that this will start *only* netman, not any other TWS processes.
  - If netman started as root and not as a TWS user, bring TWS down normally, and then start up as a TWS user through the conman command line on the master or FTA:

```
unlink <FTA name>
stop <FTA name>; wait
shut <FTA name>; wait
StartUp
```
  - If netman could not create a standard list directory:
    - If the file system is full, open some space in the file system.
    - If a file with the same name as the directory already exists, delete the file with the same name as the directory. The directory name would be in a yyyy.mm.dd format.
    - If the directory or NETMAN standard list is owned by root and not TWS, change the ownership of the directory standard list from the command line in UNIX with the `chown yyyy.mm.dd TWS` command. Note that this must be done as root user.
- *If the host file or DNS changes, it means that:*
  - The host file on the FTA or Master has been changed.
  - The DNS entry for the FTA or Master has been changed.
  - The hostname on the FTA or Master has been changed.
- *Communication processes hung*
  - mailman process down or hung on FTA
    - TWS not brought down properly

Try to always bring TWS down properly via conman command line on the master or FTA using the following commands:

```
unlink <FTA name>
stop <FTA name>; wait
shut <FTA name>; wait
```

- If the mailman read corrupted data, try to bring TWS down normally. If this is not successful, kill the mailman process with the following steps.

Using command line UNIX:

1. Use `ps -ef | grep maestro` to find the process ID.
2. Issue `kill -9 <process id>` to kill the mailman process.  
or with command line NT (commands in unsupported directory):

1. Use `listproc` to find the process ID.
2. Run `killproc <process id>` to kill the mailman process.

- Batchman hung

Try to bring TWS down normally. If not successful, kill the mailman process as explained in the previous bullet.

- *If the writer process for FTA is down or hung on the Master, it means that:*

- FTA was not properly unlinked from the Master
- The writer read corrupted data
- Multiple writers are running for the same FTA

Use `ps -ef | grep maestro` to check writer processes running. If there is more than one process for the same FTA, perform the following steps:

1. Shut down TWS normally.
2. Check the processes for multiple writers again.
3. If there are multiple writers, kill them.

- *netman process hung*

- If multiple netman processes are running, try shutting down netman properly first. If this is not successful, kill netman using the following commands:

using command line UNIX:

1. Use `ps -ef | grep maestro` to find the running processes.
2. Issue `kill -9 <process id>` to kill netman process.

or with command line NT (commands in unsupported directory):

1. Use `listproc` to find the process ID.
  2. Run `killproc <process id>` to kill the mailman process.
- hung port / socket; FIN\_WAIT2 on netman port
    1. Use `netstat -a | grep <netman port>` for both UNIX and NT systems to check if netman is listening.
    2. Look for FIN\_WAIT2 for the TWS port
    3. If FIN\_WAIT2 does not time out (approximately 10 minutes), reboot.
- *Network problems to look for outside of TWS include:*
    - The router is down in a WAN environment
    - The switch or network hub is down on an FTA segment
    - There has been a power outage
    - There are physical defects in the network card / wiring

***Batchman not up or will not stay up (batchman down)***

- *If the message file has reached 1 MB:*
  - Check the size of \*.msg files in the TWShome directory (48 bytes minimum).
    1. Use the `EVTSIZE` command to expand temporarily, and then try to start TWS:
 

```
evtsize <filename> <new size in bytes>
```

For example, `evtsize mailbox 2000000`
    2. If necessary, remove the message file (only after failing with the `EVTSIZE` and start).

Note that you must do the remove operation only as a last resort; all data in message file will be lost.
  - Check size of \*.msg files in /pobox directory
 

Use the `EVTSIZE` command to expand temporarily, and then try to start TWS; remove it as a last resort.
- *jobman not owned by root*

Check the ownership of jobman in /TWShome/bin by issuing the following command with root user ID:

```
chown root /TWShome/bin/jobman
```
- *Read bad record in Symphony file*

This can happen for the following reasons:

- There is a byte order problem between UNIX and Intel platforms (requires patches)
- Initialization process interrupted or failed
- Cannot create Jobtable
- Corrupt data in Jobtable

- *Message file corruption*

This can be for the following reasons:

- Bad data
- File system full
- Power outage
- CPU hardware crash

### ***Jobs not running***

- *Jobs not running on NT*

- If NT authorizations for TWS user are not in place, you can try the following:

- Act as part of the operating system
- Increase quotas
- Log on as a batch job
- Log on as a service
- Log on locally
- Replace a process level token

- Valid NT or Domain user for FTA not in the TWS user database

Add TWS user for FTA in the TWS user database. Do not fill in CPU name field if TWS user is a Domain account.

- Password for NT user has been changed

Do one of the following:

- Change the password on NT to match the one in the TWS user database.
- Change the password in the TWS user database to a new password.

Note that changes to the TWS user database will not take effect until Jnextday.

If the user definition existed previously, you can use the `altpass` command to change the password for production day.

- *Jobs not running on NT or Unix*

- batchman down

See the section, "Batchman not up or will not stay up (batchman down)" on page 151.

- Fence limit set to 0

To change the fence limit via the `conman` command line:

for single FTA: `lc <FTA name>;10`

for all FTAs: `lc @;10;noask`

- Fence set above the limit

To change fence priority via the `conman` command line:

For all FTAs: `f @;10;noask`

- If dependencies are not met, it could be for the following reasons:

- Start time not reached yet, or UNTIL time has passed
- OPENS file not present yet
- Job FOLLOW not complete

***Jnextday is hung or still in EXEC state***

- *Does not have exclusive access to Symphony*

batchman and/or mailman was not stopped before running Jnextday from the command line

- *Jnextday not able stop all FTAs*

- Network segment down and cannot reach all FTAs
- One or more of the FTAs has crashed
- netman not running on all FTAs

- *Jnextday not able to start or initialize all FTAs*

- The Master or FTA was manually started before Jnextday completed stageman

Reissue a link from the master to the FTA

- The Master was not able to start batchman after stageman completed

See the section "Batchman not up or will not stay up (batchman down)" on page 151.

- The Master was not able to link to FTA

See the section “FTAs not linking to the Master” on page 149.

### ***Jnextday in ABEND state***

- *Jnextday not completing compiler processes*

This may be due to bad or missing data in schedule or job. You can perform the following actions:

- Check for missing calendars
- Check for missing resources
- Check for missing parameters

- *Jnextday not completing stageman process*

This may be due to bad or missing data in the CARRYFORWARD schedule. You can perform the following actions:

- Run show jobs or show schedules to find the bad schedule
- Add missing data and rerun Jnextday
- Cancel the schedule and rerun Jnextday

- *Jnextday not completing logman process*

The reason may be one of the following:

- Negative runtime error (requires patch)
- The Master was manually started before logman completed

### ***FTA still not linked after Jnextday***

- *Symphony file corruption*

Corruption during transfer of Sinfonia file

- Byte order problem between UNIX and Intel
- Apply patches for byte order.

- *Symphony file, but no new run number, date, or time stamp*

You can perform the following actions:

- Try to link FTA. See the section, “FTAs not linking to the Master” on page 149.
- Remove Symphony and message files (on FTA only) and link from the Master again

- *Run number, Symphony file, but no date or time stamp*



You can perform the following actions:

- Try to link FTA. See the section, “FTAs not linking to the Master” on page 149.
- Remove Symphony and message files (on FTA only) and link from the Master again.



---

## Chapter 7. Tivoli Workload Scheduling Agent for SAP R/3

In this chapter, we will explain the Extended Agent concept and cover the implementation of Tivoli Workload Scheduling Agent for SAP R/3 as an example Extended Agent.

---

### 7.1 The anatomy of an Extended Agent

An Extended Agent (X-agent) is a system or application that is dependent on its TWS host to receive scheduling instructions. Extended Agents use open scheduling APIs and protocols and provide an effective mechanism for extending TWS scheduling capabilities to foreign platforms and applications. X-agents also allow segregation of applications at the workstation level by providing an alternative method to the *jobmanrc* process. This allows some applications to be treated differently using an alternative job scheduling method.

The connection between TWS and X-agent is shown in Figure 94. X-agent is installed on one of the TWS hosts. TWS accepts information from the X-agent. The Interface between TWS and an X-agent is called the access method. The access method is a shell script or program that resides in the *~TWSHOME/methods* directory.

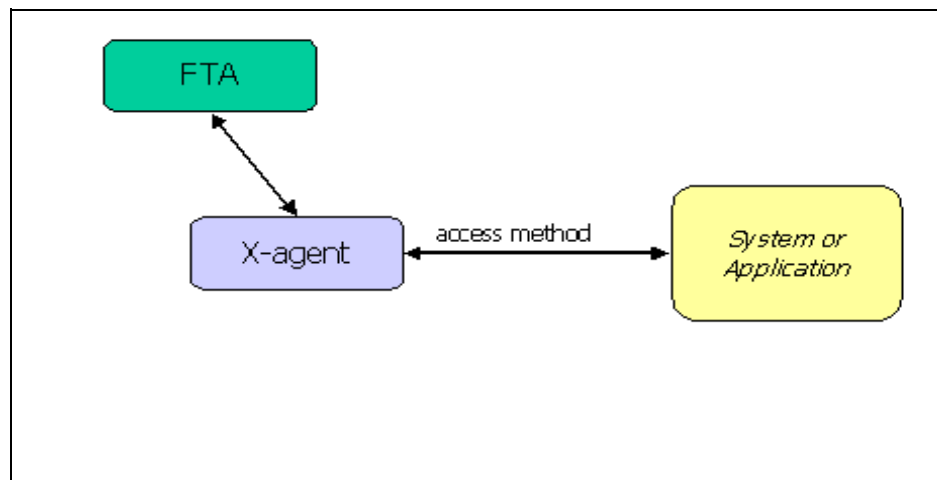


Figure 94. X-Agent network

X-agent processing is depicted in Figure 95 on page 159. The sequence of operations is as follows:

**Note**

This explanation is given without consideration for network communication issues. It is accepted that the X-agent resides on a TWS Server. If the X-agent resides on an FTA, the sequence of operations for communication with the X-agent will be same but there will be additional processes for communication between the TWS Master and the FTA.

1. Batchman process on the FTA talks to the jobman (jobman.exe on an NT system) that is owned by the root user ID.
2. Jobman invokes a JOBMAN (jobmon.exe on an NT system) process in the context of the TWS user (maestro, for example).
3. JOBMAN talks to the access method.
4. The access method invokes a Remote Function Call (RFC).
5. The access method consults with the <method.opts> file.
6. The access method talks to the system or application's job.
7. The access method and the job communicate with each other.
8. The method passes the information back to the TWS host through JOBMAN.

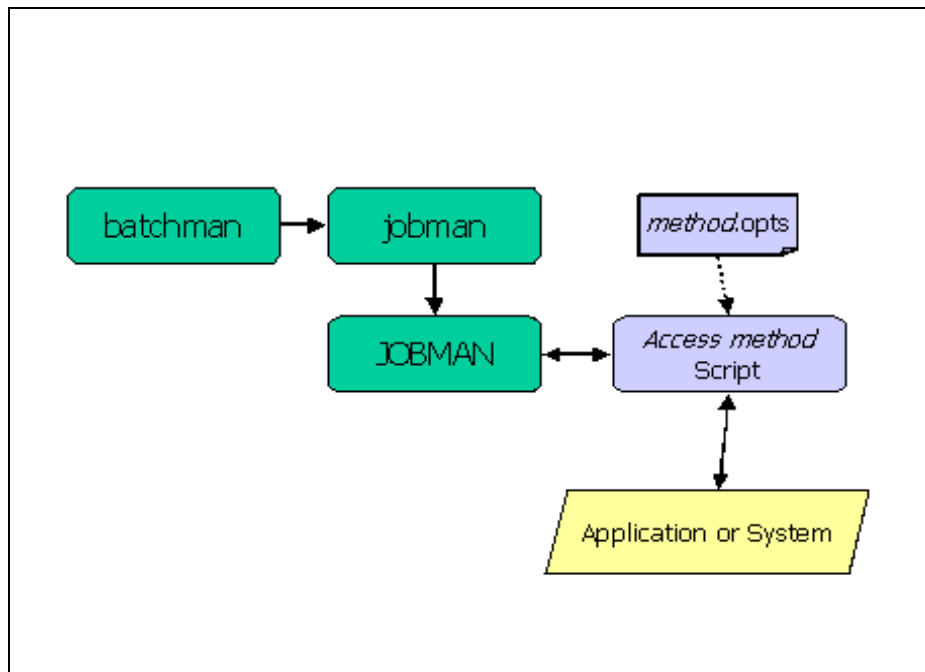


Figure 95. Extended Agent processing

The JOBMAN process launches the method script to perform one of the following tasks:

- Launch a job
- Manage a job
- Check for the existence of a file to satisfy an OPENS dependency
- Get the status of an external job

The syntax of the method execution is as follows:

```
methodname -t task options -- taskstring
```

where:

task is one of the following:

- LJ (Launch a job)
- MJ (Manage a previously launched job)
- CF (Check availability of a file – OPENS dep)

- GS (Get the status of an external job)

`options` are the list of job properties

`taskstring` is the string to execute from the job definition.

The following are the options (related to the job's properties) that should be passed to the access method:

- Workstation/Host/Master
- Workstation's node definition
- Workstation's port definition
- The current production run number
- The job's run number
- The job's schedule name
- The date of the schedule (in two formats)
- The user ID (logon) for the job
- The path to the output (stdlist) file for the job
- The job ID
- The job number
- The string from the SCRIPTNAME or DOCOMMAND entry in the job definition

The following screen shows an example method invocation where job TEST is executed on the X-agent workstation, ITS06, using the user ID, `itso`, and the method name, `wgetmethod`.

```
wgmethod -t LJ
-c ITS06,ITS07,ITS07
-n ITS06
-p 31111 -r 143,143 -s MACDSH91
-d 20000410,955381986 -l itso
-o /opt/maestro/stdlist/2000.04.10/013676.1053
-j TEST,13676 -- /home/itso//batchjobs/killer
```

The following are the current X-agents provided by TWS:

- UNIX Remote Shell
- UNIX Local
- MVS (JES,OPC, CA7)

- SAP R/3 Batch
- PeopleSoft
- BAAN
- Oracle Applications

**Note**

In addition to TWS provided X-agents, you can always write your own X-agents for platforms or applications that have not been supported by TWS using the open API that is documented in the X-Agent Programmer's Reference.

---

## 7.2 What are the benefits of using TWS agent for SAP R/3?

By using the Tivoli Workload Scheduling Agent for SAP R/3, you can enhance R/3 job scheduling in the following ways:

- Eliminate the need in R/3 to repeatedly define jobs, even if the content of the job does not change. With TWS, you define the job in R/3 once and then reuse it.
- Manage jobs that have dependencies with other R/3 instances or even other resources or jobs originating from other sources on different systems, such as Oracle Applications, PeopleSoft, and Baan IV. You can create dependencies that allow you to structure your workflow and model your business process.
- Define recovery jobs in case your R/3 job fails. This helps in network automation and helps you take quick action to recover from a failed job.
- Define interdependencies between R/3 jobs and jobs that run on other platforms including UNIX, Windows NT, MVS, and HP MPE.

---

## 7.3 SAP R/3 Extended Agent details

The SAP R/3 Extended Agent is defined in a standard Tivoli Workload Scheduler workstation definition, which gives the X-agent a name and identifies the access method as *r3batch*. When invoked, the method reads configuration information from an options file called *r3options*. Each instance of R/3 is defined as a separate Extended Agent, and each has a separate entry in the *r3options* file.

To launch a job on an SAP R/3 X-agent, the TWS host executes r3batch, passing it information about the job. Using the Extended Agent's workstation name as a key, r3batch looks up the corresponding entry in the r3options file to determine which instance of R/3 will run the job. r3batch makes a copy of the template and then marks the job in SAP R/3 as able to run and sets its start time to now. It then monitors the job through completion, writing job progress and status information to the job's standard list file.

R3batch uses Remote Function Call (RFC) methods to handle R/3 jobs. RFC is provided by SAP to call R/3 functions from external programs. RFC is implemented as a platform-dependant library.

**Note**

R3batch 4.0 gained SAP certification for R/3 release 4.0 and 4.5.

### 7.3.1 Tivoli Workload Scheduler and R/3 job states

When an SAP R/3 job is launched by Tivoli Workload Scheduler, you can monitor its progress. The status transitions in TWS (internal status), and the corresponding R/3 statuses are listed in Table 15.

*Table 15. Comparison of TWS and SAP R/3 Job States*

<b>TWS Internal Job State</b>	<b>R/3 Job State</b>
INTRO	n/a
WAIT	ready
EXEC	active
SUCC	finished
ABEND	cancelled

The INTRO state indicates that TWS is in the process of introducing the job, but, in R/3, the job has not yet entered the ready state. Because it takes some time to get a job queued and into the ready column, the INTRO state may last a few minutes if the R/3 system is particularly busy.

Although a job may be finished in R/3, TWS will keep it in the EXEC state if its BDC sessions are not complete and you have not selected the Disable BDC Wait option.



**Note**

The BDC acronym stands for Batch Data Collector and with the BDC wait option, you can specify that an R/3 job launched by TWS will not be considered complete until all of its BDC sessions have completed. This prevents other TWS jobs that are dependent on the R/3 job from being launched until all of the related BDC sessions for the R/3 job have complete.

---

## 7.4 Our environment and implementation steps

The environment we have set up reflects a typical business design where Tivoli Workload Scheduler is used to control and schedule SAP R/3 batch jobs.

Since the SAP R/3 system we have been using is located in Raleigh, NC, we decided to install the Tivoli Scheduling Agent for SAP R/3 on one of our local FTAs and let the Extended Agent point to the system in Raleigh. Usually, the FTA and the Extended Agent are installed on the same system that is running the SAP application server.

We performed the following steps to set up the environment:

1. Install the Tivoli Scheduling Agent for SAP R/3 on an already existing FTA.
2. Set up the Extended Agent to point to the correct SAP R/3 application server and the correct instance.
3. Install the SAPGUI on one of our NT workstations (not mandatory for TWS X-agent to work but recommended for troubleshooting purposes).
4. Define a workstation in TWS, which represents the connection to the SAP R/3 system.
5. Use the Job Scheduling console to define and launch SAP R/3 batch jobs.

Figure 96 on page 164 shows a diagram of our SAP environment.

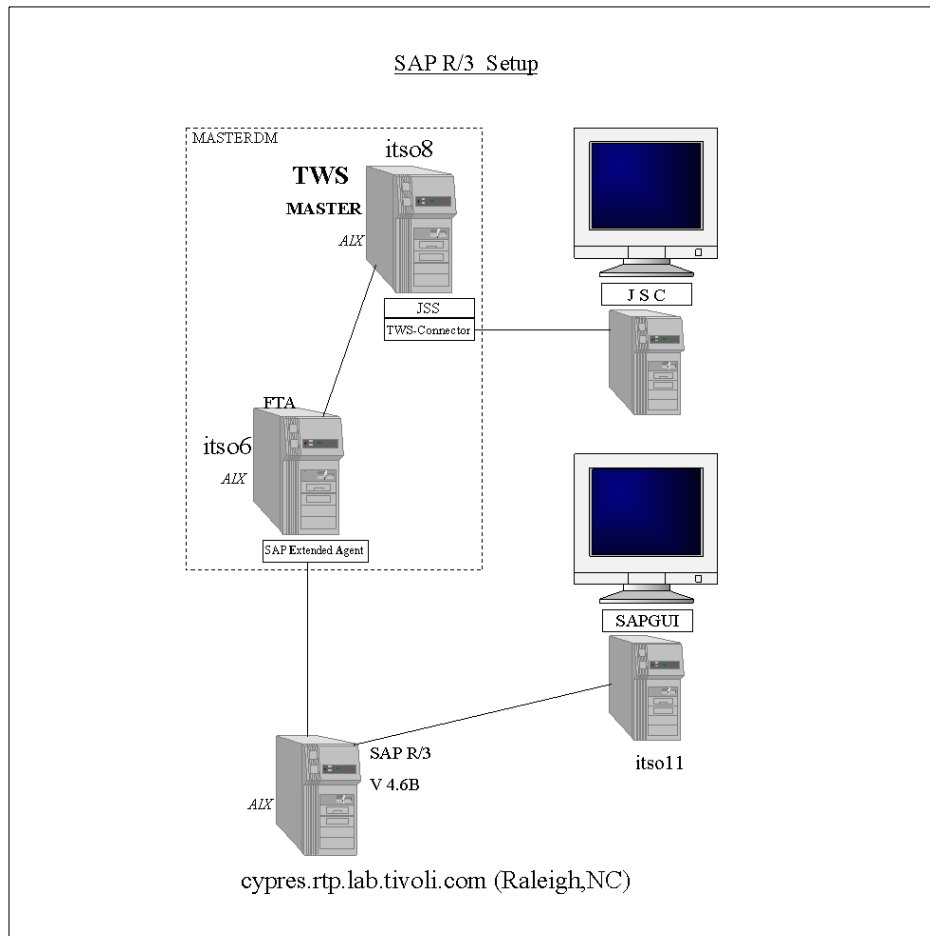


Figure 96. SAP R/3 environment

## 7.5 Installing of the TWS Extended Agent for SAP R/3 on AIX

We installed the Tivoli Workload Scheduling Agent for SAP R/3 on one of our AIX-systems, as most SAP-systems being controlled by TWS run on AIX. For a detailed description of the installation, its use with Tivoli Workload Scheduler as well as the setup of SAP R/3 versions other than 4.6 B, refer to the *Tivoli Scheduling Agent for SAP R/3*, GC31-5147.

### 7.5.1 Software Requirements

To use the Tivoli Workload Scheduling Extended Agent for SAP R/3 Version 4.0, you must be running the following:

- TWS 5.2 or later
- SAP R/3 Version 3.0f, 3.1i, 4.0x, 4.5x, and 4.6x
- SAP R/3 Hot Package 8 is required for SAP R/3 4.5b installations

### 7.5.2 Installation

1. Log in as *root* and change to *TWSHome*, in our case, */opt/maestro*.
2. Extract the software from the *SAP.TAR* file, which can be found either on the installation CD-ROM or on your fileserver.

```
tar xvf path/SAP.TAR
```

This creates two files in the current directory: *r3setup* and *r3btar.Z*

3. Execute the *r3setup* script to decompress the file, *r3btar.Z*, perform the initial setup, and create the *r3options* file.

```
/bin/sh r3setup -new | -update
```

where:

*-new*

is used for new installations

*-update*

is used when installing over a previously-installed version of the TWS Extended Agent for SAP R/3

The following screen is a snapshot of the installation we have done on our FTA ITS06. It includes all input and output seen on the screen. The letters with the bold format are the inputs that we supplied running this script.

```

root@itso6 > pwd
/opt/maestro
root@itso6 > tar xvf /Tivoli/SAP-XA/SAP.TAR
x r3setup, 17606 bytes, 35 media blocks.
x r3btar.Z, 1235330 bytes, 2413 media blocks.
root@itso6 > /bin/sh r3setup -new
R3SETUP: R3BATCH Installer -- Version 2.0
=====
This script installs the Tivoli method R3BATCH in the appropriate directory Installation time is five
minutes, and less than 5 Mbytes additional disk space will be used. This script creates or updates the
r3options file that R3BATCH uses to determine how to open a connection to R/3.
IMPORTANT: An appropriate scheduling agent must be installed BEFORE this script is run.
=====
Proceed (yes or no)? y
Installation proceeding...
The default username is maestro.
Is this correct (yes or no)? yes
Installation proceeding...
x r3batch.r3ins, 2644432 bytes, 5165 media blocks.
x r3batch.msg.r3ins, 11625 bytes, 23 media blocks.
x K000119.tvl.r3ins, 155 bytes, 1 media blocks.
x R000119.tvl.r3ins, 80727 bytes, 158 media blocks.
x enigma.r3ins, 22300 bytes, 44 media blocks.
x VERSION.aix-4-1.4.0.0.10.r3ins, 104 bytes, 1 media blocks.
Installing Tivoli Workload Scheduler - SAP R/3 Extended Agent, aix-4-1 version 4.0.0.10version
4.0.0.10
+++ Copying r3batch into ./methods/
+++ Copying enigma into ./bin/
+++ Copying r3batch.msg into ./catalog/
+++ Copying R/3 transport files into ./
... copying K000119.tvl ...
... copying R000119.tvl ...
You should remember to copy the R/3 transport files listed above to your R/3 database server and
follow the instructions in the Install guide Until this is done R3BATCH will not be operational!
Beginning the update of the r3options control file...
What is the host CPU name for this R/3 system? r3xal
What is the host name of your R/3 application server? cypress.rtp.lab.tivoli.com
PING cypress.rtp.lab.tivoli.com: (69.205.182.174): 56 data bytes
64 bytes from 69.205.182.174: icmp_seq=0 ttl=121 time=55 ms
64 bytes from 69.205.182.174: icmp_seq=1 ttl=121 time=48 ms
---cypress.rtp.lab.tivoli.com PING Statistics---
2 packets transmitted, 2 packets received, 0% packet loss
round-trip min/avg/max = 48/51/55 ms
Certain network topologies require specifying a separate R/3 gateway hostname, different from the
from the application server name. If you are not sure whether a gateway host is needed, then then e
then you should answer no to the following question
Do you wish to specify a separate R/3 gateway host (yes/no)? no
What is your three-character R/3 system id (e.g. OSS or TRN)? D10
What is your two-digit R/3 instance number (e.g. 00 or 01)? 00
What is your three-digit R/3 client number (e.g. 100 or 900)? 100
What is your R/3 user id for RFC calls (e.g. rfcuser)? maestro
What is the password for your R/3 rfc user id (e.g. sesame)?xxxxx
Your password is being encrypted...
JOB STATUS INTERVAL: [Defaults used: 30 seconds/300 seconds]
While the host is running R/3 jobs it will periodically wake up processes to check on the status of
the jobs. The interval between these status checks

```

```

is configurable, and is specified as a pair of values: the minimum
interval and the m interval. As a job begins running the status interval is initially
set to the minimum value, and as the job runs the interval is gradually creased up
increased up to the maximum value.
If you wish to change the intervals from the defaults (30/300),edit the r3options file after this
after this script completes.
(Press return when ready to continue) Enter
The next entry is for R/3 interface audit level.
This is only applicable for R/3 version 3.1G or higher.
Use default audit level of 0 for normal operation.
Edit the r3options file if you want to change the audit level later.
What is your R/3 interface audit level (e.g. 0, 1, 2 or 3) 0 ? 0
+++ New r3options file entry is:
r3xal cypress.rtp.lab.tivoli.com D10 00 100 maestro #p42-xZU5-MY2A-mX 30 300 0
The /etc/services file does not need updating
=====
The R3BATCH software update has completed.
To finish this installation, please make sure that you follow the
additional installation steps such as running the R/3 Correction
and Transport commands to load the supplementary R/3 function
modules.
Additional installation steps are in the Installation Guide

```

## 7.5.3 r3options file reference

The r3options file must reside on the TWS host workstation in the *TWSHOME/methods* directory. It contains a set of options for each SAP R/3 Extended Agent workstation. During initial installation or after entering `r3setup -maintain`, you are prompted to enter eleven attributes for each SAP R/3 Extended Agent.

### 7.5.3.1 Attributes in r3options file

The options for each SAP R/3 Extended Agent are as follows:

- **Host Workstation/CPU Name** - The TWS workstation name of the Extended Agent for this instance of R/3. This must be the same as the workstation Name entry in the x-agent's workstation definition.
- **R/3 Host** - The host name of the R/3 application server.
- **R/3 Gateway Host** - The host name of the R/3 gateway system (if it differs from the R/3 host).
- **R/3 System Name** - The three-digit R/3 system identifier.
- **R/3 Instance** - The two-digit R/3 instance number on R3 Host.
- **R/3 Client Number** - The three-digit R/3 client number for starting jobs.
- **R/3 Userid for RFC-user** - The R/3 user name that TWS will use to release R/3 jobs and connect to the R/3 servers. This user must have

batch administration privileges. When R/3 jobs are released by TWS, the job log output in R/3 is found under this user name.

- **Password for RFC-User** - The password for Userid. The R/3 user should be given a password that does not expire.
- **Short Interval** - The minimum interval in seconds for R/3 job status updates. The default is 30.
- **Long Interval** - The maximum interval in seconds for R/3 job status updates. The default is 300.
- **Audit level** - The Audit level is used to log TWS activities on R/3. A higher level means more messages are logged. Valid values are 0 to 5. For R/3 versions earlier than 3.1G, enter a value of 0.

The following is a sample r3options file entry for an SAP R/3 Extended Agent workstation called sap001:

```
SAP001 SYS01 SYS01 SAP 00 100 R3JOBBER <ENCRYPTED>30 300 0
```

### 7.5.3.2 Modifying the r3options file

The r3options file can be modified by any text editor or by using the following:

- r3setup -maintain on UNIX
- R3OPT.EXE on Windows NT

### 7.5.4 r3batch.opts file reference

Create the r3batch.opts file. This file must reside in the TWShome /methods directory. Make the following entries in the file:

```
LJuser=launch_user  
IFuser=info_user  
JobDef=jobdef
```

The entries are defined as follows:

- **launch\_user** - This is the UNIX user name that executes the r3batch method. This must be a valid UNIX account.
- **info\_user** - This is the UNIX user name that executes the r3batch method to retrieve job information. It must be the same as the LJuser.
- **Jobdef** - Enter r3batch as the job definition type to access SAP-specific job definition windows in TWS. Enter Generic for the standard composer job definition GUI.

The contents of the r3batch.opts file that we used are as follows:

```
LUser=maestro  
IFuser=maestro  
JobDef=r3batch
```

**Note**

If JobDef=r3batch is missing, JS Console will not allow to work with SAP job definitions

## 7.5.5 SAP R/3 configuration

To communicate and control job execution on SAP R/3, Tivoli Workload Scheduler requires you to install correction and transport files on the SAP R/3 database system.

**Note**

These steps require an R/3 BASIS Administrator authority. You also need SAP expertise to complete these steps. Otherwise consult your SAP Administrator.

### 7.5.5.1 Overview of required steps

These procedures add new ABAP/4 function modules to your R/3 system, and several new internal tables as well. No existing R/3 system objects are modified.

Here is an overview of the procedure:

1. Create the authorization profile.
2. Create the TWS RFC user ID.
3. Copy the correction and transport files from the TWS server to the SAP R/3 server.
4. Import correction and transport files into SAP R/3.
5. Verify the installation.

### 7.5.5.2 Creating the authorization profile for SAP R/3 V 4.6B

Before you create an RFC user ID for TWS batch processing, you need to create the profile of the authorizations that the TWS user requires. The authorizations required differ depending on your version of SAP R/3. The SAP-defined authorizations are all found under the Object Class Basis: Administration.

**Note**

It may be necessary to give additional authorizations to the MAESTRO user for some special jobs.

1. Create a profile called `Z_MAESTRO`
2. Insert the correct authorization according to the information in Table 16.

Table 16. Authorizations

Object	Text	Authorization
S_RFC	Authorization check for RFC access	S_RFC_ALL
S_XMI_PROD	Authorization for external management interfaces (XMI)	X_XMI_ADMIN
S_BTCH_ADM	Batch processing: Batch administrator	S_BTCH_ADM
S_BTCH_NAM	Batch processing: Batch user name	S_BTCH_ALL
S_BTCH_JOB	Batch processing: Operations on batch jobs	S_BTCH_ALL
S_XMI_LOG	Internal access authorization for XMI log	S_XMILOG_ADM
S_SPO_DEV	Spool: Device authorization	S_SPO_DEV_AL

### 7.5.5.3 Creating the Tivoli Workload Scheduler RFC user

For TWS to communicate with SAP R/3, you need to create a user ID in R/3 for TWS batch processing. For security reasons, Tivoli suggests using a new user ID rather than an existing one.

1. Create a new RFC user ID.
2. Give this new RFC user ID the following attributes:
  - The user type should be CPIC.
  - A password at least six characters in length; TWS requires this password to start or monitor SAP R/3 jobs. If this password changes in SAP R/3, you must update the `r3options` file with the new password.
  - Assign the appropriate security profiles depending on your version of SAP R/3.



#### 7.5.5.4 Copy the correction and transport files

The setup file loads two correction and transport files into the TWS home directory. You must copy these correction and transport files to the SAP server and import them into the SAP R/3 database.

1. On your R/3 database server, log into your SAP R/3 system as an administrator.
2. Copy the control file and datafile from the TWShome directory to the following directories on your SAP R/3 database server:

```
copy controlfile /usr/sap/trans/cfiles/  
copy datafile /usr/sap/trans/data/
```

The names of the controlfile and datafile vary from release to release. They are usually called K0000xx.tv1 (the control file) and R0000xx.tv1 (the data file). In our case, the names were K000119.tv1 and R000119.tv1.

#### 7.5.5.5 Import ABAP/4 Modules into SAP R/3

This procedure generates, activates, and commits new ABAP/4 function modules to your SAP R/3 system and several new internal tables. No existing R/3 system objects are modified.

1. Change to the following directory:

```
cd /usr/sap/trans/bin
```

2. Add the transport request to the buffer:

```
tp addtobuffer transport sid
```

where *transport* is the transport request and *sid* is your SAP R/3 system ID. The name of the transport is tv1k0000xx.

3. Execute the `tp tst` command to test the import.

```
tp tst transport sid
```

After you have run this command, examine the log files in the `/usr/sap/trans/log` directory for error messages. Warnings of severity level 4 are normal.

If you have errors, check with a person experienced in Correction and Transport, or try using unconditional modes to do the import.

4. Execute the following command to import all the files in the buffer:

```
tp import transport sid
```

This command generates the new ABAP/4 modules and commits them to the SAP R/3 database. They automatically become active.

After you have run this command, examine the log files in the `/usr/sap/trans/log` directory for error messages. Warnings of severity level

4 are normal. If a problem is encountered, use unconditional mode when executing this step:

```
tp import transport sid U126
```

5. When the import has completed, check the log files to verify that the import was successful. The log files are in the /usr/sap/trans/log directory.

#### 7.5.5.6 Troubleshooting the R/3 connection

If you are unable to submit SAP R/3 jobs using TWS after the SAP R/3 configuration, perform the following tests:

1. Make sure you can ping the SAP R/3 system from the TWS system. This will show basic network connectivity.
2. Execute the following telnet command to verify connectivity:

```
telnet systemname 33xx
```

where *systemname* is the system name or IP address of the SAP R/3 server and *xx* is the R/3 instance.

3. Log into the SAP R/3 system as an administrator and verify that the TWS RFC user exists.
4. Create a new Workload Scheduler job to be run under R/3, and select the following options:

```
debug  
trace
```

Using the debug option, additional debug information is included in the job's stdlist file (in Tivoli Workload Scheduler).

Using the trace option on UNIX, a trace file, *dev\_rfc*, is created in the TWS home directory. Using the trace option on Windows NT, a trace file, *rfcxxxxx.xxxxxtrc.file*, is created in the Tivoli Workload Scheduler's home directory. Be sure to delete the trace option from the job after you perform debug procedures. The *dev\_rfc* file can become very large and unmanageable.

5. Verify that an SAP R/3 gateway process is active on the SAP R/3 system to which you are connecting. Verify that a *sapgwNN* entry is present in the TWS system's services file. Note that if YP/NIS (or any other directory service) is managing the services entries, you should verify that the master services file on your Workload Scheduler system is being used rather than the local services file.
6. If you are installing on an AIX system not using U.S. English, note that the U.S. Language Environment must be installed on the TWS workstation and the SAP R/3 database workstation. Otherwise, the error,

BAD-TEXTENV (or a similar error message), may appear in the dev\_rfc trace file, and connections to R/3 will fail.

#### 7.5.5.7 Changing the TWS RFC user ID password

If the password of the Workload Scheduler RFC user ID is modified after initial installation on the SAP R/3 system, the r3options file must be updated with this change. There are two ways to do this:

##### **Option 1**

- For a UNIX Tivoli Workload Scheduler Extended Agent host, log in as root and run the following command:

```
r3setup -maintain
```

- For Windows NT Workload Scheduler Extended Agent host, log in as administrator, and from the Start menu, select the following program: **Start > Maestro > R3options Editor**.

This allows you to specify a new password for the TWS RFC user ID.

##### **Option 2**

1. In UNIX, log in as root to the system where TWS is installed; for Windows NT, log in as an administrator and start a DOS shell on the system where TWS is installed.
2. Generate an encrypted version of the new password. To do this, use the utility command, `enigma`, in TWShome/bin.
3. In a command shell, type:

```
enigma newpass
```

where `newpass` is the new password for the TWS RFC user ID. The `enigma` command prints an encrypted version of the password.

4. Copy this encrypted password into the r3options file. The r3options file is located in the TWShome/methods directory. The file can be edited with any text editor. Be sure to copy the password exactly, preserving upper/lower case and punctuation. The encrypted password will look something like:

```
#TjM-pYm#-z82G-rB
```

If the encrypted password is mistyped, TWS will not be able to start or monitor R/3 batch jobs.

---

## 7.6 Setting up for Tivoli Workload Scheduler 7.0

This chapter provides information on how to do the following:

- Create the Extended Agent for the SAP R/3 workstation in the TWS network
- Create jobs in Workload Scheduler that correspond to jobs already in the SAP R/3 database
- Create jobs in the SAP R/3 database controlled by TWS
- Create job streams that use Extended Agent for SAP R/3 jobs

### 7.6.1 Defining an SAP R/3 Extended Agent workstation

SAP R/3 Extended Agent workstation definitions are required for each instance of R/3 that will execute TWS-controlled jobs.

To define a SAP R/3 Extended Agent workstation, perform the following steps:

1. From the Tivoli Job Scheduling Console, highlight the **TWS engine instance**.
2. From the Selected menu, select the **New Workstation...** command. This opens the Properties -Workstation in the Database window as shown in Figure 97 on page 175.

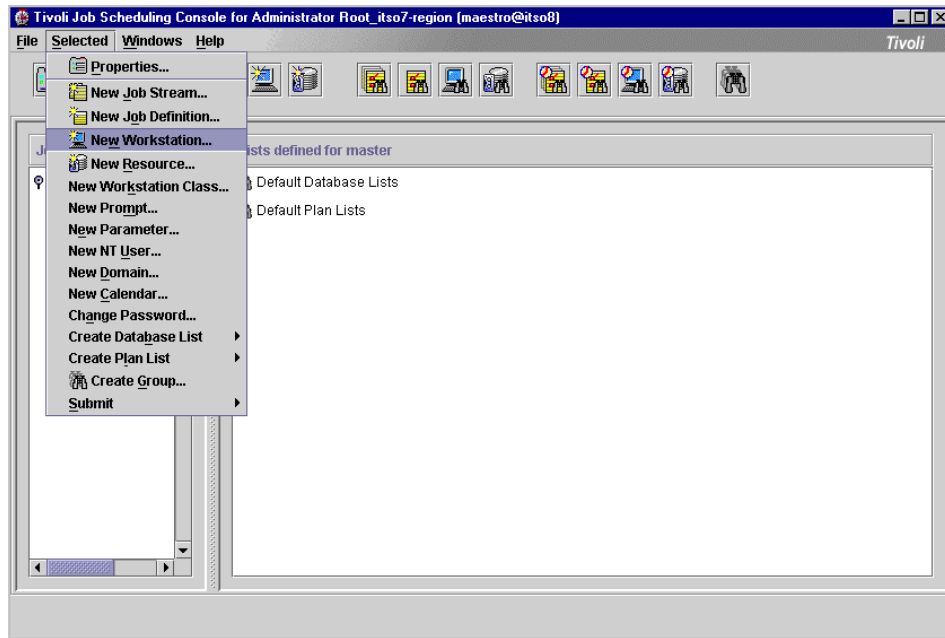


Figure 97. Defining a new SAP R/3 Extended Agent Workstation

3. Fill in the necessary fields for creating a new SAP R/3 Extended Agent workstation as shown in Figure 98 on page 176.
  - In the *name* field, enter the workstation name that corresponds to the entry in the *r3options* file
  - In the *node* field, enter `null`.
  - In the *TCP Port* field, enter any value between 1 and 65535. This field is not valid for an SAP Extended Agent but requires a number greater than zero to be entered.

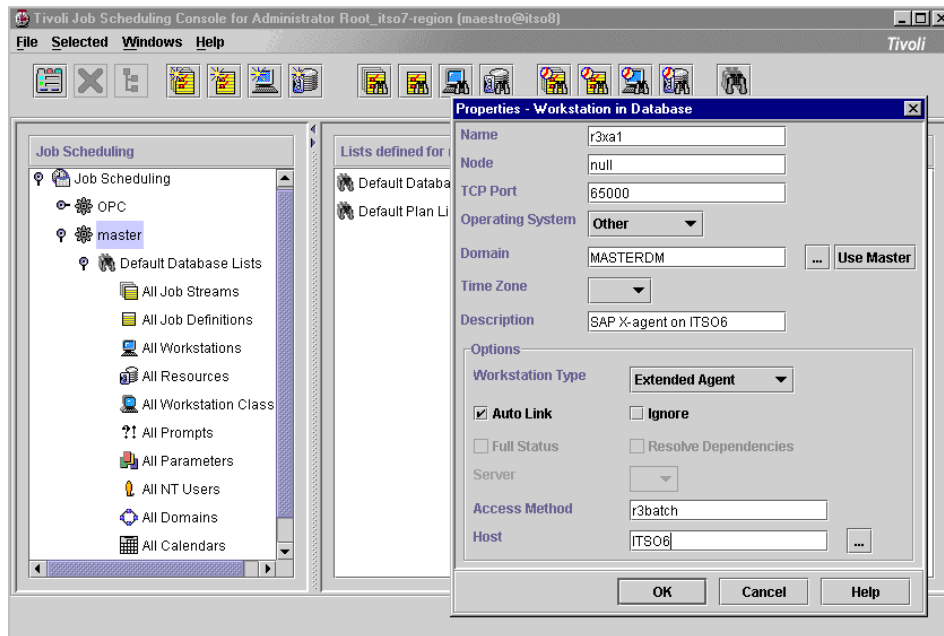


Figure 98. Fill in the necessary fields in the Properties window

4. In the All Workstations window, shown in Figure 99, you can see the newly-defined SAP R/3 Extended Agent Workstation.

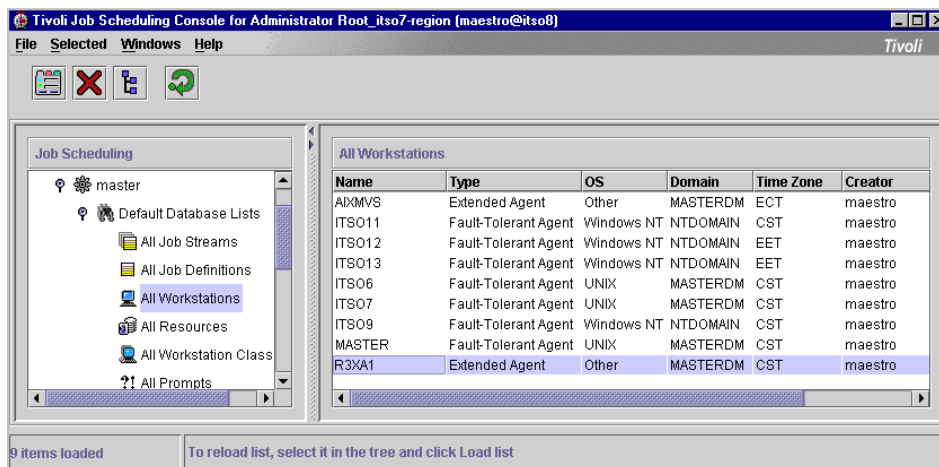


Figure 99. Displaying the newly-defined SAP R/3 Extended Agent Workstation

Remember that only *after having run the Jnextday procedure*, will this workstation be available for scheduling.

## 7.6.2 Defining SAP R/3 Extended Agent jobs

To control job execution on the SAP R/3 workstation from Tivoli Workload Scheduler, you must do the following:

- Define jobs in SAP R/3 that you want to run under TWS control. You can define these jobs using standard SAP R/3 tools or the Tivoli Job Scheduling Console.
- Define jobs in Tivoli Workload Scheduler that correspond to the jobs in SAP R/3. The TWS job definitions are used in scheduling and defining dependencies, but the SAP R/3 jobs are actually executed.

### 7.6.2.1 Defining a job in Tivoli Workload Scheduler

The following procedure creates a new job definition in the TWS database for a job that already exists in the SAP R/3 database:

1. From the Job Scheduling Console, highlight **TWS engine instance**.
2. From the Selected menu, select the **New Job Definition...** command as shown in Figure 100.

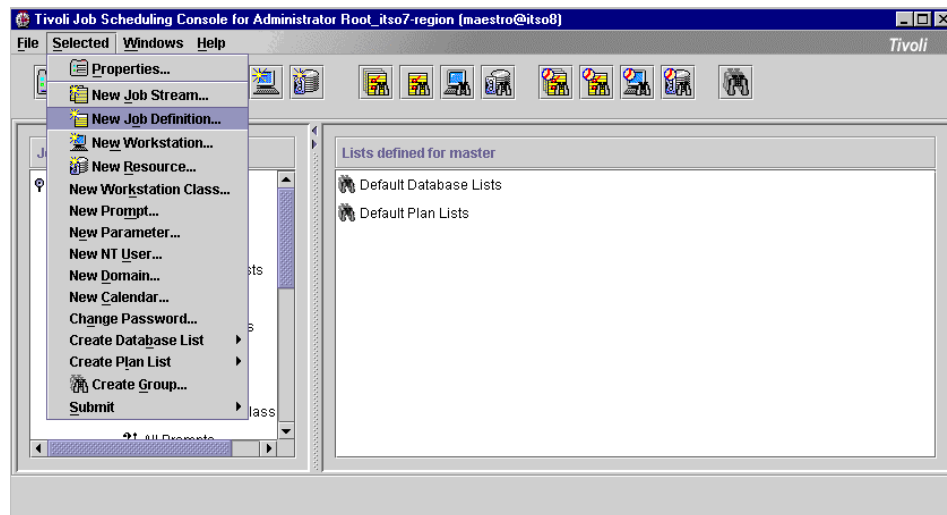


Figure 100. Defining a new job

The Select a Task Type window is displayed as shown in Figure 101 on page 178.

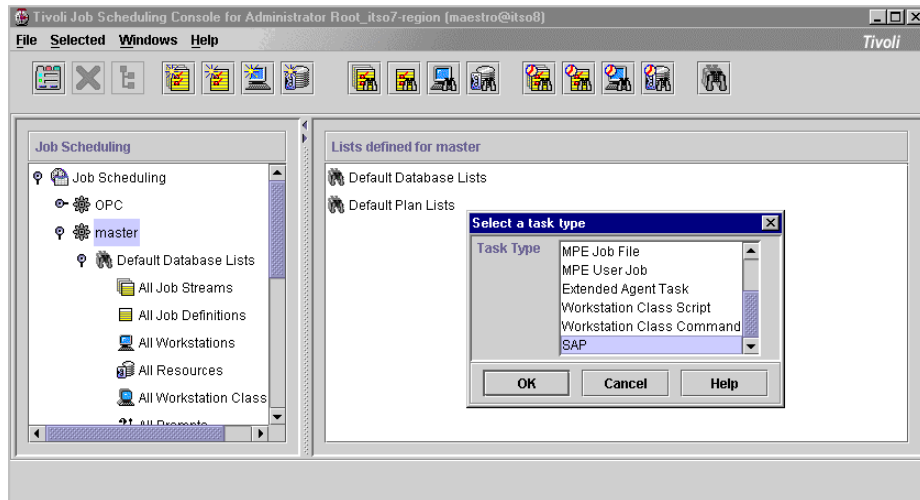


Figure 101. Selecting the Task type

3. From the Task Type scroll down menu, select the **SAP** job type and click **OK**. The Properties - Job Definition window, shown in Figure 102 on page 179, is displayed.
  - a. In the Name field, enter a Tivoli Workload Scheduler name for the SAP R/3 job. The name can contain eight alphanumeric characters or 40 alphanumeric characters if you are using the expanded database option. The job name must start with a letter.
  - b. In the Workstation field, use the ellipse (...) button to open a find window to search and select an available workstation.
  - c. In the Description field, enter a description for the job. This field is an optional text description of the job and can consist of up to 64 alphanumeric characters.
  - d. In the Login field, enter the TWS login ID that is used to execute the job. Click the **Add Parameter...** button to add any predefined parameters to the login ID.
  - e. In the *Recovery Options* fields, specify any recovery options for the SAP R/3 job. Refer to the Tivoli Workload Scheduler User's Guide for information about recovery options.



**Properties - Job Definition**

**General**

Task Type: SAP

Name: SAP-testjob-20

Workstation: R3XA1

Description: Sap-testjob20-description

Login: maestro

Add Parameter...

Recovery Options

Action: ☒ Stop ☐ Continue ☐ Rerun

Message:

Job:

Workstation:

OK Cancel Help

Figure 102. Job Definition (general)

4. Click the **Task** tab. Next to the Job Name field, click the ellipse (...) button. This opens the SAP Pick List window as shown in Figure 103 on page 180.

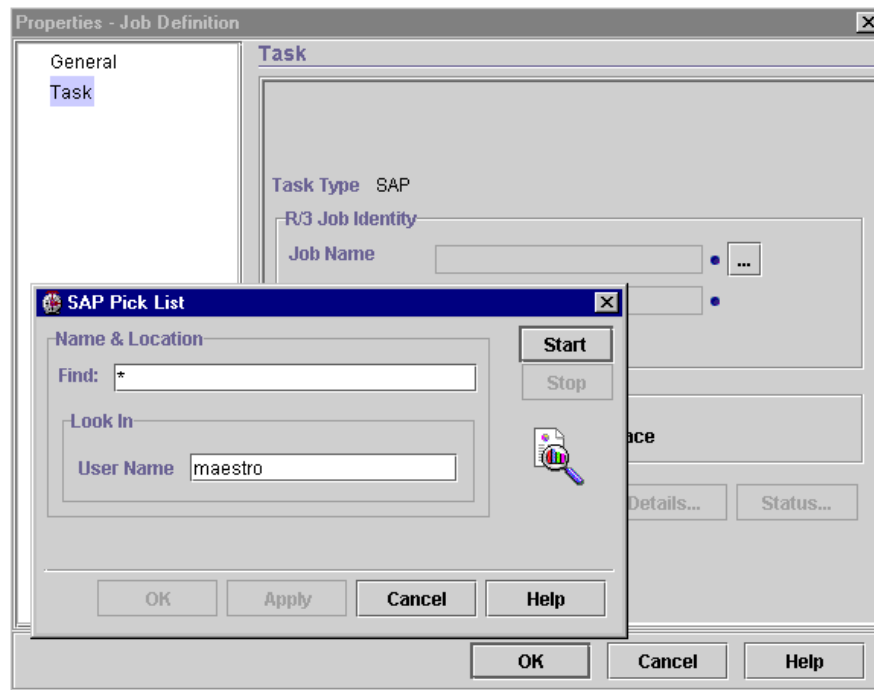


Figure 103. Job Definition Task, Select SAP Pick List

5. Use the SAP Pick list to find and select an SAP job, and click the **OK** button. The job information will be propagated to the Task tab. These can be seen in Figure 104 and Figure 105 on page 181.

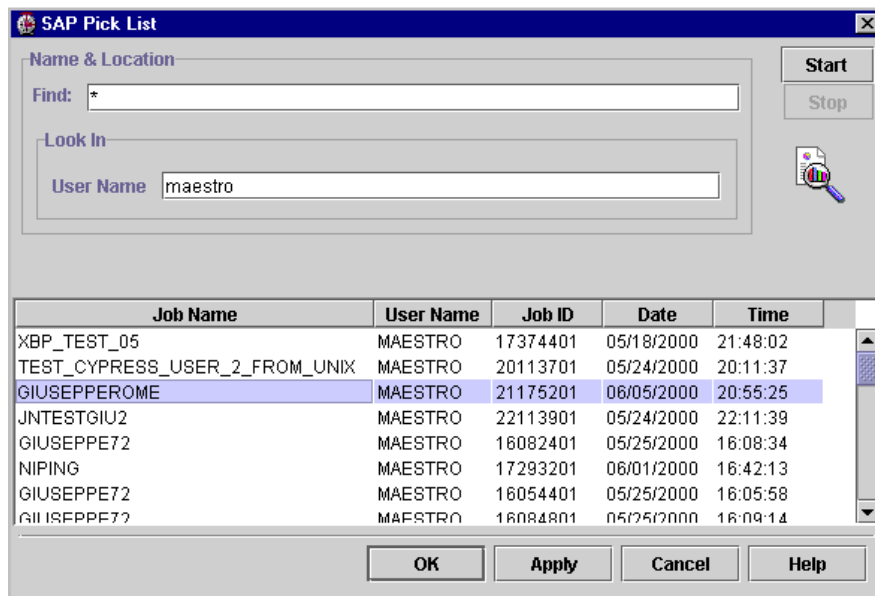


Figure 104. Showing all SAP jobs on SAP R/3 application server for user maestro

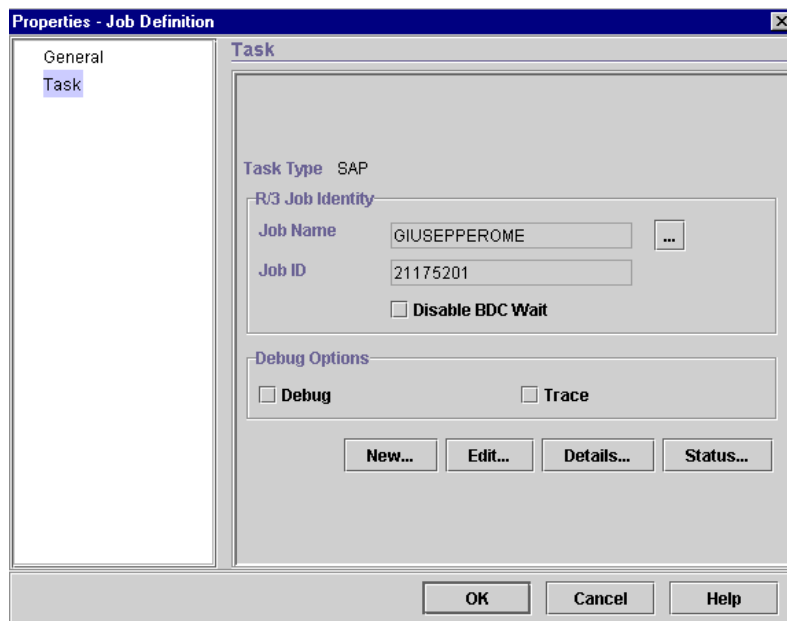


Figure 105. Propagated Information from SAP Pick List in Task tab

- Click the **OK** button in Figure 105 on page 181 to save the job definition in the TWS database. You can see the newly-defined job in Figure 106.

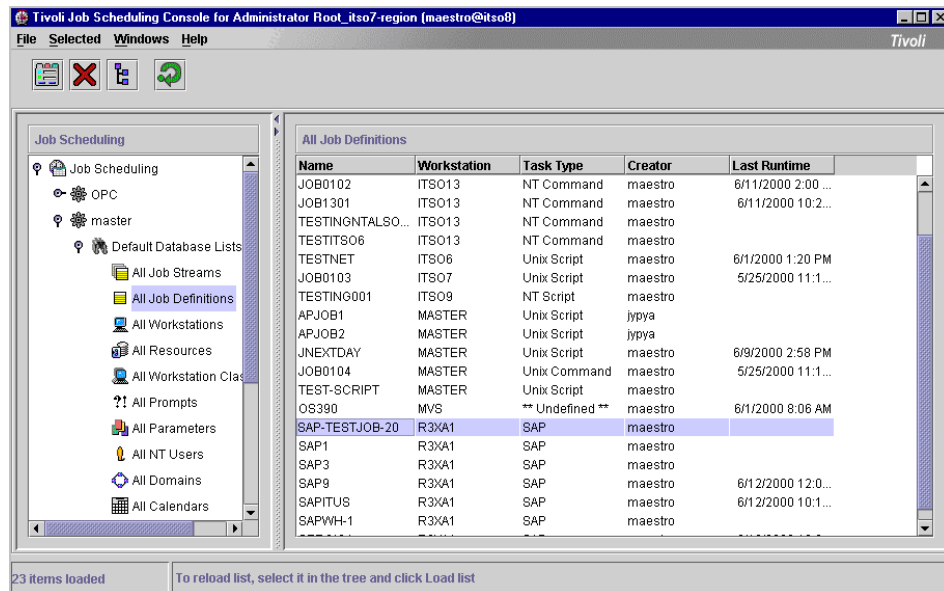


Figure 106. Displaying the new job in Job Definitions List

#### 7.6.2.2 Define a job in SAP R/3

To create SAP R/3 jobs that are controlled by Tivoli Workload Scheduler, use either of the following:

- The standard R/3 tools for defining jobs
- Tivoli Job Scheduling Console

Refer to SAP R/3 documentation for information on creating jobs with R/3 tools. When creating jobs on SAP R/3, do not assign a start time and do not specify any dependencies; these are specified and controlled in TWS. The job is identified in TWS by its R/3 job name and job ID. The same job can be scheduled to run repeatedly by TWS without having to redefine it each time in R/3.

#### Note

For R/3 jobs controlled by TWS, the job log output in R/3 is found under the user name defined in the r3user option of the r3options file.

### 7.6.2.3 Defining a job in TWS and SAP R/3

This procedure creates a new job definition in the Tivoli Workload Scheduler database *and* a new job definition in the SAP R/3 database.

1. From the Job Scheduling Console, highlight the **TWS engine instance**.
2. From the Selected menu, click the **New Job Definition...** command as shown in Figure 107.

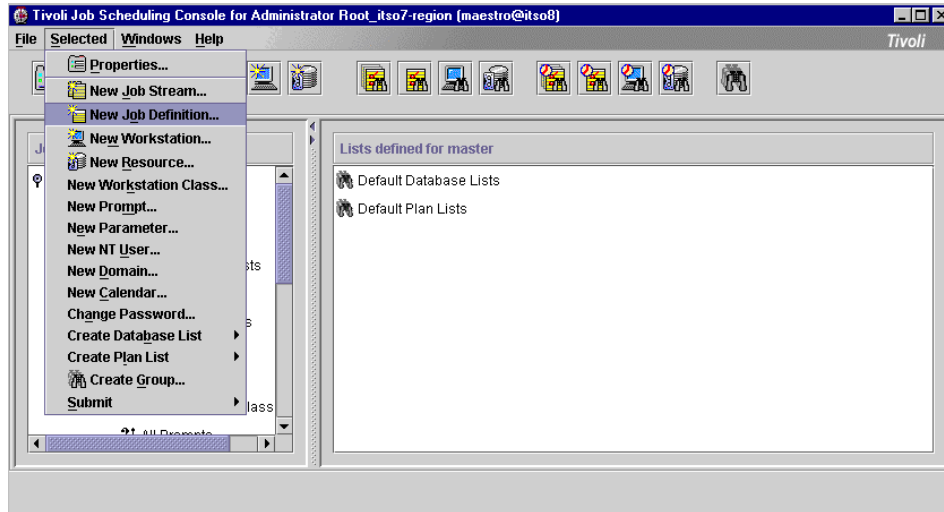


Figure 107. Defining a new Job

The Select a Task Type window is displayed as shown in Figure 108 on page 184.

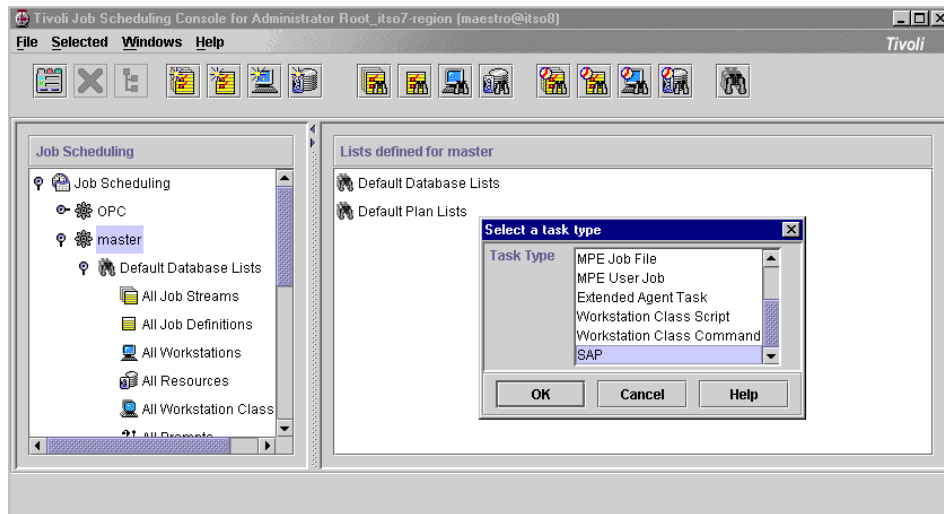


Figure 108. Selecting the task type

3. From the task type scroll-down menu, select the **SAP** job type and click **OK**. The Properties - Job Definition window, shown in Figure 109 on page 185, is displayed.
  - a. In the Name field, enter a Tivoli Workload Scheduler name for the SAP R/3 job. The name can contain eight alphanumeric characters (or 40 alphanumeric characters if you are using the expanded database option). The job name must start with a letter.
  - b. In the Workstation field, use the ellipse (...) button to open a find window to search and select an available workstation.
  - c. In the Description field, enter a description for the job. This field is an optional text description of the job and can be up to 64 alphanumeric characters long.
  - d. In the Login field, enter the TWS login ID that is used to execute the job. Click the **Add Parameter...** button to add any predefined parameters to the login ID.
  - e. In the Recovery Options fields, specify any recovery options for the SAP R/3 job. Refer to the *Maestro UNIX User's Guide V6.0*, GC31-5136, for information about recovery options.

**Properties - Job Definition**

**General**

Task Type: SAP

Name: SAP-testjob-30

Workstation: R3XA1

Description: SAP-testjob-30 description

Login: maestro

Add Parameter...

Recovery Options

Action: ☒ Stop ☐ Continue ☐ Rerun

Message:

Job:

Workstation:

OK Cancel Help

Figure 109. Job Definition (general)

- Click the **Task** tab from the window shown in Figure 110 on page 186.

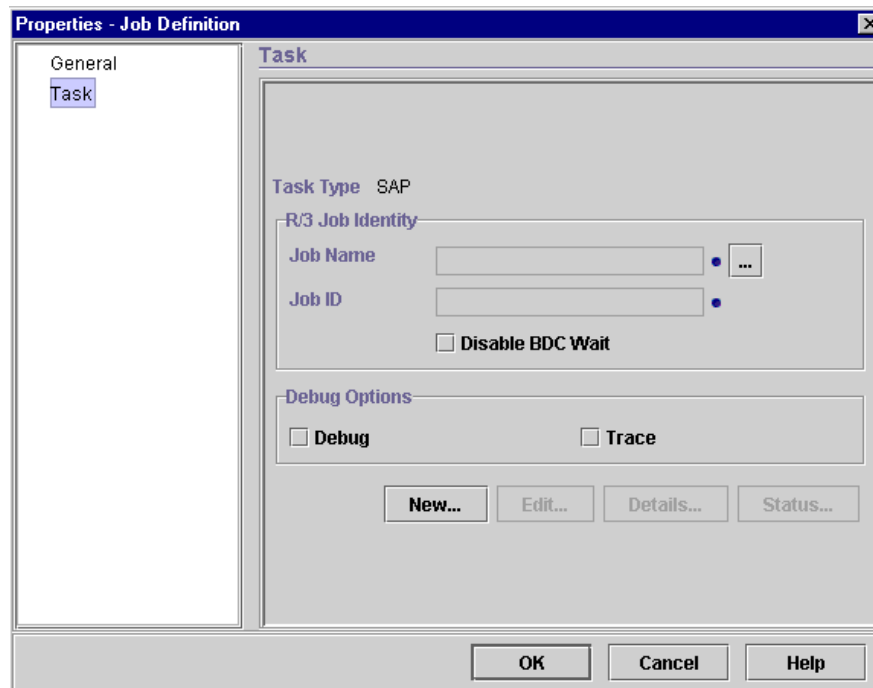


Figure 110. Job Definition (task)

5. Click the **New...** button. This opens the SAP Job Definition window. On the SAP Job tab, enter information about the SAP job you are creating:
  - a. In the Job Name field, enter an SAP job name.
  - b. In the Target Host field, enter the name of the target workstation where this job is executed. In our case, since we executed just an external dummy program (sleep), this was left empty. It will be specified in the next step.



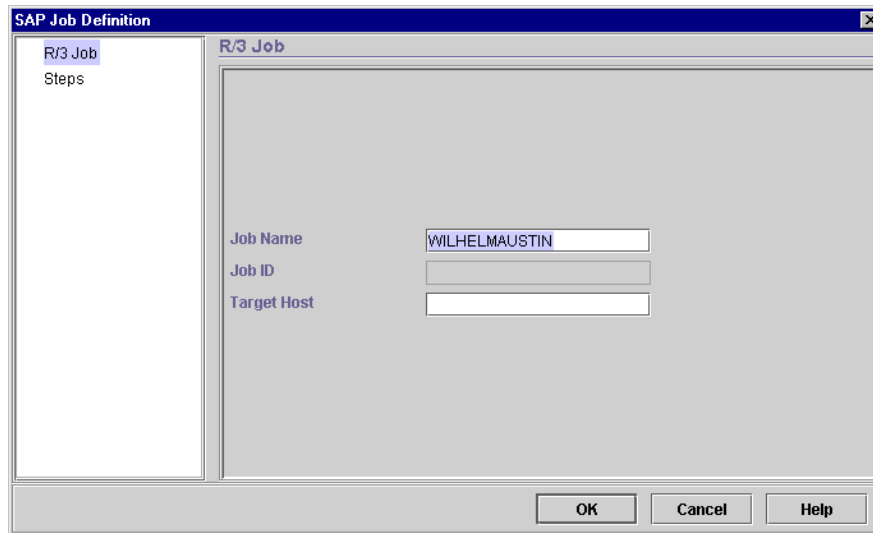


Figure 111. SAP R/3 Job Definition (general R/3 Job)

6. On the **Steps** tab, you must add an ABAP or External program to the SAP job:
  - a. In the Type field, select **ABAP** or **External step** from the drop down menu.
  - b. Press the **Add step** button (the green plus sign).
  - c. In the Name field, enter an ABAP name or a fully-qualified path and filename for an External program.
  - d. In the User field, enter the name of the SAP user who executes this step.
  - e. In the Var/Parm field, enter a variant name or a parameter, if necessary. Variants are used with ABAPs, and parameters are used with External programs, but both are optional. Not all ABAPs require variants, and not all External programs require parameters.
  - f. In the Target Host field, enter the SAP workstation where this step executes. Target Hosts are only required for External programs.
  - g. Select any print parameters (for ABAPs) or control flags (for External Programs). Refer to the corresponding SAP User Manuals for more information.

These settings are shown in Figure 112 on page 188.

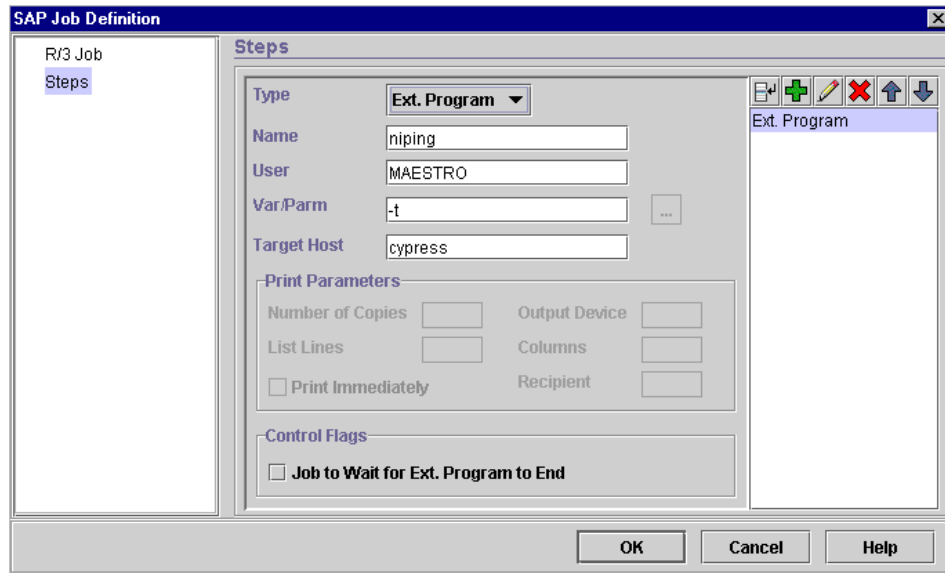


Figure 112. Defining an SAP R/3 job (steps tab)

7. When you have completed creating the ABAP or External program steps, click the **OK** button. The SAP job definition is saved to the R/3 database, and the window is closed. If the SAP job was successfully saved to the R/3 database, the Task tab for the Properties - Job Definition window should display an SAP job ID.

This is shown in Figure 113 on page 189.

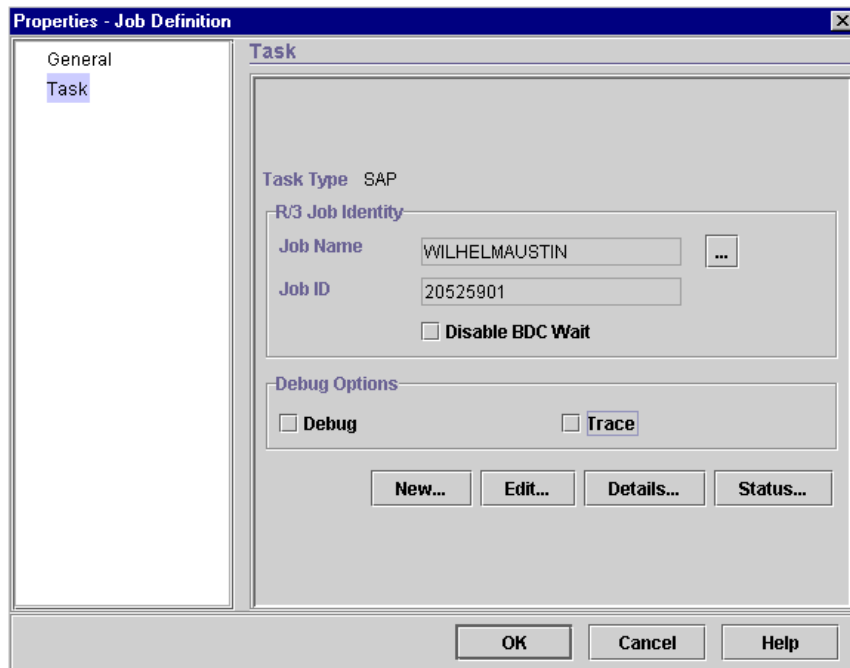


Figure 113. Job Definition Window (Job-ID after saving to R/3 database)

8. Click the **OK** button in the Properties - Job Definition window, shown in Figure 113, to save the job definition to the Tivoli Workload Scheduler database.

### 7.6.3 Defining SAP R/3 Extended Agent job streams

SAP R/3 job streams are scheduled in the same manner as other TWS job streams and can include dependencies, run cycles, time restrictions, and other controls, but not Files dependencies. Before creating job streams, you must define the SAP R/3 jobs to the Tivoli Workload Scheduler database.

To define a job stream using SAP R/3 jobs:

1. From the Tivoli Job Scheduling Console, select **TWS engine instance**.
2. From the Selected menu, click the **New Job Stream...** as shown in Figure 114 on page 190.

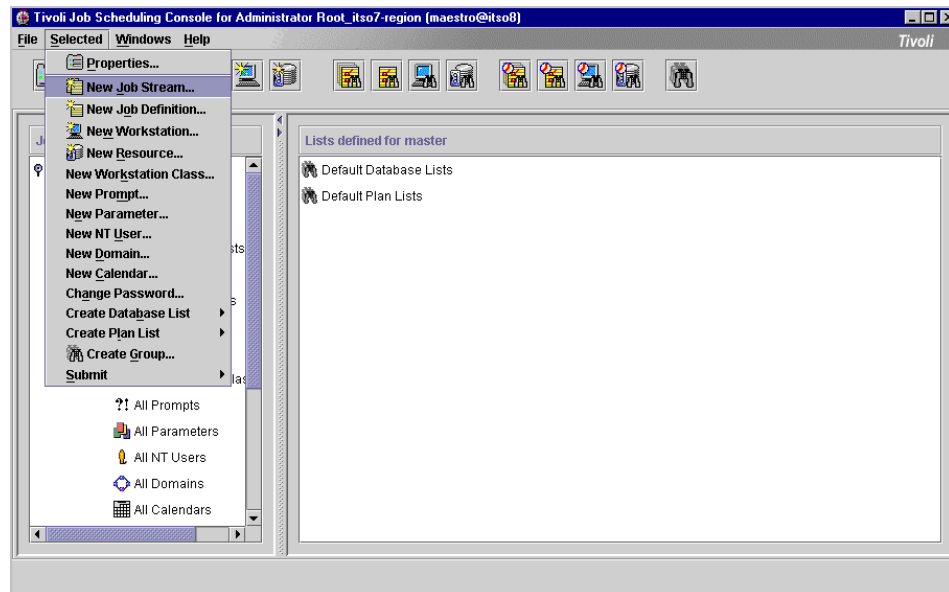


Figure 114. Defining a Job Stream

3. The Job Stream Editor and Properties - Job Stream windows are opened. We will first show the Properties - Job Stream window because the Job stream Editor contains data only after the first job within the job stream is defined.
4. In the Name field of the Properties - Job Stream window, enter a name for the job stream. The name can contain eight alphanumeric characters, or, if you are using the expanded database option, it can contain 16 alphanumeric characters. The job stream name must start with a letter.
5. In the Workstation field, enter the name of the SAP R/3 Extended Agent workstation that executes this job stream, or use the ellipse (...) button to search for an SAP workstation.
6. Click **OK** to close the Properties - Job Stream window after you have completed all the desired fields. All fields and tabs on the Properties - Job Stream window, other than Name and Workstation, are optional.

Depending on your general setup, you usually define a certain priority for a job stream as well as for a job to put it into a certain execution window.

These settings can be seen in Figure 115 on page 191.

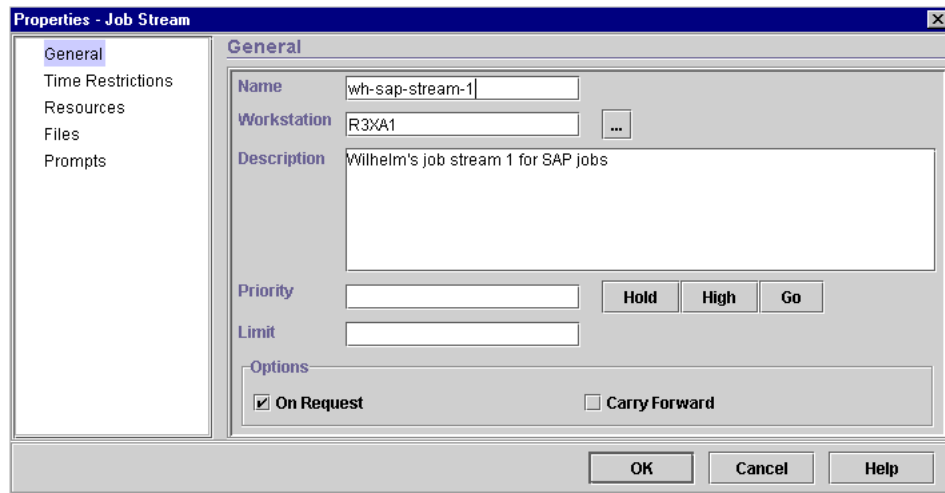


Figure 115. Defining an SAP R/3 Job Stream

7. Add the SAP R/3 jobs that you want to be part of this job stream:
  - a. In the Job Stream Editor, select the **Actions > Add Job > Job Definition** menu. After double-clicking with the left mouse button in the white area of the window, the Properties - Job Definition window, shown in Figure 116, is displayed.

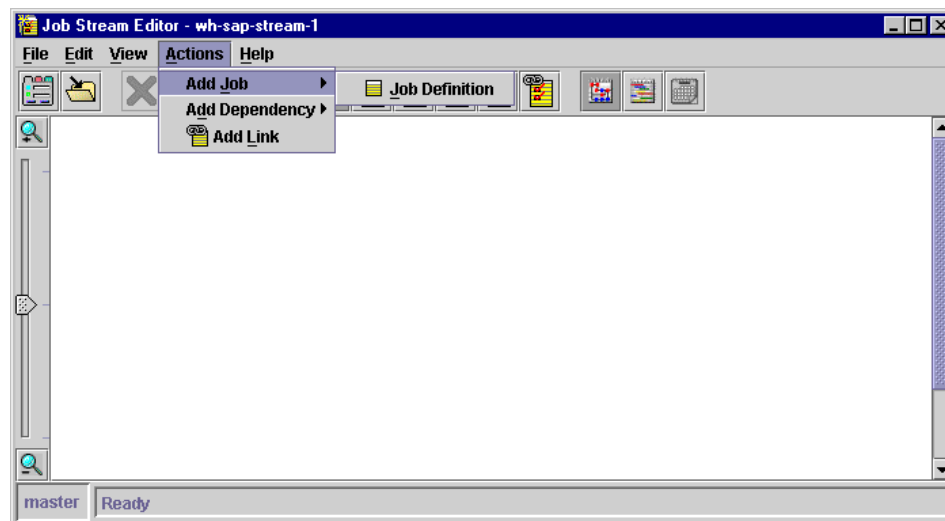


Figure 116. Adding jobs into an empty job stream

- b. In the Name field, enter the Tivoli Workload Scheduler name of the SAP R/3 job you want to add to the job stream.
  - c. Repeat these steps to add all the desired SAP R/3 jobs.
- This is shown in Figure 117.

The screenshot shows a Windows-style dialog box titled "Properties - Job". On the left is a vertical list of tabs: "General", "Time Restrictions", "Resources", "Files", and "Prompts". The "General" tab is selected. The main area of the dialog has the following fields and controls:

- Name:** A text field containing "SAP-TESTJOB-30" followed by a small button with three dots.
- Workstation Name:** A text field containing "R3XA1".
- Priority:** A text field that is currently empty.
- Buttons:** To the right of the Priority field are three buttons labeled "Hold", "High", and "Go".
- Description:** A large, empty rectangular text area.
- Requires Confirmation:** A checkbox located at the bottom left of the main area, which is currently unchecked.
- Footer Buttons:** At the bottom right of the dialog are three buttons: "OK", "Cancel", and "Help".

Figure 117. Entering jobs into a job stream

- 8. The defined jobs show up as icons in the Job Stream Editor as shown in Figure 118 on page 193.

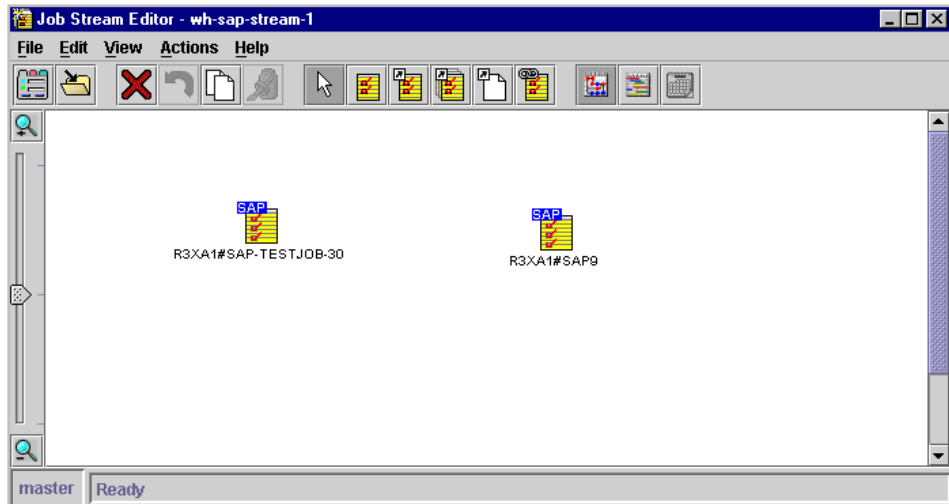


Figure 118. Jobs (represented by icons) within a job stream

9. If you add two or more jobs, you can specify dependencies between these jobs, such as adding links or specifying external job / external job stream dependencies. This is shown in Figure 119 and Figure 120 on page 194.

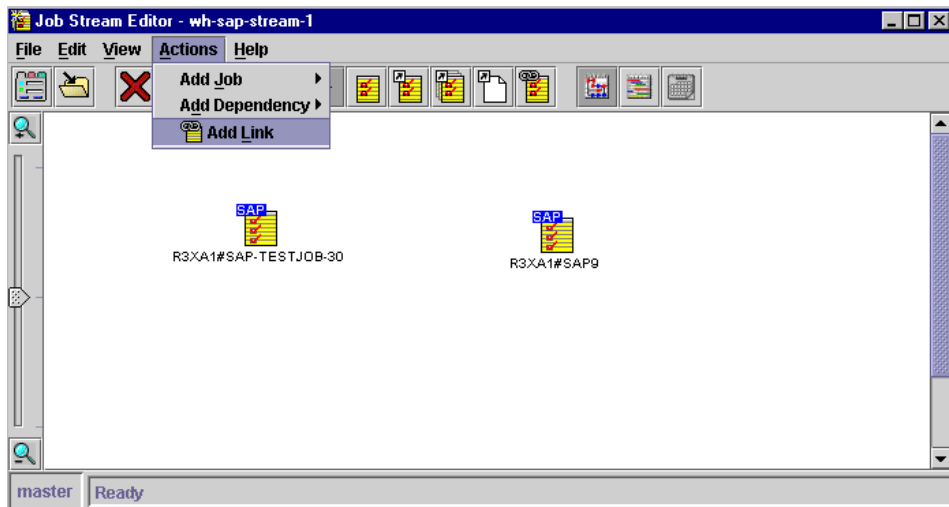


Figure 119. Adding dependencies between jobs

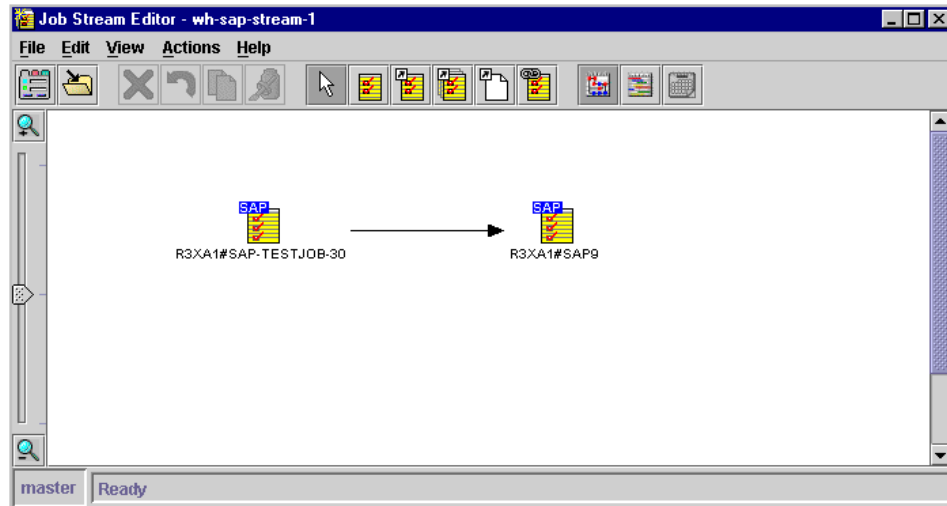


Figure 120. Showing job dependencies

10. Create run cycles that determine the days that this job stream will execute.
11. Save the job stream by clicking **Save** from the File menu.

## 7.7 Troubleshooting SAP R/3 Extended Agent problems

This section will give you some ideas for solving SAP R/3 Extended Agent problems you may encounter in your environment.

We can make the following assumptions when working with SAP R/3 Extended Agent problems:

- It is, usually, a setup problem.
- It is, usually, a new install or upgrade situation.
- Upgrading to the latest version is not a problem for 6.0, 6.1, or 7.0.

### 7.7.1 Critical files

The three files that make up a complete SAP Extended Agent installation are:

- /TWShome/methods/r3options
- /TWShome/methods/r3batch.opts (this file must be created manually)
- /TWShome/methods/r3batch



#### 7.7.1.1 r3options

It is found on the Host FTA as defined in the X-agent workstation definition. This file contains the connection parameters for each defined X-agent to make an RFC connection to an R3 Host FTA. The r3batch method reads down the r3options file until it finds a matching Xagent name and uses the parameters found on that line.

#### 7.7.1.2 r3batch.opts

This file must be created according to the manual and stored on the Host FTA. The file defines the users authorized to run the r3batch method and retrieve job information from the R3 Host FTA. In addition, it is also required to invoke the new r3batch job definition window.

#### 7.7.1.3 r3batch

This is the TWS method that runs and passes job information between TWS and its RFC connection to the R3 Host FTA.

### 7.7.2 Information necessary for troubleshooting

The following information is crucial when troubleshooting SAP R/3 Extended Agent problems. This is also the same information that you will be asked to send if you report your problem to Tivoli Customer Support.

#### Note

You may want to consult the SAP Basis Administrator if you do not know some of the information that follows

1. What version of SAP X-agent are you using?

The following command will tell you what version is installed:

```
r3batch -v
```

2. What version of SAP R3 system are you using?

There are two pieces to this: the SAP Application version and the SAP Kernel version. For example, the SAP Application might be 3.1f, and the SAP Kernel version might be 3.1g.

Although it is very common to have different versions, r3batch is passed the SAP Kernel version and assumes that the SAP Application version is the same. This can cause a potential mismatch between r3batch and the R/3 system when the SAP Kernel version may have modules that are not included in the lower SAP Application version (see XMI errors).

### 3. How is your TWS network set up within the R/3 System?

- Usually, the r3options file will tell enough about the setup; so, make sure to get the r3options file. Note that if you make any changes to the r3options file, it will take place immediately. You do not need to run Jnextday or restart TWS. The r3batch method reads the r3options file every time it runs.
- At a minimum, only one hosting FTA is required for any given number of X-agents. Most customers choose to host SAP X-agents on multiple FTAs. This prevents losing all the SAP X-agents with one single Host FTA failure.
- In networks with multiple X-agent Host FTAs, there must exist an r3options file on each Host FTA. In practice, it is advisable to keep a master copy of an r3options file on the TWS Master that includes all the Xagents in the network. It does not matter if the r3options file on one Host FTA has an r3options entry for an Xagent that is hosted by a different FTA. This provides several advantages and less confusion when troubleshooting or making changes to X-agents. For example, moving an X-agent to another Host FTA is simply one change to the Host field in the Xagent workstation definition on the TWS Master.
- It is not necessary every time, but a good idea to get workstation definitions from the Master along with output from the `conman sc;I` command.

### 4. If you are past setup, is there a problem with the connection between X-agent and R/3 System?

The following command will give useful information (run from methods directory on Host FTA):

```
./r3batch -t PL -c xagentname -l r3loginname -j * -- "-debug -trace" > r3debug
```

where:

-t is a flag for 1 of 3 tasks that r3batch can call. PL refers to the picklist task.

-c is a flag for any of the XAgent names as defined in the Maestro CPU database.

-l is a flag for the r3login name. The login setup on the R3 System for TWS.

-j is a flag for the job name. The " \* " means any job (you may need to add double quotes on some systems).

- - (dash dash) is a delimiter.

- debug creates the information sent to the r3debug file.
  - trace creates a dev\_rfc file (optional).
  - Redirecting to a file, called r3debug, is optional because output is piped to the display by default. If the problem is still not clear, you may refer to the r3debug file and the dev\_rfc file that are created as a result of the above command.
  - The -trace option will append to an existing dev\_rfc file if present and can grow quickly to a large file. Therefore, the trace flag should never be left within a documented recurring job.
5. Check that the user listed in the r3batch.opts file is a valid UNIX account on the Hosting FTA, preferably the same user that was used to install TWS. Both LJuser and the IFuser must be the same.
  6. Check that the jobs that TWS runs on the R/3 system are in a scheduled state with no start time. TWS simply creates a new copy of the job and runs the copy. While this bypasses having to define a job every time, it is not the same as creating a job in R/3 (sm36).
  7. If the jobs are actually starting on the R/3 side, the system log (sm21) and RFC log (rz15) could be useful. Ask the Basis Administrator to get a copy of the logs for the specific jobs.

### 7.7.3 Common problems and solutions

If you follow the steps that are listed in Section 7.7.2, “Information necessary for troubleshooting” on page 195, you will have almost everything you could possibly need to resolve SAP R/3 Extended Agent problems. However, in the following section, we will give you some common *symptom - solution* scenarios:

#### **XMI errors**

XMI errors are usually due to incorrect XMI authorization setup on the R/3 system side. The R/3 Basis Administrator must be involved to help fix this. The correct authorizations for all the SAP versions we support are in the SAP Xagent release notes. Giving the SAP\_ALL authorization to the user is a quick way to test if the problem is strictly due to authorizations.

Mixed versions of the SAP Application and Kernel can be a problem for r3batch to interpret. Early SAP Application versions did not have XMI Function Modules until Version 3.1i and later. When r3batch queries the R3 System for its version it is passed the Kernel version. If the Kernel version is 3.1i or later, r3batch will assume that the SAP Application is also, in which case, r3batch will make calls to non-existent functions in the SAP Application

environment. The most common problem is a call to the XMI\_LOGON function module; so, if the SAP Application is less than 3.1i, it will not be able to negotiate with the XMI logon functions. The solution for r3batch 3.2.x versions is to install r3batch Version 3.2.2 and make sure the SAP Application version is added to the end of the line in the r3options file. This software "switch" is not available in r3batch Version 3.2.1. For more information, see the Release Notes included for r3batch Version 3.2.2. Since r3batch Version 3.2.3 was intended for SAP Versions 4.x, this has not been an issue with any of the 3.2.3.x revisions. The latest version, r3batch Version 4.0, has not had this issue either.

***Printing customizations are not being retained in an R3 job***

You may be using r3batch Version 3.2.1. This version does not preserve printing options. You should upgrade to r3batch Version 3.2.2. Versions 3.2.3.x have the same problem as 3.2.1; however, r3batch 4.0 fixes this problem and allows printer configurations in the TWS 7.0 R/3 job definition GUI.

***Not able to see the new r3batch job definition window***

You must be running SAP X-agent Version 3.X to utilize the new functionality in the r3batch job definition window.

See if you have loaded an updated gcomposer patch for SAP X-agents (only available for Maestro Versions 5.2 and 6.0). Maestro V 6.1 has the updated gcomposer. Currently, the only option as of now is to upgrade to 6.1 or TWS 7.0. The patch for 5.2 is no longer available.

***My job is not running with the correct R/3 Job Class.***

SAP Certification mandates not to change the Job Class from what it was originally in R/3 when the job was created; therefore, the Job Class cannot be set or changed in gcomposer. Job Class is a display-only field in the 6.x gcomposer r3batch job definition window, and any changes attempted in the GUI are ignored.

#### **Note**

The TWS Extended Agent works with SAP only through RFC (remote function calls) commands. With SAP Versions 3.1f to 4.0b, the TWS r3batch method is required to use the SAP SXMI\_XBP\_\* RFC commands for certification. With SAP Versions 4.5 to 4.6b, the TWS r3batch method is required to use the BAPI\_XBP\_X\* RFC commands, again for certification with SAP. While using the SAP RFC commands ensures certification and stability, they limit the flexibility and features available for the r3batch method. For example, when the r3batch method tries to run a class A job in R3, it makes a copy of a job already in a scheduled state and runs the copy. The method uses the RFC command that copies the job; however, the SAP RFC interface defaults to class C. There is no provision for the RFC command to copy a class A job, nor for the r3batch to run an R3 job directly outside of the SAP BAPI\_XBP interface. The TWS r3batch method runs copies of jobs in a scheduled state to eliminate redefining jobs in R3 every time a job needs to be run. This makes running jobs multiple times through TWS very quick and easy. If TWS used its own ABAPs to do the job copy, we would lose certification and the reliability that our code would always work with changes that occur in SAP. The best solution is to make the SAP BAPI\_XBP interface more flexible. For instance, allow class A jobs to be copied via the RFC interface. As it stands right now, a copy of a job via the RFC interface is not the same as creating a job via sm36.

#### ***Cannot connect to R/3 system after upgrading Host FTA Operating System***

The r3batch setup adds default r3gateway information to the /etc/services file. The upgrade created a new /etc/services file. Just replace the info from the old /etc/services file.

#### ***Codepage errors***

Errors of this type pertain to different language support, typically Japanese, Chinese, and Korean. These problems require the creation of codepage conversion tables to convert instructions from one language to another and back to the original language. For example, 1100 (English) to 6300 (Chinese), and 6300 (Chinese) back to 1100 (English).

To create a codepage conversion table, perform the following steps:

1. Log in R/3, user SAP\*. client 000.
2. Call SM59 menu option RFC -> Generate "conv. tab" or start the report "RSRFCPUT" using SE38. Include the following parameters:

Source code page: "1100"

Target code page: "8500"

Path: Save the file in the "\$HOME/codepage/" directory

**Note**

The path must end with a path identifier (UNIX = "/" ; NT = "\").

This generates a *<Source-code-page><Target-code-page>.CDP* file, such as 11008500.CDP.

In addition to 11008500.CDP, generate a code page for each of the following:

85001100.CDP

01008500.CDP

85000100.CDP

In the .CDP file format, each line contains a number in Hex format. For example, during a conversion in which all characters are transferred identically, each line contains the line number in Hex format as shown in the following:

0x00 (Where "Character 0x00=Line 0" is changed to character "0x00")

0x01

0x02

....

0xFF

3. Modify the "\$HOME/.cshrc" file and add the following line:

```
set PATH_TO_CODEPAGE $HOME/codepage/
```

After setting the PATH\_TO\_CODEPAGE to the directory name where the codepage conversion files (\*.CDP) are located, the R/3 instance must be restarted. An environment variable can also be added in the r3options file for r3batch Versions 3.2.x as follows:

The first line in the r3options file should be:

```
set environment PATH_TO_CODE_PAGE $HOME/codepage/
```

**Note**

There should be a space at the end of the line.

---

## 7.8 Summary

In this chapter, we covered the Extended Agent concept, which is, basically, used for extending TWS scheduling capabilities to foreign platforms and applications.

As a sample X-agent, we discussed the implementation of Tivoli Workload Scheduling Agent for SAP R/3. We covered the installation and gave examples of scheduling SAP R/3 jobs using the Tivoli Workload Scheduling Agent for SAP R/3. We also discussed some troubleshooting tips and techniques.

Using the Tivoli Workload Scheduling Agent for SAP R/3, you will have the benefits of eliminating the need in R/3 to repeatedly define jobs, managing jobs that have dependencies with other R/3 instances or other systems/applications, and being able to define recovery jobs in case your R/3 job fails.





---

## Chapter 8. Enterprise scheduling scenarios

This chapter describes four enterprise scheduling problems. Some were derived from customer questions, and we feel that our solutions offer examples that can be utilized in many similar situations. We re-created these scenarios on our system, which consists of OPC running on an OS/390 machine in Mainz, Germany, TWS running work on AIX and NT machines in Austin, Texas, and an AIX machine running SAP/3 in Raleigh, North Carolina. Our system is more fully described in Section 8.1.2, “Environments and connections” on page 205.

---

### 8.1 Detailed description of our environment

In this section, we will describe the environment we used for the scenarios.

#### 8.1.1 Systems used

First, we will detail the different systems used.

##### 8.1.1.1 AIX-systems

All three AIX-systems used here in Austin are RS6000 43P model 150s with 256 MB of Memory, an 8.6 GB hard disk, and at least 256 MB of swap space. We were running the standard version of AIX 4.3.3, and the only modification we made was to increase the maximum number of processes allowed per user to 512 on itso7 and itso8. We have a TCP/IP connection to the AIX-system, cypress.rtp.lab, running SAP R/3 V 4.6B and located in Raleigh, North Carolina, in order to access it via SAPGUI and TWS Extended Agent for SAP.

### Note

For running the Job Scheduling Console only, you just need one workstation running the Tivoli Framework(TMR Server), Job Scheduling Services, and the appropriate Connectors for TWS and OPC. In addition, if the system running TWS Master is different from the TMR Server, you have to make the TWS Master a Managed Node and install the Job Scheduling Services and the TWS connector onto it as well.

However, if you intend to use the Tivoli Plus Module for TWS, you will have to install additional products on the TMR Server running the Tivoli Framework, such as TEC-Server, TEC-Console, Distributed Monitoring, Software Distribution, additional Tivoli products on the TWS Master, and TMA endpoints on all FTAs you want to be monitored. You also need access to a DB instance.

In our case, we have installed all the components required to be able to use the TWS Plus module.

- itso8:  
TWS MASTER, running TWS Engine V 7.0, TWS Extended Agent for OS/390  
Tivoli Framework 3.6.2 as Managed node, Job Scheduling Services, TWS connector, TMA Endpoint
- itso7:  
FTA, running TWS Engine V 7.0  
Tivoli Management Region SERVER, running Framework 3.6.2, TEC Server 3.6.2, TEC Console 3.6.2., Log File Adapter 3.6.2, DM 3.6.1, SWD 3.6.2, Endpoint Gateway, TMA Endpoint, Job Scheduling Services, TWS connector, and OPC connector
- itso6:  
FTA, running TWS Engine V 7.0, OPC Tracker Agent, TWS Extended Agent for SAP R/3  
TMA Endpoint
- cypress.rtp.lab:  
SAP R/3 V4.6B system. We just needed a working TCP/IP connection to itso6, which is running the TWS Extended Agent for SAP R/3. In a customer environment, you usually install the FTA and the Extended

Agent on the system running the SAP application; so, for monitoring the SAP system, you would need at least the TMA Endpoint on it.

#### **8.1.1.2 NT systems**

These systems, named itso9, itso11, itso12 and itso13, were running NT 4.0 with Service Pack 5. Itso11 was the manager of the TWS domain, NTDOMAIN, with its members, itso12 and itso13. Itso9 was a member of the TWS MASTERDM.

#### **8.1.1.3 OS/390 and OPC Environment**

This is a two-member SYSPLEX running OS/390 Rel 2.6, JES2, and Tivoli OPC 2.3 and is located in Mainz, Germany.

### **8.1.2 Environments and connections**

Since there are various connections and relations between the aforementioned systems, we have decided to split them by application; so, in each of the following diagrams, we will show only the application-relevant information.

#### **8.1.2.1 Job Scheduling Console environment**

The following diagram, shown in Figure 121 on page 206, shows all the connections in our environment necessary to use the Job Scheduling Console for the TWS and OPC environment.

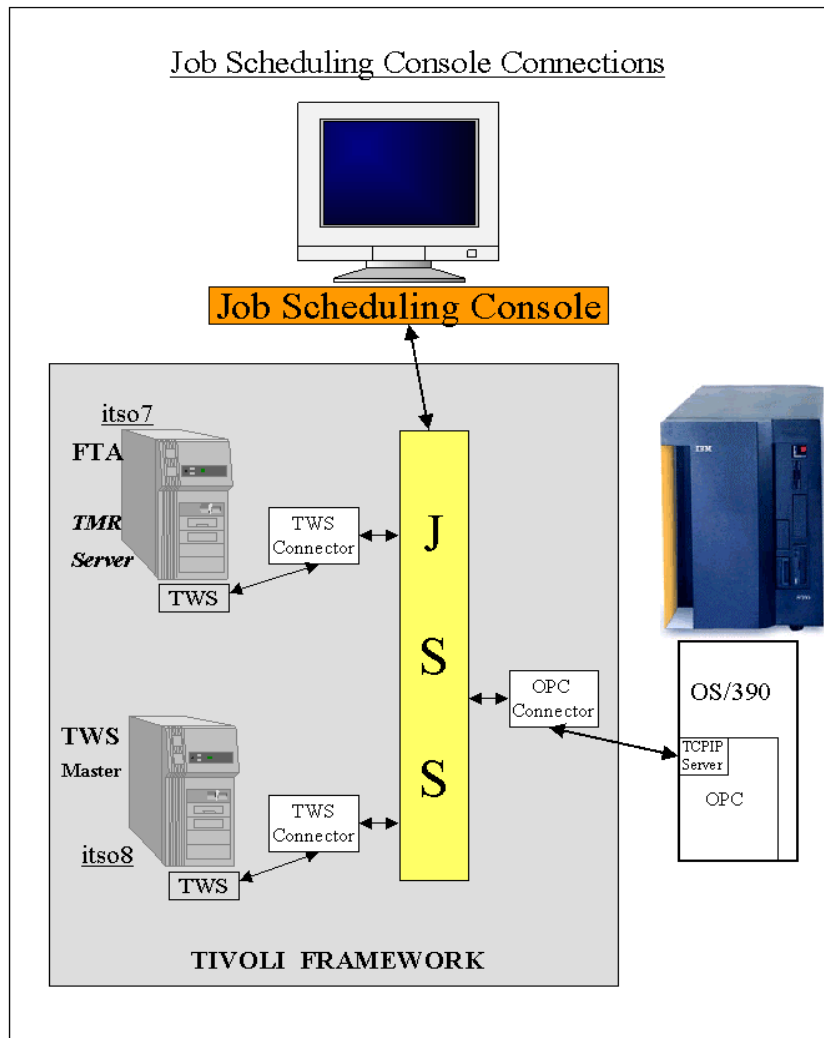


Figure 121. Connections needed for JSC operation

#### 8.1.2.2 Tivoli Framework setup

Figure 122 on page 207 shows the Framework setup necessary for using JSC as well as using the TWS Plus Module for monitoring the TWS environment.

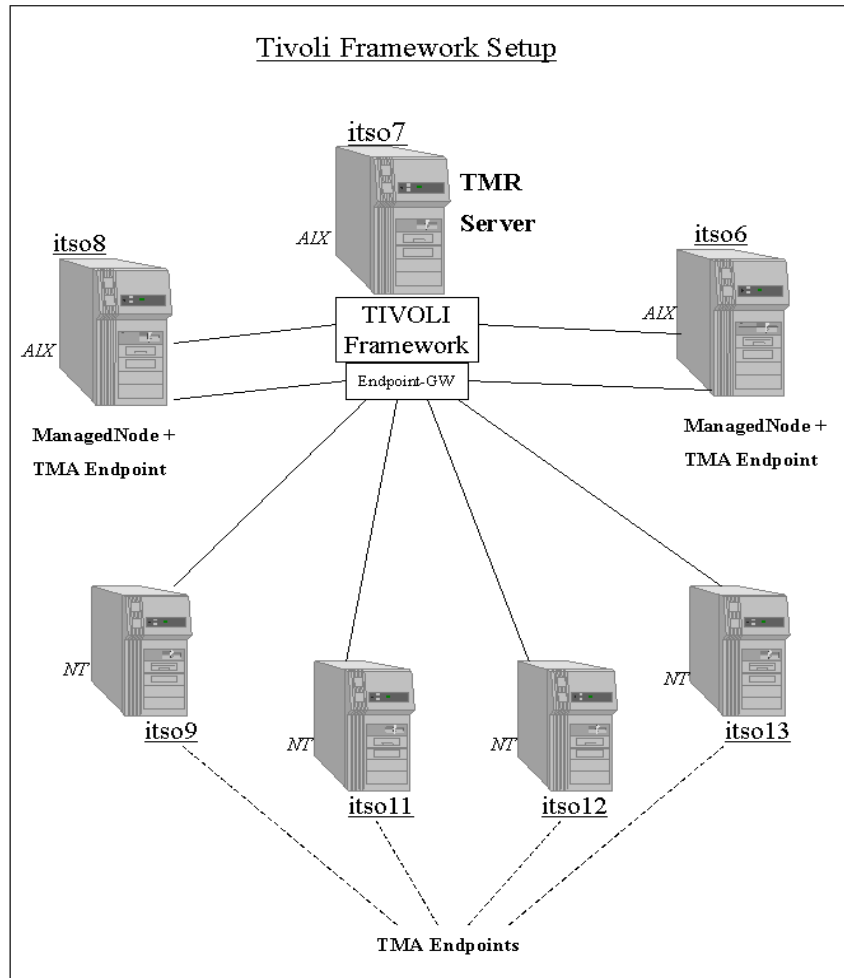


Figure 122. Framework setup

### 8.1.2.3 Tivoli Workload Scheduler setup

The diagram, shown in Figure 123 on page 208, shows all the necessary connections for a successful operation of the TWS environment as well as the connections to OPC and SAP.

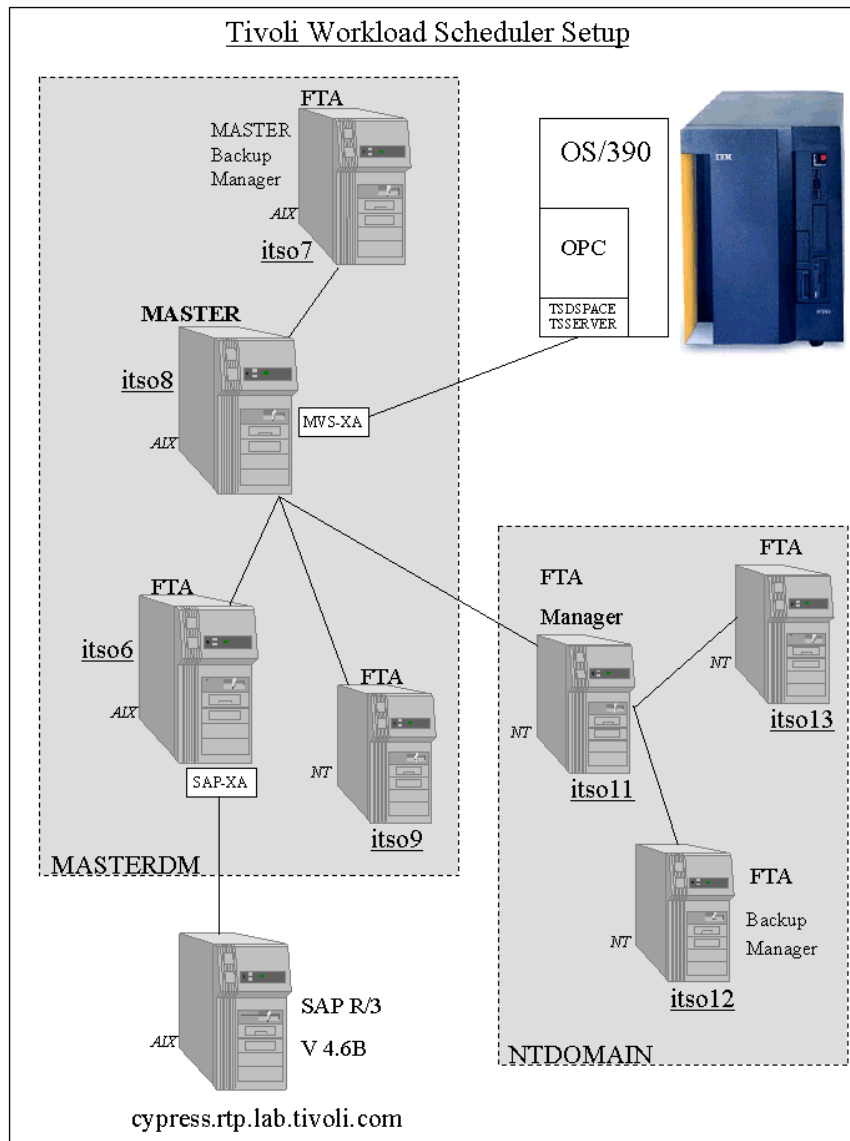


Figure 123. TWS network and external connections

#### 8.1.2.4 Connections regarding interaction between TWS and OPC

Figure 124 on page 209 shows the relevant parts necessary to make the interaction between TWS and OPC work.

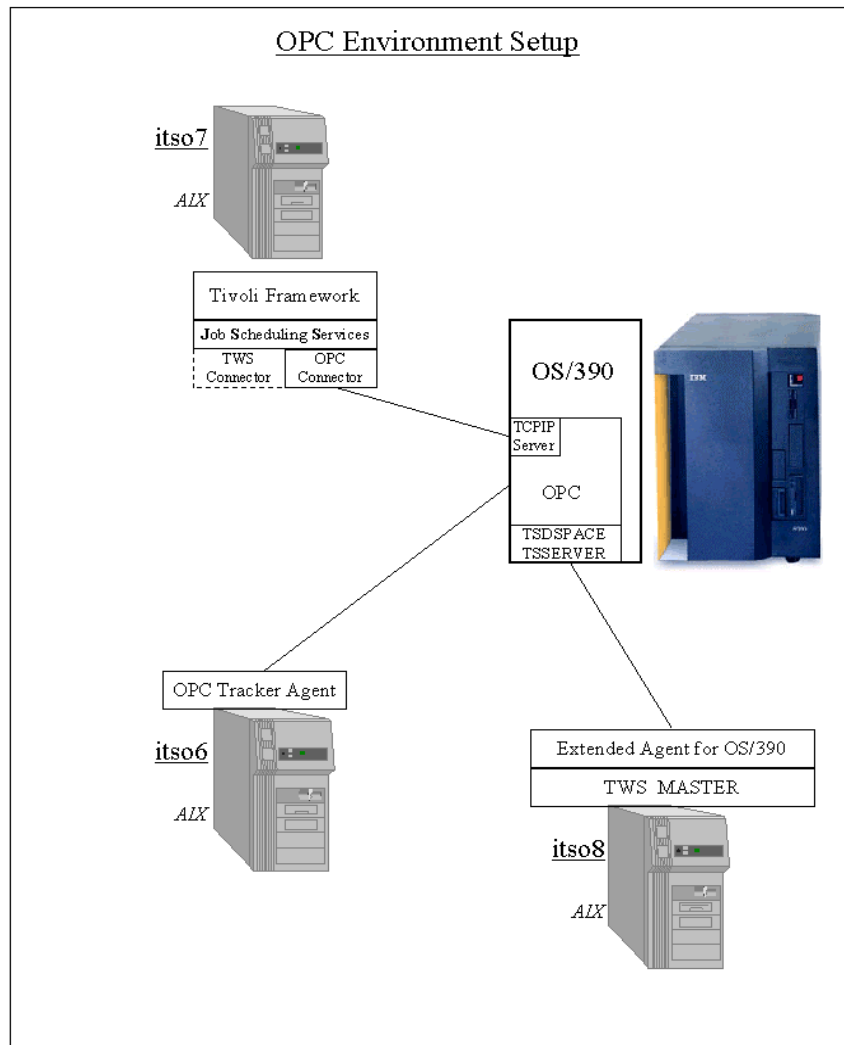


Figure 124. Environment for TWS (OPC interaction)

## 8.2 Resource synchronization between OPC and TWS

A common requirement is making the cross-dependencies between OPC and TWS. We decided to use the resource dependencies. Both OPC and TWS utilize resources. We can cause work to run by making a resource available, and we can cause work to be held back by making a resource unavailable.

The challenge is to get OPC to change the TWS resource availability, and TWS to change the resource availability on OPC. In this section, we will describe how we achieved this.

### **8.2.1 OPC resource availability**

An OPC special resource is a logical representation of some resource that an operation needs. Special resources are defined in the OPC special resource database. The current plan keeps a copy of the resource settings and stores them in part of the current plan known as the special resource monitor. Before OPC submits operations, it checks the monitor to see that the resources required by the operation are available. By updating monitor settings, we can affect work in the current plan. We can change these settings through the ISPF dialog panels, by using the JSC, or by issuing the OPC TSO command, `SRSTAT`.

We intend for TWS to cause the `SRSTAT` command to be issued in the OS/390 environment.



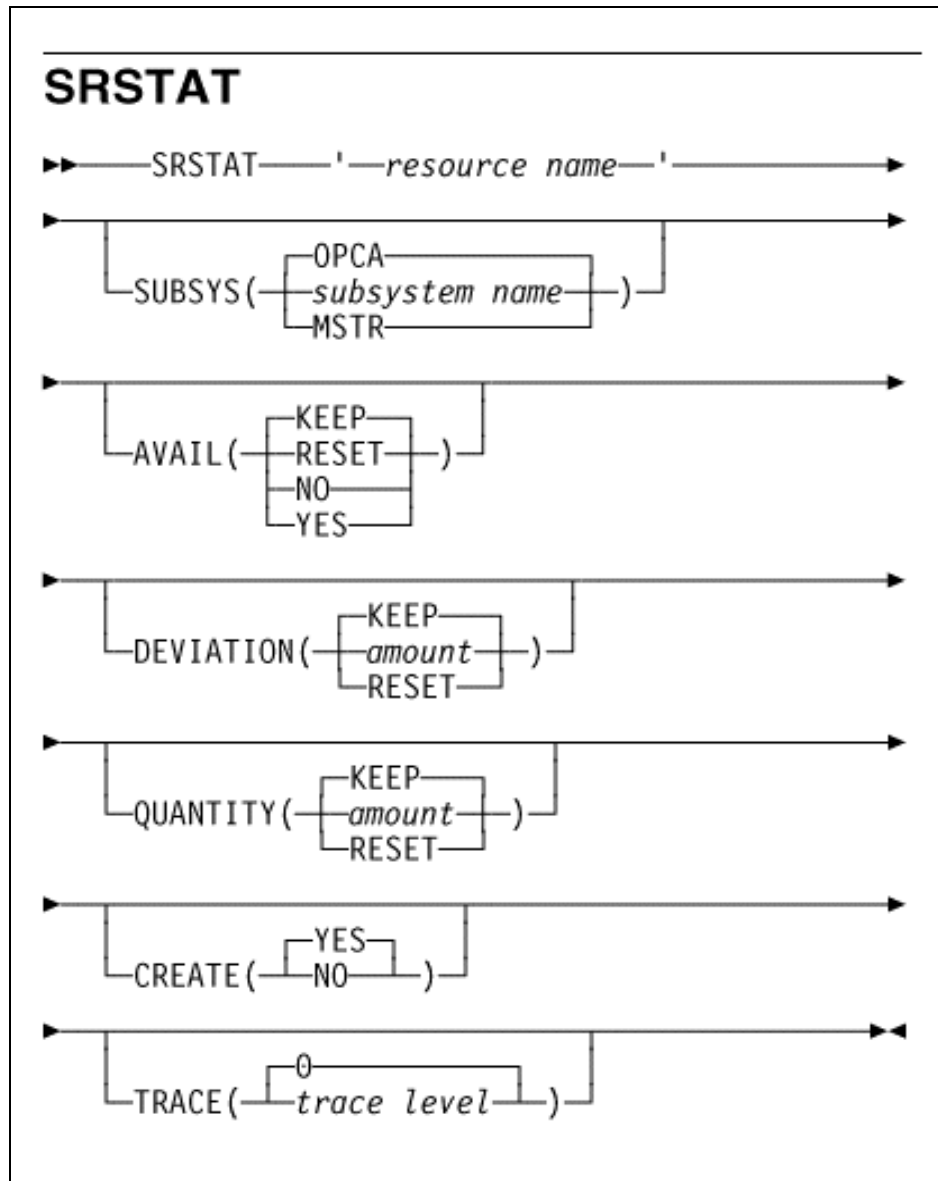


Figure 125. SRSTAT command

The name of the resource is required and is enclosed in single quote marks. SUBSYS is usually required because it identifies the tracker that will process the command; unless specified, the default name, OPCA, will be used. We can use AVAIL to make the entire resource available or unavailable, and we

can use `DEVIATION` to make a temporary addition or subtraction to the current default quantity. By using `QUANTITY`, you can change the default quantity. An example of the `SRSTAT` command is

```
SRSTAT 'MICHAELA' AVAIL(NO) SUBSYS(OPCT)
```

Figure 126. Coding the `SRSTAT` command

This command is making a resource called `MICHAELA` unavailable. The started task identifier of the tracker is `OPCT`. The `SRSTAT` command is documented in the *Tivoli Operations Planning and Control*, SH19-4376.

While the `SRSTAT` command can be used to release or hold back jobs, those jobs must already be scheduled in the current plan. We can link special resources to Event Triggered Tracking (ETT) and cause a non-scheduled application to be added into the current plan.

The ETT database holds the names of special resources and the application that is to be added into the current plan when the `SRSTAT` command makes the special resource available.

### 8.2.2 Creating the special resource

Follow these instructions to create the special resource:

1. Use the right mouse button to click on the **OPC logo** in the left pane and select **New resource** from the pop-up menu as shown in Figure 127 on page 213.

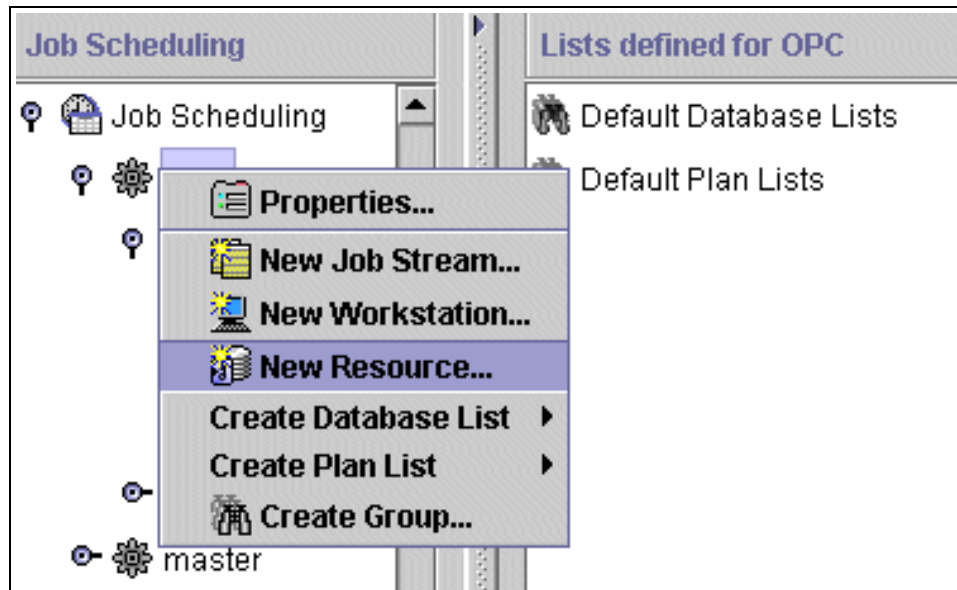


Figure 127. Creating a New Logical Resource

2. Since this resource will be used solely for communication from TWS to OPC, you need only to give it a name and description as shown in Figure 128 on page 214.

The screenshot shows a Windows-style dialog box titled "Properties - Resource in Database". On the left is a sidebar with three options: "Resource" (selected), "Default Workstations", and "Availability Intervals". The main area is titled "Resource" and contains several input fields and checkboxes. The "Name" field contains "MICHAELA". The "Description" field contains "TWS make this ava". Under a "Default" section, the "Quantity" field contains "1". There is an unchecked checkbox labeled "Is available". Below that is an empty "Group ID" field and an unchecked checkbox labeled "Hiperbatch". There are two dropdown menus: "Used for" set to "Planning" and "On error" set to "Free". At the bottom of the main area, it says "Updated by SFRA4 on 13/06/2000 at 22:19". At the very bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

Figure 128. Naming the resource

3. The default is a tick in the box indicating that the resource is available; you should click the **tick box** to make it unavailable.

#### Important

You *must* change the supplied Used For default value Planning to Planning and Control. If you leave the default value, OPC will ignore the availability of the resource when submitting jobs. This JSC default differs from the ISPF dialog default, which is Both, meaning Planning and Control.

4. OPC resources are logical; so, the name you use is your choice. Since this is used solely for TWS-to-OPC communication, it could be named TWS2OPC as a reminder. We have called our resource MICHAELA because that is the name of the AIX box.

You need to define this special resource in the OPC job that is dependent on the completion of a TWS job. Click on **Resources** in the left pane of the job menu. A list of resources is displayed. You have to click on **Target Resources** at the top, and select the **Logical Resources** (Special Resources) pull-down menu as shown in Figure 129 on page 215.

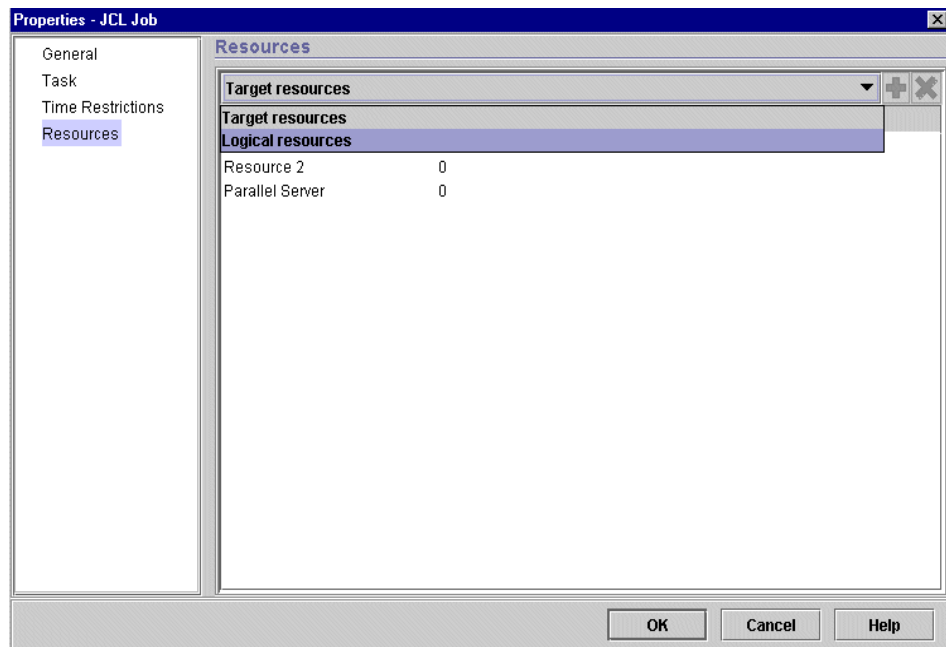


Figure 129. Selecting Job Resources

5. Click the green cross on the logical resources panel to create a new resource dependency as shown in Figure 130 on page 216.

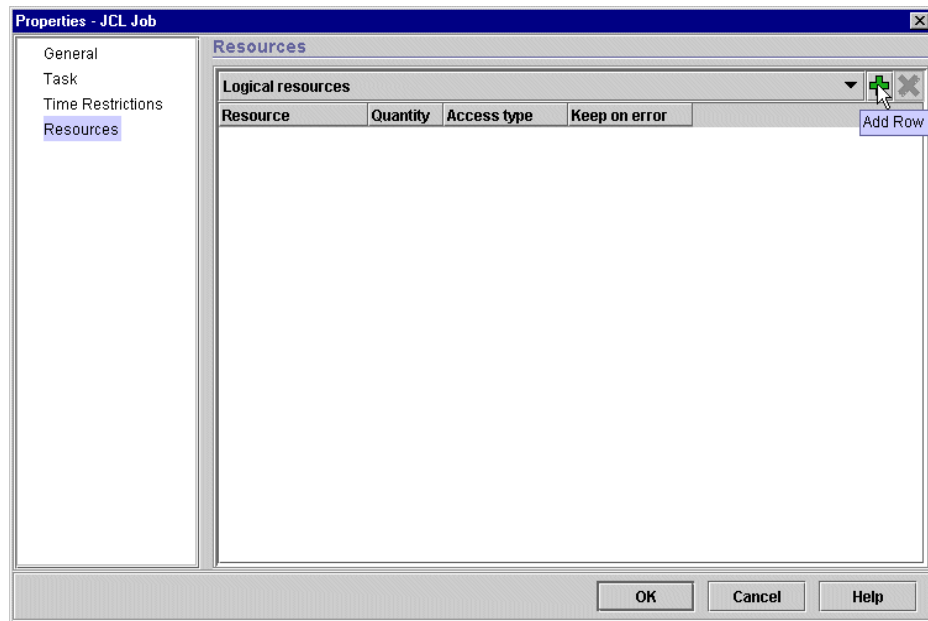


Figure 130. Adding a Logical Resource Dependency

6. Define that this job needs the logical resource that you previously created as shown in Figure 131 on page 217.

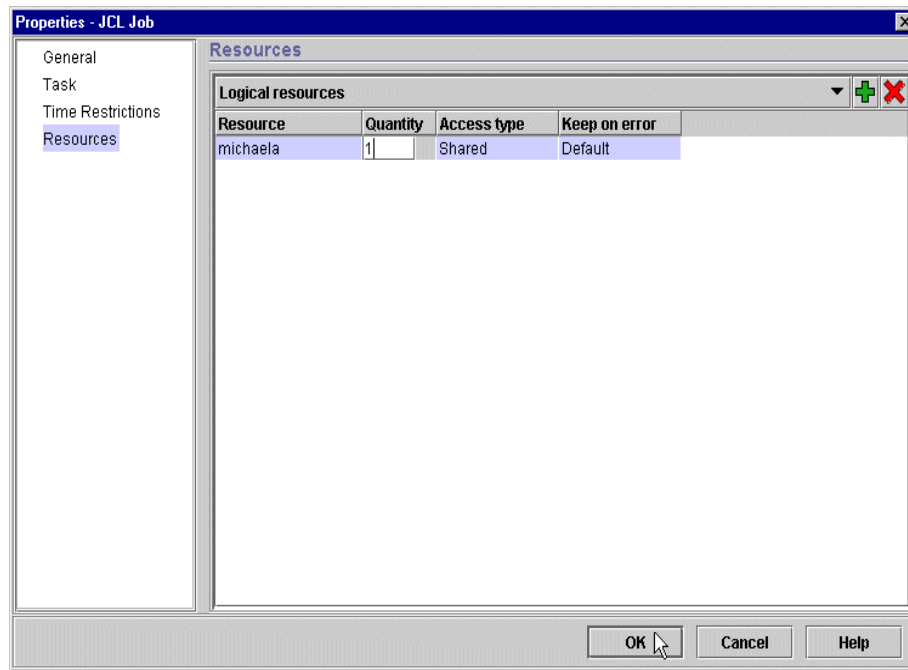


Figure 131. Defining Job Logical Resource Dependency Settings

7. The job can now be saved.

When the job stream is in the plan, the job needing the special resource will not start while that resource is unavailable.

8. Monitor the progress of the job by selecting **All Scheduled Jobs** in the left pane, and choose the **List view** on the tab at the bottom of the right pane as shown Figure 132 on page 218.

**Job Scheduling**

- Job Scheduling
  - OPC
    - Default Database Lists
    - All Job Streams
    - All Workstations
    - All Resources
    - Default Plan Lists
    - All Scheduled Job Streams
    - All Scheduled Jobs**
    - Status of all Workstations
    - Status of all Resources
  - master

**All Scheduled Jobs**

Job Stream Name	Job ID	Description	Task Name	Workstation	Status	Internal...	Status details
TESTHUFF41	20	ADD JOB	TSETVS...	CPUC	Waiting	Waiting	
TESTHUFF41	30		TSETVS...	CPUC	Waiting	Waiting	
TESTHUFF41	40		TSETVS...	CPUB	Waiting	Waiting	
MAY#TEST#APP1	4	non-rep-st...	DUMY		Successful	Complete	
MAY#TEST#APP1	5		TM42J01	CPUB	Successful	Complete	
MAY#TEST#APP1	10		TM42J02	CPUB	Successful	Complete	
MAY#TEST#APP1	15		TM42J03	CPUB	Successful	Complete	
MAY#TEST#APP1	20		TM42J04	CPUB	Successful	Complete	
MAY#TEST#APP1	25		TM42J25	CPUB	Successful	Complete	
MAY#TEST#APP2	4		TM42J09	GX	Waiting	Arriving	
MAY#TEST#APP2	5		TM42J09	CPUB	Waiting	Waiting	
MAY#TEST#APP2	10		TM42J08	CPUB	Successful	Complete	
MAY#TEST#APP2	15		TM42J15	CPUB	Waiting	Waiting	
MAY#TEST#APP2	20		TM42J12	CPUB	Waiting	Waiting	
MAY#TEST#APP2	25		TM42J10	CPUB	Successful	Complete	
MAY#TEST#APP2	4		TM42J09	GX	Waiting	Arriving	
MAY#TEST#APP2	5		TM42J09	CPUB	Waiting	Waiting	
MAY#TEST#APP2	10		TM42J08	CPUB	Successful	Complete	
MAY#TEST#APP2	15		TM42J15	CPUB	Waiting	Waiting	
MAY#TEST#APP2	20		TM42J12	CPUB	Waiting	Waiting	
MAY#TEST#APP2	25		TM42J10	CPUB	Successful	Complete	
TESTHUFF41	10		TESTVS4	CPUB	Successful	Complete	
TESTHUFF41	10		TESTVS...	CPUB	Error	Error	
TESTHUFF41	20	ADD JOB	TSETVS...	CPUC	Waiting	Waiting	
TESTHUFF41	30		TSETVS...	CPUC	Waiting	Waiting	
TESTHUFF41	40		TSETVS...	CPUB	Waiting	Waiting	
MAYJARI#TWS20...	5		TM42J01	CPUB	Waiting	Arriving	Waiting for resources

Timeline View List View

131 items loaded To reload list, select it in the tree and click Load list

Tivoli Job Scheduling Console for Administrator may (may@tso7)

Figure 132. Listing scheduled jobs

The job is shown on the last line with an internal (OPC) status of Arriving. The status details column shows the reason, *Waiting for resources*, as shown in Figure 133 on page 219.



All Scheduled Jobs						
Job Stream Name	Job ID	Task Name	Workstation	Status	Internal Status	Status details
MAY#TEST#APP2	20	TM42J12	CPU8	Waiting	waiting	
MAY#TEST#APP2	25	TM42J10	CPU8	Successful	Complete	
MAY#TEST#APP2	4	TM42J09	GX	Waiting	Arriving	
MAY#TEST#APP2	5	TM42J09	CPU8	Waiting	Waiting	
MAY#TEST#APP2	10	TM42J08	CPU8	Successful	Complete	
MAY#TEST#APP2	15	TM42J15	CPU8	Waiting	Waiting	
MAY#TEST#APP2	20	TM42J12	CPU8	Waiting	Waiting	
MAY#TEST#APP2	25	TM42J10	CPU8	Successful	Complete	
MAY#TEST#APP2	4	TM42J09	GX	Successful	Complete	
MAY#TEST#APP2	5	TM42J09	CPU8	Successful	Complete	
MAY#TEST#APP2	10	TM42J08	CPU8	Successful	Complete	
MAY#TEST#APP2	15	TM42J15	CPU8	Ready	Ready	Waiting for resources
MAY#TEST#APP2	20	TM42J12	CPU8	Waiting	Waiting	
MAY#TEST#APP2	25	TM42J10	CPU8	Successful	Complete	
MAY#TEST#APP2	4	TM42J09	GX	Waiting	Arriving	
MAY#TEST#APP2	5	TM42J09	CPU8	Waiting	Waiting	
MAY#TEST#APP2	10	TM42J08	CPU8	Successful	Complete	
MAY#TEST#APP2	15	TM42J15	CPU8	Waiting	Waiting	
MAY#TEST#APP2	20	TM42J12	CPU8	Waiting	Waiting	
MAY#TEST#APP2	25	TM42J10	CPU8	Successful	Complete	
MAY#TEST#APP2	4	TM42J09	GX	Waiting	Arriving	
MAY#TEST#APP2	5	TM42J09	CPU8	Waiting	Waiting	
MAY#TEST#APP2	10	TM42J08	CPU8	Successful	Complete	
MAY#TEST#APP2	15	TM42J15	CPU8	Waiting	Waiting	
MAY#TEST#APP2	20	TM42J12	CPU8	Waiting	Waiting	
MAY#TEST#APP2	25	TM42J10	CPU8	Successful	Complete	
MAYJAR#TWS20...	5	TM42J01	CPU8	Waiting	Arriving	Waiting for resources

Figure 133. Job status details

9. Another way to view jobs waiting on a logical resource is by selecting **Status of all resources** in the left pane, as shown in Figure 134.

Tivoli Job Scheduling Console for Administrator may@itso7

File Selected Windows Help

Job Scheduling

- Job Scheduling
  - OPC
    - Default Database Lists
    - Default Plan Lists
    - All Scheduled Job Streams
    - All Scheduled Jobs
    - Status of all Workstations
    - Status of all Resources

Status of all Resources

Name	Status	Adjusted quantity	Shared	Exclusive	Jobs Waiting
MICHAELA	Not Available	1	0	0	Yes

Figure 134. Listing resource status

10. Click the right mouse button on the selected resource line, and select **List Jobs => Waiting for Resource** from the pull-down menus as shown in Figure 135.

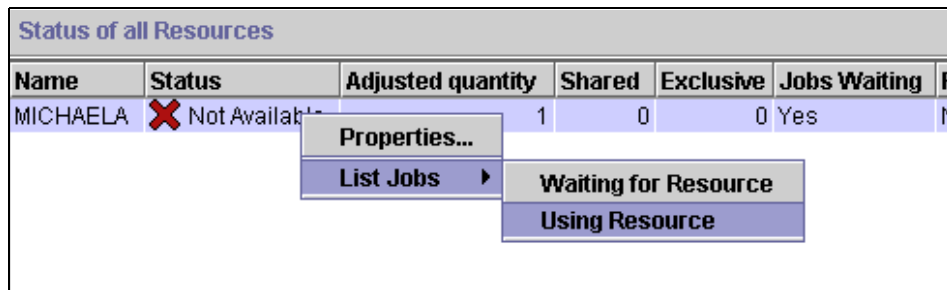


Figure 135. Selecting Jobs Waiting for Resource

The jobs waiting for this resource are listed with the reason they are waiting. In Figure 136, the reason given is **UNAVL**.

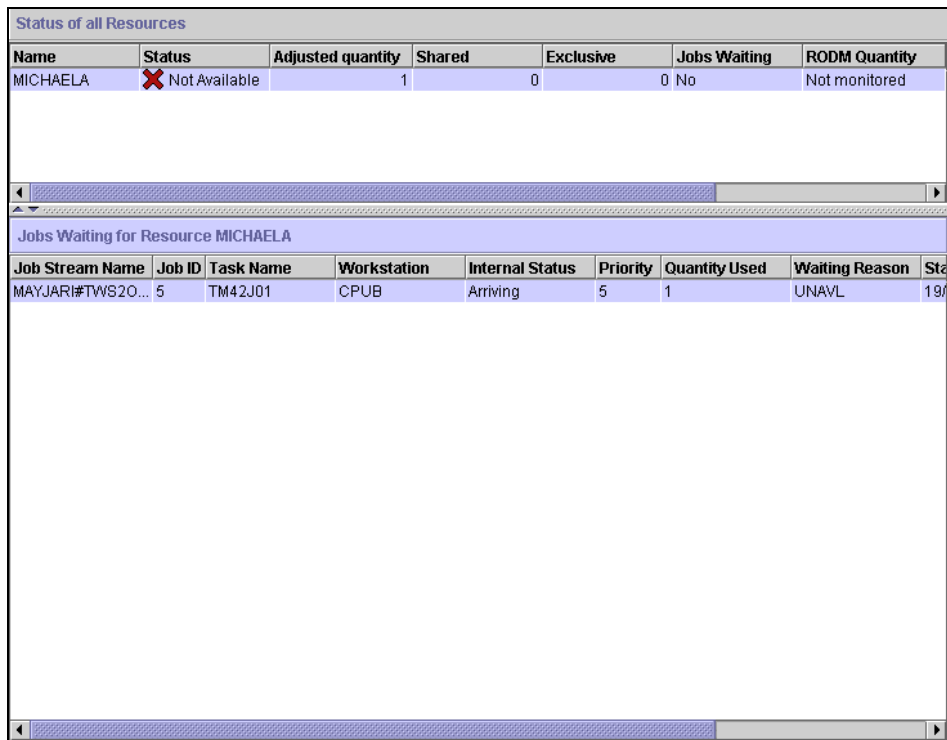


Figure 136. Jobs Waiting on Resource

This operation will not be submitted by OPC until the logical resource is made available. This job should not be started until a job controlled by TWS completes, and when it does, TWS will cause the availability status to be reset, as we describe next

### **8.2.3 Changing OPC Resources status from TWS**

Sometimes it is very useful to be able to modify OPC resources from TWS. At the moment, this task should be performed indirectly. You need to create scripts. Here we give you few useful examples. We found three way to do this and these are:

1. Create JCL script on OS/390 which modifies OPC resources and then launch the JCL script via X/AGENT using MVSJES method. This is explained in Section “Modify OPC Resources from TWS (Solution 1)” on page 221.
2. TWS uses X/AGENT and MVSOPC method to add an OPC application into the current plan. The added application changes resources by using SRSTAT command. This is explained in Section “Modify OPC Resources from TWS (Solution 2)” on page 226.
3. TWS uses X/AGENT and MVSJES method to transfer and submit a batch job in OS/390 which issues the SRSTAT command. Batch job is stored in TWS Workstation (UNIX). This is explained in Section “Modify OPC Resources from TWS (Solution 3)” on page 230.

#### **8.2.3.1 Modify OPC Resources from TWS (Solution 1)**

1. Create a JCL script on OS/390 that modifies OPC resources and then launch the JCL script via X/AGENT using the MVSJES method. See Figure 137 on page 222.

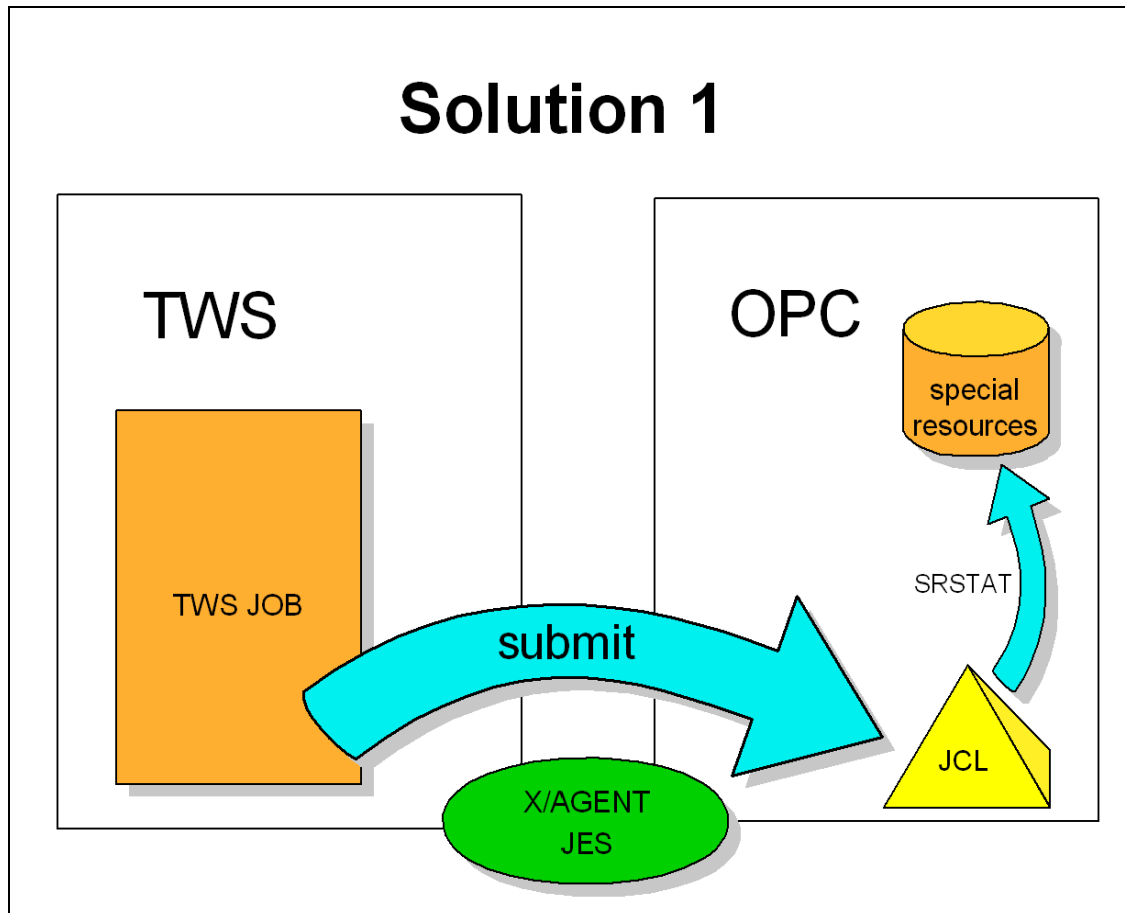


Figure 137. Changing OPC resources from TWS (Solution 1)

2. On OS/390, create a job that will issue the SRSTAT command. In Figure 138 on page 223, we use the program, IKJEFT01, to make the TSO environment for the SRSTAT command. Instead, you could use the program, EQQEVPGM.

```
//YARI      JOB MSGLEVEL=(1,1),MSGCLASS=K,
//          CLASS=A
//STEP01    EXEC PGM=IKJEFT01
//SYSTSPT DD SYSOUT=*
//EQMLIB DD DISP=SHR,DSN=OPCESA.V2R3M0.SEQMSG0
//EQMLOG DD DISP=SHR,DSN=OPC.V2R3M0.MLOGA
//SYSTSIN DD *
// * Change resources in next line
//          SRSTAT 'MICHAELA' AVAIL(YES) SUBSYS(MSTR)
//          /*
```

3. Create the X/AGENT Workstation on TWS. Use the access method, mvsjes, as shown in Figure 138.

Figure 138. Define X/AGENT using the mvsjes method

4. Create a new job on TWS that launches the JCL script on OS/390. Call **LAUNCHYARI**. See Figure 139 on page 224 and Figure 140 on page 224.

**Properties - Job Definition**

**General**

Task Type: Extended Agent Task

Name: LAUNCHYARI

Workstation: AIXJES

Description: Submit JCL-script YARI

Login: maestro

Add Parameter...

Recovery Options

Action: ☒ Stop ☐ Continue ☐ Rerun

Message:

Job:

Workstation:

OK Cancel Help

Figure 139. Define new job in TWS using MVSJES method

**Properties - Job Definition**

**Task**

Task Type: Extended Agent Task

Task: opc.v2r3m0.joblib(yari) = 0004

Add Parameter...

OK Cancel Help

Figure 140. Enter job name and parameters

5. Run Jnextday to update X/AGENT into Plan. Log in to TWS Master as **maestro** user and enter the following:  
\$ Jnextday
6. Select the **TWS Master**, right click, and select **Submit->Job** as shown in Figure 141.

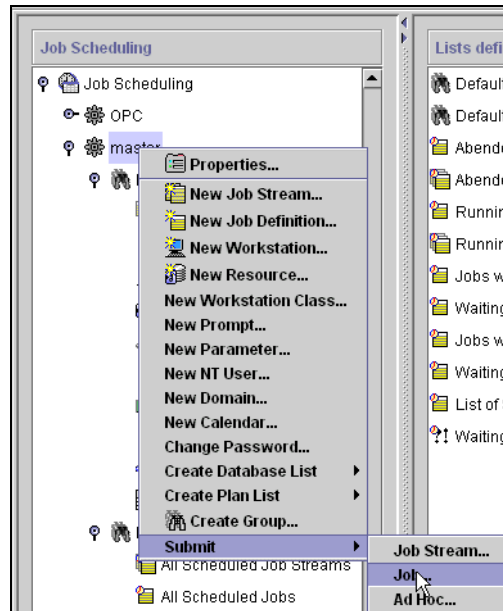


Figure 141. Submit job from TWS

7. Press **OK** to submit job from TWS as shown in Figure 142.

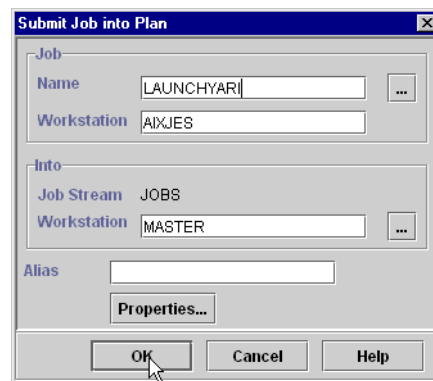


Figure 142. Select **LAUNCHYARI** job

### 8.2.3.2 Modify OPC Resources from TWS (Solution 2)

TWS uses the X/AGENT and MVSOPC methods to add an OPC application into the current plan. The added application changes resources by using the `SRSTAT` command. See Figure 143.

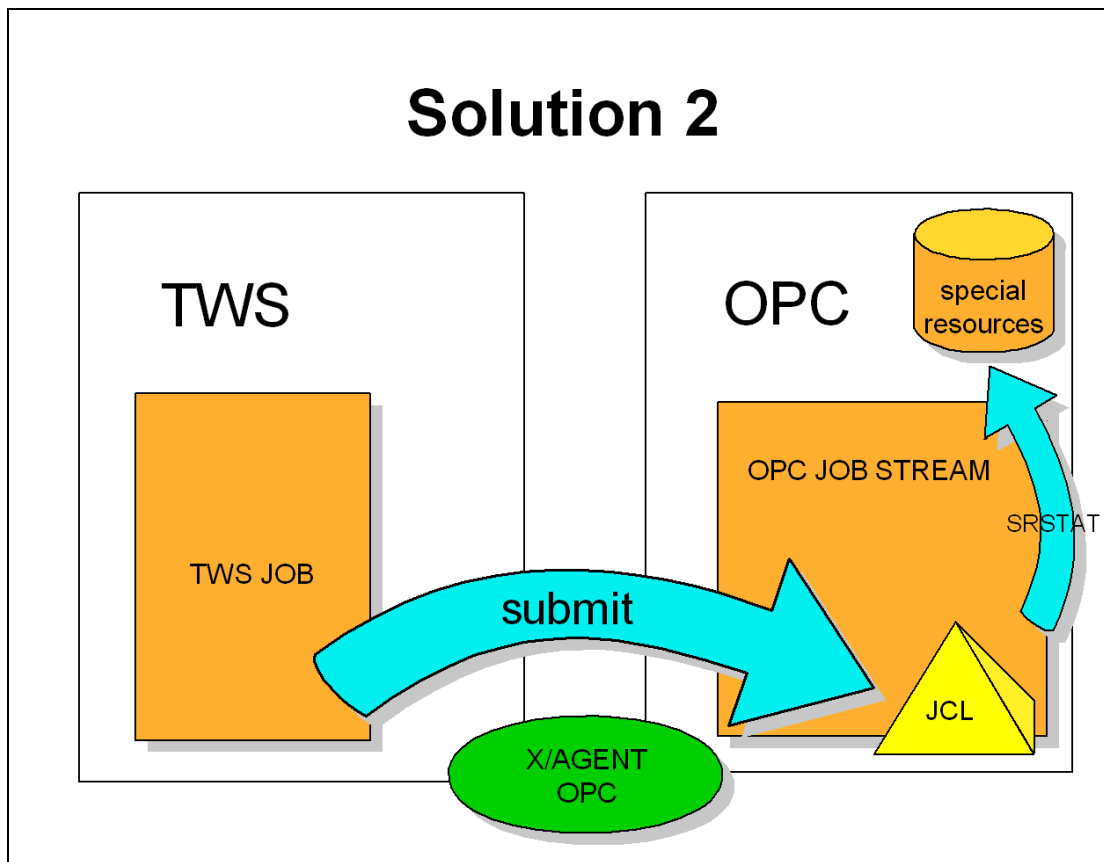


Figure 143. Changing OPC resources from TWS (Solution 2)

1. Create an OPC Job Stream, called OPCYARI. this will contain one job, called YARIJOB.
2. Create a member, called YARIJOB, in the OPC job library. This member should have the JCL that runs the program, EQQEVPGM, and which will issue the `SRSTAT` command. The `SRSTAT` command can be coded to change OPC special resources.
3. Our job modifies the special resource, called MICHAELA, to make it available as shown in the following screen.



```
//YARIJOB JOB MSGLEVEL=(1,1),MSGCLASS=K,
//          CLASS=A
//STEP01 EXEC PGM=EQQEVPGM
//SYSTSPT DD SYSOUT=*
//EQQLIB DD DISP=SHR,DSN=OPCESA.V2R3M0.SEQMSG0
//EQQLOG DD DISP=SHR,DSN=OPC.V2R3M0.MLOGA
//SYSTSIN DD *
/* Change resources in next line
SRSTAT 'MICHAELA' AVAIL(YES) SUBSYS(MSTR)
/*
```

4. Create an X/AGENT Workstation on TWS. Use the access method, MVSOPC. See Figure 144 on page 227 and Figure 145 on page 228.

Figure 144. Create X/AGENT workstation using MVSOPC method

5. Create a TWS job call that submits the OPC Job Stream, OPCYARI, as shown in Figure 145 on page 228 and Figure 146 on page 228.

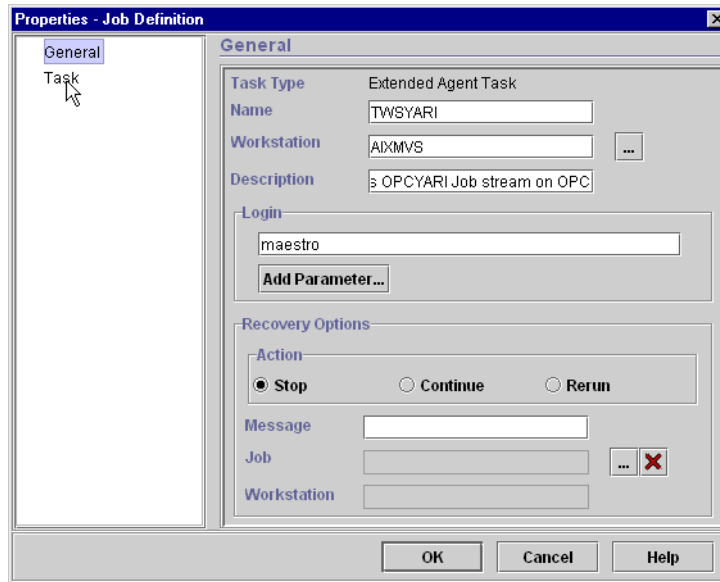


Figure 145. Create new TWS job with MVSOPC method

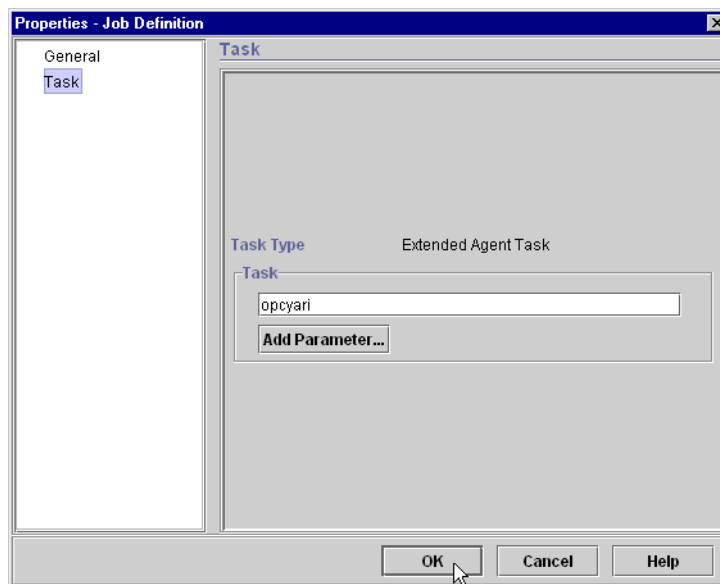


Figure 146. Enter OPC Job Stream (Application) name

6. Run Jnextday to update X/AGENT into Plan. Log in to TWS Master as **maestro** user and enter the following:

\$ Jnextday

7. Submit the TWS job, TWSYARI, from TWS as shown in Figure 147 and Figure 148.

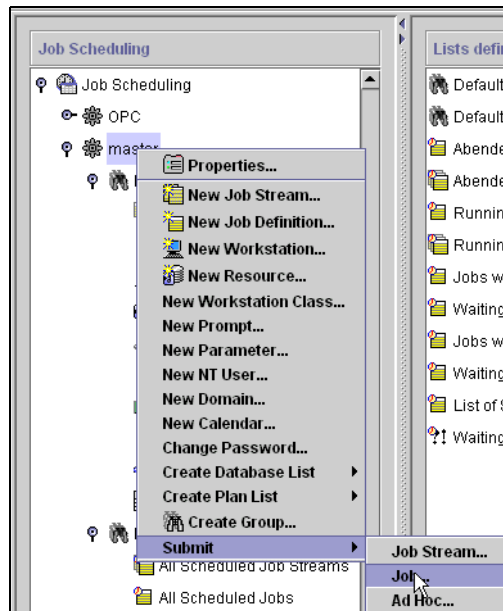


Figure 147. Submit job from TWS

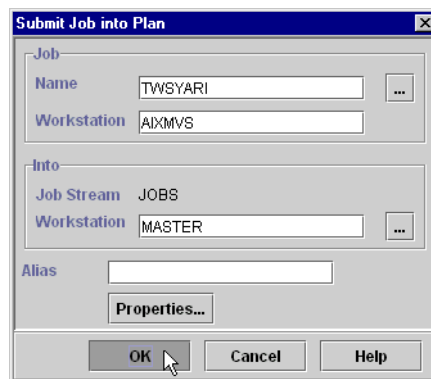


Figure 148. Select TWS job TWSYARI

### 8.2.3.3 Modify OPC Resources from TWS (Solution 3)

TWS uses X/AGENT and MVSJES method to transfer and submit a batch job in OS/390 which issues the SRSTAT command. Batch job is stored in TWS Workstation (UNIX).

We assume that X/AGENT will be installed on UNIX. This example uses FTP and FTP server, so make sure you have FTP server running on OS/390 and proper user privileges to that FTP server. On UNIX site the FTP command is provided as default.

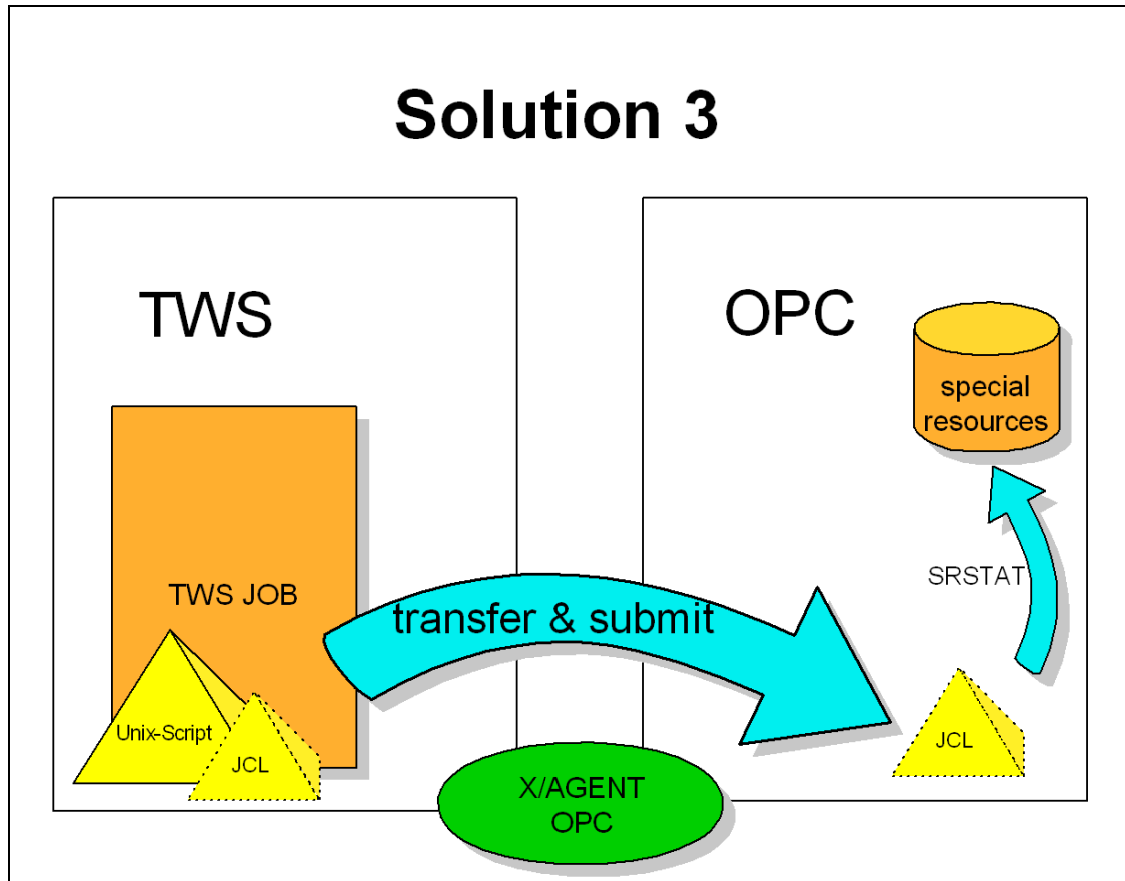


Figure 149. Changing OPC resources from TWS (Solution 3)

1. Create X/AGENT Workstation on TWS. Use the MVSJES access method. Our X/AGENT host is TWS Master, but it can be another TWS Workstation. See Figure 150 on page 231.

Figure 150. Define X/AGENT using vmsjes method

2. Login X/AGENT host as user maestro and create a JCL script. The following script changes the state of the resource in OPC. The resource must be predefined in OPC.

```
$ vi script.jcl
```

```
//YARI      JOB MSGLEVEL=(1,1),MSGCLASS=K,
//          CLASS=A
//STEP01    EXEC PGM=IKJEFT01
//SYSTSPT   DD SYSOUT=*
//EQQLIB    DD DISP=SHR,DSN=OPCESA.V2R3M0.SEQMSG0
//EQQLLOG    DD DISP=SHR,DSN=OPC.V2R3M0.MLOGA
//SYSTSIN   DD *
/* Change resources in next line
  SRSTAT 'MICHAELA' AVAIL(YES) SUBSYS(MSTR)
/*
```

3. In the maestro user home directory in UNIX, create a .netrc file that automates the FTP command. This automation only affects the user maestro and this specific IP-address. The file, yari.jcl, is transferred from UNIX to OS390. In this example, the dataset is OPC.V2R3M0.JOBLIB, and the member is YARI.

The following is the .netrc file syntax:

- **machine** - IP address of OS/390 node

- **login** - User name
- **password** - User's password (Password is not encrypted; so, be careful!)
- **macdef init** - Defines initialization macro for FTP command
  - **prompt** - Disables the interactive mode
  - **ascii** - Changes the transfer mode to ascii
  - **put** - Transfers the file. (put [source] '[dataset(member)]')
  - **quit** - Quits the FTP application

```
$ vi .netrc
```

```
machine 9.39.62.19 login sfra1 password wh2t3v3r
macdef init
prompt
ascii
put /opt/maestro/script/yari.jcl 'opc.v2r3m0.joblib(yari)'
quit
```

After the `macdef init` command, you can add normal FTP commands.

#### 4. Modify .netrc file attributes:

```
$ chmod 600 .netrc
```

#### 5. Create a new shell script that starts an FTP transfer and submits a JCL script on OS390:

```
$ vi opcresources.sh
```

```
#!/bin/ksh

#Connect to OS390 host. Initialazion script is taken from file .netrc
ftp 9.39.62.19

#Test if ftp command is success.
if [ $? != 0 ]
then
    exit 1
fi

#Submit OPC job through TWS and give it random alias with prefix "res".
conman "sbd aixjes#'opc.v2r3m0.joblib(yari) = 0004';alias=res$$"
```

#### 6. Modify the attributes of opcresources.sh so that it will be executable.

```
$ chmod 750 opcresources.sh
```

#### 7. Modify the .jobmanrc file to get your profile settings from the .profile file:

```
$ vi .jobmanrc
```

```
#!/bin/ksh
. ~/.profile
```

8. Modify the `.jobmanrc` file attributes to make it executable:  

```
$ chmod 755 .jobmanrc
```
9. Create a job onto TWS that launches the `opcresources.sh` script. Call it **changeres**. See Figure 151 and Figure 152 on page 234.

The screenshot shows a 'Properties - Job Definition' dialog box with the 'General' tab selected. The 'Task Type' is 'Unix Script'. The 'Name' field contains 'CHANGERES'. The 'Workstation' field contains 'ITS06'. The 'Description' field contains 'Send jcl script to OS390 and lau'. The 'Login' field contains 'maestro'. There is an 'Add Parameter...' button. Under 'Recovery Options', the 'Action' section has three radio buttons: 'Stop' (selected), 'Continue', and 'Rerun'. There is a 'Message' field. The 'Job' and 'Workstation' fields are empty. At the bottom are 'OK', 'Cancel', and 'Help' buttons.

Figure 151. Create a new job definition

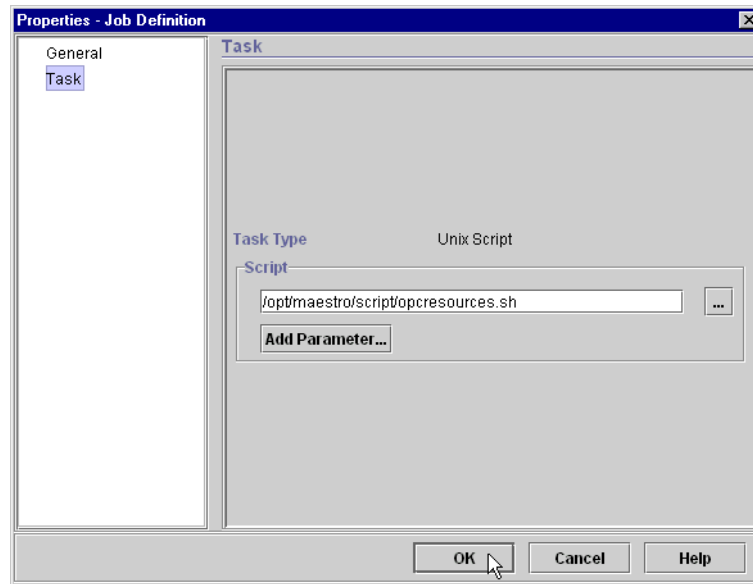


Figure 152. Enter name of the script

10. Run Jnextday to update X/AGENT into Plan. Log in to TWS Master as maestro user and enter the following:

```
$ Jnextday
```

11. Submit the job from TWS. See Figure 153 on page 235 and Figure 154 on page 235.



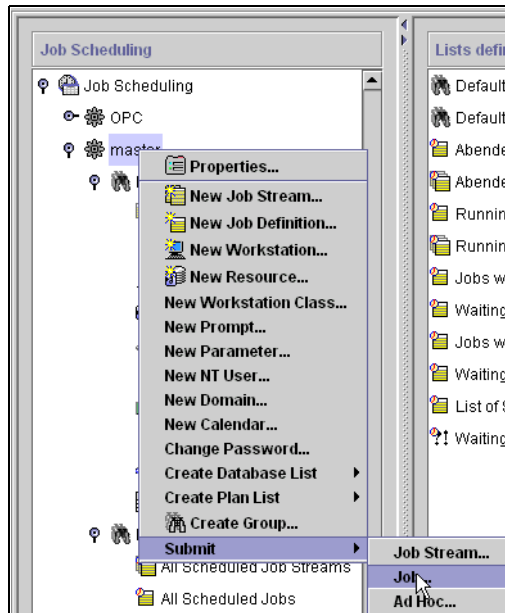


Figure 153. Submit job

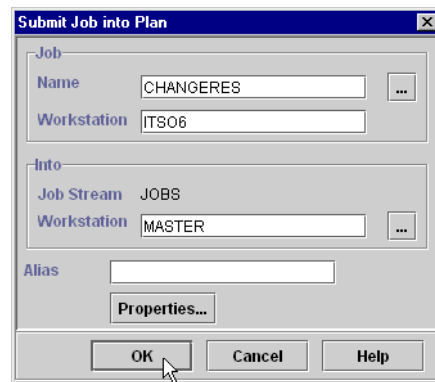


Figure 154. Enter Job name

Submitting the CHANGERES job causes the YARI.JCL file to be transferred onto OS390 after you get a new job, named RES[and numbers], which submits a YARI.JCL script on OS390 using the MVSJES method.

**Important**

At least Version 1.4.6. of X/AGENT on OS390 is required to get the job exit code correctly with the MVSJES method.

## 8.2.4 Conclusion

For the preceding examples we assumed that your OS/390 system had the TWS Extended Agent for OS390 installed. You may refer to Chapter 11, “TWS Extended Agent for MVS and OS/390” on page 331 for installation of TWS Extended Agent for MVS and OS/390.

We implemented three different solutions to synchronize resources between OPC and TWS. Each solution has its merits and we believe that the decision to choose which method to use depends on the specifics of the environment.

We will give you some guidelines below:

*Solution 1* is very simple to implement on the TWS side. It requires that you predefine jobs on OS/390 with hardcoded SRSTAT commands, which cannot be controlled dynamically from scripts on the TWS side. If you know exactly what and how resources should be changed, this solution will be appropriate.

*Solution 2* is also easy to implement from the TWS side. It also requires that you predefine jobs on OS/390 with hardcoded SRSTAT commands, which cannot be controlled dynamically from scripts on the TWS side. You could have a job that makes the resource available and another job that makes it unavailable. In this case these jobs are under OPC control and thus this solution can also be used to make dependency links with other OPC jobs. It also gives you an example of how to integrate with an OPC Jobstream.

*Solution 3* gives you the ability to control resource changes completely from TWS. This is most flexible solution of these three as it allows the SRSTAT command to be coded at the TWS side. You can also influence other OPC jobs if you use the MVS OPC X-agent method. The main disadvantage is possible problems when checking the FTP script exit code. In the FTP client there is no way to get an exit code if the transfer fails. This can be circumvented if your script uses grep to parse error texts from FTP output. One alternative would be to use the confirm feature in the FTP job stream, requiring an operator to check the output manually. The operator could then set the job stream status to success orabend.

We recommend using either solution 2 or solution 3 depending on whether you want to be able to code the SRSTAT command in the OPC or TWS environment.

**Note**

Keep in mind that in all cases, if something goes wrong during execution of conman submit you may not get error information through exit codes. You may need to create your own scripts if you want to account for this possibility.

---

## **8.3 Changing TWS Resources from OPC**

In this section we explain how to operate TWS Resources from OPC. We will provide one method to do this. You may find different ways to do it, but the following should give you an idea of possible solutions.

### **8.3.1 Resources in TWS**

Resource functionality in TWS is very simple. Resources in TWS are logical, and they need not to be tied to real physical resources (although usually they do closely match a real resource status). This gives you a flexible way to control jobs by using a resource as a semaphore.

When creating a resource, you can specify a resource name, which can contain eight characters and a numeric value (quantity) that can be between 0 and 1024. Resources are specified on TWS Workstations.

Figure 155, on this page, and Figure 156 on page 238 show how to create a new resource in TWS.

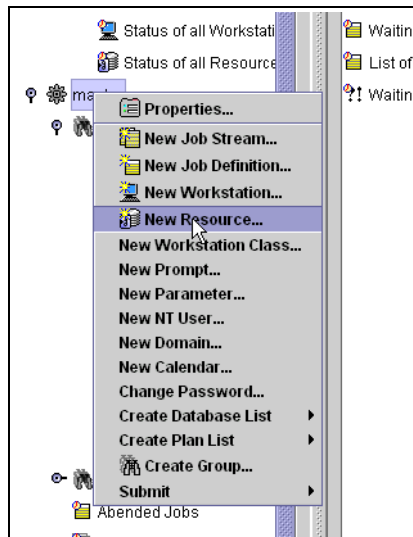


Figure 155. Create new resource

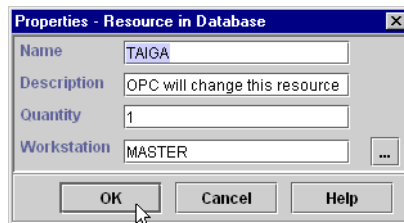


Figure 156. Properties of a resource

### 8.3.2 Changing TWS resources from OPC

Figure 157 on page 239 shows an example of how to change TWS resources from OPC.

The resource will appear to the TWS plan only when a job needs that specific resource. You cannot change the status of a resource from the plan view or from the command line when it is not in the current plan. This is important to know if you are writing scripts because using the `conman resource` command only affects resources if a specific resource is in the plan.

You may need to change a resource that is in the TWS database from scripts. You cannot do this with just one command; the following shows how we did it.

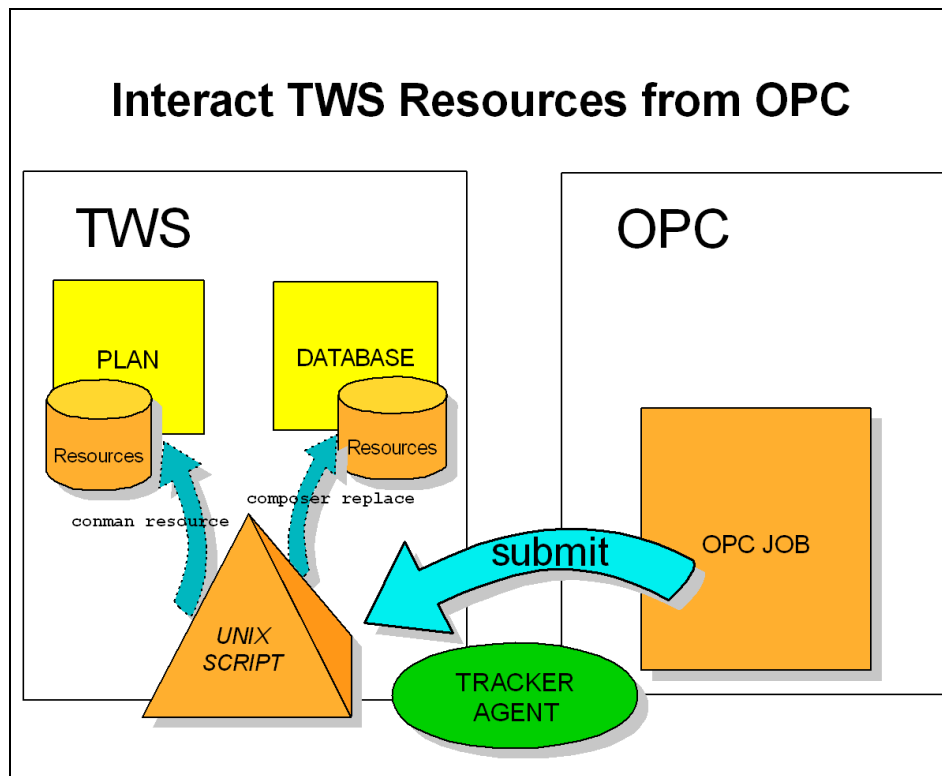


Figure 157. Interact TWS Resources from OPC

The *modres* script takes input from command line parameters and modifies an existing resource's status in the TWS database. At first, the script checks if a specific resource is in the plan and it invokes the `conman resource` command to change its value. In case the resource is not in the plan `composer replace` is used. You can run this script only on the TWS Master or on a TWS Workstation connected to the TWS Database via NFS. Modres does not create new resources; so, make sure that the resource you are using is already defined in the TWS Database.

```

#!/bin/ksh
RESOURCE=$1
RESNUM=$2
RESOPT=$3
umask 000
. /opt/maestro/.profile

function composerreplace {
    composer "create /tmp/res.txt from resources"
    grep "${RESOURCE}" /tmp/res.txt >/dev/null
    if [ $? != 0 ]
    then
        echo "Resource $RESOURCE was not found!"
        exit 1
    fi
    echo $RESOURCE $RESNUM
    cat /tmp/res.txt |sed "s/^\($RESOURCE\) [ 0-9 ]*/\1 $RESNUM /g" | tee /tmp/res.txt
    composer "replace /tmp/res.txt"
}

function conmanresource {
    conman "resource $RESOURCE;$RESNUM;noask"
}

if [ -n "$RESOURCE" ] && [ -n "$RESNUM" ]
then
    RE=`conman "sr $RESOURCE" 2>&1|grep $RESOURCE |wc -l`
    RE=`expr $RE`
    if (( $RE==2 )) && [[ $RESOPT = "r" ]]
    then
        RE=3
    fi
    case $RE in
        1) composerreplace
            ;;
        2) conmanresource
            ;;
        3) conmanresource
            composerreplace
            ;;
    esac
else
    echo "Need parameters!\nexample: modres {CPU#RESOURCE} {0-1024} [r]"
    exit 1
fi
exit 0

```

Notice that there is one *tabulator* and *no whitespaces* right after \$RESOURCE before backslash \ in the sed command. You can do this example by pressing **Ctrl-I**.

1. Save the script as /opt/maestro/scripts/modres.

2. Next, modify the file properties as follows:

```
$ chmod 750 modres
```

3. Run the `modres` command only as `maestro` user or as a user who has sufficient privileges. *This command is case-sensitive.* You need at least *TWS modify access rights* to the TWS database.

Command usage:

```
modres [CPU]#[RESOURCE] [0-1024]
```

For example, to modify logical resource **TAIGA** to **5** do the following:

```
$ /opt/maestro/scripts/modres MASTER#TAIGA 5
```

### **Steps to change TWS resources from OPC**

Perform the following steps to change TWS resources from OPC:

1. Install OPC Tracker Agent for AIX on the AIX machine. We recommend that you install the OPC Tracker Agent for AIX on the TWS Master. Refer to Section 9.6, “Installing and customizing the Tracker Agent” on page 283 for more information.
2. If your UNIX machine with the Tracker Agent is not the TWS Master, make NFS connections for the `/opt/unison` and `/opt/maestro/mozart` directories with the following steps:
  - a. Make sure that the UID and GID numbers of user `maestro` are the same in both systems. Otherwise, it may claim problems with file accesses and owners of files.
  - b. Export directories on the TWS Master. Log in to the TWS Master as root user.
  - c. Execute `smitty nfs`.
  - d. Select **Network File System (NFS)**.
  - e. Select **Add a Directory to Exports List**, and enter `/opt/maestro/mozart` in the PATHNAME of Directory to Export field.

Add a Directory to Exports List

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
* PATHNAME of Directory to Export	/opt/maestro/mozart	
* MODE to export directory	read-write	+
HOSTS & NETGROUPS allowed client access	[]	
Anonymous UID	[-2]	
HOSTS allowed root access	[]	
HOSTNAME list. If exported read-mostly	[]	
Use SECURE OPTION?	no	+
Public filesystem?	no	+
* CHANGE export now, system restart or both	both	+
PATHNAME of alternate Exports file	[]	

F1=Help

F2=Refresh

F3=Cancel

F4=List

Esc+5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do

- f. Press **F3** and then **Enter**, and, again, select **Add a Directory to Exports List**. Enter `/opt/unison` in the PATHNAME of Directory to Export field.

Add a Directory to Exports List

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
* PATHNAME of Directory to Export	/opt/unison	
* MODE to export directory	read-write	+
HOSTS & NETGROUPS allowed client access	[]	
Anonymous UID	[-2]	
HOSTS allowed root access	[]	
HOSTNAME list. If exported read-mostly	[]	
Use SECURE OPTION?	no	+
Public filesystem?	no	+
* CHANGE export now, system restart or both	both	+
PATHNAME of alternate Exports file	[]	

F1=Help

F2=Refresh

F3=Cancel

F4=List

Esc+5=Reset

F6=Command

F7=Edit

F8=Image

F9=Shell

F10=Exit

Enter=Do



- g. Press **Enter** and then **F3**
- h. Select **Configure NFS on This System**
- i. Select **Start NFS**

Start NFS

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
* START NFS now, on system restart or both	both	+

- j. Press **Enter**. NFS should now be running on the TWS Master.
- k. Log in to the AIX machine where Tracker is installed as root user.
- l. Use smitty to mount the NFS file systems:
 

```
$ smitty nfs
```
- m. Select **Network File System (NFS)**.
- n. Select **Add a File System for Mounting**. Press **Enter** and then **F3**.
- o. Again, select **Add a File System for Mounting**. Enter `/opt/unison` in the **PATHNAME of mount point** and **PATHNAME of Remote Directory** fields. Set the other options as shown in the following screen.

### Add a File System for Mounting

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
* PATHNAME of mount point	/opt/maestro/mozart	
* PATHNAME of Remote Directory	[/opt/maestro/mozart]	
* HOST where remote directory resides	[itso8]	
Mount type NAME	[]	
* Use SECURE mount option?	no	+
* Remount file system now,	both	+
update /etc/filesystems or both?		
* /etc/filesystems entry will mount the directory	yes	+
on system RESTART.		
* MODE for this NFS file system	read-write	+
* ATTEMPT mount in foreground or background?	background	+
NUMBER of times to attempt mount	[]	#
Buffer SIZE for read	[]	#
Buffer SIZE for writes	[]	#
NFS TIMEOUT. In tenths of a second	[]	#
NFS version for this NFS filesystem	any	+
Transport protocol to use	any	+
Internet port NUMBER for server	[]	#
* Allow execution of SUID and sgid programs	yes	+
in this file system?		
* Allow DEVICE access via this mount?	yes	+
* Server supports long DEVICE NUMBERS?	yes	+
* Mount file system soft or hard	hard	+
Allow keyboard INTERRUPTS on hard mounts?	yes	+
Minimum TIME, in seconds, for holding	[]	#
attribute cache after file modification		
Maximum TIME, in seconds, for holding	[]	#
attribute cache after file modification		
Minimum TIME, in seconds, for holding	[]	#
attribute cache after directory modification		
Maximum TIME, in seconds, for holding	[]	#
attribute cache after directory modification		
Minimum & Maximum TIME, in seconds, for	[]	#
holding attribute cache after any modification		
The Maximum NUMBER of biod daemons allowed	[]	#
to work on this file system		
* Use acls on this mount?	no	+
Number of NFS retransmits	[]	#
* Exchange POSIX pathconf information?	no	+
* Inherit group IDs?	no	+
F1=Help	F2=Refresh	F3=Cancel
Esc+5=Reset	Esc+6=Command	Esc+7=Edit
Esc+9=Shell	Esc+0=Exit	Enter=Do
		F4=List
		Esc+8=Image

3. Create an OPC Job that submits the /opt/maestro/scripts/modres command with parameters on the AIX machine through the OPC Tracker Agent for AIX.

Now, you can simply change the parameters for command line parameters and affect any resource in TWS from OPC. You cannot create new resources with the modres script.

---

## 8.4 Synchronization between OPC and TWS

In this scenario, we envisage a company with branch offices all over the world. The company HQ has an OS/390 host with OPC that schedules and controls the enterprise workload.

### 8.4.1 Requirements

Each country has its own IT architecture (host-centric or distributed) and software products. Each country has a central office that controls all the national branch offices. Branch offices have several HP, SUN, NT, and AIX servers running Oracle database. In the country's central office, TWS controls the workload in the country's local environment.

The company HQ controls the central offices of all the countries.

The central marketing department wants to receive detailed sales reports from the central offices of all countries, whether they are in one country or a number of countries. A remarkable growth or decline in business is registered.

The following process is used to produce this report:

1. Every Saturday, OPC schedules a jobstream, ACCOUNT#01, which is an accounting application that runs in each national central office. This is done by having the OPC Tracker Agent execute the script, submit1, which submits a TWS jobstream to the MASTER workstation. This jobstream, STATISTC, consists of two jobs.
2. The first job, sales\_data, updates the sales data.
3. The second job, calculate, calculates the variation in the country's business. This process has to be finished by 0800 hours of the following Monday. When a variation exceeding a defined threshold is detected, notification is sent to the central office. In this case, the TWS job starts the application, COLLECT#01, in OPC by submitting the job, NOTIFY, to the Extended Agent for OS/390 workstation, AIXMVS.
4. This NOTIFY job starts an OPC application, called COLLECT#01, which starts a collection application running Monday night after 1900 hours and which is to be finished before 2300 hours.

In Figure 158 on page 247, you will find a process flowchart of our scenario showing the interaction between OPC and TWS.

## Synchronisation between OPC and TWS

### Flowchart

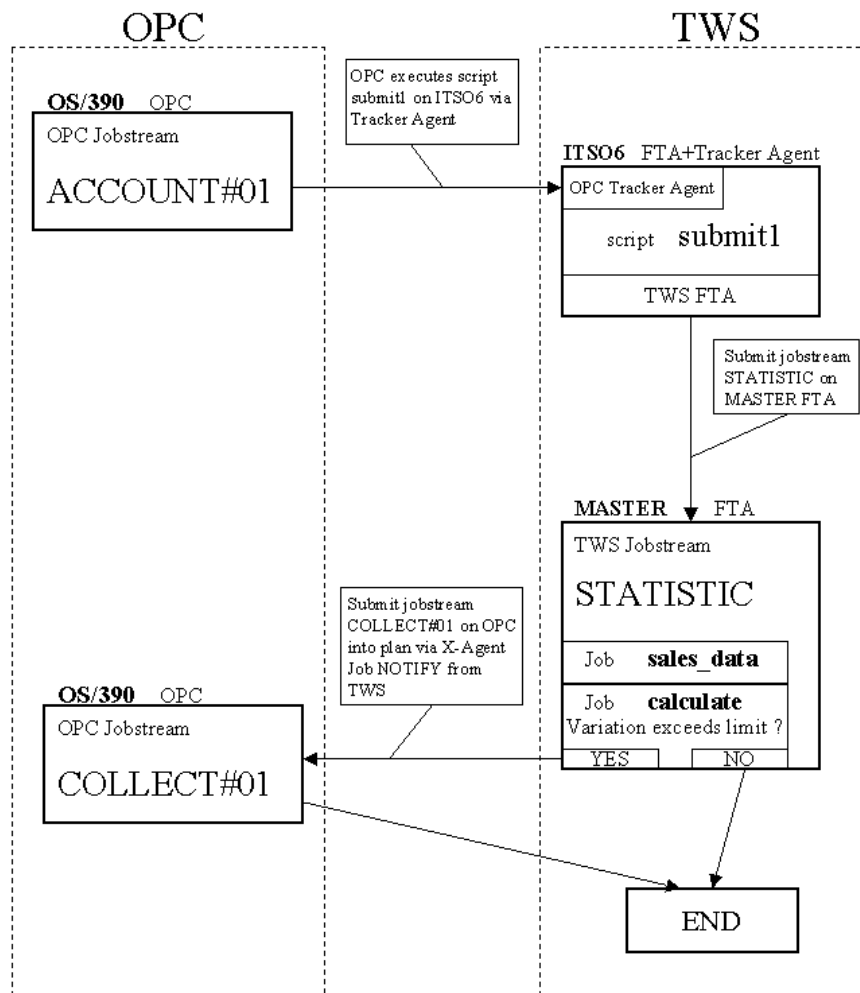


Figure 158. Interaction between OPC and TWS

### 8.4.2 Submitting a Job Stream/Job/Command into TWS via CLI

If you want to be able to submit a job stream/job/command in TWS via CLI from an FTA other than the MASTER itself, you need to have access from this FTA to the contents of the following directories on the MASTER.

- *TWShome/mozart*
- *TWShome/./unison*

*TWShome* points to the TWS home directory.

You can accomplish this by doing one of the following:

- Copy the contents of these two directories from the MASTER to the client FTA, either at regular intervals or whenever you add new jobs, jobstreams, calendars, prompts, and so on. The size of the directories can go up to a few Megabytes depending on the number of daily jobs, job streams, and other entries in your daily plan.
- Mount the two directories via NFS from the master.

Which one you choose depends on the platform(s) used, the size of your mozart-directory, and how often you add/modify your MASTER TWS database. Copying the contents from the MASTER gives you more redundancy in case of MASTER failure but adds more complexity in order to keep the copy of the mozart-directory up-to-date. NFS-mounts are easy to set up and very common in a UNIX-environment, and you are always at the same version of the files in the TWS/mozart and TWShome/./unison-directory.

In our environment, we used NFS-mounts from MASTER itso8 to the FTA itso6, which is also running the OPC Tracker Agent for AIX and needs to submit a jobstream on the MASTER.

#### Note

When submitting jobs or jobstreams through CLI with conman, you should be aware of the fact that conman does not return the correct return code if the submit failed; so, you have to provide the script with additional code to search in the output of the conman submit... for strings, such as error, failed, or similar words that cover situations where submissions failed.

### 8.4.3 Create jobs for TWS Extended Agent for OS/390

The workstation of the extended agent for OS/390 is already defined to the TWS engine. If you need more information, see Chapter 2 of the *TWS Extended Agent for MVS and OS390*, GC32-0642.

1. From Pull-down menu, select **Selected - New Job Definition** as shown in Figure 159.

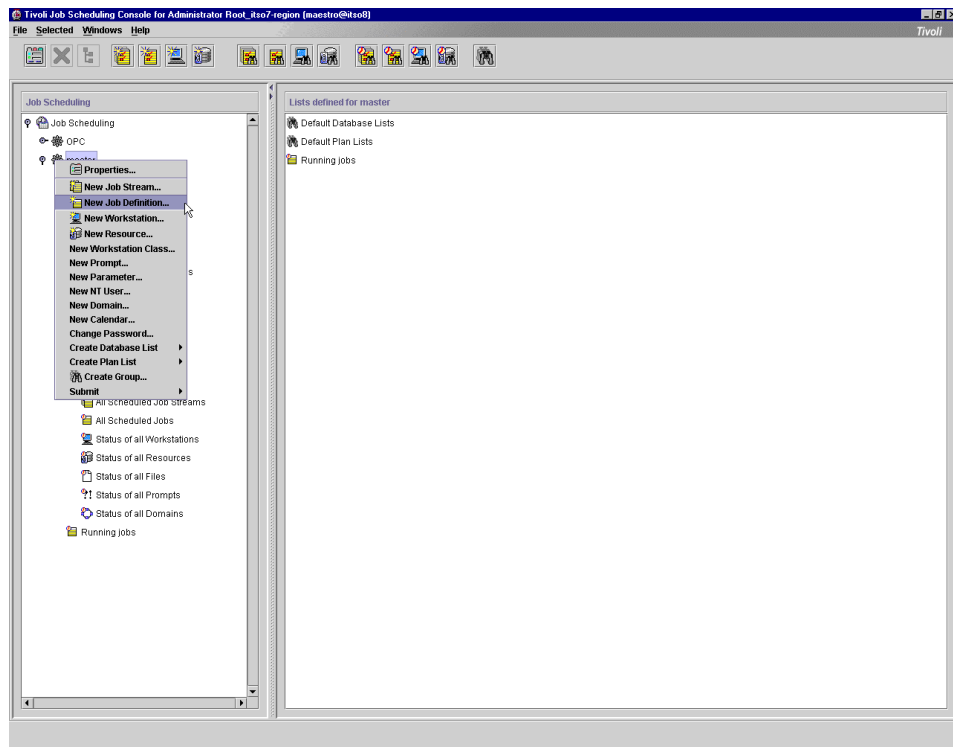


Figure 159. Create a new job definition in TWS

2. Select **Extended agent** task type as shown in Figure 160.

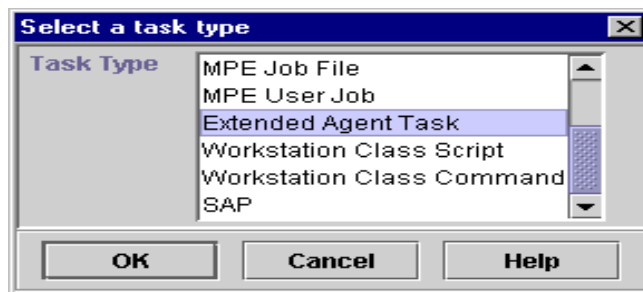


Figure 160. Task type selection panel

You will get a list of already-defined Extended Agent Workstations (except for the SAP Extended Agents, which have a task type of their own). Make sure that you select the right one. The list shows two extended agent workstations: AIXJES and AIXMVS. We named the workstation for the JES method AIXJES, and AIXMVS represents the OPC method.

Different methods are explained in detail in Chapter 2 of the *TWS Extended Agent for MVS and OS390*, GC32-0642.

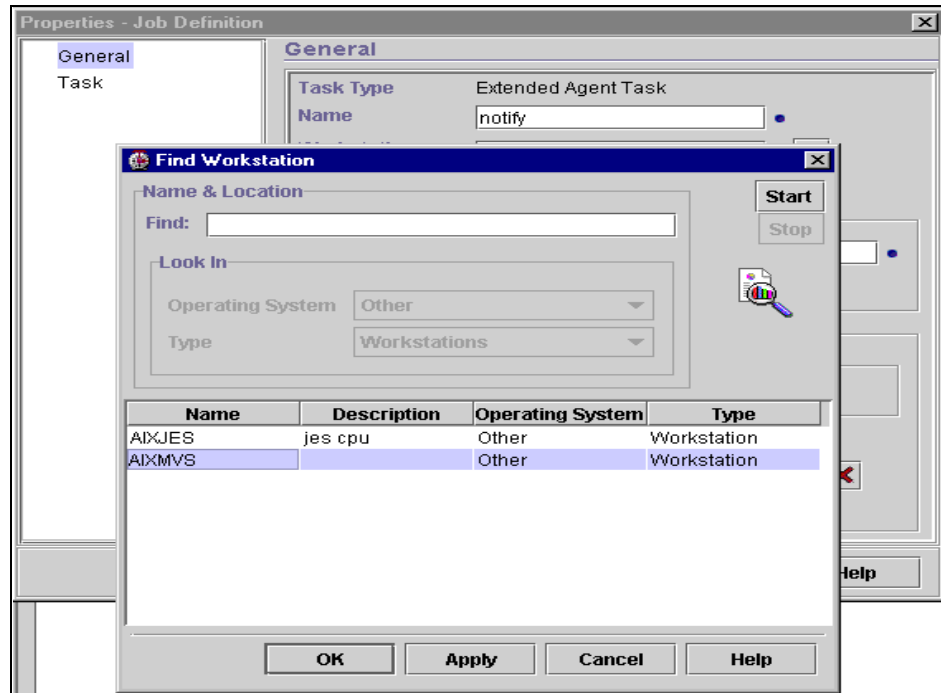


Figure 161. Extended Agent selection list

3. Follow the syntax provided by the Extended Agent manual exactly; otherwise, the job will abend.

In our example, the TWS job submits an OPC job stream, COLLECT#01, with an input arrival time of 19.00h and a deadline of 23.00h, with the highest priority (9) in the current plan. Figure 162 and Figure 163 on page 251 show the details of the Extended Agent job, NOTIFY.



**Properties - Job Definition**

**General**

Task Type: Extended Agent Task

Name: NOTIFY

Workstation: AIXMVS

Description: starts OPC appl. COLLECT#01

Login: ^LOGON^

Add Parameter...

Recovery Options

Action: ☒ Stop ☐ Continue ☐ Rerun

Message:

Job:

Workstation:

OK Cancel Help

Figure 162. General Job definition properties for Extended Agent Job, NOTIFY

**Properties - Job Definition**

**Task**

Task Type: Extended Agent Task

Task: collect#01 iatime(1900) deadlinetime(2300) priority(9)

Add Parameter...

OK Cancel Help

Figure 163. Syntax for adding OPC application, COLLECT#01

4. Press **OK** to save and exit the job definition.

This usually ends the definition of an Extended Agent job. You can include this job in any jobstream you like and make it a predecessor or successor of other jobs or jobstreams.

#### 8.4.4 Implementation

The following is a description of how we implemented the solution.

On the OPC side, we created the accounting application as a job stream and called it *ACCOUNT#01*. We used the rule-based run cycle to insert the job stream into the plan on Saturday only as shown in Figure 164.

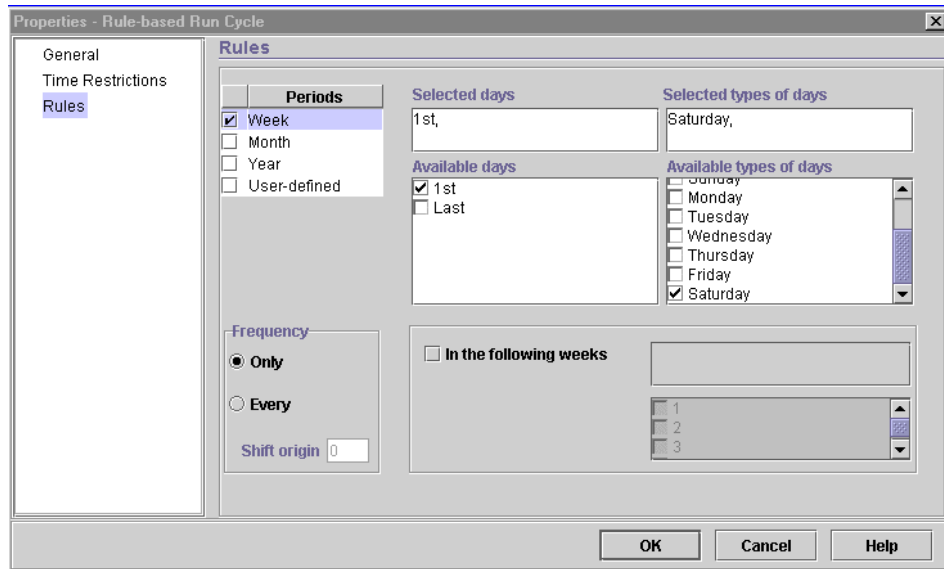


Figure 164. Rule-based run cycle for job stream account#01

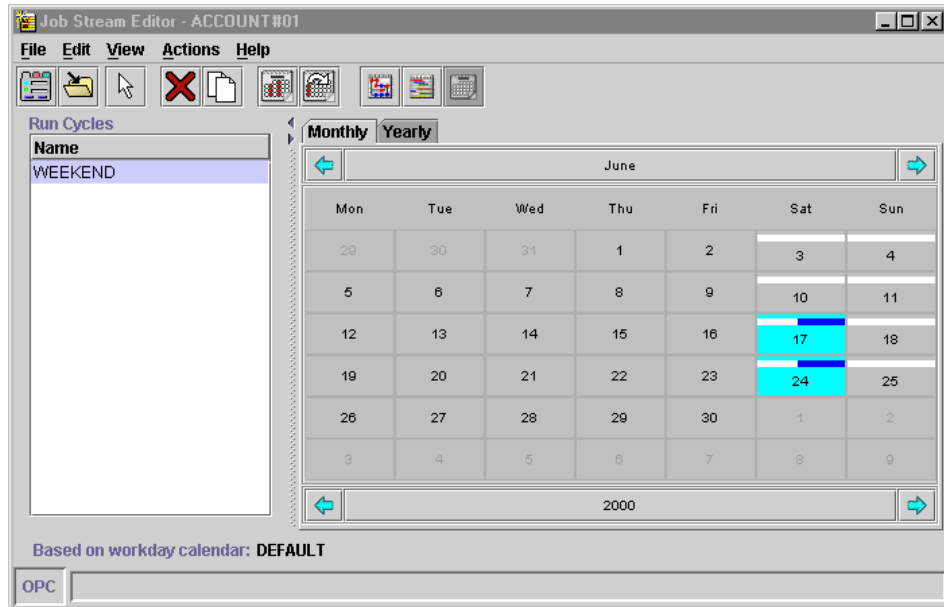


Figure 165. Graphical display of the run cycle used

The accounting application must update the sales data in TWS, which means that OPC must submit a TWS job stream. This can be realized with the help of an OPC Tracker Agent, connected to the OPC Controller, where the executed script accesses the TWS engine through its command line interface (CLI).

You can either let the OPC Controller transfer the whole script to the tracker agent or just have the Controller execute the script already residing on the agent as we did. We used the explicit path name to execute the submit1 script.

```
000001 # SCRIPT THAT LIES ON A AIX TA, WHICH ACCESS THE CLI OF TWS
000002 # AND SUBMIT a job stream
000003 /opt/maestro/scripts/submit1
```

Figure 166. Content of script submitted by OPC Controller to Tracker Agent

We used the above-mentioned script invoked by OPC via the Tracker Agent to submit the schedule *statistic* to the MASTER. The following screen contains the contents of the script, `/opt/maestro/scripts/submit1`.

```
#!/bin/ksh
#
# submit job stream STATISTIC
su - maestro -c "conman submit sched=MASTER#statistic"
rc=$?
exit $rc
```

Figure 167. Contents of the /opt/maestro/scripts/submit1 script

On the TWS side, we defined a job stream, called STATISTIC, running on the MASTER. It includes a *sales\_data* job, which is the predecessor of a *calculate* job as shown in Figure 168.

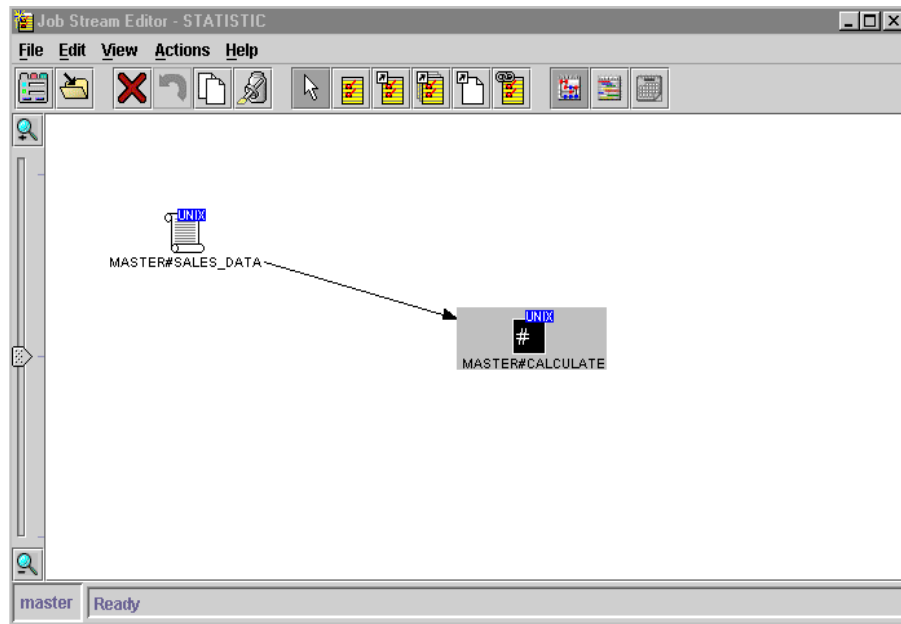


Figure 168. Job stream STATISTIC

The *sales\_data* job is a dummy job and just does a sleep for 10 seconds.

The *calculate* job checks the variation against a defined threshold. When the threshold is reached, it sends a notification to OPC, which, in turn, starts a country-wide collection application. A possible solution could be that the *calculate* job submits a TWS job on an Extended Agent for OS/390 workstation when the variation exceeds a certain level. This extended agent job adds an OPC job stream into the current plan.

We have simulated this behavior by having the CALCULATE job execute the shell-script *threshold*, which submits a job on the Extended Agent for OS/390 workstation in TWS if the variation exceeds a certain limit. A detailed description follows.

Figure 169 and Figure 170 on page 256 show the General and Task Properties of the job, CALCULATE, as defined in TWS.

The screenshot shows a window titled "Properties - Job Definition" with a close button in the top right corner. On the left is a tree view with "General" selected and "Task" below it. The main area is the "General" tab, which contains the following fields and options:

- Task Type:** Unix Command
- Name:** CALCULATE
- Workstation:** MASTER (with a browse button "...")
- Description:** calculate variation
- Login:** maestro (with an "Add Parameter..." button below it)
- Recovery Options:**
  - Action:** Three radio buttons: ☒ Stop, ☐ Continue, ☐ Rerun
  - Message:** (empty text box)
  - Job:** (empty text box with a browse button "...")
  - Workstation:** (empty text box)

At the bottom are three buttons: OK, Cancel, and Help.

Figure 169. General job properties of the CALCULATE job

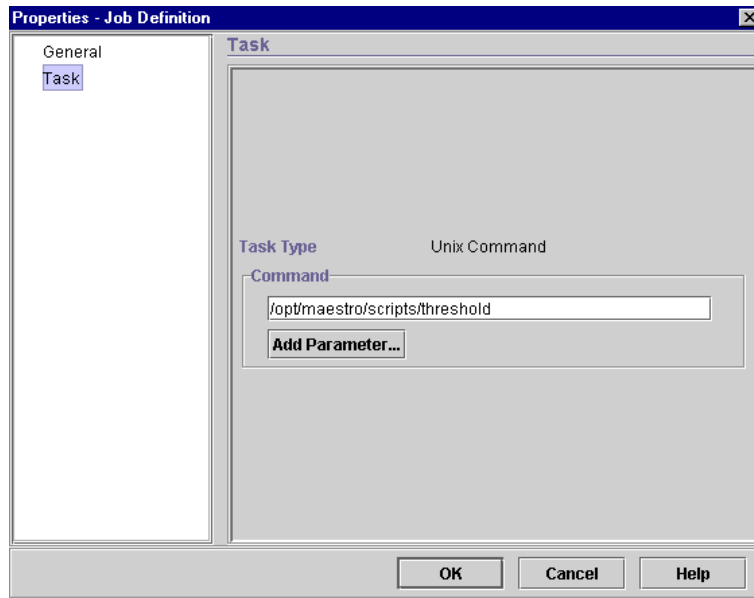


Figure 170. Task job properties of the CALCULATE job

The CALCULATE job needs to be finished by 0800 hours on the following Monday. Therefore, it needs a deadline and a delay of two days. Figure 171 on page 257 shows the Properties - Job / Time Restrictions window of the job, CALCULATE, invoked from the Job Stream Editor of the jobstream, STATISTIC.

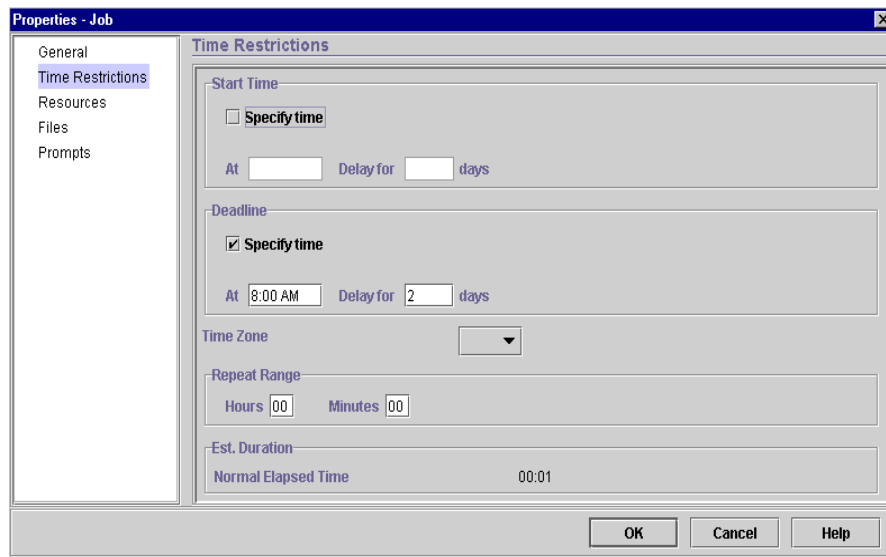


Figure 171. Deadline of CALCULATE job as defined in Job properties job stream

Next, we describe the script, threshold, which is executed by the CALCULATE job shown in Figure 170 on page 256. In Figure 172, you find a listing of the shell script, /opt/maestro/bin/threshold. As you can see, we submit the job, NOTIFY, on the Extended Agent for OS/390 workstation-AIXMVS when the variation exceeds the limit. In our script, it always does.

```
#!/bin/ksh
#
# script used for job "calculate"
# if (simulated) variation is higher than 5, than
#   submit job "notify" on AIXMVS (Ext. Agent for OS/390)
#
# echo "dummy job - calculate variation VAR - "
VAR=6

if [ "$VAR" -gt "5" ]
then
  echo "Variation > 5, submitting job AIXMVS#NOTIFY"
  /opt/maestro/bin/conman submit job=AIXMVS#NOTIFY
fi
rc=$?
exit $rc
```

Figure 172. Listing of the /opt/maestro/scripts/threshold script

Figure 173 and Figure 174 on page 259 describe the NOTIFY job as defined in TWS.

The screenshot shows a Windows-style dialog box titled "Properties - Job Definition". On the left is a tree view with "General" selected under a "Task" category. The main area is the "General" tab, which contains the following fields and controls:

- Task Type:** Extended Agent Task
- Name:** NOTIFY
- Workstation:** AIXMVS (with a browse button "...")
- Description:** start application collect#01
- Login:** ^LOGON^ (with an "Add Parameter..." button)
- Recovery Options:**
  - Action:** Three radio buttons: ☒ Stop, ☐ Continue, ☐ Rerun
  - Message:** (empty text box)
  - Job:** (empty text box with a browse button "...")
  - Workstation:** (empty text box with a browse button "...")

At the bottom of the dialog are three buttons: "OK", "Cancel", and "Help".

Figure 173. General properties of the NOTIFY job

As you can see in Figure 174 on page 259, the job, NOTIFY, starts the COLLECT#01 application in OPC with an input arrival time of 19:00h, a deadline time of 23:00h, and the highest priority.



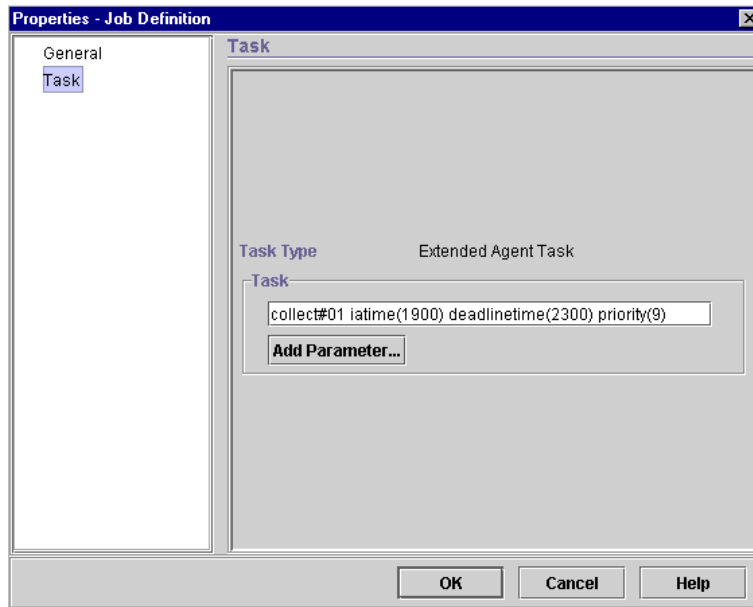


Figure 174. Task properties of the NOTIFY job

Back in OPC, the OPC jobstream, COLLECT#01, must run Monday night after 19.00h and be finished before 23.00h. We use a special resource, called AMI\_MAX, which is only available between 19 and 23h. The jobstream, called COLLECT#01, depends on this special resource. Figure 175 and Figure 176 on page 260 show the details of this special resource.

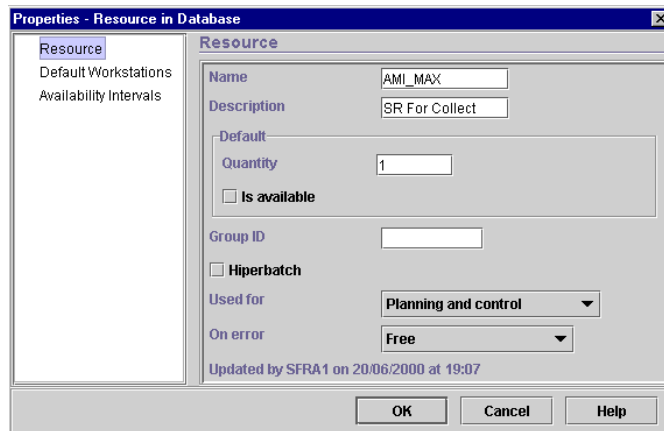


Figure 175. Resource properties of the AMI\_MAX Special Resource

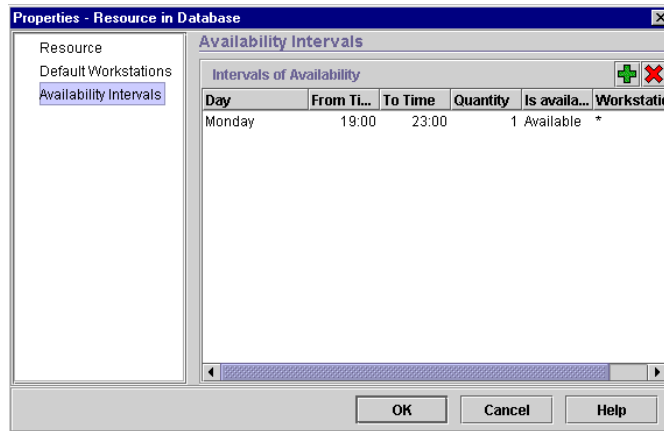


Figure 176. Availability interval properties of the AMI\_MAX special resource

### 8.4.5 Demonstration run

In the following figures, we will show in detail what happened on OPC and TWS when starting the above-mentioned sequence by manually submitting the OPC jobstream, ACCOUNT#01, through ISPF.

In Figure 177, we see the running application, ACCOUNT#01, on the OPC console.

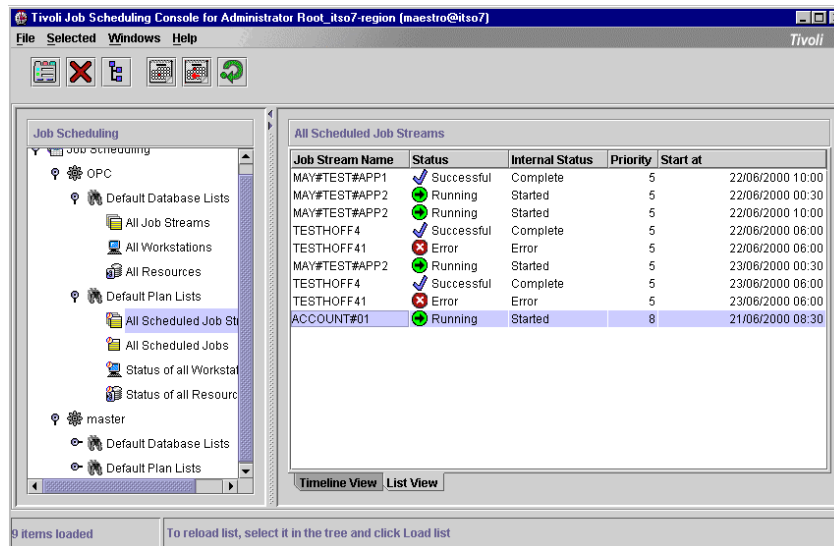


Figure 177. Running the ACCOUNT#01 application on OPC

In Figure 178, we see the jobstream, STATISTIC, running on the MASTER.

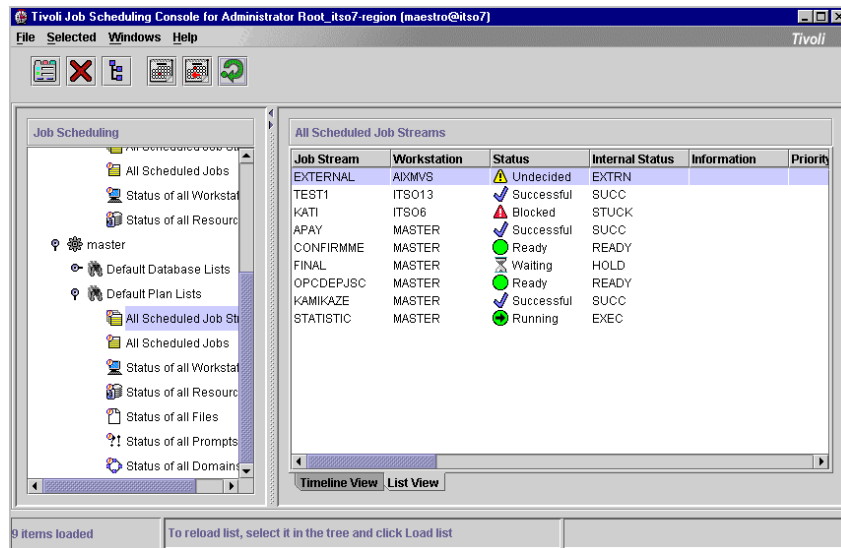


Figure 178. Running the STATISTIC jobstream on TWS (last line)

Figure 179 shows the jobs, sales\_data and calculate, of the STATISTIC jobstream finished and the NOTIFY job waiting.

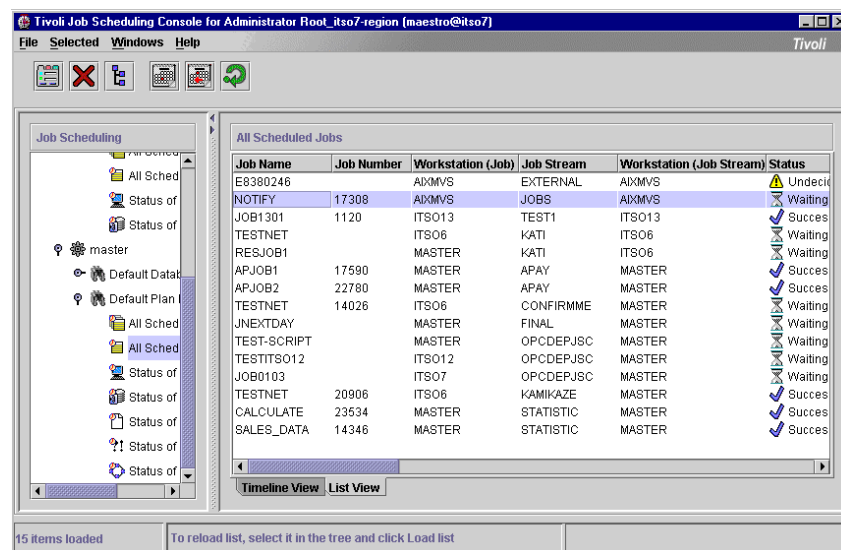


Figure 179. Status of the STATISTIC jobstream and job notify on TWS

Why is the notify job waiting? This job is running on the Extended Agent for OS/390 workstation-AIXMVS and submits the OPC jobstream, COLLECT#01, which is waiting on a special resource, AMI\_MAX, which is only available on Monday between 1700 and 2300 hours (see Figure 176 on page 260); so, both the TWS job notify as well as the OPC jobstream, COLLECT#01, are waiting for this resource to become available. In real life, this resource will, eventually, become available. In our test case, since it was not Monday evening, we enabled it manually.

Figure 180 and Figure 181 on page 263 show the status of the OPC jobstream, COLLECT#01, as well as the OPC special resource, AMI\_MAX, before the special resource was made available.

Job Stream Name	Status	Internal Status	Priority	Start at
MAY#TEST#APP1	Successful	Complete	5	22/06/2000 10:00
MAY#TEST#APP2	Running	Started	5	22/06/2000 00:30
MAY#TEST#APP2	Running	Started	5	22/06/2000 10:00
TESTHOFF4	Successful	Complete	5	22/06/2000 06:00
TESTHOFF41	Error	Error	5	22/06/2000 06:00
MAY#TEST#APP2	Running	Started	5	23/06/2000 00:30
TESTHOFF4	Successful	Complete	5	23/06/2000 06:00
TESTHOFF41	Error	Error	5	23/06/2000 06:00
ACCOUNT#01	Successful	Complete	8	21/06/2000 08:30
COLLECT#01	Waiting	Waiting	9	22/06/2000 19:00

Figure 180. Status of COLLECT#01 OPC jobstream waiting for special resource

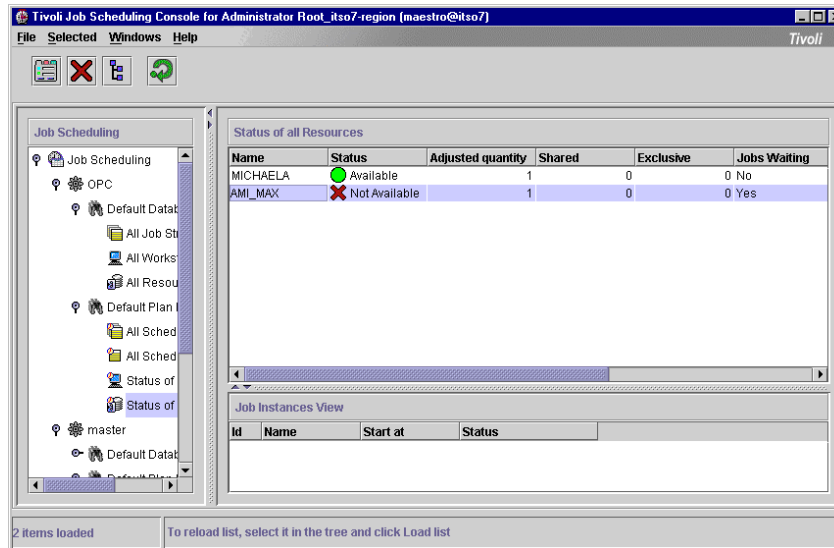


Figure 181. Status of the AMI\_MAX OPC special resource

After making the special resource available, both the OPC jobstream and the TWS job will finish successfully as shown in Figure 182.

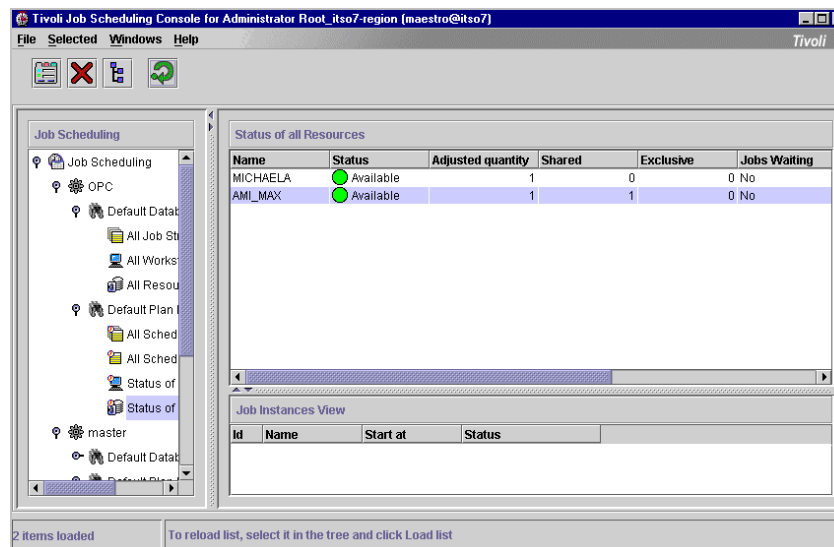


Figure 182. OPC special resource made available

The OPC jobstream finished successfully as shown in Figure 183.

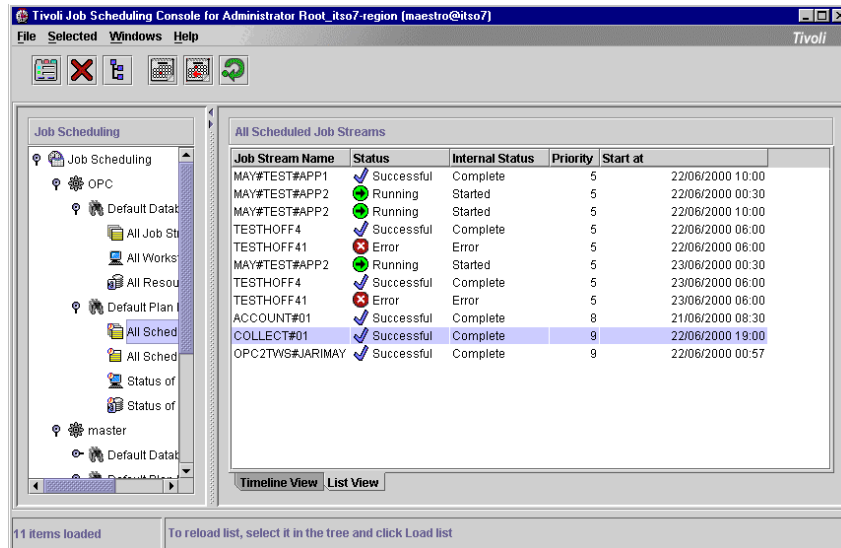


Figure 183. OPC jobstream, COLLECT#01, finished successfully

As a result of the successful completion of the COLLECT#01 OPC jobstream, the TWS job also finishes as shown in Figure 184.

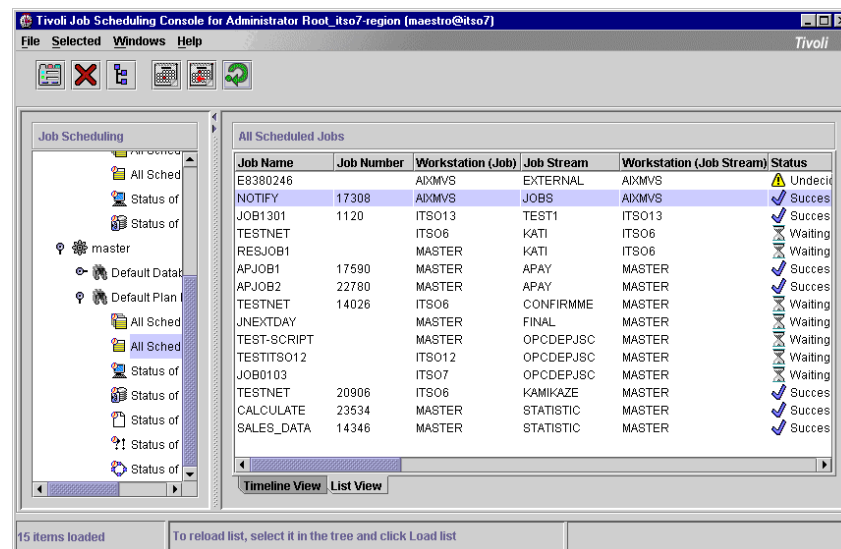


Figure 184. Notify TWS job finished successfully

This demonstration is supposed to show you one way of controlling TWS through OPC as well as controlling OPC through TWS. We tried to make the example as simple as possible yet include all the major steps necessary to implement a customer solution.

---

## **8.5 User interaction with OPC and TWS**

This scenario considers an industry that has a central accounting office and several technical departments in various cities.

The accounting office has an OS/390 parallel sysplex, DB/2 database, several business applications, and OPC that schedules and controls the enterprise workload.

The technical department's administrative offices have several PCs and a UNIX server running several business applications and TWS that schedules and controls the workload in the department environment.

An administrator wishes to run a salary procedure in the departmental area as soon as a business application on the central mainframe ends. The administrator wants to control the job execution on both environments from a central site.

The administrator has to perform the following tasks:

1. Define a job to OPC to execute the business application during the last weekend of every month when the required data is available in the central database.
2. Define a job to the TWS Job Scheduler to execute the salary procedure when the OPC has terminated the execution of the business application.
3. Monitor the status and the execution of those activities on the two job schedulers from a central point of control.

### **8.5.1 Network for the scenario**

Figure 185 on page 266 shows the part of our network used for this scenario.

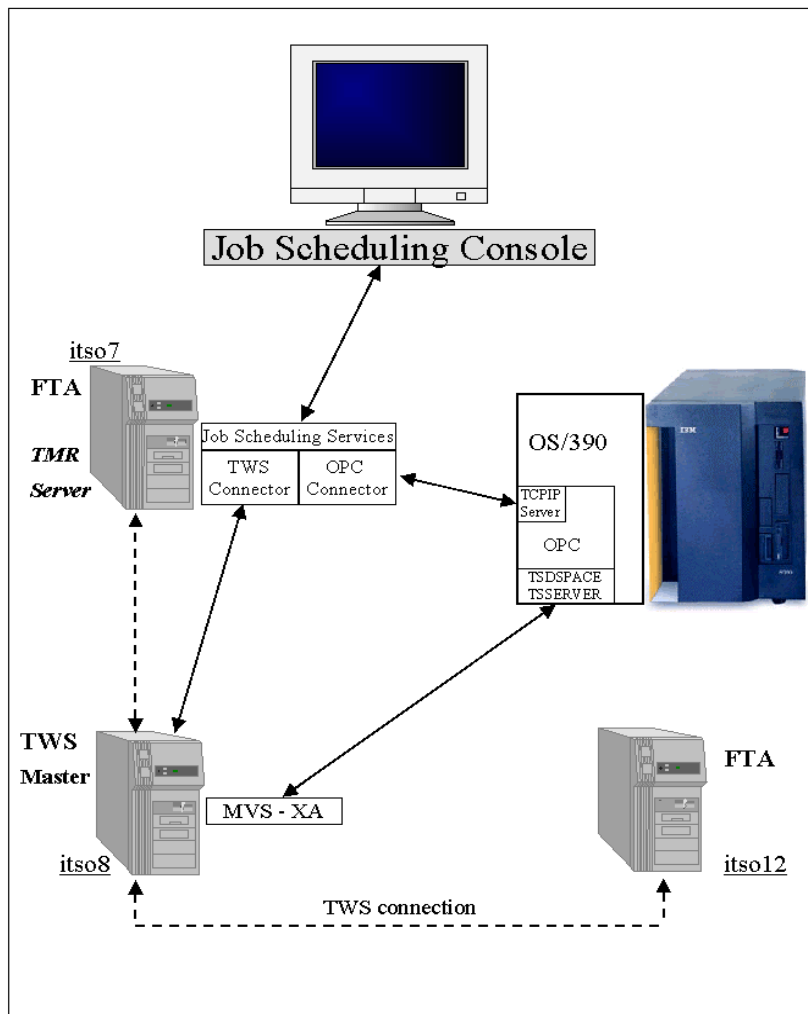


Figure 185. Network used for this scenario

The MVS extended agent is running on the TWS Master, and events on OPC are passed from the TSSERVER and TSDSPACE started tasks, thus enabling monitoring of OPC jobs from JSC. The NT machine where one of the TWS jobs runs is itso12, and the UNIX jobs are running on its07 and its08, which is the TWS Master.



## 8.5.2 The solution

The solution for this problem is quite simple. We can use the JSC to create an internetwork dependency and make the TWS salary procedure dependent on the successful completion of the OPC application.

The JSC also enables us to monitor the status and execution of both the OPC and the TWS job streams, thus, being our central control point.

### 8.5.2.1 Defining the jobs

We used the JSC to define an OPC job stream and scheduled it to run as requested, and we used the JSC to define a TWS job stream that is dependent on the successful conclusion of the OPC job stream. The creation of job streams is documented elsewhere in this book and in the respective manuals; so, we will not take up space repeating that information here. However, making the dependency link from TWS to OPC is not clearly documented; so, the next section will show the creation of that dependency.

### 8.5.2.2 Making an internetwork dependency

Click on the white internetwork dependency icon on the menu bar as shown in Figure 186.

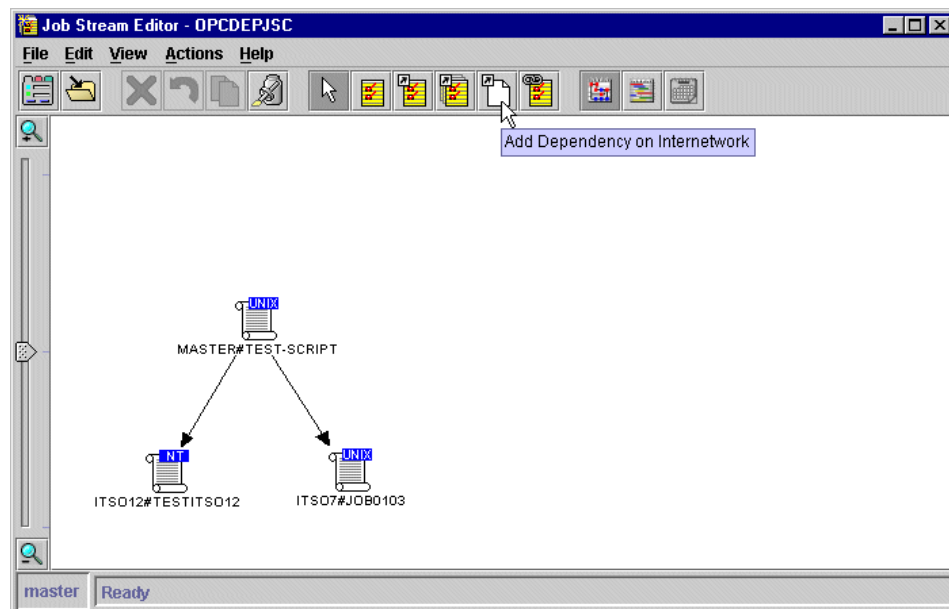


Figure 186. Selecting internetwork dependency

As you move the cursor down into the frame, the pointer changes to a cross. Click where you want the icon for the predecessor to be placed, and a dialog box will appear as shown in Figure 187.

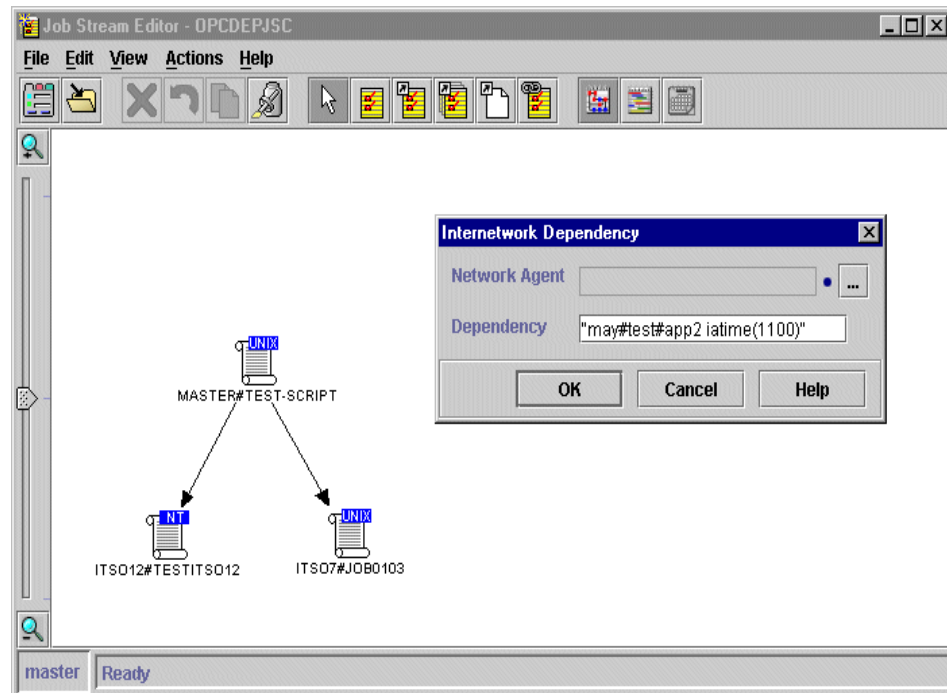


Figure 187. Creating the internetwork dependency

To identify the OPC job stream, you can use the application name and input arrival time as we did here. The application name is required, and you can use any combination of the following:

- **JOBNAME** - This is the OPC operation name.
- **OPNO** - This is the operation sequence number.
- **IA** This is the input arrival date and time in a *yyymmddhhmm* format. It is unlikely you will use this parameter because it will hardcode the dependency to one specific date.
- **IATIME** - This is the Input Arrival Time in an *hhmm* format.

**Note**

The entire string must be enclosed in double quotation marks.

The TWS workstation that represents OPC must also be named. Click on the box at the right of the *Network agent* field, and a Find workstation dialog window opens as shown in Figure 188.

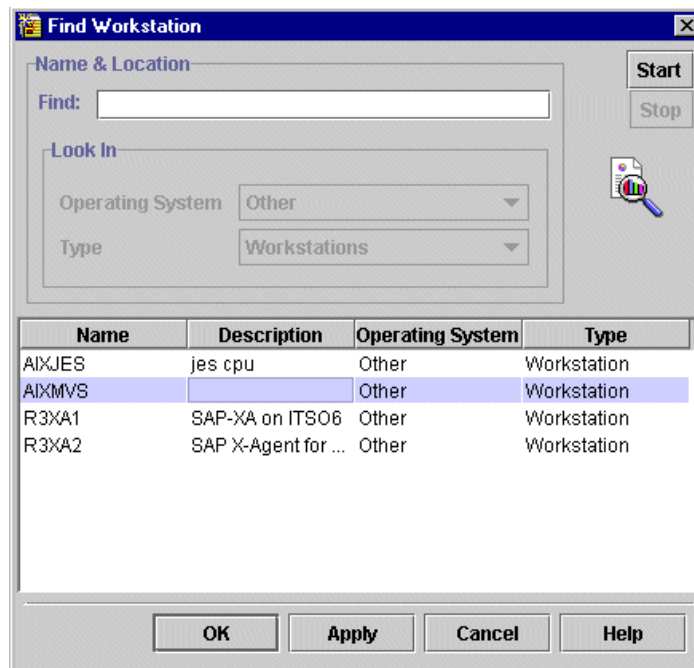


Figure 188. Selecting the OPC workstation

Clicking on **Start** to show a list of workstations, and select the workstation that represents the extended agent on the OPC processor. On our system, that is named AIXMVS.

The internetwork dependency icon now appears. You must connect it to the job stream. You can do this by selecting the **Actions** item on the menu bar and then selecting **Add dependency** or by clicking on the **Link** icon on the menu bar as shown in Figure 189 on page 270.

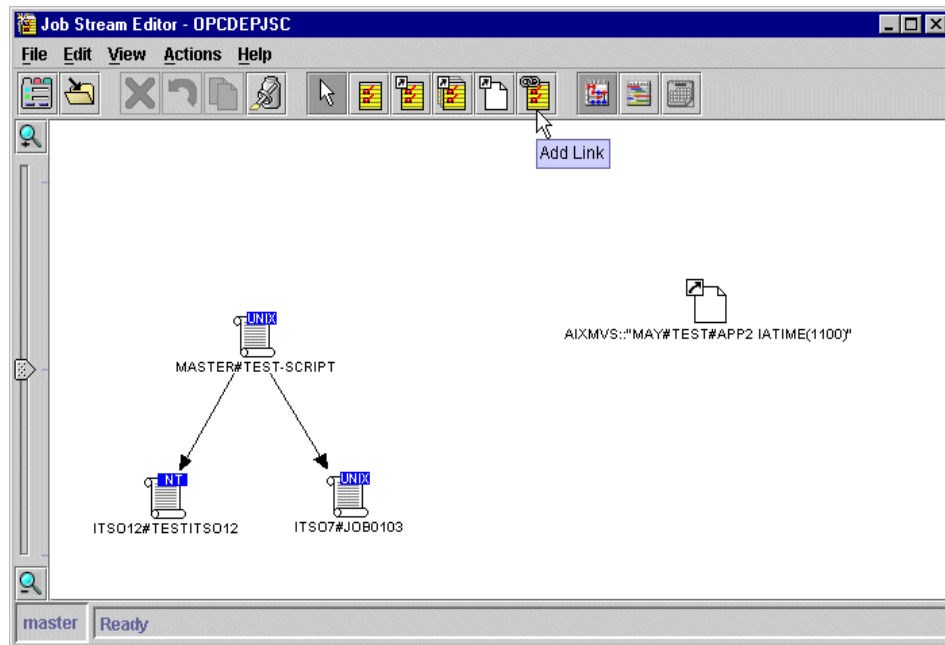


Figure 189. Linking OPC Predecessor to TWS Job Stream

Click on the **predecessor** icon, and pull the dependency arrow to the successor icon to create a link as shown in the completed job stream in Figure 190 on page 271.

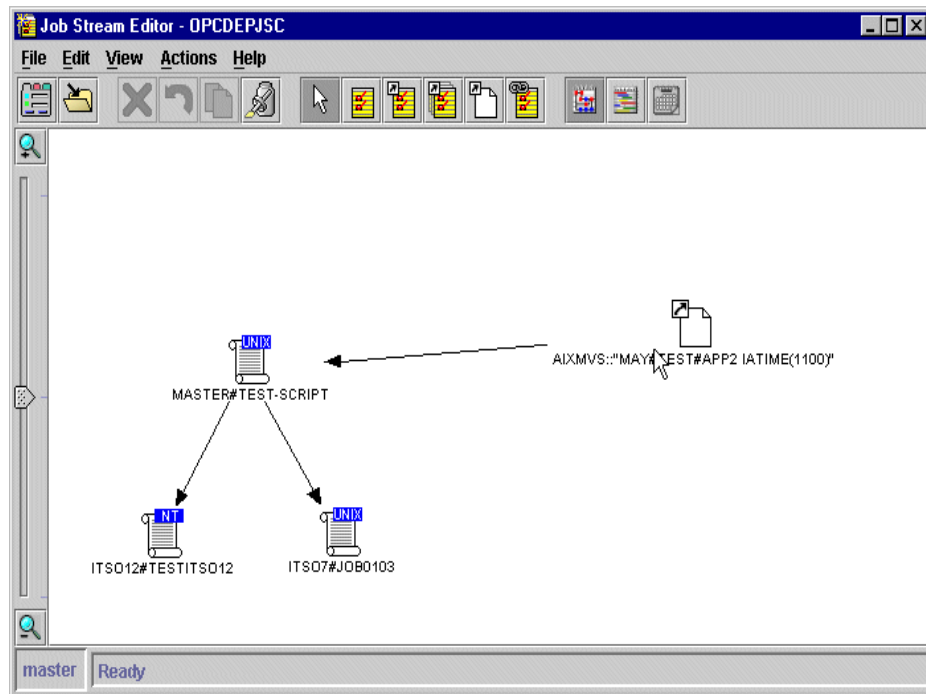


Figure 190. TWS Job Stream

You can realign the icons by dragging them into position, or you can right-click in the window and select **Arrange Icons** from the pop-up menu as shown in Figure 191 on page 272.

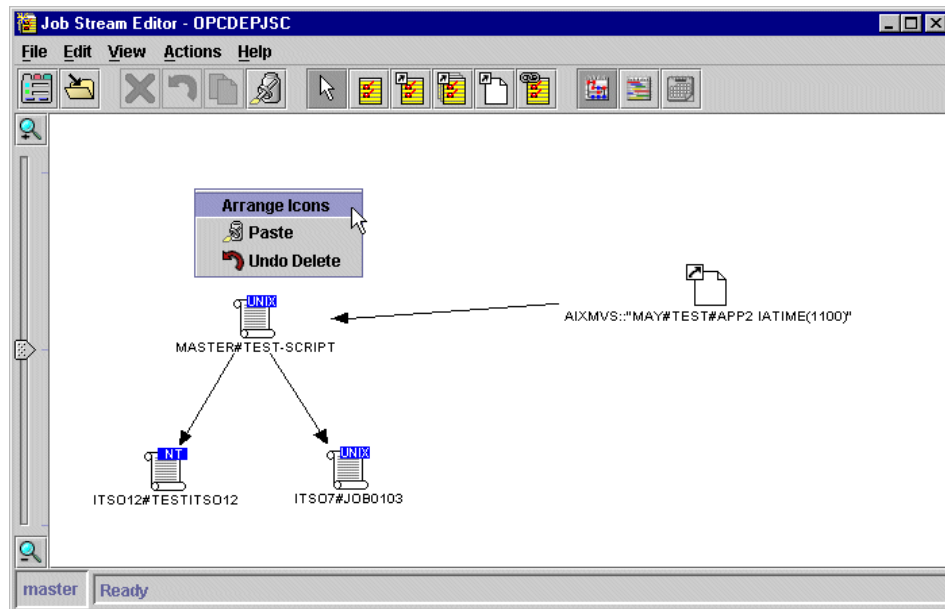


Figure 191. Arranging Job Stream icons

The job stream is now complete as shown in Figure 192 on page 273.

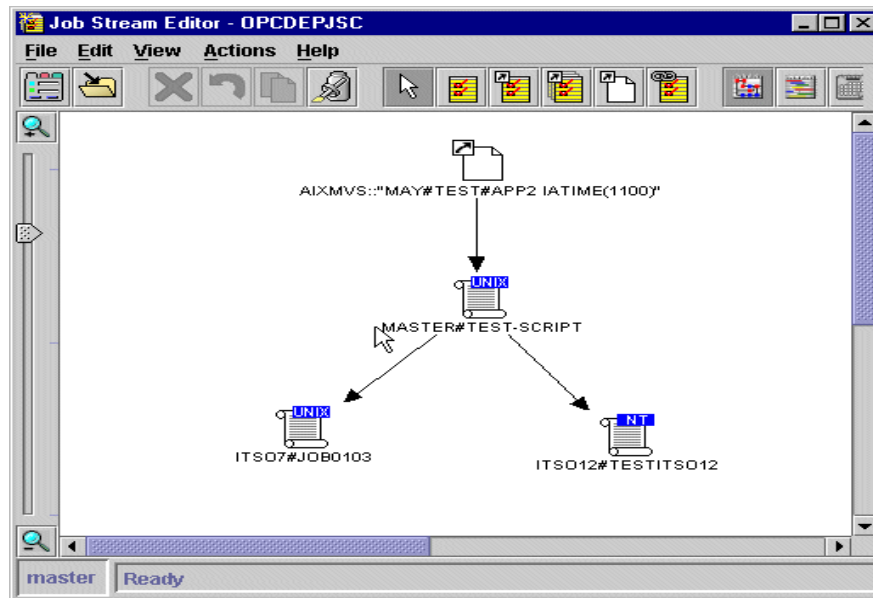


Figure 192. TWS Job Stream Dependent on an OPC application

### 8.5.2.3 Central point of control

The JSC provides the central point of control. We can monitor the status of both the OPC and TWS job streams. In Figure 193 on page 274, we can see that the OPC application is running with an OPC status of *Started*.

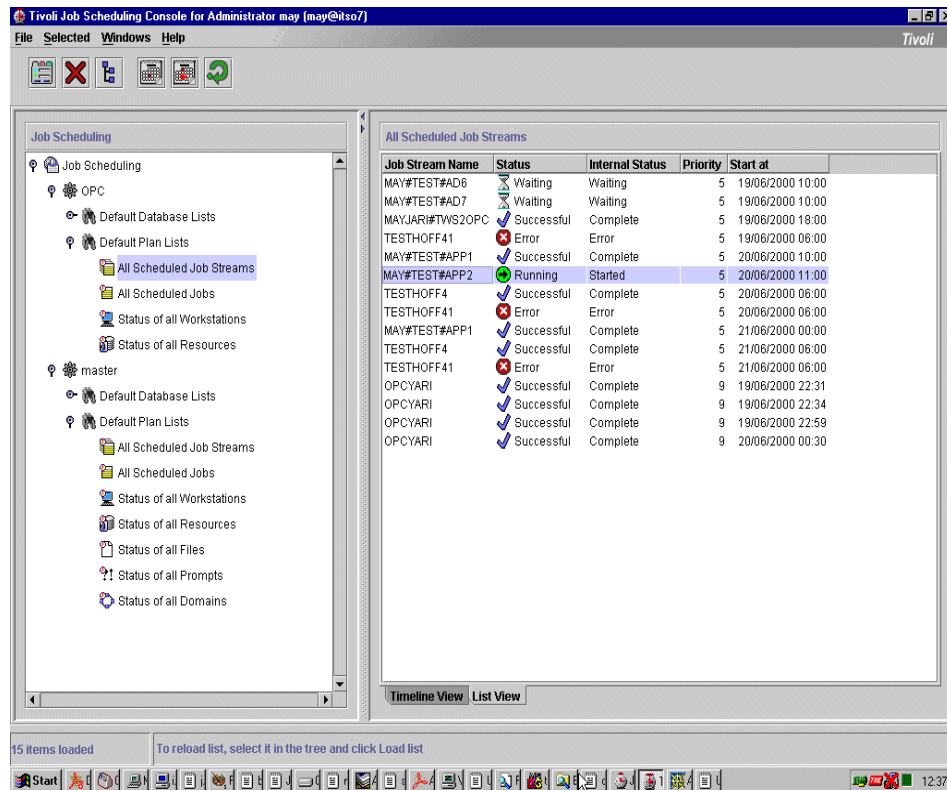


Figure 193. Monitoring the OPC Job Stream

We chose this view by selecting **All Scheduled Job Streams** from the OPC Plan Lists on the menu in the left pane. By selecting **All Scheduled Job Streams** from the Master Plan Lists, on the same menu, we can see the TWS jobs stream shown in Figure 194 on page 275.



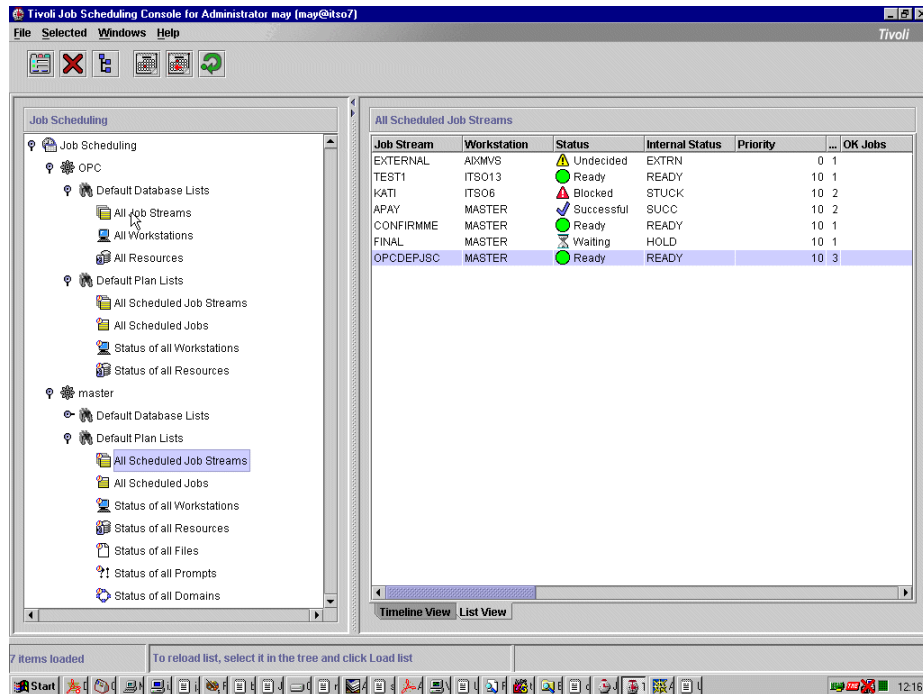


Figure 194. Monitoring the TWS Job Stream

The TWS job stream is in *Ready* status. It will not start until the OPC job stream successful completes.

#### 8.5.2.4 Conclusion

The JSC provides a simple and effective mechanism for making a TWS job dependent on an OPC job, and the JSC allows us to monitor the progress of each job stream.

In this first JSC release, the dependency link works only one way. At this moment, you cannot similarly make an OPC job depend on a TWS job, but we expect future releases to allow this.

## 8.6 Summary

In this chapter, we put together three different end-to-end scheduling scenarios using OPC and TWS:

- Resource synchronization between OPC and TWS
- Synchronization between OPC and TWS

- User interaction with OPC and TWS

These scenarios were aimed at showing you how to implement an end-to-end enterprise scheduling solution in your environment using OPC and TWS. There are several methods of passing information between both scheduling products, and these methods were explained in detail. Each method has its merits and is particularly suitable for a certain type of environment.

---

## Chapter 9. Tivoli OPC Tracker Agents

This chapter describes the Installation and Operation of OPC Tracker Agents. OPC provides agents for many platforms, and the installation procedure differs depending on the platform you use. Therefore, we decided to concentrate on the AIX version that we used in our test scenario.

Many installations run business applications on a variety of UNIX platforms. Tivoli OPC provides Tracker Agents that can manage your workload in several operating system environments. Currently, UNIX Tracker Agents for HP-UX, SunOS, Sun Solaris, OS/390, UNIX System Services, Silicon Graphics IRIX, Digital OpenVMS, Digital UNIX, AIX, NT, OS/2, and OS/390 operating systems are available. There is also a Tracker Agent for OS/400 and a base Tracker for OS/390. OPC Tracker Agents are functionally equivalent to the base OS/390 tracker with the exception that automatic dataset cleanup and the job-completion checker (JCC) functions are not provided for non-MVS platforms. This means that you can use Tivoli OPC functions, such as automatic recovery and automatic job tailoring, for your non-MVS workload. Refer to the book, *Tivoli Operations Planning and Control V2R3 Getting Started*, SH19-4481, for details about the functions provided by Tivoli OPC.

---

### 9.1 HW/SW requirements for OPC Tracker Agent for AIX/6000

In the following sections, we will give the hardware and software requirements for installing OPC Tracker Agent for AIX/6000, which will, hereafter, be referred to as the Tracker Agent.

#### 9.1.1 Hardware requirements

The Tracker Agent for AIX/6000 requires a RISC Systems/6000 computer with a minimum of 3 MB of RAM and capable of running AIX/6000 Version 4.2.1 or later. For performance reasons, 8 MB of RAM is recommended.

The Tracker Agent also requires approximately 20 MB of disk space for the components. There is additional space on the local hard disk for the log files and other temporary data it generates. *The volume of data is highly-dependent on the volume and output of jobs managed by the Tracker Agent.*

#### 9.1.2 Software requirements

The Tracker Agent for AIX/6000 requires AIX Version 4.2.1 or later.

---

## 9.2 Verifying that TCP/IP is operational

A stable connection between the OPC Controller system and every Tracker Agent is essential because the agent does not run as a fault-tolerant agent (FTA), which continues processing even if the connection breaks.

If you have selected a machine where you want to install the agent, first check if you can ping the Controller where you want to connect. To find the name or IP address of the Controller, ask your system programmer or have a look into the dataset concatenated to //SYSTCPD (TCP/IP Params) of your TCP/IP stack on the OS/390 system.

```
TCPIPJOENAME TCPIP4
HOSTNAME MCEVS4
DOMAINORIGIN NCS.MAINZ.IBM.COM
NSPORTADDR 53
RESOLVEVIA UDP
RESOLVERTIMEOUT 300
RESOLVERUDPRETURNS 1
TRACE RESOLVER
DATASETPREFIX TCPIP
```

In our example, the DNS name of the machine where the Controller resides is MCEVS4.NCS.MAINZ.IBM.COM.

Trying to ping the Controller from your AIX box, you should receive output similar to that shown in the following screen.

```
/ ping mcevs4.ncs.mainz.ibm.com
PING mcevs4.ncs.mainz.ibm.com: (9.39.62.19): 56 data bytes
64 bytes from 9.39.62.19: icmp_seq=0 ttl=63 time=25 ms
64 bytes from 9.39.62.19: icmp_seq=1 ttl=63 time=15 ms
64 bytes from 9.39.62.19: icmp_seq=2 ttl=63 time=12 ms
64 bytes from 9.39.62.19: icmp_seq=3 ttl=63 time=19 ms
64 bytes from 9.39.62.19: icmp_seq=4 ttl=63 time=17 ms
64 bytes from 9.39.62.19: icmp_seq=5 ttl=63 time=19 ms
64 bytes from 9.39.62.19: icmp_seq=6 ttl=63 time=14 ms
64 bytes from 9.39.62.19: icmp_seq=7 ttl=63 time=19 ms
```

If you want to ping the AIX box from the Controller side, you first have to allocate the dataset that contains the TCP/IP parameter to the DD name SYSTCPD. You can do it in ISPF Option 6, the TSO command processor.

Enter TSO or Workstation commands below:

```
==> Alloc fi(systcpd) da('tcpip.iv4.tcparms(tcpdata)') shr
```

Then, you can ping the AIX box in the common way. Double check that you get the following output:

Enter TSO or Workstation commands below:

```
==> ping 146.84.32.100
```

```
Ping CS V2R6: Pinging host 146.84.32.100. Use ATTN to interrupt.  
PING: Ping #1 response took 0.020 seconds. Successes so far 1.  
***
```

For additional information related to the TCP/IP configuration of other platforms, see Chapter 3 of the book, *Tivoli Operations Planning and Control V2R3 Tracker Agent*, SH19-4484.

---

### 9.3 Tivoli OPC Controller initialization statements

The Tivoli OPC Controller ROUTOPTS initialization statement defines the Tivoli OPC configuration. Update the statement to include the UNIX machines on which you will run work scheduled by Tivoli OPC. Review these parameters of ROUTOPTS, and set the values according to your configuration:

- **CODEPAGE** (host system codepage|**IBM-037**) - This keyword specifies the name of the host codepage. The value is used by Tracker Agents running on operating environments that use the ASCII character set to convert submitted input data to ASCII. Up to 8 characters can be defined. The default value, IBM-037, defines the EBCDIC codepage for U.S. English, Portuguese, and Canadian French.

#### Note

OPC internally reserves space only for one codepage. So if you are using multiple code pages the last defined one will be used by OPC.

- **TCP** (destination,...,destination) - This keyword specifies the network addresses of all TCP/IP-connected Tracker Agents that are able to communicate with the Controller for job-tracking purposes. Each

destination consists of a destination name and an IP address separated by a colon (name: nnn. nnn. nnn. nnn). The name consists of one to eight alphanumeric characters, and the first character is alphabetic. The IP address consists of four numeric values separated by periods. Each value is in the range 1 to 255; leading zeros are not required.

**Note**

If the keyword is not defined, support for TCP/IP-connected Tracker Agents will not be activated.

- **TCPIPID** (TCP/IP ID|**TCPIP**) - This keyword identifies the name of the TCP/IP address space on the MVS system where the Controller is started. If you do not specify this keyword, the default value TCPIP is used.
- **TCPIPPORT** (TCP/IP port|**424**) - This keyword defines the TCP/IP port number used by the Controller. The number is 1–5 numeric characters. The Controller reserved port number is 424. This keyword must be specified if the TCP keyword has been defined and the default port number is not used, for example, when you start more than one Controller on the same MVS image that uses TCP/IP or when the port number is already in use. If you use a port number less than 1024, you must run the tracker as root.
- **TCPTIMEOUT** (TCP/IP time-out interval|**5**) - This keyword specifies the time interval within which the Controller expects a TCP/IP-connected Tracker Agent to respond to a submit request. If the Tracker Agent does not respond in two consecutive intervals, the session is terminated and workstations that reference the destination are set offline. When the tracker becomes active again, or if the time-out was caused by poor network response, the session will be reestablished automatically. The time-out processing comes into effect after the Controller and the Tracker Agent have synchronized at startup. Specify a number of minutes from 1 to 60, or specify 0 if you do not require time-out processing. The default time-out interval is five minutes.
- **ROUTOPTS** Example

```
ROUTOPTS TCP(TWS1:146.84.32.100)
          TCPIPID(TCPIP4)
          TCPIPPORT(3112)
          CODEPAGE(IBM-037)
```

- Tivoli OPC communicates with AIX machines running the Tracker Agent. The communication method is TCP/IP. Operations that specify a computer

workstation with destination TWS1 are transmitted to IP address 146.84.32.100 for execution.

- TCPIP4 is the name of the TCP/IP address space on the MVS system where the Controller is started, and 3112 is the port number to which it connects.
- The codepage used on the MVS system where the Controller is started is IBM-037, the codepage for US English, Portuguese, Canadian, French.

You need to tell OPC what actions to be taken if a workstation fails and the maximum return code to set an operation to complete. This must be done in the job tracking options of the Controller. If you need information about the OPC initialization parameter, refer to the *Tivoli Operations Planning and Control V2R3 Customization and Tuning Guide*, SH19-4380.

#### **JTOPTS example**

```
JTOPTS WSFAILURE(LEAVE,REROUTE,IMMED)

WSOFFLINE(LEAVE,REROUTE,IMMED)

HIGHRC(0)
```

The highest return code generated in a job without causing the operation to set to error is 0.

The default is 4. For non-MVS Tracker Agents, specify 0. You can also specify this return code for each operation in the AUTOMATIC OPTIONS section of the Application Description dialog.

When you want to see the standard list (std) output via the joblog retrieval function in OPC dialog, you have to use catalog management with the storelog parameter. Both parameters have to be defined in the OPCOPTS Controller statement. See also the job\_log parameter of the tracker parm.

#### **OPCOPTS example**

```
OPCOPTS CATMGT(YES)

STORELOG(ALL)
```

---

## 9.4 Planning your Tracker Agent installation

The Tracker Agent is flexible and lets you define a configuration to suit your needs. If your enterprise has a large number of networked UNIX machines, you should run the Tracker Agent on several machines. For availability, it is recommended that you start the tracker on at least two machines and specify more than one host for each job. When the first destination (workstation) is unavailable, the Controller can send the job to the next available one. How you install the Tracker Agent depends on your configuration. If you have Network File System (NFS) or Network Information System (NIS) installed, you will be able to carry out most of the installation from one machine; otherwise, you will need to install the Tracker Agent on every machine where the Controller will start work. To install, you need superuser (root) authority. Depending on your network, you may need root authority on the other servers. Certain programs for administration and operation of the Tracker Agent can only be run by the system administrator logged in with certain user IDs. Table 17 lists the user IDs and groups that you need.

*Table 17. User IDs and groups needed*

Create user	For...	Group	Recommended home directory
tracker	running the agent	OPC	default /u/tracker

Even if you plan to run several instances of the Tracker Agent for the same type of Controller on the same machine, you should run them all under the same user ID.

You must use port numbers above 1024 to avoid running the Tracker Agent with root authority. You should use port numbers much higher than this (for example, above 5000) to avoid conflict with other programs.

Even if you plan to run several instances of the Tracker Agent for the same type of Controller on the same machine, you should run them all under the same user ID. A Tivoli OPC Controller needs one port, TCPIPPORT, which is also the tracker's Controller port. The default port number is 424. Each Tracker Agent needs two ports:

- The tracker's Controller port, which is also the Controller's tracker port (TCPIPPORT).
- The tracker's local port, which must be unique for each machine. Tracker Agents on different machines can have the same local port number. See Figure 195 on page 283.



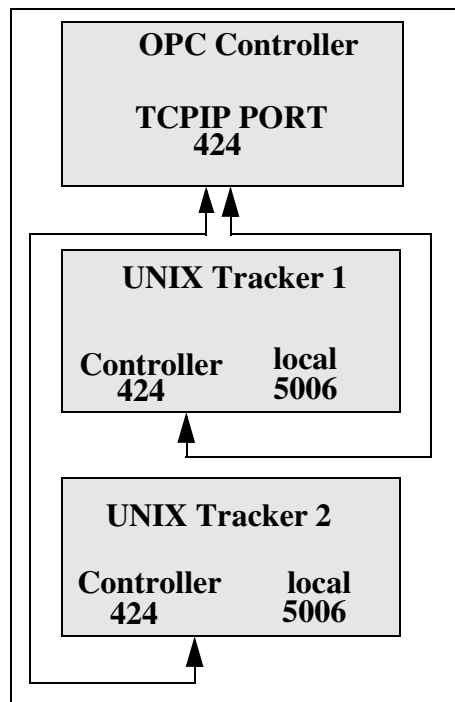


Figure 195. Ports

---

## 9.5 Creating a user group and user IDs

As a security measure, the Tracker Agent requires that users belong to the OPC user group. The administrator does this by using the UNIX file-privilege mechanism, which makes the Tracker Agent programs executable only by users in this group. The creation of UID and groups differs from each platform you use. How to create the UID and Group ID for the operating system you use is explained in detail in the “Planning your Tracker Agent” chapter of the book, *Tracker Agents for AIX, UNIX, VMS and OS/390 Open Edition*, SH19-4484.

---

## 9.6 Installing and customizing the Tracker Agent

Before installing the Tracker Agent, make sure that you have the *latest fix level* available to install. A fix level is not only a summary of known problems fixed since last level, but it also contains functional enhancements. Ask your local IBM support for the latest version. The fix level is shipped in SMP/E

format, and resides in dataset SEQQEENU. For every platform, SEQQEENU contains different member names. The member name for AIX is called EQQTXAIX.

The Tracker Agent files must be transferred from the Controller host. Any file transfer program can be used. It is recommended that you use the TCP/IP FTP program because this is a good way to test whether the TCP/IP connection works properly. You can send the Tracker Agent files to the Tracker Agent machine, or you can receive them from the Controller machine.

To receive files from the Controller machine, enter the commands listed in the following screen on the Tracker Agent machine:

```
cd /usr/sys/inst.images
ftp control
user opc
passwd xxxxx
binary
get 'OPCESA.INST.SEQQEENU(EQQTXAIX)' tracker.image.aix
quit
```

In these examples, the Controller machine is `control`. You can receive the image to any directory. `/usr/sys/inst.images` is the recommended and default directory. `tracker.image` is an installp image.

---

## 9.7 Using SMIT to install the required features (AIX only)

We used the AIX Tracker Agent in our scenarios. Now we will describe how to install the Tracker Agent using the SMIT install facility. If you are using other UNIX platforms refer to the administration tools they use, in *Tracker Agents for AIX, UNIX, VMS and OS/390 Open Edition*. Consult your administrator for assistance if you have not previously used this tools. There is more than one way to perform this task. For example, you can install all the features from the installation media, or you can copy the software to a hard disk for future installation.

When installing the different features of the Tracker Agent, consider the following points:

- The programs can be installed on only one machine if the following are true (these network setups must be performed on every machine that uses the Tracker Agent):
  - The file system is network mounted with NFS
  - The installation directory is exported

- The remote clients NFS-mount the install directory

### ***Updating the .toc file for the first install***

If this is the first time the Tracker Agent has been installed on the machine, you must update the SMIT .toc file with the tracker information. To update your .toc file, enter:

```
inutoc /usr/sys/inst.images
```

If the Tracker Agent image is stored in another directory, use that directory instead of /usr/sys/inst.images.

### ***Reinstalling the Tracker Agent***

If the Tracker Agent has previously been installed, set these values on the Install Software Products at Latest Available Level SMIT panel using the following options:

- Install Prerequisite Software: **no**.
- Overwrite existing version: **yes**.

Depending on the level of SMIT, you might have to remove all Tracker Agent files from the system when reinstalling. To remove the old files, issue the following commands:

```
cd /usr/lpp  
rm -rf tracker
```

Use SMIT to Install Images

The steps and panel names may vary slightly depending on the version of SMIT that you have.

To use SMIT to install the required features either from the installation media or the hard disk, perform the following steps:

1. Start **SMIT**.
2. Select **Software Installation & Maintenance**.
3. Select **Install / Update Software**.
4. Select **Install Software Products at Latest Available Level**.
5. Enter the device or directory name where the installation media resides.  
Enter the full path to the directory containing the file that was downloaded from the Controller machine, for example:

```
/usr/sys/inst.images
```

6. Position the pointer on the **SOFTWARE to install** line, and press **F4**.

7. Select the required features from the list, position the pointer beside the package, and press **F7**.

286 End-to-End Scheduling with OPC and TWS Mainframe and Distributed Environments

8. Press **Enter** or select **OK**
9. Select **Do** to start the installation.

When the Tracker Agent install is complete, you should see a panel similar to the following screen:

```
. COMMAND STATUS

Command: OK          stdout: yes          stderr: no

Before command completion, additional instructions may appear below.

[TOP]
installp -acgNOqwx -d /usr/sys/inst.images -f File 2>&1

File:
    tracker.obj          2.3.0.11

+-----+
+-----+ Pre-installation Verification... +-----+
+-----+
Verifying selections...done
Verifying requisites...done
Results...

[MORE...56]
```

After the installation finished, the directory structure where you installed the Tracker Agent should look like the information contained in the following screen.

```
-rwxr-xr-x  1 opc          1460 Nov 24 1999  EQPARM
dr-xr-x---  2 opc          512 Jun 06 16:48  bin
dr-xr-x---  2 opc          512 Jun 06 16:48  catalog
-rw-r--r--  1 system      322 Jun 06 16:48  copyright.master
drwxr-xr-x  2 system      512 Jun 06 16:49  deinst1
dr-xr-x---  2 opc          512 Jun 06 16:48  doc
drwxrwxr-x  2 opc          512 Jun 06 17:00  etc
drwxrwxrwx  2 opc        1024 Jun 20 12:01  log
dr-xr-x---  2 opc          512 Jun 06 16:48  methods
dr-xr-xr-x  4 opc          512 Jun 06 16:48  nls
-rw-r--r--  1 system       37 Jun 06 16:48  productid
drwxrwxr-x  2 opc          512 Jun 06 16:48  samples
drwxrwxrwx  3 opc          512 Jun 19 14:34  tmp
```

A detailed explanation for every directory can be found in the book, *Tivoli Operations Planning and Control V2R3 Tracker Agent*, SH19-4484.

---

## 9.8 Setting EQQHOME and EQQINSTANCE variables

Normally, you start the Tracker Agent under the tracker user ID and have /u/tracker as the home directory. You are strongly recommended to set the following two environment variables that tell the Tracker Agent binaries where the files are:

- EQQHOME variable
- EQQINSTANCE variable.

### 9.8.1 Setting the EQQHOME variable

This variable is the name of the home directory. To check the home directory, enter the following commands as the user ID under which the Tracker Agent runs:

- *cd*
- *pwd*

To set the environment variable, EQQHOME, in the Korn Shell (ksh), enter  
`export EQQHOME=/u/tracker`

To set the environment variable, EQQHOME, in the C Shell (csh), enter  
`setenv EQQHOME /u/tracker`

To set the environment variable EQQHOME in the Bourne Shell (sh), enter  
`EQQHOME=/u/tracker`

`export $EQQHOME`

These examples assume that the home directory is /u/tracker. In the remainder of this chapter, the home directory is referred to as \$EQQHOME. For example, if /u/tracker is the home directory, \$EQQHOME/etc refers to /u/tracker/etc.

### 9.8.2 Setting the EQQINSTANCE variable

This variable is the name of the configuration parameter file. Set it in the same way as for the EQQHOME variable:

*EQQINSTANCE=myconfig.file*

*export EQQINSTANCE*

You can also specify the name of the configuration file using the -f flag on the eqqstart script or when you start the Tracker Agent directly. The -f flag takes

precedence over the EQQINSTANCE variable. See Section 9.19, “The Tracker Agent utilities” on page 304, for more information. Put the configuration parameter file in the \$EQQHOME/etc directory. Copy the configuration file to each machine on which a Tracker Agent is installed, and ensure that the values are consistent.

You can define the necessary variables either in .profile of the tracker home directory, or, if you want to define them system-wide, you need a corresponding entry in the /etc/environment file. An automatic start of the Tracker Agent after every reboot of the AIX machine can be defined in the /etc/rc.tcpip file. The following screen shows our entries in .profile.

```
PATH=$PATH:/u/tracker/bin:
PS1='$PWD '
export PS1
export EQQHOME=/u/tracker
export EQQINSTANCE=aix1.cnf
```

---

## 9.9 Creating links between the directories

This step must be performed for all platforms except Digital OpenVMS. If you have standard directory tree naming conventions, you can create the required links using the sample script, eqqinit. Ensure that EQQHOME is set to your home directory.

*/u/tracker/bin/eqqinit -tracker*

Run the script from the tracker user ID. Use the *-tracker* parameter to create the links.

---

## 9.10 Updating the configuration parameter file

The configuration parameter file of the Tracker Agent is needed as the startup setup. It contains connection-related definitions and information about how it should operate.

The general syntax rules for the statements are as follows:

- All keywords must be lowercase.
- All user-specified values must be in the format shown.
- A pound sign (#) can be used for comments.
- Blank lines are treated as comments.

- Unknown or invalid keywords are ignored.
- Each keyword must begin on a separate line.
- If a keyword definition cannot fit on a single line, use a backslash (\) as a continuation character.
- If you code a keyword more than once, the last value is used.
- Keyword values can include the following:
  - Previously-defined configuration or environment variables (\$variable)
  - Home directories in the format ~user
  - The logged-on-user home directory (~ on its own).
  - Service names (for port numbers).
  - Host names (for IP addresses).



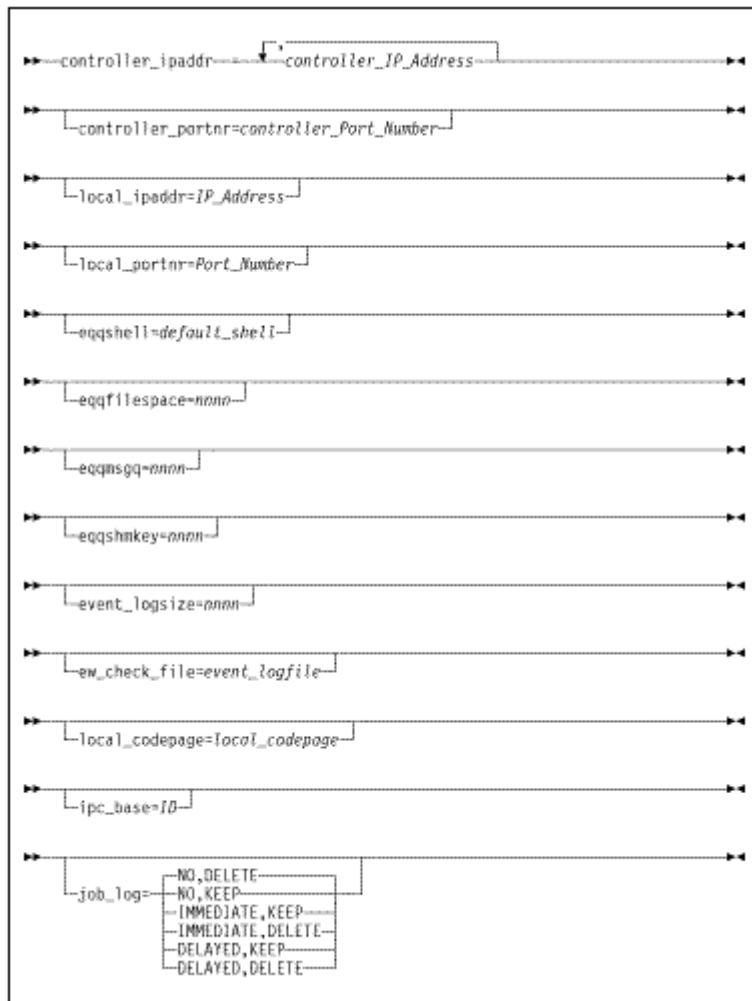


Figure 196. Controller parameter file of the Tracker Agent (Part 1)

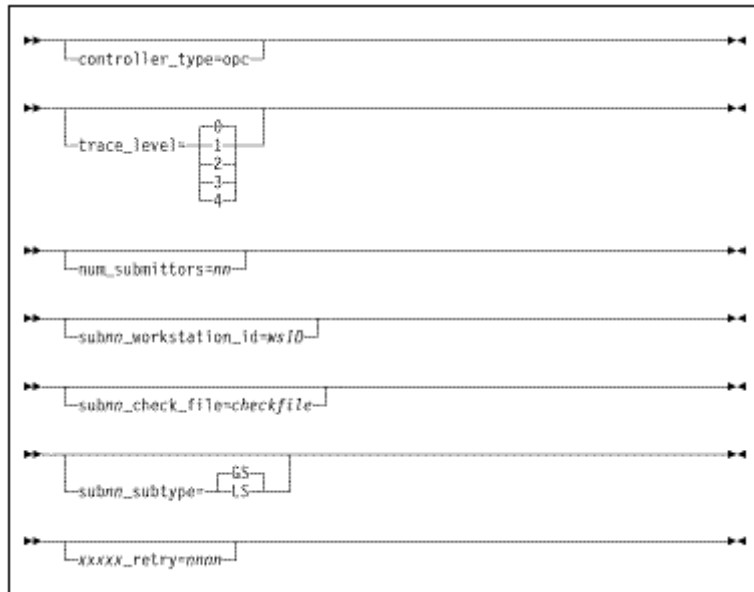


Figure 197. Controller parameter file of the Tracker Agent (Part 2)

### Keywords

The following are the keywords and their definitions:

- **Controller\_ipaddr=** Controller\_IP\_Address - Specifies the IP addresses for the systems where the Controllers are running. There can be up to 10 addresses, each in the format nnn.nnn.nnn.nnn, where nnn is in the range 1-254, or a host name. It is a required keyword; there is no default value. Separate the addresses with commas. The Tracker Agent tries the first address in the list at startup. If it is unable to make a connection, it tries the next address in the list, and so on. Only one Controller can be connected at any one time.
- **Controller\_portnr=** Controller\_Port\_Number - Specifies the port number to which the Tracker Agent TCP-Writer connects, or a services name.
- **local\_ipaddr=** IP\_Address - Specifies the IP address for the machine where the Tracker Agent is running. It must be in the format nnn.nnn.nnn.nnn, where nnn is in the range 1-254, or an environment variable such as \$HOST, or a host name.
- **local\_portnr=** Port\_Number - Specifies the port number to which the Tracker Agent TCP-Reader binds a socket.

- **eqqshell=** default\_shell - Specifies the shell under which noninterpreted scripts run. The default is the Korn shell, if there is one; otherwise, it is the Bourne shell (/bin/sh).
- **eqqfilespace=** nnnn | **1000** - Specifies the minimum number of blocks of temporary file space. The size of a block depends on the Tracker Agent machine. If there is less space available, the Tracker Agent closes down in an orderly way with a message. The default is 1000 blocks.
- **eqqmsgq=** nnnn | **16384** - Maximum size (in bytes) of a message transferred to the Controller. The default is 16384. If the kernel does not allow the size that you specify, the maximum allowable size is used, and a message is issued.
- **eqqshmkey=** nnnn | **58871** - The key to the shared memory that the Tracker Agent will use. The default is 58871.
- **event\_logsize=** nnnn | **1000** - Specifies the number of events to be logged in the Event Writer event logfile. The value defined is the number of events, not the event record size. The default is 1000, and this value is also the minimum.
- **ew\_check\_file=** event\_logfile - The name of the file used for logging events. If a file does not exist, it will be created. The default file name is ewriter\_check. You can guarantee a unique network-wide name for the log file by specifying a name in the following format:
- \$HOST.\$local\_portnr.ew.check
- **local\_codepage =** local\_codepage | **ISO8859-1** - Specifies the ASCII codepage used. The default is ISO8859-1.
- **ipc\_base=** ID | **A** - Specifies the unique ID character to be used for creating unique inter-process communication (IPC) numbers. The default value is A.
- **job\_log=** Log\_Option - Specifies how joblogs will be processed by the Tracker Agent. The following combinations are valid:
  - IMMEDIATE,DELETE
  - IMMEDIATE,KEEP
  - NO,DELETE
  - NO,KEEP
  - DELAYED,DELETE
  - DELAYED,KEEP

The values have the following meanings:

- **NO** Joblogs will not be sent to the Controller.
- **IMMEDIATE** Joblogs are immediately sent to the Controller when the job ends.
- **DELAYED** Joblogs are returned to the Controller only if a Tivoli OPC dialog user requests the joblog.
- **KEEP** Joblogs are stored on disk for all jobs.
- **DELETE** Joblogs are deleted when the job ends if NO or IMMEDIATE is also specified. When DELAYED is specified, the joblog is kept on disk until a retrieval request is received from the Controller. When saving joblogs on disk, consider the disk space available on the system. The jobs will end in error if there is no space for the log. The default value is IMMEDIATE,DELETE. That is, joblogs are sent to the Controller immediately and deleted. Joblogs are never sent to the Controller if the submitter uses LoadLeveler.
- Controller\_type= opc - Specifies the type of Controller for this Tracker Agent:
  - opc - The Controller is OPC/ESA.  
There is no default value. OPC must be specified.
- trace\_level=0 | 1 | 2 | 3 | 4 - Specifies the trace level for the component. The default is 0 (no trace). You do not need a trace for normal running.
- num\_submitters= nn | 1 - The number of submitters to initialize, from 1 to 56. By default, one submitter is started. Each submitter will have a default subtype of GS, a default workstation ID of AX nn, and a default check file of AX nn.check, but you can override these defaults with the following parameters.
- sub nn\_workstation\_id= wsID | AXnn - Connects submitter nn with a workstation. It is a character string. The length must not exceed 4 (only the first 4 characters will be used). The default value is AX nn.
- sub nn\_check\_file= checkfile | wsID.check - Connects submitter nn with the file used for job checkpointing and event logging. If a file does not exist, it will be created. The default file name is wsID.check. You can guarantee a unique network-wide name for each file by specifying a name in the following format: wsID.\$HOST.\$local\_portnr.check
- sub nn\_subtype=GS | LS - Controls whether a submitter uses LoadLeveler (LS) or not (GS). GS, which stands for generic submitter, is the default. A generic submitter is simply one that does not use LoadLeveler.
- xxxxx\_retry= nnnn | 60 - These are retry intervals, in seconds, for various Tracker Agent components. xxxxx can be:

- **eqqtr** - This is the interval that Tracker Agent will wait before attempting to communicate with the Controller if a TCP read attempt fails.
- **eqqtw** - This is the interval that Tracker Agent will wait before attempting to communicate with Controller if a TCP write attempt fails.
- **eqqdr** - This is the interval that Tracker Agent will wait before attempting to connect and revalidate the connection to the controlling system.
- **sub nn** - This is the interval that the nn submittor will wait before retrying an operation (for example, because the number of processes had reached the limit).

Example: The Tracker Agent has been installed on AIX 4.3.3. The IP interface of the box can be reached through the address, 146.84.32.100. The OPC Controller on OS/390 binds sockets with port number 3112 and IP address 9.39.62.19.

```
Controller_ipaddr = 9.39.62.19
Controller_portnr = 3112
local_ipaddr = 146.84.32.100
local_portnr = 1967
Controller_type = opc
eqqfilespace = 200
event_logsize = 1000
num_submittors = 1
job_log = delayed,keep
ew_check_file = ewriter_check
local_codepage = ISO8859-1
sub01_workstation_id = tws1
sub01_check_file = tws1.chk
```

The workstation definition, TWS1, assigns a one-to-one relationship to the OPC computer workstation definition. For a better understanding, we show the corresponding Controller Routopts statement again:

When you have defined the Tracker Agent parameter file, you can check the syntax of your parameter file when running the eqqverify utility. Eqqverify is located in the \$EQQHOME/bin subdirectory. If you defined the variable, EQQINSTANCE, properly, you don't need any subparameter. See also Section 9.19, "The Tracker Agent utilities" on page 304, for more information. Check that the verify runs successfully. The next screen shows the correct output of eqqverify.

```

EQQPARM configuration variable is /u/tracker/etc/aix1.cnf.
Verifying /u/tracker/etc/aix1.cnf
eqqsystem='its06 AIX 4 3 000681704C00'
eqqversion='FIX LEVEL 11, 1999/11/10'
EQQHOME='/u/tracker'
eqqiconv='/u/tracker/nls/loc/iconvTable'
EQQPARM='/u/tracker/etc/aix1.cnf'
msg='/u/tracker/nls/msg/en_US/eqqmsgcat.cat'
tmp='/u/tracker/tmp'
log='/u/tracker/log'
logfile='/u/tracker/log/eqqmsglog'
num_submitters='1'
trace_level='0'
local_codepage='ISO8859-1'
ipc_base='A'
job_log='DELAYED,KEEP'
event_logsize='1000'
Controller_portnr='3112'
Controller_type='opc'
local_portnr='1967'
local_ipaddr='146.84.32.100'
eqqshell='/bin/ksh'
eqqmsgq='16384'
eqqshmkey='58871'
eqqdispatch='10'
eqqfilespace='200'
ew_check_file='ewriter_check'
Controller_ipaddr='9.39.62.19'
sub01_workstation_id='TWS1'
sub01_check_file='tws1.chk'
eqqdr_executable='/u/tracker/bin/eqqdr'
eqqdr_retry='60'
eqqtr_executable='/u/tracker/bin/eqqtr'
eqqtr_retry='60'
eqqtw_executable='/u/tracker/bin/eqqtw'
eqqtw_retry='60'
eqqew_executable='/u/tracker/bin/eqqew'
eqqew_retry='60'
sub01_subtype='GS'
sub01_executable='/u/tracker/bin/eqqgs'
sub01_retry='60'
Verify status = 0
Parameter file verified successfully..

```

## 9.11 Customizing the directories

By default, the Tracker Agent creates the log, temporary, and trace files in the \$EQQHOME/log and \$EQQHOME/tmp directories. If the Tracker Agent home directory is NFS mounted, these directories will be the same for every Tracker Agent. This can cause performance problems. Also, if the network connection to the NFS server is lost, the Tracker Agent cannot function fully. Put these

directories on a local file system (or with a symbolic link to a local file system) to improve log performance. If you run several instances of the Tracker Agent, and they share the same directory, use variables in the configuration parameter file to ensure that they do not use the same checkpoint and log files. If the Tracker Agent is running from an NFS-mounted file system, it is recommended that the log and temporary directories be configured on the local file system. The `egginit` command (see Section 9.19, “The Tracker Agent utilities” on page 304) initializes a directory on the local machine:

```
egginit -v
```

This command must be run as root. This local file system must have write privileges for everyone including a root user who is logged in across the network. The recommended name of the local directory is `/var/tracker`. You might need administrator privileges to create the `/var` directory (if it does not exist) and to create the links.

The `/tmp` directory is not suitable because this file system is frequently cleaned when booting the system, and this would cause the Tracker Agent to be unable to re-create its internal status. There can be a problem if a job writes too much output: this can fill up the allocated space. To protect the system, use a logical volume for the `tmp` and `log` directories (where this is supported) or set up a separate file system for them. If this fills up, the Tracker Agent will stop submitting jobs, but the operating system will continue to work. You can use SMIT to create a logical volume. The log directory includes an event writer checkpoint file, a message log (`eqqmsglog`), a trace log (`EQQtrc.log`) for each Tracker Agent instance, and a submitter checkpoint file for each submitter instance.

---

## 9.12 Customizing the file permissions

Some of the Tracker Agent components are designed to run with root authority. These components are:

- The TCP Reader
- The Generic subtask
- The LoadLeveler submitter.

If you use port numbers lower than 1025, the TCP Reader process must have root authority. If port numbers are defined with values greater than 1024, the TCP Reader process does not need root authority.

### Running without root authority

To update the TCP Reader to run without root authority, enter the following command as root:

```
chown tracker $EQQHOME/bin/eqqtr
```

The generic and LoadLeveler submit processes must also run as root if the user ID under which submitted jobs should be started is supplied by the Controller. If the Tracker Agent is not required to run jobs under other user IDs, the submitters can also be updated to run with normal user authority.

To update the submitter processes to run without root authority, enter the following commands as root:

```
chown tracker $EQQHOME/bin/eqqls  
chown tracker $EQQHOME/bin/eqqgssub
```

### **Restoring root authority**

To update the TCP Reader and submitter processes to run with root authority, enter the following commands as root:

```
chown root $EQQHOME/bin/eqqtr  
chmod u+s $EQQHOME/bin/eqqtr  
chown root $EQQHOME/bin/eqqls  
chmod u+s $EQQHOME/bin/eqqls  
chown root $EQQHOME/bin/eqqgssub  
chmod u+s $EQQHOME/bin/eqqgssub  
chown root $EQQHOME/bin/eqqgmeth  
chown u+s $EQQHOME/bin/eqqgmeth
```

---

## **9.13 Restrictions and dependencies on system software**

This section outlines restrictions and system dependencies that you need to consider.

### ***NFS restrictions***

When running the Tracker Agent on NFS-mounted directories, the user ID running the Tracker Agent must have write access to the file system. If the Tracker Agent is running on an NFS-mounted file system, the superuser must have write access to the file system.

### ***Number of processes per user***

If the Tracker Agent is running under a user ID other than root, or if many jobs are run under one user ID, the number of processes per user ID should be increased.

The following example is given for AIX only.



To set this parameter:

1. Start SMT.
2. Select System Environments.
3. Select Change / Show characteristics of Operating System.
4. Select Maximum number of PROCESSES allowed per user.

---

## 9.14 Coordinating clock values

The value of Greenwich Mean Time (GMT) must be approximately the same for the Tracker Agent machine and the controlling system. If you set GMT as local time, both the Tracker Agent and Controller environments must set GMT as local time. The Tracker Agent will not be able to connect to the Controller if the GMT value for the Tracker Agent machine is not within 60 minutes (plus or minus) of GMT on the controlling system. If machines are in different time zones, these times must be set correctly on the different machines or coordinated through a network time service.

---

## 9.15 Running scripts for Tivoli OPC

The following sections describe the running of scripts for Tivoli OPC.

### 9.15.1 Storing scripts

The Controller schedules scripts to be executed by the Tracker Agent in the normal Tivoli OPC way, that is, a script is an operation, which is part of an application. Refer to the book, *Tivoli OPC Planning and Scheduling the Workload*, SH19-4481, for details of creating application descriptions. You can store the scripts in the Controller EQQJBLIB dataset or retrieve them with the Controller EQQUX002 exit, which is described in Customization and Tuning. If you edit the member using ISPF, make sure that you have numbers set off (UNNUM), or the editor will add sequence numbers in columns 73–80, which will cause errors.

The scripts residing in the MVS host dataset EQQJBLIB must have a logical record length of 80 (LRECL=80). The tool eqqcv80p is provided to facilitate the 80-bytes formatting. For details of the eqqcv80p utility, see Section 9.19, “The Tracker Agent utilities” on page 304.

### 9.15.2 Writing scripts

Scripts can contain Tivoli OPC variable substitution and automatic recovery directives, which are described in Planning and Scheduling the Workload, but

they cannot use MVS-specific functions, such as catalog management and step-level restart.

### **9.15.3 Determining the shell under which scripts run**

There are two ways to determine the shell under which scripts run. The value of the `eqqshell` keyword in the `EQQPARM` file determines where noninterpreted scripts run. If you do not give this keyword a value, it takes the default, which is `/bin/ksh` (the Korn shell) for AIX systems, and `/bin/sh` (the Bourne shell) for other UNIX systems. Interpreted script files, that is, files that begin with the line, `#! pathname`, run under the shell indicated in the `pathname`.

### **9.15.4 Specifying a user ID**

If the Controller supplies user IDs for submitted jobs, the user ID must exist on the Tracker Agent machine. Ensure that the user ID is supplied in the correct format, with lowercase and uppercase characters as defined on the Tracker Agent machine. You cannot specify a user ID if the script will run under `LoadLeveler`; the `LoadLeveler` runs all scripts under its own user ID.

### **9.15.5 Getting output from scripts**

If an empty script is sent to the Tracker Agent or a script is too long, the operation completes with error code `JCLI`. If there is an environment error during execution of the script, the operation completes with error code `JCL`. If you use the generic submitter, you can browse the standard out and error files from the script (the job log) using the Controller dialogs. You cannot browse the job log of scripts submitted using `LoadLeveler`. If you have very large script output, check the `event_logsize` configuration parameter. Every 512 bytes of output (approximately) causes an event; so, the parameter must be set large enough for the largest expected output. If the output is too big, the Tracker Agent writes an error message to the message log. The event log wraps round, and the Tracker Agent must scan the whole file for events; so, do not make the file unnecessarily large, or this will impact performance. There is a limit of approximately 64 KB on the job log output that can be retrieved by the Tivoli OPC Controller. You find your script output in `$EQQHOME/tmp`.

### **9.15.6 Testing for errors from commands**

The generic submitter monitors the return code from the script, and the return code sent to the Controller is the return code from the execution of the script. If you have a multiline script, such as the following:

*date*

*touch /tmp/file*

*date*

and the `touch` command fails, the return code in the shell is set. On the next command, `date`, the return code is reset to 0; so, the return code from the `touch` is gone and the script will return 0 (job successful). If you want to verify each step in the script, add tests after each call in the script to verify the shell return code:

*date*

*(test rc) - if rc nonzero exit with rc*

*touch /tmp/file*

*(test rc) - if rc nonzero exit with rc*

*date*

*(test rc) - if rc nonzero exit with rc*

The test depends on the shell used to run the job; the syntax for `/bin/sh` is different than for `/bin/csh`. If you want to monitor a single command directly, specify the command as a single-line script. In the case of scripts that are only one line long, the submittor monitors the actual command, and the return code sent to the Controller is the return code from the execution of the command. In this case, the command is run using a standard UNIX `exec`; so, you cannot use shell syntax. Of course, testing return codes will only work if the command returns a bad code when it fails and a zero code when it works. If you are not sure, try the command from the UNIX command line, and echo the return code from the shell. If the script is more than one line, the generic submittor submits the script and monitors the shell for a return code. This means that, in very rare cases, the script can have run without error, but an error in the shell can result in an error return code. It is important to note that, if more than 256 error codes are produced from the execution of a command or program, they are processed modulus 256. This means that return code multiples of 256 are treated as return code zero. For example, return code 769 ( $256 \times 3 + 1$ ) is treated as return code 0001, and so on. Make sure that the correct code page is set for your terminal emulator. If the code page is incorrect, such characters as £, \$, and # in scripts sent from the Tivoli OPC Controller might be mistranslated, causing jobs to not run correctly.

### 9.15.7 Specifying the path

The default path and the sequence in which the libraries are searched for commands to be executed depends on the user ID with which the tracker is started. To be sure that a program or script is loaded from the correct library, it is advisable to specify the full path in the script or command.

---

## 9.16 Starting the Tracker Agent

A sample script has been provided to start the Tracker Agent. Log in as the Tracker Agent user ID (normally tracker), and enter the following:

```
eggstart [-f filename]
```

You can use the `-f` flag to specify the configuration file. This overrides the `EQQINSTANCE` environment variable. You must also set the `EQQHOME` environment variable to point to the home directory. The following screen appears when starting the Tracker Agent:

```
/u/tracker eggstart
Starting tracker from /u/tracker EQQINSTANCE=aix1.cnf
/usr/lpp/tracker
06/21/00 10:51:38 /*****
06/21/00 10:51:38 /*   Licensed Materials - Property of IBM          */
06/21/00 10:51:38 /*   5695-007 (C) Copyright IBM Corp. 1994, 1999. */
06/21/00 10:51:38 /*   5697-OPC (C) Copyright IBM Corp. 1997, 1999. */
06/21/00 10:51:38 /*   All rights reserved.                          */
06/21/00 10:51:38 /*   US Government Users Restricted Rights -      */
06/21/00 10:51:38 /*   Use, duplication or disclosure restricted    */
06/21/00 10:51:38 /*   by GSA ADP Schedule Contract with IBM Corp. */
06/21/00 10:51:38 /*****
06/21/00 10:51:38 Starting OPC Tracker Agent
06/21/00 10:51:38 FIX LEVEL 11, 1999/11/10
06/21/00 10:51:38 Hostname is itso6 AIX 4 3 000681704C00
06/21/00 10:51:38 ****
06/21/00 10:51:38 ***          TRACKER DAEMON STARTING UP          ***
06/21/00 10:51:38 ***          STARTED BY          tracker          ***
06/21/00 10:51:38 ****
06/21/00 10:51:38 REAL          UID : 7
06/21/00 10:51:38 EFFECTIVE UID : 7
```

The method for automatically starting the Tracker Agent depends on your operating system. Edit the `/etc/rc.tcpip` file. This file is processed at startup to initiate all TCP/IP-related processes. To add the Tracker Agent to the `/etc/rc.tcpip` file, perform the following steps:

1. Log in as root.
2. Edit `/etc/rc.tcpip`, using an editor, such as `vi`.
3. At the bottom of the file add the following section:

```
set EQQINSTANCE=myconfig
set EQQHOME=/u/tracker
export EQQINSTANCE
export EQQHOME
/u/tracker/bin/eqqstart
```

Daemons cannot control TTY consoles. Therefore, if you are running applications that access TTY consoles, either start AIX Tracker by issuing the `eqqstart` command, or use local applications to avoid the need for console access.

---

## 9.17 Shutting down the Tracker Agent

The Tracker Agent can be shut down in an orderly manner by using the `eqqstop` command on a command line. Only the Tracker Agent administrator or root user can shut down the Tracker Agent. If the Tracker Agent has been started as root, only the root user can request a shutdown.

```
/u/tracker eqqstop

Process KEY : 58871
EQQPARM configuration variable is /u/tracker/etc/aix1.cnf.
Eqqmon option is 5.

Sent SIGTERM to daemon process.
/u/tracker
```

---

## 9.18 Restarting after an abnormal termination

Use the `eqqclean` command to tidy the temporary files.

Sometimes, depending on how the Tracker Agent terminates, the shared memory key might not be deleted, preventing a subsequent restart of the Tracker Agent. When this happens and the Tracker Agent is restarted, the security mechanism prevents the Tracker Agent from starting. A warning message informs you that a Tracker Agent might already be active, and a message describing a shared memory allocation failure is issued to the message log. If the problem is not due to a tracker with the same key already running, you must remove the shared memory segment before the Tracker Agent can restart. To do this:

- Enter the `ipcs -mob` command.
- In the generated output, find the identifier of the segment with:

- **Owner** tracker
- **Group** opc
- **Size** 6048
- **66** Tivoli OPC Tracker Agents for AIX, UNIX, VMS, OS/390
- Remove the segment using the `ipcrm -m <identifier>` command.
- Perform the first step again to ensure that the segment is no longer listed.

---

## 9.19 The Tracker Agent utilities

The bin and samples directories supplied with the Tracker Agent contain utilities and samples to help you operate and maintain the Tracker Agent. These utility programs and scripts are located in the bin directory. If you need to modify a script, save a copy of the original in the samples directory.

The following are the most important utilities:

- **eqqverify** - This utility shows the Tracker Agent settings. The Tracker Agent configuration parameters are written to `$EQQHOME/log/eqqmsglog`. The verified values are also written to the terminal. If a parameter cannot be verified, a message is generated describing the problem.
- **eqqstart** - This script starts the Tracker Agent. If the environment variable `EQQHOME` is set, the Tracker Agent starts from `$EQQHOME/bin`.

The following are descriptions of the flags for start-up.

- **-dID** - Engine timeout, in minutes. When a process signals that it is active, this value specifies the maximum time allowed before it must send another active or idle signal. If the default value is used, a process cannot be active for more than five minutes without sending another active or idle signal, or the Tracker Agent will automatically shut down and restart.
- **-iII** - Maximum inactivity time for server. This is the time allowed for a process to become active. Change it only in special cases.
- **-fIF** - The configuration parameter file name. You need not specify this if you have set the `EQQINSTANCE` variable, or if the configuration parameter file is `$EQQHOME/EQQPARM`.
- **-rIR** - Time between automatic restart in minutes. At these intervals (for example, once a day), the daemon will refresh each process.
- **-sIS** - Sleep between restart, in seconds. It takes about five seconds to complete a clean shutdown of the Tracker Agent processes. Sometimes, it can be useful to delay the time between automatic restart attempts.

- **-vIV** - The number of times, within a number of minutes, that a process is allowed to restart before the Tracker Agent terminates abnormally. For example, the value, -v4:10, specifies that the Tracker Agent will restart up to four times in a 10-minute period. There are two reasons for restart:
  - A process has been active more than five minutes without a signal.
  - A process has abnormally terminated.

The eqqstart sample script passes these flags unaltered to the Tracker Agent daemon. You can change the eqqstart sample script if you want to change the defaults. The flags can be entered either as uppercase or lowercase characters. There is no difference in how they are treated.

- **eqqstop** - This script stops the Tracker Agent. The current tracker processes are searched for by the daemon. The daemon is sent a SIGHUP, (1) and exits normally.
- **eqqclean** - This script tidies the log files after the tracker has terminated abnormally. Next screen shows eqqclean example:

```
/u/tracker eqqclean
Cleaning tracker from /u/tracker.
./EQQ5ab8.ENV
./EQQ5ab8.PGM
./EQQ5ab8.TRC
./eqqmsglog
./EQQtrc.log
./EQQ56dc.ENV
./EQQ56dc.PGM
./EQQ56dc.TRC
./CLISCR__000614_151913_00000003.OUT
./CLISCR__000614_151913_00000003.TMP
./LS__000619_114505_0000000A.OUT
./LS__000619_114714_0000000B.OUT
./CLISCR__000619_142652_0000000C.OUT
./CLISCR__000619_143418_0000000D.OUT
./CLISCR__000619_143418_0000000D.TMP
/u/tracker
```

---

## 9.20 Which agent to choose? OPC Tracker Agents or TWS Agents?

OPC Trackers and TWS Agents development is converging. For example, the OPC Tracker agent and the TWS X-agent for SAP R/3 use a common code core.

TWS Agents provide fault tolerance and a rich set of scheduling functionality in the distributed arena. Therefore, new development is likely to further move in that direction.

On the other hand, master-slave agents like OPC tracker agents have inherent design constraints that make them less flexible, and less likely to be developed further.

So, today, if you have an OPC environment and are considering which agent to use, the following guidelines could be useful to you:

The following is the rationale for using TWS Agents:

- You need fault tolerance at the Agent and Domain level.
- You do not need to use applications with some jobs on the mainframe and some jobs on the distribute site right away.

**Note**

Remember that, today, it is possible to schedule and control TWS and OPC jobs from the same console (JSC). You can also trigger the execution of jobs from each scheduling engine on the other. This is shown in Chapter 8, “Enterprise scheduling scenarios” on page 203. However, you need to have two different application definitions (one on the mainframe and one on the distributed environment) and link them.

- You need to support platforms that are not currently supported by OPC Tracker Agents but are supported by TWS Agents.

The following is the rationale for staying with the OPC Trackers:

- You do not need fault tolerance.
- You do not need any platforms other than the ones currently supported by OPC Tracker Agent, and you will not need any others in the near future.
- You want to mix distributed and mainframe operations (jobs) on the same OPC application right now in the easiest way possible.



---

## Chapter 10. Best practices

Scheduling across the enterprise involves the potential management of vast amounts of jobs and resources. In this chapter, we make suggestions for best practices.

---

### 10.1 OPC

OPC is a mature stable scheduling platform used worldwide by thousands of companies. One would expect to find them making similar use of OPC, but we have observed wide variations in practices between companies. This section discusses some considerations in making the best use of OPC.

#### 10.1.1 Moving to OPC

When moving to OPC, it is worth looking at your business practices again. Your way of working is, most likely, dictated by the scheduling package you are currently using. All scheduling programs, OPC included, work to a process model. A common mistake is to aim to duplicate in OPC the exact process model you currently use. No one likes change; we are comfortable with what we know and do not want to learn new ways of doing things, but, to gain the most benefit from OPC, it is a good idea to utilize its method of working.

Set up a migration group comprised of schedulers and operations staff to consider what your business purpose is, and think about what you really want to achieve. Then, look to see how best to do this with OPC. One difficulty you may have at this stage is that no one knows much about OPC. We recommend that you send key personnel to a formal IBM/Tivoli training course prior to installation. Such courses offer hands-on laboratory sessions where OPC skills and knowledge can be quickly and easily gained, and, in addition to “picking the brains” of the instructor, your delegates will meet participants from other companies currently using OPC and learn from their experiences. Tivoli and/or IBM professional services and migration specialists are available to help you install, upgrade, or migrate to OPC. Contact your Tivoli sales representative for access to Tivoli Professional Services.

Check the following web page for more information on migration services:

Software Migration Project Office (SMPO)

<http://www.ibm.com/solutions/softwaremigration/tivmigteam.html>.

### **10.1.2 OPC roles and responsibilities**

The number of people with direct responsibility for OPC depends on the size of your data center. The following are the main OPC functions once the initial loading of information about work and resources and calendars in the OPC databases is completed.

#### **10.1.2.1 Scheduler**

The scheduler performs the following functions:

- Maintains the calendar and period databases
- Checks that scheduling is correct
- Checks the current plan
- Authorizes extensions and modifications of the plans

#### **10.1.2.2 Application builder**

The application builder performs the following functions:

- Maintains and creates new applications and job descriptions in the database
- Creates resources and ETT and OPC variables as required in the databases

#### **10.1.2.3 Operations**

Operations perform the following functions:

- Primarily responsible for managing work in the current plan
- Corrects and reruns failed operations
- Ensures that deadlines are met

#### **10.1.2.4 Systems programmer**

The systems programmer performs the following functions:

- Installs and maintains OPC to latest release
- Ensures high availability and performance
- Maintains OPC exits
- Designs and maintains security access and allocates user IDs
- Ensures system backups and recovery processes are in place

#### **10.1.2.5 OPC administrator**

The OPC administrator performs the following functions:

- Point of contact for OPC
- Manages audit trail log
- Authorizes access to OPC
- Documents and maintains naming standards
- Documents operating standards
- Arranges training for new staff
- Supports the local User Group and attends its meetings with key staff

In some data centers, one person will cover several functions; in others, a person may be dedicated to that responsibility.

### 10.1.3 Naming Standards

Using good naming standards for applications can pay dividends. OPC is rich with filter panels. You can combine application names, owner and authority group ID fields, and use wild card characters to easily find related applications.

Since the application name can be up to 16 characters in length, meaningful names can be used. Some sites use the first character to indicate whether the application is a production or test application. The second character indicates the frequency it runs; the next three characters identify the department, and so on. The following are some examples:

```
PWPAYTX#WKPAYTX
PMACCNC#MONTHACC
TXEDU###PETETEST
```

- PWPAYTX#WKPAYTX - The first character, P, indicates that this is a production application. The second character, W, indicates that it is scheduled weekly. PAY indicates that it is a payroll job, and the next two characters, TX, show that it is for Texas. A name follows all that.
- PMACCNC#MONTHACC - The first character, P, indicates that this is a production application. The second character, M, indicates that it is scheduled monthly. ACC means it is an accounts application, and the next two characters, NC, show that it is for North Carolina. The name follows the # separator character.
- TXEDU###PETETEST - The first character, T, indicates that this is a test application. The second character, X, means that it is not scheduled. EDU indicates that it belongs to the education department, and the next two characters, ##, show that it is not linked to any area. The name follows the # separator character.

Using this naming standard, it is easy to set up a search for related applications. Figure 198 on page 310 shows the setup of a JSC view of accounting applications with an owner name of MAY.

Figure 198. Filtering a Job stream View in the JSC

The JSC uses the question mark symbol (?) to replace one alpha character. The percent sign (%) replaces one numeric character, and an asterisk (\*) replaces zero or more alphanumeric characters. In the ISPF dialog panels, the percent sign (%) replaces one alphanumeric character. Whatever naming standard you choose, it should be consistent and simple to understand.

#### 10.1.3.1 Internal documentation

OPC has many description fields that can be used for documentation within the application, run cycle, and so on. Full use should be made of these fields, especially for items, such as special resources and JCL variables. The name of the responsible person or department will be useful; so, in the future, you can check whether the item is still needed. There are data centers with

hundreds of orphan special resources in the database, and no one really knows if they are still needed.

#### **10.1.3.2 Jobs**

Having naming standards for jobs is essential. Submitting non-OPC jobs with the same names as those in OPC's current plan will cause problems.

Depending on the customization, OPC may recognize the job has started, but it knows it has not submitted it, and the OPC job will be marked with the error code, OSEQ, indicating that it has run out of sequence.

#### **10.1.3.3 Operation sequence numbers**

All OPC operations must have a number that is unique within the application. The number can range from 1 to 255. It is used to make dependency links when building an application and to uniquely identify the operation.

Consider standardizing these numbers. For example, some sites use numbers ending in 0 or 5 for OS/390 CPU operations, numbers ending with 4 as JCL prep operations, and numbers ending in 6 for printer workstations. Other numbers can be used for other types of workstations.

The problem with this method is that it limits the number of OS/390 operations in your applications. The maximum number of operations in one application is 255. If your applications have a larger number of OS/390 CPU operations, you need to consider a different numbering system or use application description groups.

#### **10.1.3.4 Operator instructions**

Operator instructions are text files that are linked to OPC operations. It can be used for the comments that would traditionally be coded in JCL, but operator instructions offer many advantages over JCL comments. They avoid the need to edit JCL. They can have in-effect times, which means that they will be displayed only in a certain time frame and they can be bulk updated and bulk uploaded to the database.

### **10.1.4 Resource modelling**

Resources give you the ability to model your data center resources in OPC. OPC uses this information when scheduling, providing you with more accurate plans, and when submitting jobs, thus avoiding resource contention. You can use workstation resources and special resources to represent limited resources. We recommend you do not use workstation resources, except for parallel servers.

Workstation resources are limited to two per workstation, with a maximum of 99 of each resource. Everything you can do with workstation resources (and more) can be done with special resources. You can define an unlimited number of resources, each having up to 999,999 units. Special resource usage is easily monitored in real time using the Special Resource Monitor in the Modify Current Plan ISPF dialog or in the JSC plan view.

OPC resources are a logical representations of real resources, but you can link special resource names to real resources defined in the Resource Object Data Manager (RODM). RODM will vary the amount of units of the special resource as the real resource numbers change.

You can use special resources to represent parallel servers, but we recommend that you use the workstation parallel servers. The reason is that every operation you create has one parallel server defined by default. If you choose to use special resources instead, you will have to manually define the special resource for every operation, and you will have thousands of operations.

### **10.1.5 Input arrival time**

The meaning of the input arrival time is not widely understood. In OPC, it means the expected start time, but the job can run before or after this time. There are two input arrival times. The generally-used one is the input arrival time for the application. This is defined by the run cycle. By default, all operations within the application use the same time. It is possible, optionally, to define a different input arrival time for individual operations.

The input arrival time is used for the following:

- For scheduling purposes - The long-term plan uses this to determine the run day
- When making dependencies - The long-term plan uses the input arrival time and day to make dependency links.
- To uniquely identify an occurrence - You may have multiple occurrences of an application in a plan; OPC identifies each by a combination of its name, input arrival day, and time. A negative (exclusionary) run cycle identifies the run cycle it is to suppress by having the same input arrival time.

Once an application has been brought into the current plan, any of its operations that do not have predecessors or whose predecessors are complete are considered for submission. OPC checks that the requested resources are available and will then submit the jobs. If a job must not be submitted before a certain time, you should make it time-dependent. This

means OPC will not submit the job before the input arrival time, which is, by default, the same as the application input arrival time. You can define for an operation a different, later, time, but you should only do this for time-dependent operations. The operation input arrival time will be used when making external dependency links to this operation.

#### **10.1.6 Deadline time**

Once an application is in the current plan, its deadline time becomes very important. OPC will do everything it can to ensure the application is completed by that time. The current plan batch job uses the operation duration times to calculate the latest start time for every operation to meet the deadline. It assumes it can schedule the application at any time between the input arrival time and the latest start time.

To give OPC flexibility to process the work in the most efficient and effective manner, you should have the widest possible span between the input arrival and deadline times. This allows OPC more scope in scheduling the use of resources. Consider what happens if an application's input arrival time is 21:00, the deadline is 21:30, and the critical path total run time duration is 30 minutes: OPC has no leeway at all.

#### **10.1.7 Database and plan relationship**

It is very important to remember that there is no dynamic link between OPC's database and the plans. The long term plan calculates when applications should run based on information in the database. The long term plan is created, extended, and modified by running a batch job. If you make a change to scheduling information in the database, you will not see that change take effect in the long term plan until the batch job is run. Similarly, if you modify an application in one of the plans, that change is not propagated to the database.

Any change you make to an application in the plan affects just that one occurrence of the application. If you want to change the application for all future runs, you should update its definition in the database.

#### **10.1.8 Long term plan length**

The purpose of the long term plan is to calculate on which days applications should run in the future and to make inter-application dependency links. Its most important use within the data center is to show staff the result of that calculation. They can see what work will be run in the future and check that scheduling is correct. If work that should run is not listed in the long term plan

or is shown on the incorrect days, scheduling staff can make adjustments in advance.

So, the long term plan should look forward a suitable length of time to allow for checking and correction. Most data centers have a long term plan that covers a time frame from four to twelve weeks in the future. This is a horizon that is regularly extended. The maximum length of the plan is four years, and a few companies have long-term plans that cover the forthcoming twelve months, but such lengthy plans take longer to calculate and, usually, are not necessary.

#### **10.1.9 Current plan length**

The vast majority of data centers extend their current plan once a day by 24 hours. It is possible to extend over a longer time frame, but this can cause serious problems and is not recommended. If a current plan was extended to cover 48 hours, all applications scheduled for both days would be brought into the current plan, and once applications are in the current plan, their jobs are eligible to run. If you had a daily scheduled application, you would get two occurrences appearing in the plan, and both could run on the first day unless they were time-dependent.

Large data centers with a heavy workload sometimes prefer to extend by a shorter period, 8 or 12 hours. This makes the current plan more manageable to work with.

#### **10.1.10 Calendar maintenance**

A calendar allows you to define working days and free (non-working) days and is one of the factors used by the long term plan batch job when calculating what days applications will be scheduled. Even data centers that work 24/365 are likely to want to vary processing on official holidays, such as the new year, because their customers and service suppliers are not working; so, a calendar is used to define official holidays. Multiple calendars may be defined with different holiday dates. This is useful if you are responsible for scheduling work for different countries.

Because official holiday dates are not announced by governments more than two years in advance, some mechanism should be put in place to annually remind the scheduler to update the calendar. A simple effective solution is to create an application with one operation at a manual workstation. This should be scheduled to run annually near the end of the year. The operation text and operator instructions will contain the reminder about updating the calendar.



When the calendar has been updated, the operation status is manually changed to complete, and the application is not seen for another year.

### 10.1.11 Complex operation dependencies

Consider simplifying complex dependencies within an application by using non-reporting workstations. The job stream, shown in Figure 199, has eight jobs, the first four are predecessors of every one of the second group of four.

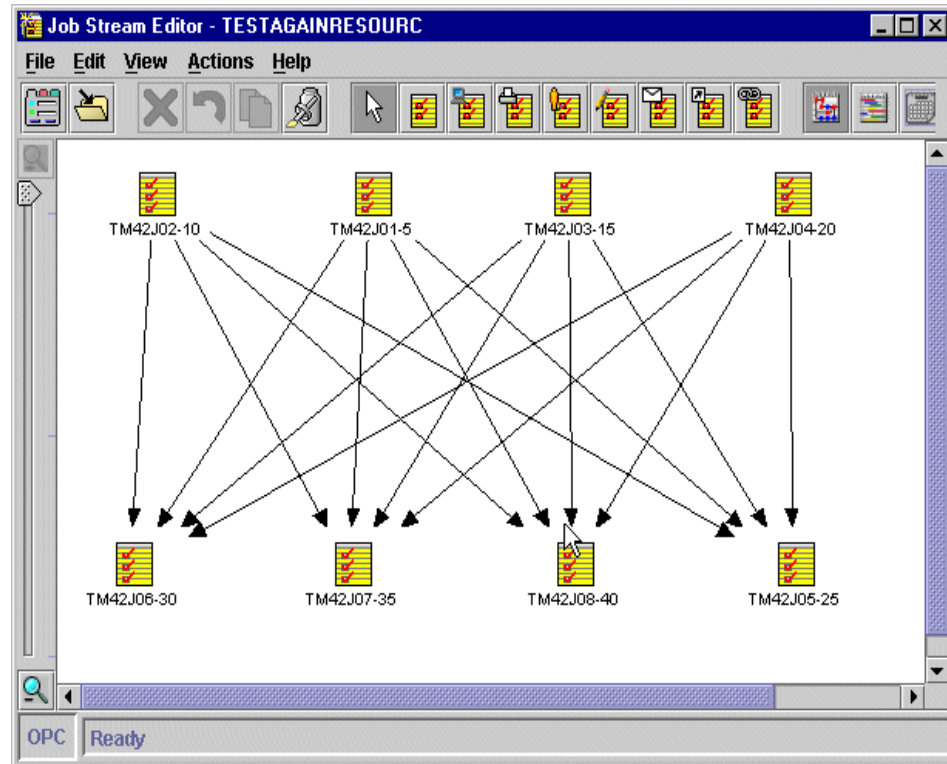


Figure 199. Complex dependency

Even in such a small job stream, the dependencies are getting complicated. By inserting a non-reporting work station, as shown in Figure 200 on page 316, you can simplify the dependencies.

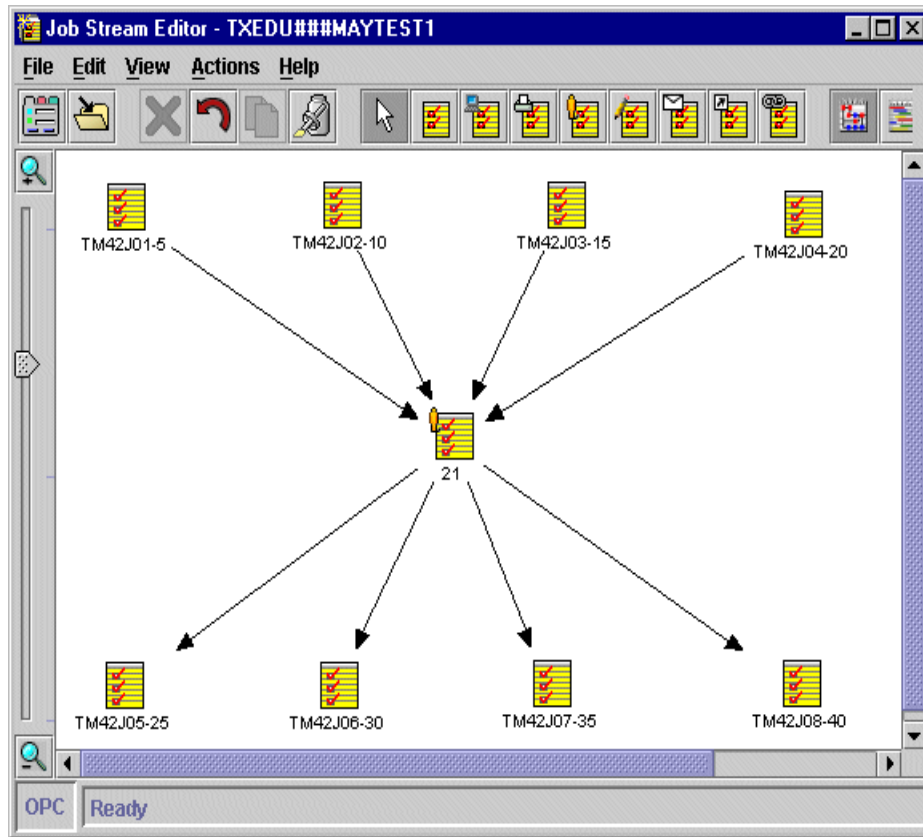


Figure 200. Dependency simplified with non-reporting workstation

Job number 21 is a non-reporting workstation. A non-reporting workstation acts as a dummy job; when its predecessors are completed, its status changes to complete, thus, releasing its successors. No real work takes place.

Another use is for satisfying the OPC requirement that all operations in an application must be linked by dependencies. If you have fifty small jobs that are not dependent on each other and you do not want to create fifty separate applications, you can put them in one application where the first operation is at a non-reporting workstation. All fifty jobs would use the non-reporting workstation as a predecessor. As soon as the application comes into the current plan, the status of the non-reporting workstation turns to complete, releasing the fifty jobs.

Creating a non-reporting workstation is simple. It needs to be a General workstation with the non-reporting attribute set as shown in Figure 201.

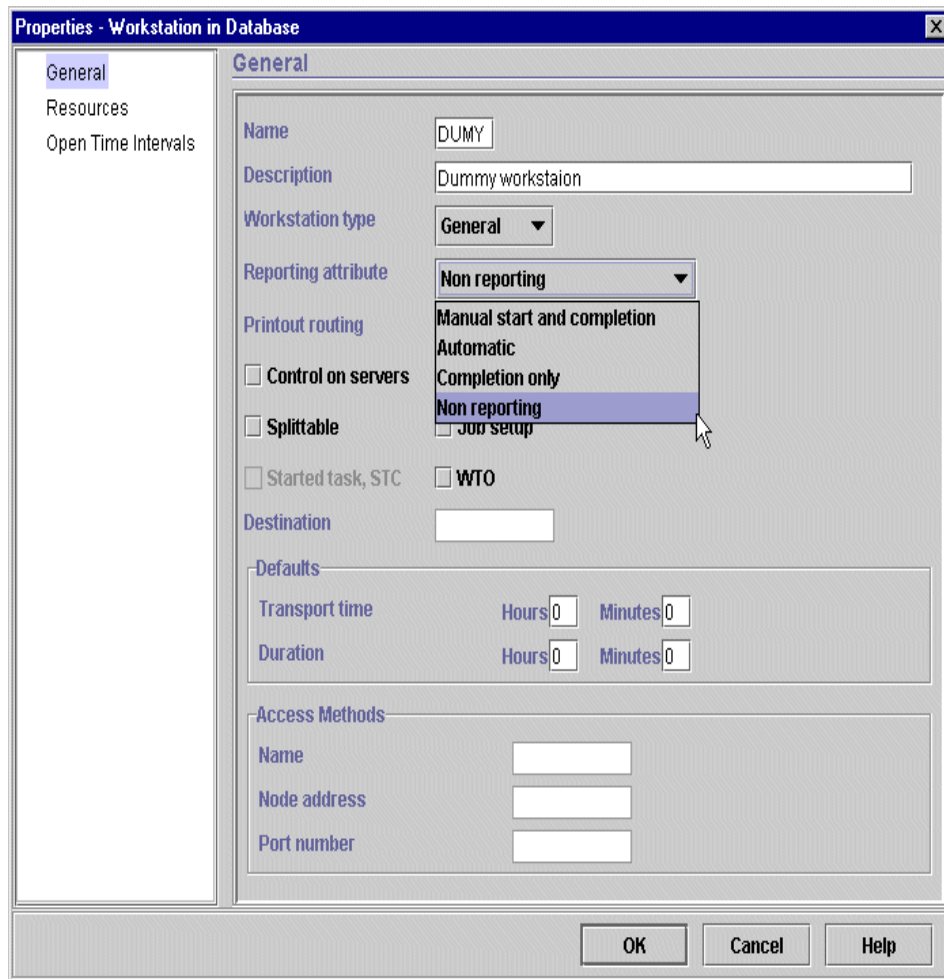


Figure 201. Creating a non-reporting workstation

Some data centers always use a non-reporting workstation as the very first operation from which all other operations in the application are dependent, and the very last operation is a non-reporting workstation. These are used to provide the only connection points for external dependency links and are intended to simplify coding inter-application dependencies.

#### 10.1.11.1 Using the EX command

We have observed operations staff using the `EX` command to force OPC to submit jobs. The `EX` command is very powerful. It causes a job to be submitted even when:

- Required OPC resources are not available
- Job submission is switched off
- There is a time dependency

The `EX` command overrides all restrictions other than non-completed predecessor dependencies.

While there may be legitimate occasions when the `EX` command is appropriate, it must be clearly understood that OPC always has a good reason for not submitting a job, and while OPC shows a reason why a job in A R or \* status has not been submitted, it can only show one reason. For instance, if you look at the operation, shown in Figure 202 on page 319, the reason stated is that it is *waiting for resources*.

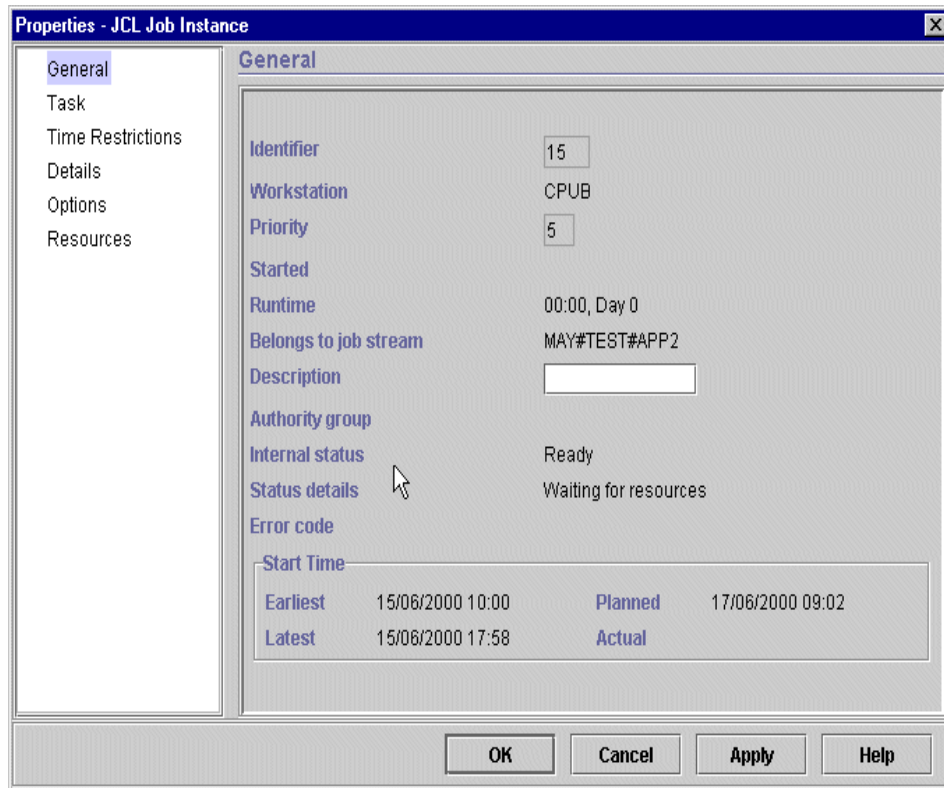


Figure 202. Job waiting for resources

The operator discovers that the job requires a special resource that is set as unavailable and decides that it is no longer a reason to hold back this job. Because it is fast and easy to use, the `EX` command is issued. A right-mouse click against the job on the plan view gives the window shown in Figure 203 on page 320.

MAY#TEST#APP2	10	TM42J08	CPUB	✓	Successful	Complete
MAY#TEST#APP2	15	TM42J15	CPUB	●	Ready	Ready
MAY#TEST#APP2	20			⌚	Waiting	Waiting
MAY#TEST#APP2	25			✓	Successful	Complete
MAY#TEST#APP2	4			⌚	Waiting	Arriving
MAY#TEST#APP2	5			⌚	Waiting	Waiting
MAY#TEST#APP2	10			✓	Successful	Complete
MAY#TEST#APP2	15			⌚	Waiting	Waiting
MAY#TEST#APP2	20			⌚	Waiting	Waiting
MAY#TEST#APP2	25			✓	Successful	Complete
TESTHOFF41	10			✗	Error	Error
TESTHOFF41	20 ADD JOB	TSETVS41	CPUC	⌚	Waiting	Waiting
TESTHOFF41	30	TSETVS41	CPUC	⌚	Waiting	Waiting

Figure 203. Issuing the EX command

OPC submits the job, but what was not apparent is that the job was also waiting on parallel servers. OPC only shows one reason why the operation has not been submitted.

It is best to let OPC decide when best to submit jobs, and to change resource settings in the plan when necessary. Use EX only as the last resort.

### 10.1.12 Education

To realize the most effective use of OPC, you should book your schedulers, operation staff, and others that may use OPC and the JSC on training courses. IBM/Tivoli regularly schedule public OPC courses worldwide. An alternative is to request a private course for your company, to be run at the nearest IBM/Tivoli location or on your premises. It is beneficial to send one or two key personnel on a public class when you first start the project to move to OPC and arrange to train the rest of the staff when you install OPC. Set up a separate OPC system to be used for in-house training and testing so staff can practice and familiarize themselves with using the product.

Information about IBM and Tivoli classes can be found on the Web at:

- [www.ibm.com/services/learning](http://www.ibm.com/services/learning)
- [www.tivoli.com](http://www.tivoli.com)

Courses are run worldwide and you can find the training courses nearest you by selecting your country. Information about an OPC course on the UK Learning Services Web-site is shown in Figure 204 on page 321.

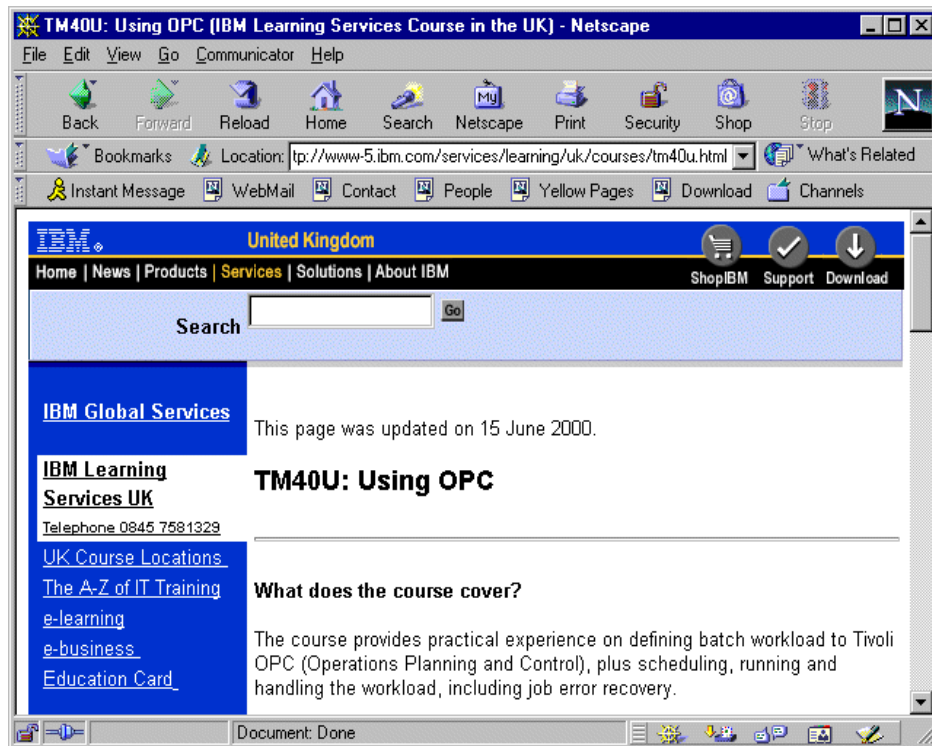


Figure 204. Course description on the Web

## 10.2 Tivoli Workload Scheduler

In this section, we will cover best practices when implementing a TWS network.

### 10.2.1 TWS architecture

One of the most important aspects of deploying a successful TWS network is its architecture. When designing the network, the goal should be to achieve the maximum degree of balance and concurrency. A balanced TWS network is one in which mailman servers and Domain Managers all have roughly similar workloads to process throughout the day. Concurrency is achieved by having multiple mailman servers or Domain Managers simultaneously distributing and processing data from the Master.

Another critical architectural goal should be to insulate the Master mailman process from being connected directly to the agents. The Master or default

mailman process (also known as the " " serverID) is critical to TWS processing on the Master. If it is also serving FTAs or Standard Agents, the Master is more vulnerable to outages due to network or machine hangs out on the TWS network. It is important to note that the serverIDs should only be assigned to FTAs, standard agents, or Domain Managers. They should never be assigned to Extended Agents.

#### **10.2.1.1 Small - Less than eight agents with one Master**

Two mailman servers can handle 6 - 10 agents, all in the same domain. Each agent should be attached to a mailman process other than the Master mailman process. To enable this, the serverID field should contain a character (A-Z) or a number (0-9). In cases where agents are running large numbers of short jobs, more mailman servers can be added to distribute the resulting message load.

#### **10.2.1.2 Medium - 8 to 100 agents with one Master**

Each group of eight agents in the TWS network should have a dedicated mailman server. All agents are still in a single domain.

#### **10.2.1.3 Large - More than 100 agents with one Master**

Domain Managers should be used with each hosting 60-100 agents. Each Domain Manager should have a dedicated mailman server on the Master. Multiple mailman servers must be running on each Domain Manager. There is one server ID for every eight agents hosted by the Domain Manager.

#### **10.2.1.4 serverID**

The Tivoli Workload Scheduler documentation recommends that each mailman server process should serve eight agents. This is only intended as a guide, and the tuning of this number can have a significant influence over the performance of the TWS network. Values in the range of 4 - 12 are reasonable, and which one is right for a particular network depends on many factors including hardware size, network performance and design, and TWS workload.

As a TWS network grows, this ratio should be tuned to minimize the initialization time of the network. Initialization time is defined as the time from the beginning of the Jnextday job until all functional agents are fully-linked.

### **10.2.2 What version of TWS should be used?**

TWS V7.0 with patch 7.0-MAE-0001 and TWS V6.1 with patches 6.1-MAE-0130 and 6.1-MAE-0140 are the only valid choices for new implementations of Tivoli Workload Scheduler at the time of this writing. Tivoli



Maestro V6.0 is not appropriate for any TWS production networks, and TWS customers should migrate to V6.1 or V7.0 as soon as possible. TWS V6.0 has problems that do not exist in later releases and no further patches will be released for it.

### **10.2.3 Choosing platforms**

UNIX has advantages over Windows NT in the areas of scalability and reliability. Properly-managed environments with AIX, Solaris, and HP-UX are extremely available. For the roles of Masters and Domain Managers, UNIX is probably a better choice. It is this writer's experience that systems based on AIX V4.3.3 running on Enterprise Server class IBM RS/6000 systems configured for High Availability host TWS Masters and Domain Managers very effectively.

TWS versions prior to TWS V6.1 with patch 6.1-MAE-130 and 6.1-MAE-140 have some problems when running on Windows NT 4.0. If you are experiencing problems on the NT platform, it is recommended that you remove the TWS Windows NT FTA and install the latest version of TWS. Before even starting the node, apply the latest patches, which are available at <http://ftp.tivoli.com/support/patches>.

### **10.2.4 Operating system resources**

In this section, we will cover operating system resources used in various TWS configurations and their ramifications in terms of performance.

#### **10.2.4.1 Processes**

A Master or Domain Manager will have one writer process for each agent it hosts and one mailman server process for each unique serverID specified. At least one process will exist for each job running locally on the workstation. This can add up to considerably more processes than most servers are initially configured to handle from a single user. Work with your system administrator to make sure, in advance of a TWS deployment, that the system has been configured to host the number of processes that TWS can generate. Make sure to check both the system-wide and per-user process limits. If the operating system hits one of these limits while TWS is running, TWS will eventually come down completely on the node. If this happens on a Master or Domain Manager, scheduling on multiple nodes may be affected.

#### **10.2.4.2 Open files**

TWS processes on UNIX Masters and Domain Managers can use a significant and somewhat unpredictable number of open files at any given

time. Customers have not commonly reported this failure on Windows NT. This failure produces an error message similar to the following:

```
MAILMAN:08:44/+ Too many open files on events file line = 1400. [1106.2]
```

The number of file descriptors that are pre-allocated for this purpose is a tunable operating system parameter. A TWS administrator can use the UNIX utility `lsof` to determine the approximate number of files that TWS is using. `lsof` binaries for a wide variety of UNIX operating systems can be obtained at the following URL:

```
ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/binaries/
```

Work with your system administrators or consult your operating system vendor documentation to determine the specific procedure for tuning this parameter on your platform. Ensure that an ample number of file descriptors are allocated at boot time. If the operating systems run out of available file descriptors, TWS and other applications running on the machine will be halted, production data may be lost or corrupted, and scheduling on multiple nodes may be affected.

#### **10.2.4.3 File locks**

When TWS coexists with other large applications on Master or Domain Managers, the availability of file locks can be an issue. In particular, Database applications can use large numbers of locks. Work with system administrators or consult vendor operating system documentation to determine how to tune this resource on your platform.

#### **10.2.4.4 Disk space**

TWS can consume large amounts of disk on networks that run large numbers of jobs. On production systems, disk space availability should be monitored using an automated tool. TWS should reside on its own filesystem to minimize the effect of other applications on TWS operation and vice versa. In medium and large implementations, TWS should reside on its own physical disk. This configuration will enhance performance and reliability.

The amount of disk space that TWS uses is greatly dependent on the workload. Peaks in disk usage can be minimized by regularly removing files from the `~maestro/schedlog` and `~maestro/stdlist` directories. The built-in command, `rmstdlist`, is provided to remove aged job output and TWS logs.

Disk space utilized by the TWS scheduling object or mozart database can be minimized by regularly running the `composer build` commands on the Master. This will also correct some database errors that could cause `composer` or `Jnextday` to fail.

**Note**

Before running any of these commands, make sure that you have a verified and available backup of at least the TWS file system(s).

More information about `rmstdlist`, `composer build`, `Jnextday`, and the `~maestro/schedlog` and `~maestro/stdlist` directories can be found in the *Tivoli Workload Scheduler 7.0 Reference Guide*, GC32-0424.

**10.2.4.5 Inodes**

TWS can consume large numbers of inodes when storing large numbers of job output on UNIX systems in `~twuser/stdlist`. Inodes are not an issue on Microsoft Windows operating systems. On an FTA, which runs large numbers of jobs, inode consumption can grow quickly. If TWS aborts due to a lack of inodes, production data may be lost. Inode usage can be checked using the UNIX `df` command. Consult your operating system documentation for information on using this command.

**10.2.5 .msg files**

TWS uses files with the `.msg` extension as part of a mechanism for local and remote interprocess communication. These files are located in the TWS home directory (`~twuser`) and in the `~twuser/pobox` directory. The various TWS processes to communicate vital production data with each other via this mechanism. Because this information is vital, it is critical that these files not be corrupted or deleted unless absolutely necessary. If in doubt, contact Tivoli Customer Support before deleting. These files have a default maximum size of 1MB. When this limit is exceeded, TWS will shut itself down on the local node. The size of these files will grow if any of the following conditions exist:

- An agent is down and no operator intervention is performed.
- Network connectivity has been lost without operator intervention.
- Performance problems on the Master cause it to be unable to keep up with the pace of messages from the agents.

In some situations, it is necessary to increase the size of these files. In most cases, however, the TWS administrator should work to eliminate the cause of the message backlog rather than implementing the workaround of increasing message file size.

The size of the `.msg` files can be increased using the following command:

```
#evtsize filename.msg nnnnnnn
```

Where `nnnnnnn` is the new size of the message file. This change will remain until the file is deleted and re-created. To change the default creation size for `.msg` files, add the following line to `~maestro/StartUp` and `~maestro/.profile`:

```
export EVSIZE=nnnnnnn
```

Where, again, `nnnnnnn` is the size at which the `.msg` files are created after being deleted.

### 10.2.6 Ad hoc job/schedule submissions

The amount of space preallocated for ad hoc jobs and schedules is tunable. The following error can occur when too many jobs are submitted using `gconman` or with `conman submit` commands:

```
BATCHMAN:05:17/* *****
BATCHMAN:05:17/* Too many jobs are scheduled for BATCHMAN to handle
[2201.12]
BATCHMAN:05:17/* *****
```

The space allocated is controlled in the `~maestro/Netconf` file in the line similar to:

```
2002      son      bin/mailman  -parm value
```

By placing a `-1` in the field marked `value`, TWS is instructed to allocate the maximum available space for job records. Smaller allocations can be used, but be careful to leave TWS a wide margin of safety. The number of records used by each job varies based on the length of the job definition; so, it can be difficult to predict the size of this requirement. If this failure occurs, scheduling will be halted on the local node.

### 10.2.7 Timeouts

False timeouts can occur in large TWS networks when mailman disconnects from a remote writer process because no response has been received from the remote node over a time interval. The agent may have responded, but the message has become caught in traffic in the `.msg` file and does not arrive before the timeout expires.

The MAESTRO log in `~twuser/stdlist/(date)` would contain messages, such as the following:

```
MAILMAN:06:15/+
+++++
MAILMAN:06:15/+WARNING: No incoming from <cpu> - disconnecting.
[207 3.25]
MAILMAN:06:15/+
+++++
```

This timeout is controlled by a parameter in `~twuser/loclopts` called `mm unlink`. `mm unlink` is expressed in seconds, as are most of the other parameters in this file. The default value for `mm unlink` is 960 and should not be less than the value of `mm response`. Increase it in small increments until the timeouts cease. This could also be a sign of performance problems on the Master. If the Master is unable to handle the pace of incoming records from the agents, timeouts can occur on FTA to Master communications. A hardware upgrade or system reconfiguration might be considered.

### 10.2.8 Mailman and writer

TWS can experience performance problems when too many writer processes are running concurrently on the Master. Some releases of TWS can experience a slowdown of the Master mailman process when more than ~75 writer processes are running simultaneously on the Master. These writer processes compete against the Master mailman process for a file lock on the `mailbox.msg` file. This reduces the mailman process' ability to read records from the file and, thereby, limits the amount of incoming records that the Master can process.

This limitation has been removed in TWS 7.0 and in TWS V6.1 with patch 6.1-MAE-0140. It is always important, however, to use TWS domains to reduce the number of writer processes running simultaneously on the same CPU. The writer process for an FTA in a subdomain runs on the Domain Manager immediately above it and not on the Master.

### 10.2.9 Job Scheduling Console and Remote Console

Large TWS environments tend to have many operators, schedulers, and administrators who require access to the Master. The use of `gconman` with its display exported to an X display or an X emulator on a Microsoft Windows 98/2000/NT system can cause performance problems. Each of these `gconman` sessions consumes almost 10 MB of virtual memory on a UNIX Master. It also causes a large amount of disk activity and network traffic between the GUI client and the host. `gconman` is also a heavy user of CPU and network bandwidth. Depending on the size of the system, too many of

these processes running concurrently on the Master can affect the overall performance of the TWS network.

For Tivoli Workload Scheduler V6.1 implementations, TWS Remote Console V6.1 fully patched and running on Windows remote workstations provides a much more efficient way to give users access to TWS on the Master. Remote Console does not provide composer functionality, and, in cases where it is needed, gcomposer should be used. gcomposer does not contain an auto-refresh cycle and, therefore, does not consume the amounts of CPU and network bandwidth that gconman does. Be sure to use filters to reduce the amount of unnecessary data that is transmitted across the connection.

For Tivoli Workload Scheduler V7.0 implementations, the Job Scheduling Console can be used to reduce the amount of resources on the Master that are consumed by console monitoring activity. Limit views to only what the operator or administrator needs or wants to see in the JSC. Views of excessively large numbers of objects can cause delays in GUI operation.

### **10.2.10 Hardware considerations**

The hardware used to run the Master for a large TWS environment should usually be a midrange UNIX server. Some examples in this range are:

- IBM RS/6000 H50 and F50
- Sun E250 and E450
- HP 9000 K and N classes

It is important to remember that the TWS engine is not a CPU-intensive application. In most configurations, disk I/O is the limiting factor in its performance. Measures, such as putting TWS on its own physical disks, on a separate disk adapter, or on a RAID array, can boost performance in a large high workload TWS environment. Ultra2 or Ultra160(IBM Ultrastar) SCSI storage components can also relieve I/O bottlenecks. If a server is more than two years old, it may have storage interfaces and components with performance that falls well below current standards and may not perform well in particularly demanding environments.

### **10.2.11 Monitoring**

Automated monitoring is an essential part of a successful TWS implementation. Tivoli Distributed Monitoring and Tivoli Enterprise Console (TEC) are good examples of products that can be linked to TWS to monitor status and events from TWS. Distributed Monitoring can be used to monitor critical systems and resources that TWS depends on to complete its

production. TEC's logfile adapters can be used to transmit selected TWS events to a centralized IT console.

Some resources that are candidates for monitoring are:

- Swap Space
- Disk Space
- TWS Process Status
- TWS Events
- Network status
- System Load Avg
- .msg file size
- Free Inodes

### **10.2.12 High availability**

A Backup Master is a critical part of a highly-available TWS environment. If the production Master or a Domain Manager fail and cannot be immediately recovered, a backup Master will allow production to continue. See the “Creating a Backup Master” section in Appendix A of the *Tivoli Workload Scheduler 7.0 Planning and Installation Guide*, GC32-0422, for more information about how to set up a Backup Master.

Server hardware manufacturers and third parties offer a variety of high availability features and solutions that can be used to greatly increase the uptime of a critical TWS node. Some examples of are:

- IBM HACMP
- HP ServiceGuard
- Windows NT Clusters

All will work with TWS but the installation and configuration can be tricky. Consider Tivoli Services for large implementations if expertise is not in-house, or contact your hardware or software vendor for information about high availability servers.

### **10.2.13 Deployment**

Tivoli products, such as the Tivoli Management Agent and Tivoli Software Distribution, can assist in the deployment of large numbers of TWS agents. Such a solution is built on the unattended install mode provided by Install Shield.

In UNIX, unattended install is a matter of script writing.

In Windows, the setup.exe file on the TWS media supports record mode (-r) and silent mode (-s). The record mode produces an answer file of type .iss, which can be used as an input in silent mode. A complete explanation of the InstallShield options for setup.exe can be obtained from Tivoli Customer Support.

#### **10.2.13.1 Q/A Processes**

Good change control is a requirement for a large TWS environment. Test new workload and Tivoli software updates before deploying into production. Some attempt should be made to construct a dedicated test environment that, to some degree, mirrors the production TWS environment.

Operating System and/or TWS security should be used to tightly control user access to the production TWS Master. TWS should be able to get 90 percent or more of the machine's resources whenever it needs to. This is particularly true during the Jnextday job.



---

## Chapter 11. TWS Extended Agent for MVS and OS/390

Tivoli Workload Scheduler (TWS) Extended Agent for MVS and OS/390 gives you the ability to schedule and control MVS and OS/390 jobs using TWS job scheduling features.

---

### 11.1 TWS Extended Agent for MVS and OS/390 features

You may use TWS Extended Agent for MVS and OS/390 for the following purposes:

- Use the TWS scheduling functions to schedule MVS and OS/390 jobs to run at specific times and in a prescribed order.
- Define dependencies between TWS-launched jobs running on different systems and platforms including MVS and OS/390, UNIX, Windows NT, HP 3000, and SAP R/3.
- Define dependencies for TWS-launched jobs based on the completion of MVS and OS/390 jobs that were not launched by TWS.
- Define dependencies for TWS-launched jobs based on the existence of files on an MVS and OS/390 system.

---

### 11.2 Software prerequisites

TWS Extended Agent for MVS and OS/390 requires the following software versions:

*Table 18. Software requirements*

Software requirements	Version
TWS	Maestro 5.2, 6.x and TWS 70. Note: Use the most recent release (1.4.6) if you are running on OS/390 2.5 and above. Use Version 1.4.4 for Client or 1.4.3 from Tape.
Operating system	MVS/ESA OS/390
TCP/IP version	One of the following: IBM 3.1 or higher Interlink 3.1 or higher Open Connect 2.3 or higher
Job Scheduling Interface	JES/2 or JES/3, OPC/ESA 1.3.1 or higher

---

## 11.3 Installation on the TWS side

The Extended Agent for MVS and OS/390 software consists of the MVS-methods that reside on the TWS host and MVS gateway software that resides on MVS or OS/390. The MVS gateway software is installed *separately from a 3480 tape cartridge*.

We will cover installation to both UNIX and NT TWS hosts.

### 11.3.1 Installation on TWS (Unix host)

The TWS-MVS methods are delivered as tar files on the CD. You may perform the following steps to install them:

1. Stop TWS on the host workstation.
2. Log in as root and change your directory to TWSHOME.
3. Mount the installation CD, and restore the tar file:

```
tar xvf cd_folder/TWS/platform/MVS.TAR
```

Table 19 lists the tar parameters and their descriptions.

Table 19. Supported platforms

tar parameter	Description
cd_folder	The pathname of your CD drive or folder
Platform	The target platform from the list: <b>HPUX</b> Hewlett-Packard <b>AIX</b> IBM <b>MIPS</b> MIPS-based <b>INTEL</b> Intel-based <b>SOLARIS</b> Sun Solaris <b>SUNOS</b> SunOS <b>DGUX</b> Data General UNIX <b>DECUX</b> Digital UNIX

4. Execute customize as follows:

```
/bin/sh customize -update -uname[user's name]
```

5. With conman or gconman, start TWS on the host workstation. After installation, you can obtain version information about the software using the version command. For example:

```
version -f twshome/methods/version.info
```

### 11.3.2 Installation on TWS-(Windows NT host)

The TWS-MVS methods are delivered on CD. Perform the following steps to install them:

1. Stop TWS on the host workstation.
2. Log in as a user in the Administrators group or the Domain Administrators group, as appropriate.
3. Close any open Windows applications including the File Manager.
4. Insert the CD and copy the files to the TWS methods directory. For example, if TWS is installed in C:\WIN32APP\maestro, and the CD drive is D:, enter the following:

```
copy D:\MAESTRO\I386NT\*. *  
C:\WIN32APP\MAESTRO\METHODS
```

5. Run the Setup program. For example, if TWS is installed in C:\WIN32APP\MAESTRO, enter the following:  

```
C:\WIN32APP\MAESTRO\SETUP\SETUP.EXE
```
6. On the Welcome screen, click **Next**.
7. On the Account Information screen, enter the account name and password and click **Next**.
8. On the Setup Information screen, verify the information you entered and click **Next**.
9. A message informs you that TWS is being customized. When the *Workload Scheduler has been installed* message appears, click **OK**.
10. Start TWS on the host workstation.

---

### 11.4 Installation on the OS/390 side

The MVS and OS/390 gateway module are delivered on a 3480 tape cartridge written using the non-IDRC (uncompressed) format. After you have unloaded the tape, you will find two datasets: The *control library* and the *load library*.

The loadlibrary must be APF authorized. This can be done with the SETPROG command. For example:

```
SETPROG APF,ADD,DSN=YOURID.UNISON.LOADLIB,  
VOL=xxxxxxx
```

where: xxxxxxxx is the volume serial number where the load library is located.

or:

```
SETPROG APF,ADD,DSN=YOURID.UNISON.LOADLIB,VOL=SMS
```

which indicates a volume under the control of SMS.

The two MVS program components of the TWS MVS gateway are:

- *TSITCP02* - This program establishes that TWS is tracking on the MVS system. The program is started by the TSDSPACE job.
- *TSITCP00* - This is the gateway program that manages TCP/IP communications between TWS and the MVS system. It is started by the TSSERVER job. TSITCP00 translates Tivoli Workload Scheduler commands to MVS equivalents, routes MVS information back to TWS, and performs EBCDIC-ASCII data conversions. Both of the programs run as started tasks, with a TIME=NOLIMIT parameter. TSITCP02 is always started first and followed by TSITCP00. If the programs must be terminated for any reason, they should be stopped, not cancelled, to ensure that they shut down gracefully without impacting other programs that use the IEFU84 Exit.

The control library contains the JCL for STC TSDSPACE and TSSERVER.

#### Installation Note

TSDSPACE must be up before TSSERVER is started. To shutdown, shutdown TSSERVER first before shutting down TSDSPACE.

TSDSPACE will create the Data Space and install the IEFU84 exit. Add TSITCP00 to the AUTHCMD NAMES section of the SYS1.PARMLIB member IKJTSoxx, where the suffix xx completes the member name that must be active either after the next ipl or dynamically. The following screen shows the definition.

```
/* LIB: CPAC.PARMLIB(IKJTSo00) */
/* DOC: THIS MEMBER IS USED AT IPL TIME TO DEFINE THE AUTHORIZED */
/* COMMAND LIST, THE AUTHORIZED PROGRAM LIST, THE NOT */
/* BACKGROUND COMMAND LIST, THE AUTHORIZED BY THE TSO SERVICE */
/* FACILITY LIST, AND TO CREATE THE DEFAULTS THE SEND COMMAND */
/* WILL USE. */
/*
/*
AUTHCMD NAMES(          /* AUTHORIZED COMMANDS */ +
    TSITCP00           /* FRANKE */ +
    BINDDATA BDATA     /* DMSMS COMMANDS */ +
    LISTDATA LDATA      /* */ +
```

Modify the following JCL for TSDSPACE that to suit your installation. It is recommended that this job be a started task, rather than a submitted job stream. *It is important that the job not be canceled.*

```
//TSDSPACE PROC MEMBER=TSPARMS
//IEFPROC EXEC PGM=TSITCP02,
//          REGION=4M,
//          TIME=NOLIMIT
//STEPLIB DD DSN=SFRA4.UNISON.LOADLIB,DISP=SHR
//SYSTCPD DD DISP=SHR,DSN=TCPIP.IV4.TCPPARMS(TCPDATA)
//SYSTSIN DD DSN=SFRA4.UNISON.CNLT(&MEMBER),DISP=SHR
//SYSTSPRT DD SYSOUT=*
```

Modify the TSSERVER JCL as follows:

```
//TSSERVER PROC MEMBER=TSPARMS
//IEFPROC EXEC PGM=IKJEFT01,
//          PARM='TSITCP00',
//          REGION=4M,
//          TIME=NOLIMIT
//EQQMSG DD SYSOUT=*
//EQQDUMP DD DISP=SHR,DSN=OPCESA.XAGENT.DUMP
//EQQMLIB DD DISP=SHR,DSN=OPCESA.V2R3M0.SEQQMSG0
//EQQYPARM DD DISP=SHR,DSN=SFRA4.UNISON.CNLT(OPCSINIT)
//STEPLIB DD DISP=SHR,DSN=SFRA4.UNISON.LOADLIB
//SYSTCPD DD DISP=SHR,DSN=TCPIP.IV4.TCPPARMS(TCPDATA)
//          DD DISP=SHR,DSN=TCPIP.SEZATCP
//SYSTSIN DD DISP=SHR,DSN=SFRA4.UNISON.CNLT(&MEMBER)
//SYSTSPRT DD SYSOUT=*
```

The SYSTSIN allocation points to the parameter library for both started tasks. See Table 20 for a description of OPC-related sysstin parameters.

Table 20. OPC-related sysstin parameters

Variable	Description
<b>Debug(no)</b>	If set to YES, it causes the MVS gateway to output diagnostic messages.
<b>MAXWAIT</b>	The maximum amount of time, in hundredths of a second, to wait for a response to commands. Do not set it above 500.
<b>MCSSTORAGE</b>	The amount of storage, in megabytes, used by each extended console used by the MVS gateway.
<b>OPCMMSGCLASS(*)</b>	The message class for the dynamically allocated message logs used by OPC. The asterisk causes the class to be set the same as TSSERVER. Ignored if OPC is not used.
<b>OPCSUBSYSTEM(OPCS)</b>	The subsystem name used for communications with the OPC subsystem.
<b>PEERADDRESS(0 0 0 0)</b>	The default, 0.0.0.0, permits access by any host. For better security, enter the IP address of the Workload Scheduler host of the MVS x-agent. Note: Do not include the "." (period) between the 0's in your code. The periods will show up in the display!
<b>PORT(5000)</b>	The TCP port number used by TWS and the MVS gateway for communications. This must be the same as the value entered in the TCP Address field of the MVS x-agent workstation definition.
<b>PUTLINE(YES)</b>	When set to YES, it directs trace information to DDNAME SYSTSPRT.
<b>SUBSYS(UNIS)</b>	The prefix used by the Extended Agent for MVS as the first four characters of extended console names. It is also used as the first four characters of internal reader DDNAMES. Do not change it.

Table 21 lists the OPC-related SYSTSIN variables.

Table 21. OPC-related SYSTSIN variables

Variable	Description
<b>PUTLINE</b>	When set to YES, it directs trace information to DDNAME SYSTSPRT. (Recommended)
<b>SVCDUMP(NO)</b>	When set to YES, abends will cause an SVC dump. We recommended to set it to yes.
<b>TCPIPCNAME(TCPIP)</b>	The name of the TCPIP address space when the IBM version of TCPIP stack is used.
<b>TCPIPSTACK(IBM)</b>	The vendor of TCPIP stack (IBM, INTERLINK or OPENCONNECT).
<b>TERMINATOR(X'25')</b>	The transaction termination character. Do not change it.
<b>WTP(NO)</b>	When set to YES, it directs trace information to SYSLOG as write-to-programmer information.

---

## 11.5 Defining an OS/390 Extended Workstation

Tivoli Workload Scheduler (TWS) launches jobs on an Extended Agent for MVS and OS/390 workstation. The Extended Agent for MVS and OS/390 is defined in a standard TWS workstation definition, which gives it a name and identifies the access method to be used. The Extended Agent for MVS and OS/390 workstation is a workstation definition linked to an instance of OS/390 Host. To launch a job on an Extended Agent for MVS and OS/390 workstation, the TWS host executes the access method and passes it information about the job. The access method communicates with the instance of OS/390 host and monitors the job through completion and writing job progress and status information to the job's standard list file. Extended Agent for MVS and OS/390 workstation definitions are required for each MVS and OS/390 entity through which TWS will schedule and launch jobs. They are defined in the standard manner and include the name of the x-agent's host and the method name.

The following is the procedure to create a Workstation using the Job Scheduling Console.

To define an MVS and OS/390 Extended Agent workstation on the JS Console, follow these steps:

1. From the main JS Console window, either, from the top toolbar, select the **New Workstation** icon and continue from Step 2 below, or, from the main JS Console window, right-click on **TWS7.0**, and select **New Workstation** from the drop-down menu. This is shown in Figure 205.

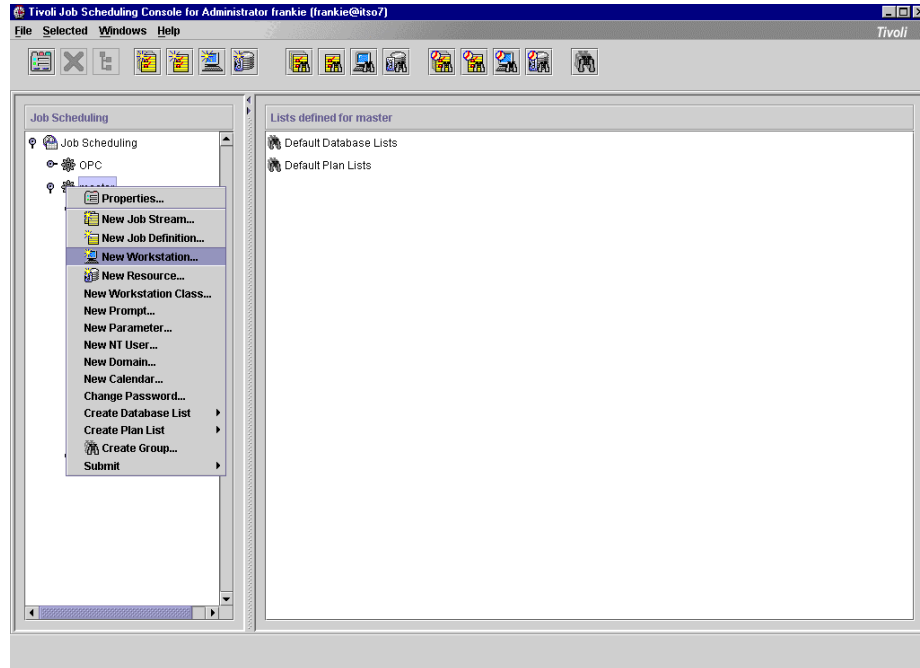


Figure 205. Creating Xagent workstation

2. In the Properties screen, complete the following fields: Workstation Name, Node (host or IP address), and TCP port.



**Properties - Workstation in Database**

Name: OS390

Node: 9.39.62.19

TCP Port: 5000

Operating System: Other

Domain: MASTERDM ... Use Master

Time Zone:

Description: Os/390 Xagent WS

**Options**

Workstation Type: Extended Agent

☒ Auto Link ☐ Ignore

☐ Full Status ☐ Resolve Dependencies

Server:

Access Method: mvsopc

Host: MASTER ...

OK Cancel Help

Figure 206. Workstation definitions

3. Select the relevant operating system. Use **Other** for the MVS and OS/390 agent.
4. Enter the Domain or select it from the Master.
5. Enter the Time Zone and Description (optional).
6. In the Options area, select the Extended Agent. The screen is defaulted with Autolink checked.
7. Enter access method MVSOPC, MVSCA7, or MVSJES. Enter the host name. Click on **OK** and then **File->Save**. Close the window.

Workstation definitions are described in Table 22 on page 340.

Table 22. Workstation definition parameter

Field	Description
Name	The TWS workstation name of the Extended Agent for MVS.
Node	The node name or IP address of the MVS system. This can be the same for more than one Extended Agent for MVS. (Appears in the JS Console Only)
TCP Port	The TCP address (port number) of the MVS gateway on the MVS system. Enter the same value as the PORT parameter described in the SYSTSIN variable table.
Operating System	Select other
Domain	Use masterdm
Time Zone	Used to set the required time zone.
Description	An optional free-form textual description of the workstation (up to 40 characters).
Workstation type	Select Extended Agent. (JS Console GUI only)
Resolve dependencies	Not used
Full status	Not used
Autolink	Not used
Ignored	Select if you want TWS to ignore this workstation definition.
Server	Not used
Host	The TWS name of the Extended Agent's MVS host. This is the TWS Master, a fault-tolerant agent, or a standard agent.

Database Workstation panel shows the defined Xagent CPU:

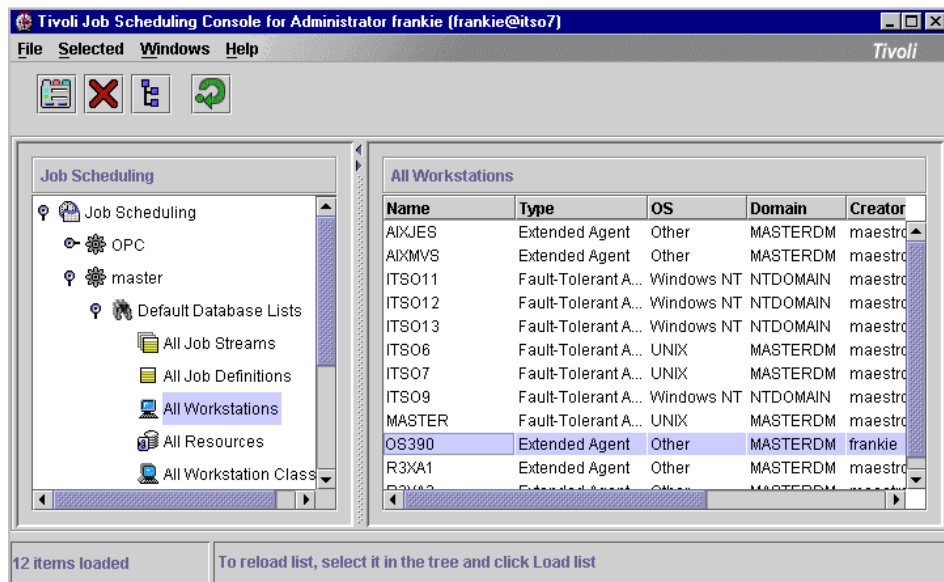


Figure 207. Workstation view

You need to run the Jnextday script to activate the new workstation. The workstation must then be in active status.

## 11.6 Creating a method options file

To select options for a method, create an options file having the same path name as the method. This should have an *opts* extension. For example, create the following files on UNIX (assuming the TWS home directory is /usr/lib/maestro):

```
/usr/lib/maestro/methods/mvsopc.opts
```

For Windows NT create the following files (assuming TWS is installed in the path, C:\WIN32APP\maestro):

```
C:\WIN32APP\maestro\METHODS\MVSOPC.OPTS
```

Table 23 on page 342 shows the parameter for the method option file:

Table 23. Option files

Option file entries	Description
LJuser=name	(Required) Assigns the user name used by the access method to launch jobs. This must be a valid UNIX user who is able to connect to Workload Scheduler's MVS gateway on the MVS system and submit jobs.
Cfuser=name	(Required) Assigns the user name used by the access method to check file dependencies. It must be the same as LJuser.
Gsuser=name	(Optional) Assigns the user name used by the access method to check on-TWS-launched jobs on MVS that are used as dependencies. The default is root.
Checkinterval=min	(Optional) Defines the polling rate, in minutes, for checking the status of MVS jobs that were launched by the method. Fractional values are accepted; for example, .5 for 30 seconds or 1.5 for one minute and 30 seconds. The default is 2. When checking non-TWS-launched jobs on MVS that are used as dependencies, the method uses the TWS local option bm check status instead of CheckInterval to determine its polling rate.
Blocktime=min	(Optional) Defines the amount of time, in minutes, the method will wait for a response to a status check before timing out. This value must be less than the value of the CheckInterval (described above) and the TWS local option bm check status. Fractional values are accepted; for example, .5 for 30 seconds or 1.5 for one minute and 30 seconds. The default is 2.
Retrycount=count	(Optional) Defines the number of times a status check is attempted before TWS writes a timeout message to a job's stdlist file and marks it in the abend state. See CheckInterval and BlockTime above for more information. The default is 10.

## 11.7 Defining internetwork dependencies for OPC with the new JSC

TWS job definitions are required for each MVS and OS/390 job you intend to schedule and launch with TWS. They are defined similarly to other TWS jobs and include job name, user name, special script name options, and optional recovery options. There are two possibilities to define internetwork dependencies for OPC methods: One *launches* OPC jobstreams and waits for its completions, and the other *monitors* only a predefined OPC jobstream until completion. Both complete events can be used to start successors at the TWS side.

### 11.7.1 Definitions for OPC Jobstreams launched by TWS

Perform the following steps to define OPC Jobstreams launched by TWS:

1. Create a new job definition for TWS as shown in Figure 208.

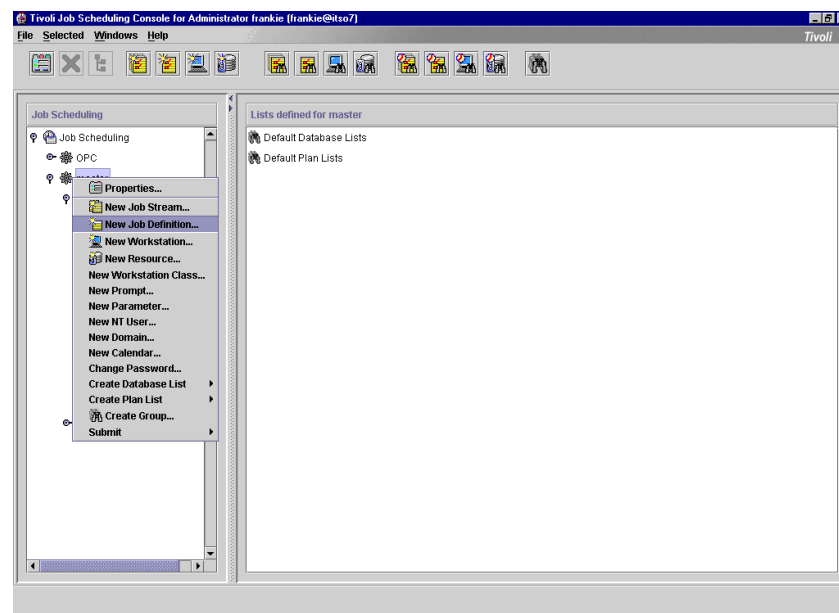


Figure 208. create new job definition

2. Select **Extended Agent Task** for the task type as shown in Figure 209 on page 344.

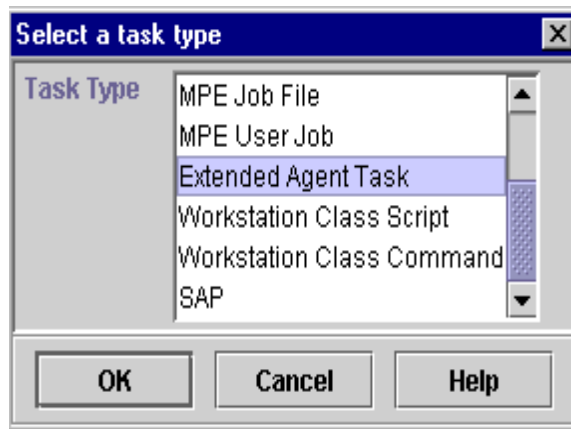


Figure 209. Task type selection

3. Define the OPC parameter in the task field as shown in Figure 210. At least, the application name is required.

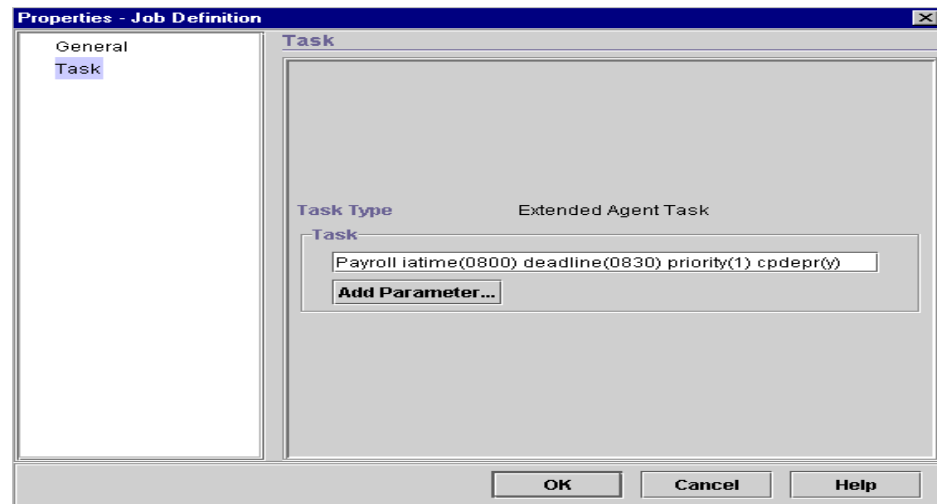


Figure 210. Internetwork definitions

The syntax of the OPC parameter is as follows:

```
appl [IA (yymmddhhmm) | IATIME (hhmm) ] [ ... ]
[DEADLINE (yymmddhhmm) | DEADLINETIME (hhmm) ]
[PRIORITY (pri) ]
[CPDEPR (Y|N|P|S) ]
```

where:

- **appl** is the name of the OPC application to be inserted into the current plan.
- **IA** is the input arrival date and time in the form: yymmddhhmm.
- **IATIME** is the input arrival time in the form: hhmm.
- **DEADLINE** is the deadline arrival date and time in the form: yymmddhhmm.
- **DEADLINETIME** is the deadline arrival date and time in the form: hhmm.
- **PRIORITY** is the priority (1-9) at which to run the application.
- **CPDEPR** is the current plan dependency resolution selection.
  - Y - Add all successor and predecessor dependencies
  - N - Do not add any dependencies (default)
  - P - Add predecessor dependencies
  - S - Add successor dependencies

**Note**

All TWS Jobs that launch OPC applications must run on the extended agent workstation.

### 11.7.2 Definitions for OPC jobstreams monitored by TWS

These internetwork dependencies can be created based on a job or schedule. The following figures show the job-based dependency. Two jobs are already defined in TWS. In order to create an internetwork dependency, perform the following steps:

1. Click on the **white sheet icon** as shown in Figure 211 on page 346.

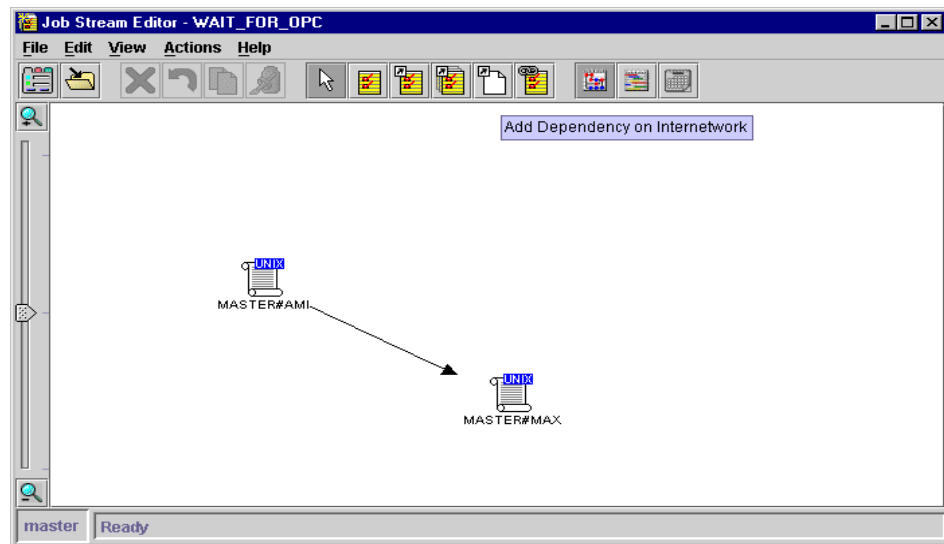


Figure 211. Add Internetwork dependencies

2. Select **OS/390 Extended Agent workstation** as shown in Figure 212.

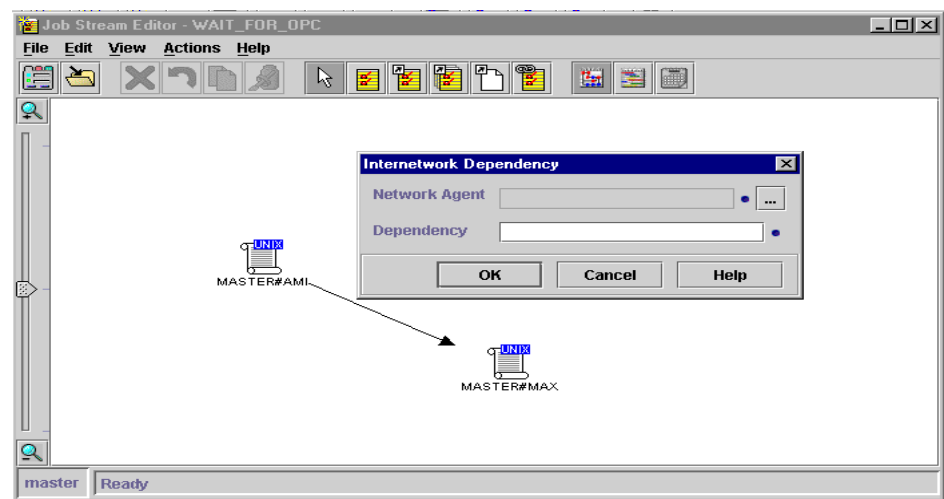


Figure 212. Finding the Network agent

3. Define the OPC parameter in the dependency field as shown in Figure 213 on page 347.



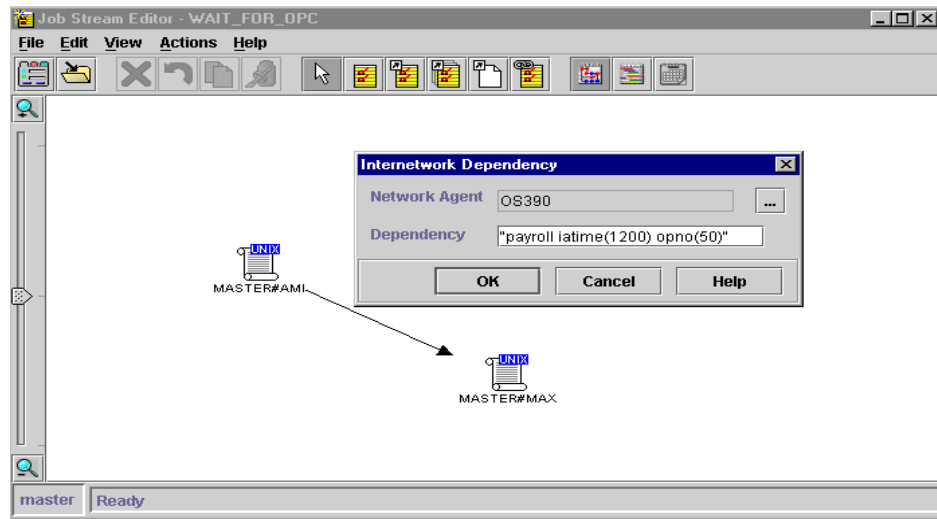


Figure 213. Define the internetwork dependency

#### Note

All definitions must be enclosed in double quotes.

For OPC jobs:

```
"application[IA(yymmddhhmm) | IATIME(hhmm)] [...] "  
[JOBNAME(jobname)]  
[OPNO(num)]
```

where:

- **application** is the name of the OPC application in the current plan.
- **IA** is the input arrival date and time.
- **IATIME** is the input arrival time.
- **JOBNAME** is the MVS job name.
- **OPNO** is the operation number (1-99). If included, the job is considered completed when it reaches this operation number.

You can only add a link in the direction in which the internetwork dependency becomes a predecessor as shown in Figure 214 on page 348.

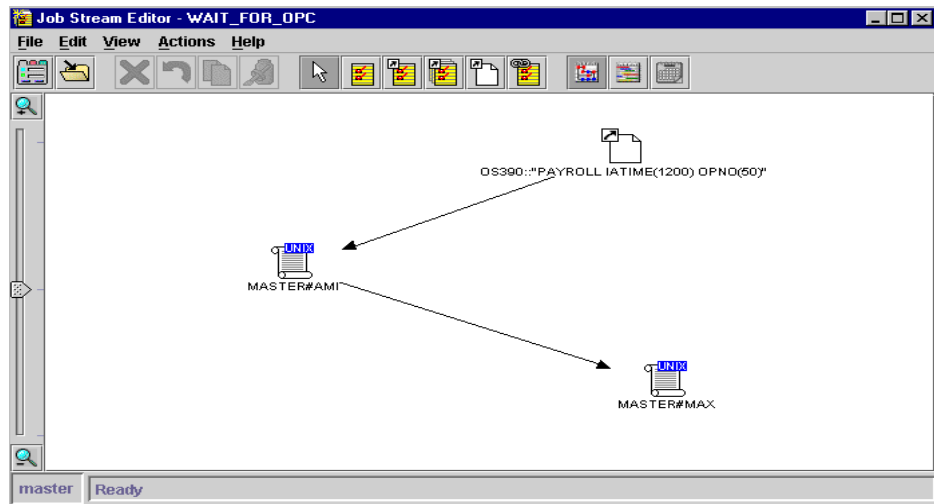


Figure 214. Creating links to the TWS network

4. If you need the dependency at a schedule-based level, click the **File** pull-down menu and select **External Dependencies** as shown in Figure 215.

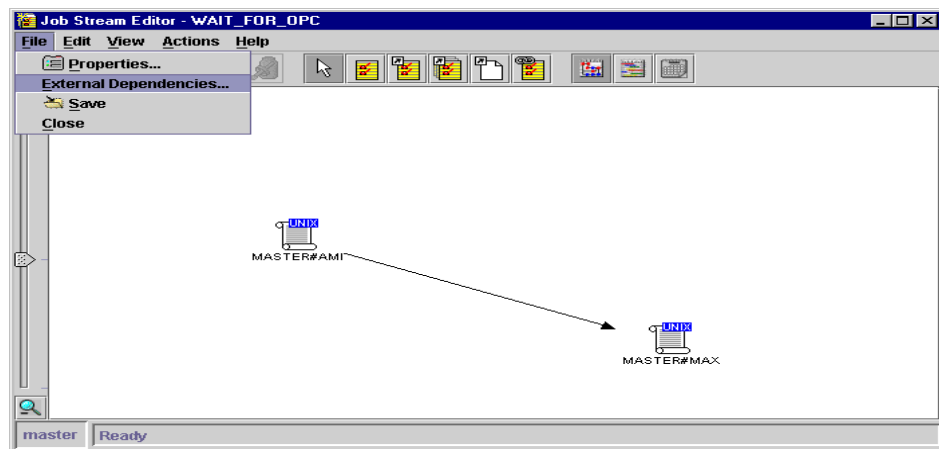


Figure 215. Internetwork dependency at the Schedule level

5. Select **Internetwork** and then **green cross** in the upper right corner as shown in Figure 216 on page 349.

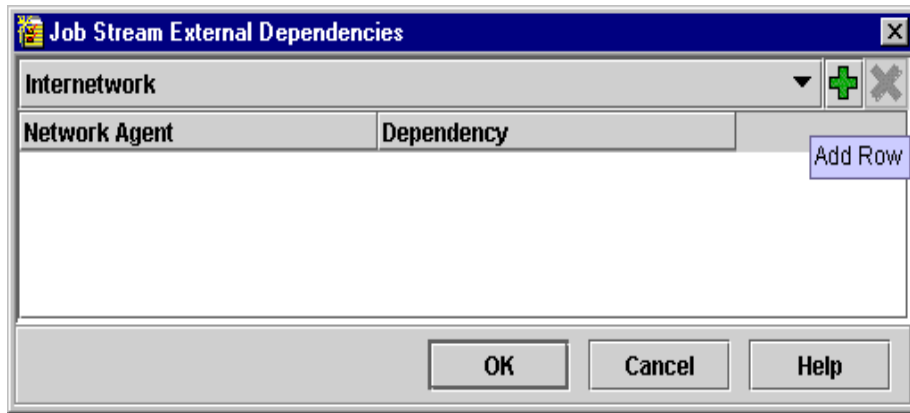


Figure 216. Add rows

6. Define the dependency as shown in Figure 217.

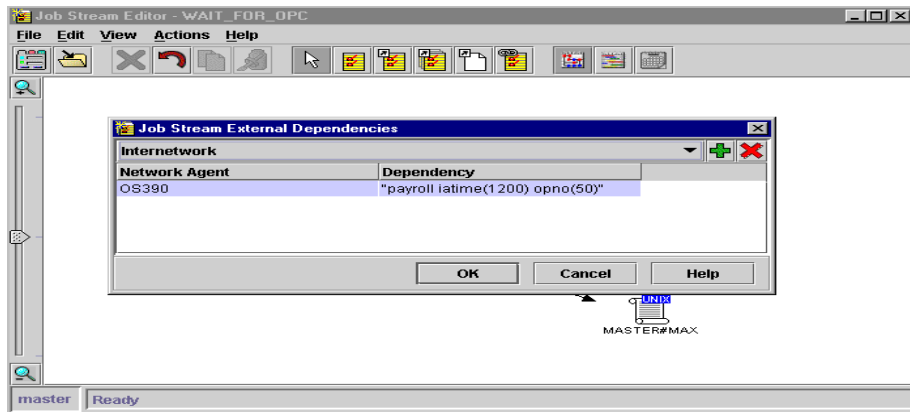


Figure 217. Complete the dependency



---

## **Appendix A. Tivoli OPC Version 2 Release 3 enhancements**

This appendix summarizes the Tivoli OPC Version 2 Release 3 enhancements.

---

### **A.1 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 administrative and operational tasks in a graphical manner and the ability to access multiple OPC Controllers from a single console.

The JSC can do the following:

- It can display lists of objects already defined to OPC from the database and from the current plan by using flexible filtering criteria.
- It can work with application descriptions including jobs and their dependencies, time restrictions (input arrival time, deadline, duration), and run cycles.
- It can work with special resource and workstation definitions.
- It can 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 back-end component supporting the JSC, is available for various UNIX platforms and Windows NT.

---

### **A.2 Catalog Management - Data Availability feature**

The new Catalog Management - Data Availability feature improves OPC performance for job restart and job log retrieval functions. Job runtime information, such as the sysout datasets, is maintained locally on the tracked system. The Controller retrieves this information only when it is needed for catalog management actions, thus, 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 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.

---

### **A.3 Improved management of critical jobs**

OS/390 Workload Manager, when used in goal mode, provides a new method of 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 resources 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 the latest start time.

---

### **A.4 Improved availability with 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, such as the Controller, the OS/390 Tracker, and the Server, can now be restarted automatically by the Automatic Restart Manager.

---

### **A.5 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 the following:

- Full support for special resource 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

---

## **A.6 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 of 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.

---

## **A.7 Usability enhancements**

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

- OPC can perform variable substitution within online procedures, thus, increasing the flexibility of the job setup feature. It is possible to customize OPC so that jobs are also submitted 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, such as 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.

---

## A.8 New and changed installation exits

User exit EQQUX001 has three new parameters:

- NEWREC - This is the number of JCL lines in the new JCLAREA.
- NEWJCL - This is the new JCLAREA.
- USDREC - This is the number of JCL lines used in the new JCLAREA.

The user exit, EQQUX007, has the new extended status (PEXSTAT) as part of its parameter set.

The Job Submission exit (installation exit 1) now allows changes to the size of JCL to be 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 sample set has two new samples: EQQCMX01 and EQQCMX05.

---

## A.9 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 also be resolved 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 upsubtask.



Parameters have been added to, or changed in, the JOBOPTS statement 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 whether the job-completion checker (JCC) requeues, to the reserved Data Store classes, only the sysouts belonging to those classes.

Parameters have been added to, or changed in, the OPCOPTS statement to be able 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 whether 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 (with 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, it defines the generic profile for a critical job. The profile contains the WLM service class and policy.



---

## Appendix B. TWS Version 7 Release 0 enhancements

This appendix is an introduction to some of the new features and changes to this version of the TWS. This chapter contains information about the following:

- Terminology changes
- The Job Scheduling Console, a new graphical user interface for TWS
- Time zones
- Auditing

---

### B.1 Terminology changes

The terminology used in the Job Scheduling Console differs from that used in the command line and earlier versions of TWS. Table 24 lists the old terms and their Job Scheduling Console equivalents.

Refer to the Glossary section of the *TWS 7.0 Plus Module Users Guide*, GC31-8401, for more definitions.

Table 24. Old terms and their JSC equivalents

Command line	Job Scheduling console	Definition
Schedule (1)	Job Stream	A unit of work consisting of a set of jobs and their dependencies.
Schedule (2)	Job Stream Instance	The occurrence of a job stream in the plan.
Job (1)	Job	An executable file, task or command, and its attributes. It is scheduled to run as part of a job stream
Job (2)	Job Instance	The occurrence of a job in the plan.
CPU	Workstation	A logical processor, typically, a computer, that runs jobs. Types of workstations include Domain Managers, Backup Domain Managers, Fault-Tolerant Agents, Standard Agents, and Extended Agents.

Command line	Job Scheduling console	Definition
Mozart Database Files	Database	A collection of scheduling objects including jobs, job streams, workstations, workstation classes, prompts, parameters, users, domains, calendars, and resources. These files were modified by gcomposer.
Symphony File	Plan	The scheduled activity for a period, typically, 24-hours. The plan is continuously updated to show the current status of all TWS activities. This file was modified by gconman.
AT Time	Start Time	The earliest time a job or job stream will begin.
UNTIL Time	Deadline Time	The latest time a job or job stream will begin execution.
ON and EXCEPT Dates	Run Cycles	The dates on which a job stream runs or is excluded from running.

---

## B.2 Job Scheduling Console

The Job Scheduling Console is the new graphical user interface for TWS. It provides the following features:

- The Job Scheduling Console is a multi-platform Java-based application and is integrated with the Tivoli Framework.
- The Job Scheduling Console can be used to manage both TWS and Operations Planning and Control (OPC) networks simultaneously and on the same interface. OPC is a Tivoli batch job scheduler for the OS-390 operating system.
- The Job Scheduling Console is available in multiple languages and supports worldwide time zones and localization.

## B.3 Overview of the new Job Scheduling Console

From the Job Scheduling Console, you are able to access scheduling functions for TWS and OPC from a common interface.

The left side of the console displays the job scheduling tree. The servers are in this view (TWS or OPC), and, under these servers, there are groups of default lists for the database and the plan.

There are two sets of default lists: Default Database lists and Default Plan lists.

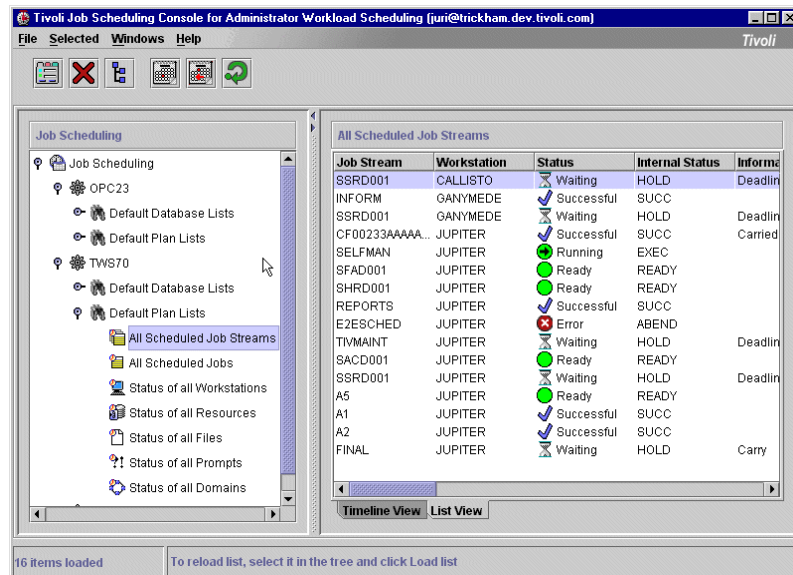


Figure 218. Default Database lists

From the left panel, you select a list icon and click the **Load List button** (a green arrow) to display the list. The right side of the window displays the list results. You can also select to detach the list into a separate window using the `Detach list` command available in the pop up menu of commands on a list icon.

When you first start TWS, there are a number of default lists for you to use. You can modify these lists or create your own groups and lists.

From the Job Scheduling Console, you can view both the configuration of objects in the database and the status of objects in the plan.

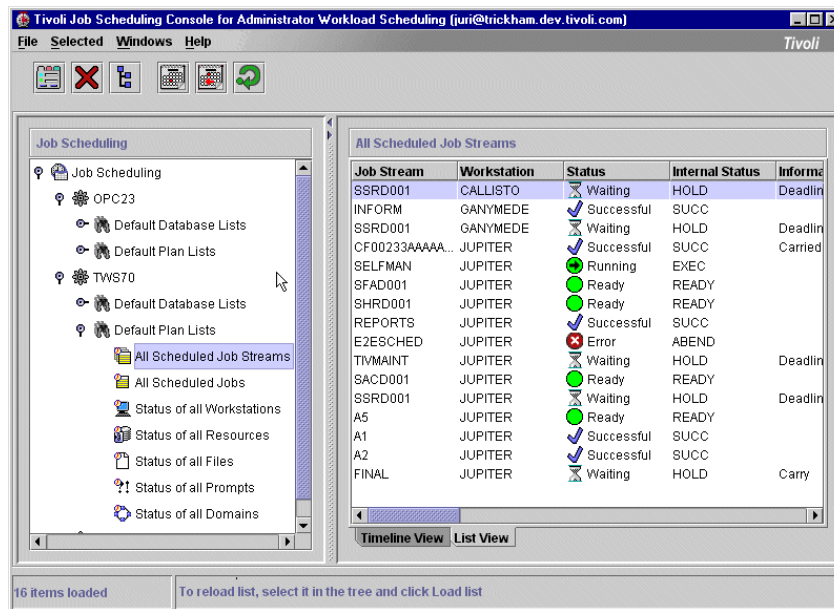


Figure 219. Default Plan lists

### B.3.1 Database and Plan

There are two types of lists: Database lists and Plan lists.

#### Database lists

A database list displays objects that have been defined in the TWS database. These can be jobs, job streams, workstations, workstation classes, parameters, prompts, resources, domains, and users. In legacy Maestro, these correspond to objects in the mozart database files that are modified using the *composer*.

#### Plan lists

A plan list displays objects that have been scheduled and are included in today's plan file. In legacy Maestro, these correspond to objects in the Symphony file that are modified using *conman*.

---

## B.4 Usage notes

The following are some tips for using the new Job Scheduling Console:

- There are three types of properties windows:
  - **Properties of a database object** - These screens edit an object in the database and affect all future uses of this object in the plan, starting with the next production day.
  - **Properties of a plan object** - These screens edit an object in the plan and affect only today's production plan. Changes to an object in the plan are not saved in the database and do not affect future runs of an object.
  - **Properties of a list** - These screens do not edit objects; they edit a list of the objects that are displayed. The properties chosen in a list window are similar to a filter on the `showschedules` or `showjobs` command in legacy Maestro.
- The order of columns in a display list can be modified. You can rearrange the columns to your personal preference by clicking and dragging the column heads. This new column order is only maintained for your current login.
- Open a list in a separate window by highlighting the list in the tree view and selecting the `Detach View` command from the pop-up menu. This enables you to open multiple lists simultaneously. For example, you can view both job instances and job stream instances. You can have up to seven detached windows at a time.
- Set the global refresh rate for your lists. This enables you to view plan lists and get updated information at a rate that you specify. You can set a long refresh rate to save processing time or a short refresh rate to always view updated information. Note that if you work with many detached views at the same time, you should avoid refreshing lists frequently in order to keep your system from slowing down.
- You can create and save customized lists for both the plan and the database. Define lists using wildcards and retrieve a specific subset of objects in the database or the plan. For example, to define a list that displays job streams, called SAP1 and SAP2, enter SAP in the list properties window. This enables you to quickly view only the database or plan objects that are important to you.
- On text entry fields, a blue dot next to a field means that this is a required field, or there has been an input error and the current input is not valid.

- Use the *Create Another* command to create a copy of any database object. This command creates a clone of the database object, opens the properties editor, and changes the name to Copyofobject. You can then modify the name and save the object.
- Note that the character restrictions listed in this manual are for double bit character languages. In some languages, the encoding of special characters will lead to more than 2 bytes and create a restriction on the maximum character length for any given field; so, normally, an eight character field will allow eight characters; however, if you use special characters or some national characters that take up more bytes, the character restrictions for that field may be as much as half of normal.
- The carat character (^) is reserved for any field that can have a parameter.

---

## B.5 Viewing TWS properties

You can view the properties of your installation of TWS by highlighting the TWS controller in the tree view and selecting the **Properties** command from the pop up menu. This displays the Properties - Scheduler window. From the Scheduler Information tab of this window, you can view such information as:

- Whether time zones are enabled
- The database mode (expanded or non-expanded)
- The workstation to which you are currently connected
- Batchman status
- The Connectors version level
- The auditing level for the database and the plan

---

## B.6 Job Scheduling Console Connectors

Connectors only become active after the Job Scheduling Console is started. There are Connectors for the TWS engine, database, and plan. After 30 minutes of inactivity, the Connectors will stop. They will automatically restart when commands in GUI are entered. For example, if you are doing work only on the database, you will not see the plan Connector process start up unless you do some plan queries or commands on the plan.

You can check if the Connectors are running for Windows NT by opening task manager and looking for the following processes:

- maestro\_engine



- maestro\_plan
- maestro\_database

You can check if the Connectors are running for UNIX by executing the following command at shell prompt:

```
ps -ef | grep "maestro_"
```

The process names listed above are displayed if they are active.

---

## B.7 Time zones

A time zone option has been implemented to support a worldwide TWS network. Time zones are enabled by entering the **timezone enable** option in the **globalopts** file.

- If time zones are enabled, the time zone fields in the Job Scheduling Console are enabled.
- If time zones are disabled, all the time zone fields are disabled throughout the Job Scheduling Console (except for workstations). In composer and conman, you get an error if you try to specify a time zone along with a time. You can still specify a time zone for workstations, but this is informational only and the time zone cannot be used for scheduling.

When enabling the time zone for the master workstation, it is recommended that you enter a time zone for any Fault-Tolerant Agents (FTAs) not in the same time zone as the master. A blank FTA time zone is assumed to be the same as the master time zone. If you leave a workstation definition time zone blank, it will default to the time zone of the master domain manager workstation.

When time zones have been enabled, you can enter a time zone along with a time in conman, composer, or the Job Scheduling Console. If you enter a time with no time zone, the time zone will be taken from the workstation on which the job or job stream executes or from the master workstation time zone if the executing workstation time zone is blank.

When time zones are enabled, you can view your plan in the timeline view from any of the supported time zones. Enabling time zones provides you with the enhanced ability to schedule jobs and job streams on a global level.

### B.7.1 Enabling the time zone feature

The time zone option is enabled by an entry in the globalopts file:

```
timezone enable = yes|no
```

Time zones are disabled by default at the installation of the product. If the `timezone enable` entry is missing from the `globalopts` file, time zones are disabled.

---

## B.8 Auditing

An auditing option has been implemented to track changes to the database and the plan:

- For the database, all user modifications are logged. However, the delta of the modifications, or the before image and after image, will not be logged. If an object is opened and saved, the action will be logged even if no modification has been done.
- For the plan, all user modifications to the plan are logged. Actions are logged whether they are successful or not.

The auditing logs are created in the following directories:

```
TWShome/audit/plan  
TWShome/audit/database
```

Audit files are logged to a flat text file on individual machines in the TWS network. This minimizes the risk of audit failure due to network issues and enables a straightforward approach to writing the log. The log formats are the same for both plans and database in a general sense. The logs consist of a header portion, which is the same for all records, an action ID, and a section of data that will vary according to the action type. All data is kept in clear text and formatted to be readable and editable from a text editor, such as `vi` or `notepad`.

### Note

For modify commands, two entries are made in the log for resources, calendars, parameters and prompts. The modify command is displayed in the log as the delete and add commands.

### B.8.1 Enabling the audit feature

The auditing option is enabled by two entries in the `globalopts` file:

```
plan audit level = 0|1  
database audit level = 0|1
```

A value of 1 enables auditing and a value of 0 disables auditing. Currently, TWS defaults to 0 or auditing disabled. If these options are not present in the globalopts file, auditing is disabled. Auditing is disabled by default at the installation of the product.

For the new settings in the global options file to take effect, you must stop the Job Scheduling Console and use the `wmaeutil` command to stop the Connectors before the changes take effect.

## B.8.2 Auditing log format

The audit log formats are basically the same for the plan and database. The log consists of a header portion, an action ID, and data sections that vary with the action type. The data is in clear text format, and each data item is separated by a vertical bar ( | ).

The log file entries will be in the following format:

```
Log Type|GMT Date|GMT Time|Local Date|Local Time|Object  
Type|Action Type|Workstation Name|User ID|Framework User|Object  
Name|Action Data fields
```

The log files contain the following information:

- **Log Type** - This field displays an eight character value indicating the source of the log record. The following log types are supported:
  - **HEADER** - The log file header
  - **CONMAN** - conman command text
  - **FILEAID** - Command that opens a file
  - **PLAN** - Plan action
  - **STAGEMAN** - stageman run
  - **RELEASE** - release command text
  - **DATABASE** - Database action
  - **PARMS** - Parameter command text
  - **MAKESEC** - makesec run
  - **DBEXPAND** - dbexpand run
- **GMT Date** - This field displays the GMT date the action was performed. The format is `yyyymmdd` where `yyyy` is the year, `mm` is the month, and `dd` is the day.

- **GMT Time** - This field displays the GMT time the action was performed. The format is hhmmss where hh is the hour, mm is the minutes, and ss is the seconds.
- **Local Date** - This field displays the local date the action was performed. The local date is defined by the time zone option of the workstation. The format is yyyyymmdd where yyyy is the year, mm is the month, and dd is the day.
- **Local Time** - This field displays the local time the action was performed. The local time is defined by the time zone option of the workstation. The format is hhmmss where hh is the hour, mm is the minutes, and ss is the seconds.
- **Object Type** - This field displays the type of the object that was affected by an action. The object type will be one of the following:
  - **DATABASE** - Database definition
  - **DBWKSTN** - Database workstation definition
  - **DBWKCLS** - Database workstation class definition
  - **DBDOMAIN** - Database domain definition
  - **DBUSER** - Database user definition
  - **DBJBSTRM** - Database job stream definition
  - **DBJOB** - Database job definition
  - **DBCAL** - Database calendar definition
  - **DBPROMPT** - Database prompt definition
  - **DBPARM** - Database parameter definition
  - **DBRES** - Database resource definition
  - **DBSEC** - Database security
  - **PLAN** - Plan
  - **PLWKSTN** - Plan workstation
  - **PLDOMAIN** - Plan domain
  - **PLJBSTRM** - Plan job stream
  - **PLJOB** - Plan job
  - **PLPROMPT** - Plan prompt
  - **PLRES** - Plan resource
  - **PLFILE** - Plan file

- **Action Type** - This field displays what action was taken against the object. The appropriate values for this field are dependent on the action being taken. For the database, the Action Type can be ADD, DELETE, MODIFY, EXPAND, or INSTALL. TWS will record ADD, DELETE, and MODIFY actions for workstation, workstation classes, domains, users, jobs, job streams, calendars, prompts, resources, and parameters in the database. The Action Type field also records the installation of a new security file. When makesec is run, TWS will record it as an INSTALL action for a Security definition object. When dbexpand is run, it will be recorded as an EXPAND action for the DATABASE object. LIST and DISPLAY actions for objects are not logged. For fileaid, TWS will only log the commands that result in the opening of a file. For parameters, the command line with arguments is logged.
- **Workstation Name** - This field displays the TWS workstation from which the user is performing the action.
- **User ID** - This field displays the logon user who performed the particular action. On Win32 platforms, it will be the fully-qualified domain name, domain\user.
- **Framework User** - This field displays the Tivoli Framework-recognized user ID. This is the login ID of the Job Scheduling Console user.
- **Object Name** - This field displays the fully-qualified name of the object. The format of this field will depend on the object type as shown here:
  - **DATABASE** - N/A
  - **DBWKSTN** - workstation
  - **DBWKCLS** - workstation\_class
  - **DBDOMAIN** - domain
  - **DBUSER** - [workstation#]user
  - **DBJBSTRM** - workstation#jobstream
  - **DBJOB** - workstation#job
  - **DBCAL** - calendar
  - **DBPROMPT** - prompt
  - **DBPARM** - workstation#parameter
  - **DBRES** - workstation#resource
  - **DBSEC** - N/A
  - **PLAN** - N/A
  - **PLWKSTN** - workstation

- **PLDOMAIN** - domain
- **PLJBSTRM** - workstation#jobstream\_instance
- **PLJOB** - workstation#jobstream\_instance.job
- **PLPROMPT** - [workstation#]prompt
- **PLRES** - workstation#resource
- **PLFILE** - workstation#path(qualifier)
- **Action Dependent Data** - This field displays the action-specific data fields. The format of this data is dependent on the Action Type field.

---

## B.9 Audit log header

Each log file will start with a header record that contains information about when the log was created and whether it is a plan or database log.

The following is the contents of the header file entry:

- **Log Type** - HEADER
- **GMT Date** - The GMT date that the log file was created.
- **GMT Time** - The GMT time that the log file was created.
- **Local Date** - The local date that the log file was created.
- **Local Time** - The local time that the log file was created.
- **Workstation Name** - The TWS workstation name for which this file was created. Each workstation in the TWS network creates its own log.
- **User ID** - The TWS user ID that created the log file.
- **Object Type** - This field reads DATABASE for a database log file and PLAN for a plan log file.
- **Object Name** - N/A.
- **Action Type** - N/A.
- **Action Dependent Data** - This field displays the version of the file.

---

## B.10 Sample audit log entries

The following are some sample log file entries:

```
HEADER |19991202|201200|19991202|131200|DATABASE| |GANGES
|RIVERS\pyasa ||1.0
DATABASE|19991202|224504|19991202|154504|DBWKSTN |ADD |GANGES
```

|RIVERS\pyasa | |JAMUNA|





---

## Appendix C. Job Scheduling Console terminology

This appendix maps terminology used by the Tivoli Job Scheduling Console (JSC) to its OPC and TWS equivalents. There are three tables: Table 25 will list the JSC terminology with TWS and OPC equivalents; Table 26 on page 375 lists OPC terms with their JSC and TWS equivalents, and Table 27 on page 378 lists TWS terms with their JSC and OPC equivalents.

---

### C.1 JSC to OPC and TWS terminology translation

If you know the JSC terminology, you can identify the OPC and TWS equivalents using Table 25.

*Table 25. JSC to OPC and TWS terminology*

Job Scheduling Console	OPC	TWS Command Line	Definition
Canceled	Status: Delete		The job or job stream has been deleted from the plan.
Database		Mozart Database Files	A collection of scheduling objects including jobs, job streams, workstations, workstation classes, prompts, parameters, users, domains, calendars, and resources. These files were modified by gcomposer.
Database	Databases		A definition of the data center, including application and job descriptions, periods, workstations, calendars, and resources.

<b>Job Scheduling Console</b>	<b>OPC</b>	<b>TWS Command Line</b>	<b>Definition</b>
Deadline Time	Time by which an application or operation should be completed.	UNTIL Time	Time Deadline Time The latest time a job or job stream will begin execution.
Engine	Controller		The OPC component that runs on the controlling system and contains the OPC tasks that manage the OPC plans and databases.
Exclusionary run cycle	Negative run cycle		Specifies when a job stream must not be run.
External Job	External Dependency		A job from one job stream that is a predecessor for a job in another job stream.
Job		Job (1)	An executable file, task, or command, and its attributes. It is scheduled to run as part of a job stream.
Job	Operation		A task performed at a workstation.
Job identifier	Operation number		A number used to uniquely identify the jobs in the job stream.
Job Instance	Operation in the current plan	Job (2)	The occurrence of a job in the plan.

<b>Job Scheduling Console</b>	<b>OPC</b>	<b>TWS Command Line</b>	<b>Definition</b>
Job Stream	Application Description	Schedule (1)	A unit of work consisting of a set of jobs and their dependencies.
Job Stream Instance	Occurrence	Schedule (2)	The occurrence of a job stream in the plan.
Job stream template	Application Group		A grouping of job streams that provides scheduling information, such as a calendar, a free-day rule, and run cycles that can be inherited by all the jobstreams that have been created using the template.
Logical resource	Special resource		A logical representation of a resource, such as tape drives, communication lines, databases, or printers, that is needed to run a job.
Offset-based run cycle	Run cycle with offsets		Includes a user-defined period and an offset, such as the 3rd day in a 90-day cycle.
Plan	Current Plan	Symphony File	The scheduled activity for a period, typically 24-hours. The plan is continuously updated to show the current status of all jobs.

Job Scheduling Console	OPC	TWS Command Line	Definition
Rule-based run cycle	Run cycle with rules	.	Includes a rule, such as the first Friday of March or the second workday of the week.
Run Cycles	Run-cycle	ON and EXCEPT Dates	The dates on which a job stream runs or is excluded from running.
Running	Status: Started		The job has started (jobs only).
Start Time		AT Time	The earliest time a job or job stream will begin.
Start Time	Input arrival time		The scheduled start time.
Successful	Status: Complete		The job or job stream has successfully completed.
Valid from	In-effect date		The first date that a run cycle applies.
Valid to	Out-of-effect date		The last date that a run cycle applies.
Workstation		CPU	A logical processor, typically a computer, that runs jobs. Types of workstations include Domain Managers, Backup Domain Managers, Fault-Tolerant Agents, Standard Agents, and Extended Agents.

Job Scheduling Console	OPC	TWS Command Line	Definition
Workstation	Workstation		A logical place where OPC controlled work runs. Typically, a computer that runs jobs, but it can also be a printer or a representation of a manual task or a WTO. Types of workstations are Computer, Printer and General.

## C.2 OPC to JSC and TWS terminology translation

Table 26. OPC to JSC and TWS terminology

OPC	Job Scheduling Console	TWS Command Line	Definition
Application Description	Job Stream	Schedule (1)	A unit of work consisting of a set of jobs and their dependencies.
Application Group	Job stream template		A grouping of job streams that provides scheduling information, such as a calendar, a free-day rule, and run cycles that can be inherited by all the jobstreams that have been created using the template.

OPC	Job Scheduling Console	TWS Command Line	Definition
Controller	Engine		The OPC component that runs on the controlling system, and that contains the OPC tasks that manage the OPC plans and databases.
Current Plan	Plan	Symphony File	The scheduled activity for a period, typically 24-hours. The plan is continuously updated to show the current status of all TWS activities. This file was modified by gconman.
Databases	Database		A definition of the data center, including application and job descriptions, periods, workstations, calendars, and resources.
External Dependency	External Job		A job from one job stream that is a predecessor for a job in another job stream.
In-effect date	Valid from		The first date that a run cycle applies.
Input arrival time	Start Time	AT Time	The scheduled start time.
Negative run cycle	Exclusionary run cycle		Specifies when a job stream must not be run.

<b>OPC</b>	<b>Job Scheduling Console</b>	<b>TWS Command Line</b>	<b>Definition</b>
Occurrence	Job Stream Instance	Schedule (2)	The occurrence of a job stream in the plan.
Operation	Job		A task performed at a workstation.
Operation in the current plan	Job Instance	Job (2)	The occurrence of a job in the plan.
Operation number	Job identifier		A number used to uniquely identify the job in the job stream.
Operation occurrence	Job instance		Operation in the plan.
Out-of-effect date	Valid to		The last date that a run cycle applies.
Run cycle with offsets	Offset-based run cycle		Includes a user-defined period and an offset, such as the 3rd day in a 90-day.
Run cycle with rules	Rule-based run cycle		Includes a rule, such as the first Friday of March or the second workday of the week.
Run-cycle	Run Cycles	ON and EXCEPT Dates	The dates on which a job stream runs or is excluded from running.
Special resources	Logical resources		A logical representation of a resource, such as tape drives, communication lines, databases, or printers, that is needed to run a job.

OPC	Job Scheduling Console	TWS Command Line	Definition
Status: Complete	Successful		The job or job stream has been completed.
Status: Delete	Canceled		The job or job stream has been deleted from the plan.
Status: Started	Running		The job has started (jobs only).
Time by which an application or operation should be completed	Deadline Time	UNTIL Time	Time Deadline Time. The latest time a job or job stream will begin execution.
Workstation	Workstation		A logical place where OPC controlled work runs. Typically, a computer that runs jobs, but it can be printer or a representation of a manual task or WTO. Types of workstations are Computer, Printer, and General.

### C.3 TWS to JSC and OPC translation

Table 27. TWS to JSC and OPC

TWS Command Line	Job Scheduling Console	OPC	Definition
AT Time	Start Time	Input arrival time	The earliest time a job or job stream will begin.



<b>TWS Command Line</b>	<b>Job Scheduling Console</b>	<b>OPC</b>	<b>Definition</b>
CPU	Workstation		A logical processor, typically a computer, that runs jobs. Types of workstations include Domain Managers, Backup Domain Managers, Fault-Tolerant Agents, Standard Agents, and Extended Agents.
Job (1)	Job		An executable file, task or command, and its attributes. It is scheduled to run as part of a job stream.
Job (2)	Job Instance	Operation in the current plan	The occurrence of a job in the plan.
Mozart Database Files	Database		A collection of scheduling objects including jobs, job streams, workstations, workstation classes, prompts, parameters, users, domains, calendars, and resources. These files were modified by gcomposer.
ON and EXCEPT Dates	Run Cycles	Run-cycle	The dates on which a job stream runs or is excluded from running.
Schedule (1)	Job Stream	Application Description	A unit of work consisting of a set of jobs and their dependencies.

<b>TWS Command Line</b>	<b>Job Scheduling Console</b>	<b>OPC</b>	<b>Definition</b>
Schedule (2)	Job Stream Instance	Occurrence	The occurrence of a job stream in the plan.
Symphony File	Plan	Current Plan	The scheduled activity for a period, typically 24-hours. The plan is continuously updated to show the current status of all jobs.
UNTIL Time	Deadline Time	Time by which an application or operation should be completed	Time Deadline Time. The latest time a job or job stream will begin execution.

---

## Appendix D. Migrating TWS smoothly from 5.X to 6.1

This document is intended to provide Tivoli Workload Scheduler administrators with a step-by-step guide for migration from Tivoli Maestro 5.x to TWS 6.1. These guidelines have been developed in the field during numerous upgrades of large TWS networks.

---

### D.1 Why do I need to upgrade TWS?

Tivoli will withdraw support for Tivoli Maestro V5.2 and V6.0 in the next year (2001). Defect support for these releases has already been scaled back. Beyond the 0130 patch set, problems in these releases will only be fixed in case of emergency. Going forward, Tivoli Workload Scheduler V6.1 and TWS 7.0 will be the only releases that will automatically be updated with patches.

A short term upgrade to TWS 6.1 will provide a stable fully-supported product release level. With the availability of TWS 7.0, an upgrade to TWS 7.0 can be planned without pressure caused by withdrawal of support or problems found in versions of TWS prior to 6.1.

Please note that this document does not include information for upgrading TWS networks running on the HP MPE operating system.

---

### D.2 Migration checklist

The following section contains the migration checklist for migrating to TWS 6.1.

#### D.2.1 Preparing for the upgrade

Perform the following steps to prepare for the upgrade:

1. When migrating a TWS production environment from Tivoli Maestro 5.x to TWS 6.1, the master should always be upgraded before any agents are upgraded. In some cases, a Tivoli Maestro 5.x UNIX master can produce a Symphony file that is not compatible with TWS 6.1 Windows NT FTAs. This is a relatively rare occurrence but should not be risked. In an all-UNIX TWS environment, the order of the upgrades is not significant.
2. Review the *Tivoli Maestro 6.1 Release Notes*, GI10-4759, and the *Maestro NT/UNIX Installation Guide V6.0*, SC31-5135, carefully and decide whether or not you wish to move to a TWS-expanded database format. See Chapter 10 of the *Maestro UNIX User's Guide V6.0*, GC31-5136, for a discussion of the `dbexpand` command. If the TWS installation is to be

converted to an expanded database format, this step should be performed after the migration is complete.

3. Consider whether TWS domain managers will be needed after the migration. In general, TWS networks of less than 100 nodes under one master do not justify the use of the domain manager architecture except where slow WAN connections are a problem. See Appendix A of the *Maestro UNIX User's Guide V6.0*, GC31-5136, for a discussion of TWS domains.
4. Bring up a TWS 6.1 test environment on your network though an upgrade of an existing 5.x test system or with a new install. Apply all general availability (GA) patches for TWS 6.1 before starting TWS in this environment. TWS patches can be found at the following URL:  
<ftp://ftp.tivoli.com/support/patches>.
5. This environment should be configured to include at least a master and an FTA and should match the operating system platforms used in the production environment as closely as possible. Identify and resolve any problems that arise in this TWS 6.1 test environment.
6. It is very important that all FTAs in any TWS network be assigned a server ID. Whenever a TWS agent is directly attached to the master mailman process, local failures on the agent or on the network segment can cause timeouts to occur in the master mailman process, effectively disabling master processing for several minutes at a time. The performance of the TWS Master in processing messages from the agents can also be severely lowered. This is a very undesirable situation, and every FTA or S-AGENT should be attached to a mailman process other than the master mailman (or " " server ID). Each of the server mailman processes on a master or domain manager can host 6 - 10 agents. Server IDs should not be assigned to X-AGENTS. See the *Maestro UNIX User's Guide V6.0*, GC31-5136, for more information about server IDs.
7. FTAs should not be defined with the Resolve Dependencies or Full Status flags turned on. Only the master, a backup master, or a remote TWS administration agent/monitoring point should have these features activated. These options cause a considerable increase in network traffic, disk usage, and I/O bandwidth usage on the local node.
8. Review and document any customizations that have been made to the TWS production environment. In particular, scripts that run TWS commands and parse output should be tested with TWS 6.1. The addition of the TWS domain and the expansion of object name lengths causes minor differences in the output of some TWS 6.1 commands when compared to their TWS 5.2 counterparts.

9. Review and document modifications that have been made to the TWS configuration files: `localopts`, `globalopts`, and `Netconf`. Save copies of these files as they will be replaced with defaults during upgrade.
10. Review the output of the Jnextday job from the production master. If there are warning messages about the workload, identify the cause and fix the offending object definitions.
11. Examine the production workload and look for occurrences of the " character in job definitions. A script called `quoter` can be obtained from Tivoli Customer Support Level 2 that will automate this process. Although it did not cause a problem in TWS 5.2, the use of this character in a TWS 6.1 job definition will cause a failure. For more information about this issue, contact Tivoli Customer Support Level 2.
12. Export the production workload to text files using the `composer create` commands, and copy these files to a safe place. For security reasons, passwords contained in TWS User definitions are not exported in this process. A list of active accounts, including passwords, should be compiled in case they are needed later.
13. Import these created files into the TWS 6.1 test environment using the `composer add` and `composer replace` commands. Correct any errors or warnings that occur during the import. Run the `schedulr` and `compiler` commands to verify that the workload is fully compatible with TWS 6.1. Correct any errors or warnings produced. You can find more information about these commands in the *Maestro UNIX User's Guide V6.0*, GC31-5136.
14. The upgrade of the production system will require that Jnextday be run after the TWS 6.1 code is installed; therefore, the upgrade must be performed at the production day rollover. On the day of the upgrade, cancel the Jnextday job.
15. Verify that a good backup of the TWS file system exists on the production master and that it is accessible to the TWS administrator performing the upgrade.
16. Ensure that the file system on which TWS is installed in the production environment has plenty of free space (150 to 200 MB).
17. Before starting the upgrade, set the CPU limit using the `conman limit CPU` command for the entire production TWS network to 0. Raise the fence on all CPUs to 101 (GO) using the `conman fence` command. You can find more information about these commands in Chapter 9 of the *Maestro UNIX User's Guide V6.0*, GC31-5136.

18. Stop all agents using the `conman stop @;noask` command, which is run on the master. Unlink all agents using the `conman unlink @;noask` command, which is run on the TWS Master.

## D.2.2 Performing the upgrade

One of the following procedures should be used according to how you wish to perform the upgrade:

### ***Updating an existing TWS 5.x Instance***

1. Copy and expand the TWS 6.1 install image into the TWS home directory on the production master. See the *Tivoli Maestro Installation Guide V6.0*, SC31-5135, for instructions.
2. Run the `/bin/sh customize -update` command. See the *Tivoli Maestro Installation Guide V6.0*, SC31-5135, for more options.
3. Apply all general availability (GA) patches for TWS 6.1 before starting TWS in this environment. TWS patches can be found at the following URL:  
<ftp://ftp.tivoli.com/support/patches>.

### ***Scratch install of TWS 6.1 to replace TWS 5.x production master***

1. Note that a scratch install of TWS will remove the run history stored in the mozart database. This run history is displayed by `conman` and several of the TWS reports.
2. Remove the following directories containing TWS, including all of their contents and subdirectories:  

```
~maestro  
~maestro/./unison
```
3. Copy and expand the TWS 6.1 install image, MAESTRO.TAR, into the TWS home directory on the production master. See the *Maestro NT/UNIX Installation Guide V6.0*, SC31-5135, for instructions.
4. Run the `/bin/sh customize -new` command. See the *Maestro NT/UNIX Installation Guide V6.0*, SC31-5135, to determine which options to specify for this command.
5. Apply all general availability (GA) patches for TWS 6.1 before starting TWS in this environment. TWS patches can be found at the following URL:  
<ftp://ftp.tivoli.com/support/patches>.
6. Import the previously saved text files into the new TWS 6.1 database using the `composer add` and `composer replace` commands. Remember that, for security reasons, the passwords in TWS User definitions is not exported into these files. These passwords will need to be updated manually.

### D.2.3 Post installation tasks

1. Start composer or gcomposer and examine various scheduling objects. Make sure that there are no errors accessing the database and that everything looks good.
2. Apply any customizations that were documented above to the new TWS 6.1 production environment, particularly changes to localopts, globalopts, and Netconf, which are replaced during the upgrade.
3. Run `./Jnextday` from the command line.
4. When all agents have fully linked, ask users to log in to master using the Remote Console to examine their jobs and schedules.
5. If users indicate that their workloads look good, raise the limits to their original values and lower the fences.
6. Ask users to review the execution of their workloads for the next few hours. Identify, document, and resolve any issues that arise.
7. Review the TWS logs found in:

```
~maestro/stdlist/YYYY.MM.DD/MAESTRO
```

and

```
~maestro/stdlist/YYYY.MM.DD/NETMAN
```

for errors or warnings and take corrective action if possible. Use the Tivoli Customer Support web site at <http://www.tivoli.com/support> to search for solutions. Contact Tivoli Customer Support via electronic support or telephone for any issues that require our assistance.

It is highly-recommended that the composer build commands and the `rmstdlist` command are run on a regular basis in any TWS environment. The builds should be run on the TWS Master during off-shift hours every two to four weeks. The `rmstdlist` command should be run on every FTA and S-AGENT, and on the master every 7 to 21 days. You can find more information about these commands in the *Maestro UNIX User's Guide V6.0*, GC31-5136.

### D.2.4 Migrating FTAs from 5.x to 6.1

It is recommended that FTAs not be upgraded using the `/bin/sh customize -update` command. When migrating 5.x FTAs to 6.1, a new install should always be performed using the `/bin/sh customize -new` command. The only circumstances where an upgrade might be indicated is when significant customizations have been made to the FTA, which must be preserved, or if local parameters have been defined on the FTA. A new installation does not

take any longer and is no more complex than an upgrade and will ensure that the new TWS 6.1 FTA does not inherit any problems from the 5.x FTA that existed previously. A detailed procedure for removing TWS from a Windows NT system is provided in the following section.

### D.2.5 Procedure for completely removing TWS from Windows NT

Perform the following steps to completely remove TWS from WindowsNT:

1. Log into the machine as Administrator or as the TWS user.
2. Start an MS-DOS window.
3. Issue the following TWS command:  

```
conman shutdown;wait
```
4. Use the NT Task Manager to verify that all TWS tasks and services have terminated.
5. Use the **Control Panel->Add/Remove Programs** dialog to uninstall TWS from the system.
6. Start a registry editor and remove all keys containing the string *unison* or the string *netman*.
7. Delete the TWS user's home directory and the Unison directory. By default these are `c:\win32app\maestro` and `c:\win32app\unison`.
8. Reboot the machine.



---

## Appendix E. Where to find more information

In this chapter we will give you some information sources for OPC and TWS such as redbooks, forums, user groups and FAQ sites.

---

### E.1 Available redbooks

You may refer to the following redbooks for more information on TWS or OPC topics:

- *An introduction to Tivoli Enterprise*, SG24-5494-00 has a chapter (Chapter 15) which discusses basic TWS concepts.
- *Managing SAP R/3 with Tivoli*, SG24-5298-00 is a good source for implementing Tivoli Workload Scheduler Extended Agent for SAP R/3.
- *Integrated Management Solutions Using Netview Version 5.1*, SG24-5285-00 discusses TWS integration with Netview in Chapter 11.
- *Managing PeopleSoft with Tivoli*, SG24-5137-00 covers implementing Tivoli Workload Scheduler Extended Agent for PeopleSoft in Chapter 12.
- *Using Decision Support Guides*, SG24-5506-00 Chapter 4 gives a scenario where Tivoli Decision Support Guide cube building processes are scheduled by TWS.
- *Maximizing Your OPC/ESA Throughput*, SG24-2130 is a great cookbook for performance optimization in OPC.
- *Batch Processing in a Parallel Sysplex Environment*, SG24-5329-00 is a good source for customers that are using OPC in a Parallel Sysplex environment.

You can download these redbooks from the following WEB site:

<http://www.redbooks.ibm.com/>

---

### E.2 Forums, user groups and FAQ sites

There are several forums and user groups that you may get information about TWS and OPC. Some of them require membership.

- *The ASAP user group* is focusing on TWS and OPC. You need to get a membership to access this forum. The ASAP user group can be accessed from the following WEB URL:

<http://www.asapuser.com/index.cfm>

- *The EGROUPS Maestro-L list forum* is a very valuable resource to get answers and other info about TWS. The EFGROUPS Maestro-L can be accessed from the following WEB URL:

<http://www.egroups.com/group/maestro.html>

- *The Nordic GSE OPC site* is a good OPC forum to get information, but it is a member only zone. The Nordic GSE OPC site up can be accessed from the following WEB URL:

<http://www.gse.nordic.org/>.

- There is customer forum on VM called *OPCESA CFORUM*. You can access this forum from *DialIBM*. How to access to DialIBM differs in each country, so you may ask your IBM representative if you have not have access already.
- *Q & A at the Tivoli website* has public FAQs. There are, in fact, two locations, one indexed under OPC and the other indexed under OPC/ESA.

You may access OPC website from the following WEB URL:

[http://www.tivoli.com/support/faqs/Tivoli\\_OPC.html](http://www.tivoli.com/support/faqs/Tivoli_OPC.html)

OPCESA website can be accessed from the following WEB URL:

[http://www.tivoli.com/support/faqs/OPC\\_ESA.html](http://www.tivoli.com/support/faqs/OPC_ESA.html)

---

### E.3 IBM Learning Center

Finally there are courses given for OPC and TWS in IBM Learning Center.

The WEB URL for IBM Learning Center is as following:

[www.ibm.com/services/learning/](http://www.ibm.com/services/learning/)

---

## Appendix F. Migration to TWS 7.0

In this appendix, we will mention some important points for you to consider while migrating to TWS 7.0 from previous versions. For a detailed discussion about migrating to TWS 7.0, refer to Appendix A of the *Tivoli Workload Scheduler 7.0 Planning and Installation Guide*, GC32-0422.

---

### F.1 Viewing data locally

If you are running gconman or gcomposer locally, you will need to consider whether to install connectors:

- For non-Framework users, this means installing a TMR on each FTA.
- Framework users will be able install connectors on a Managed node or TMR, not on an endpoint.

Master's databases must be present (NFS mount and so on) in order to view database information or submit job streams and predefined jobs from the FTA. At the very minimum, the CPU data files should be copied over in order to allow for ad hoc submits.

---

### F.2 Jnextday and writer changes

A new command, `wmaeutil`, has been added to the new Jnextday scripts in TWS 7.0. If you have created customized scripts to handle the new day process, you will need to update those scripts also:

- Source the Tivoli Environment if present on the workstation
- Run `wmaeutil` to stop the TWS Connectors after the run of `stageman`. Failure to do this will keep the connectors active on the archived symphony file. It will appear as if Jnextday has failed to run.

Writer has been modified to look and source a Tivoli environment and run `wmaeutil` to stop Connectors if they are present on FTAs. Problems can arise if FTA has Tivoli Framework active at one time whereby the `setup_env.sh` file is still present. *Libtas* error messages will show in the log file or on the NT screen. Remove `setup_env.sh` or `setup_env.cmd` files to correct this problem.

---

### F.3 MIPS and INTEL ABI platforms

The legacy GUI (gconman, gcomposer) will not be shipped for MIPS and INTEL ABI platforms. If you have MIPS or INTEL ABI platforms, you may use one of the following options:

- Make your Master a TIER 1 machine.
- Install the connectors on a backup Master Domain Manager that can support TWS Connectors.

---

### F.4 Behavioral changes

There have been some behavioral changes with the implementation of a new GUI model:

- Workstations are always required on submits. Legacy GUI allowed users to just type in the job name or command and press Enter. The same is true with streamlogon.
- What were treated as warnings in Legacy GUI and CLI are now considered errors in the JS Console.
- JS Console and TWS Connectors will now restrict the queries to only what the user is allowed to see via the TWS Security file with the DISPLAY access keyword. With legacy GUI lists, show jobs and show schedules showed all objects (of workstations, schedules, and jobs) regardless of the access keywords.

---

### F.5 Security migration

The easiest migration is to create TME Administrators with the same name as UNIX or NT users. Otherwise, existing security files will need to be modified to include TME Administrator names. Also, do not forget that if you make a modification in the security file, the connectors will need to be stopped with wmaeutil in order for the changes to be viewed in the JS Console.

## Appendix G. TWS log file record types

The TWS log file holds the starting, stopping, and error information for the following processes: Mailman, writer, batchman, and jobman. The TWS log files have the following record types:

Table 28. TWS log file record types

Record Code	Explanation
Ad	Add dependency. Issued by CONMAN
Aj	Add job to status file
As	Add schedule to symphony file
Bl	Job BLASTED off (streamed). Issued by BATCHMAN
Cf	Check for file. Issued by BATCHMAN. (OPENS keyword)
Cj	Change job. Issued by BATCHMAN
Co	Console command. Issued by CONMAN
Cs	Confirm job successful (DCM Pack, Maestro/UX D.02)
Da	Modify add dependency
DA	Modify add dependency
Dd	Modify delete dependency
DD	Modify delete dependency
Dj	Delete jcl. Issued by BATCHMAN for AIO tempjcls
Dr	Dependency, release
RD	Dependency, release
Es	End of schedule. Issued by CONMAN (SUBMIT
Ej	End of job. Issued by CONMAN (SUBMIT)
Fy	Find file (OPENS Keyword)
Fn	Did not find file (OPENS Keyword)
Go	Job priority set to GO
Gs	Get Status of remote Job ((Inter network Dependency)
Hi	Job priority set to HI
In	Initialization record

Record Code	Explanation
Jc	Job cancelled. Issued by BATCHMAN or CONMAN
Jf	Job Fence (Job priority to low)
Jr	Job completed successfully. Issued by SIGNALER
Jt	Job completed successfully. Issued by SIGNALER
Kj	Kill job, that is, abort. Issued by CONMAN (KILL command)
Lj	Launch job. Issued by BATCHMAN for JOBMAN
Lk	LINK_CPU
Lm	Limit command.
Mj	Modify job. Issued by CONMAN (ALTPRI)
My	CPU state changes (for SHOWCPU command)
Mr	Modify resource
Ms	Modify schedule. Issued by CONMAN (ALTPRI)
Nj	New Job
Qt	Terminate immediately. Issued by CONMAN
Rd	Resolve dependency. Issued by BATCHMAN for non-full status workstations
Re	Rerun job. Issued by BATCHMAN for an "EVERY" dependency
Rf	Rerun job FROM another. Issued by CONMAN
Rj	Request job to be re-run. Issued by CONMAN
Rp	Operator replied to prompt. Issued by CONMAN
Rr	Release resource
Rs	Reply to SPOOLMATE. (DCM/Pak only)
Sc	Schedule cancelled. Issued by CONMAN
Sr	Schedule released by operator. Issued by CONMAN
Ss	Schedule Done
St	Schedule Stop

Record Code	Explanation
Su	Job successful. Issued by SIGNALER when it has not found the record in the symphony file (a short job BATCHMAN has not started yet and sent an "AJ" record for). Not a BATCHMAN streamed job
To	Tellop message
Uk	Unlink from a CPU
Us	BATCHMAN was unable to stream a job. Issued by BATCHMAN
Ua	User Action





---

## Appendix H. Special notices

This publication is intended to help customers and Tivoli professionals who are implementing end to end scheduling solutions with OPC and TWS. The information in this publication is not intended as the specification of any programming interfaces that are provided by OPC and TWS. See the PUBLICATIONS section of the IBM Programming Announcement for OPC and TWS for more information about what publications are considered to be product documentation.

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. Any functionally equivalent program that does not infringe any of IBM's intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

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 the IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785.

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, Dept. 600A, Mail Drop 1329, Somers, NY 10589 USA.

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

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers

attempting to adapt these techniques to their own environments do so at their own risk.

Any pointers in this publication to external Web sites are provided for convenience only and do not in any manner serve as an endorsement of these Web sites.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

IBM	Parallel Sysplex
RACF	RS/6000
S/390	SP
SXM	System/390
Ultrastar	VTAM

The following terms are trademarks of other companies:

Tivoli, Manage. Anything. Anywhere., The Power To Manage., Anything. Anywhere., TME, NetView, Cross-Site, Tivoli Ready, Tivoli Certified, Planet Tivoli, and Tivoli Enterprise are trademarks or registered trademarks of Tivoli Systems Inc., an IBM company, in the United States, other countries, or both. In Denmark, Tivoli is a trademark licensed from Kjøbenhavns Sommer - Tivoli A/S.

C-bus is a trademark of Corollary, Inc. in the United States and/or other countries.

Sun, Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and/or other countries.

HP and HP Service Guard are trademarks Hewlett-Packard, Inc. in the United States and/or other countries.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States and/or other countries.

PC Direct is a trademark of Ziff Communications Company in the United States and/or other countries and is used by IBM Corporation under license.

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

UNIX is a registered trademark in the United States and other countries

licensed exclusively through The Open Group.

SET, SET Secure Electronic Transaction, and the SET Logo are trademarks owned by SET Secure Electronic Transaction LLC.

Lotus Notes is a registered trademark of Lotus Development Corporation.

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



---

## Appendix I. Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

---

### I.1 IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 403.

- *An introduction to Tivoli Enterprise*, SG24-5494
- *Batch Processing in a Parallel Sysplex Environment*, SG24-5329
- *Integrated Management Solutions Using Netview Version 5.1*, SG24-5285
- *Managing PeopleSoft with Tivoli*, SG24-5137
- *Managing SAP R/3 with Tivoli*, SG24-5298
- *Maximizing Your OPC/ESA Throughput*, SG24-2130
- *Using Decision Support Guides*, SG24-5506

---

### I.2 IBM Redbooks collections

Redbooks are also available on the following CD-ROMs. Click the CD-ROMs button at [ibm.com/redbooks](http://ibm.com/redbooks) for information about all the CD-ROMs offered, updates, and formats.

CD-ROM Title	Collection Kit Number
IBM System/390 Redbooks Collection	SK2T-2177
IBM Networking Redbooks Collection	SK2T-6022
IBM Transaction Processing and Data Management Redbooks Collection	SK2T-8038
IBM Lotus Redbooks Collection	SK2T-8039
Tivoli Redbooks Collection	SK2T-8044
IBM AS/400 Redbooks Collection	SK2T-2849
IBM Netfinity Hardware and Software Redbooks Collection	SK2T-8046
IBM RS/6000 Redbooks Collection	SK2T-8043
IBM Application Development Redbooks Collection	SK2T-8037
IBM Enterprise Storage and Systems Management Solutions	SK3T-3694

---

### I.3 Other resources

These publications are also relevant as further information sources:

- *Maestro NT/UNIX Installation Guide V6.0*, SC31-5135
- *Maestro UNIX User's Guide V6.0*, GC31-5136
- *OS/390 V2R9.0 MVS Initialization and Tuning Guide*, SC28-1751
- *TCP/IP V3R2 for MVS API Reference*, SC31-7187
- *Tivoli Maestro 6.1 Release Notes*, GI10-4759
- *Tivoli OPC V2R3 Installation Guide*, SH19-4379
- *Tivoli OPC V2R3 Job Scheduling Console Release Notes*, GI10-9233
- *Tivoli OPC V2R3 Job Scheduling Console User's Guide*, GC32-0402
- *Tivoli Operations Planning and Control*, SH19-4376
- *Tivoli Operations Planning and Control V2R3 Customization and Tuning Guide*, SH19-4380
- *Tivoli Operations Planning and Control V2R3 Diagnosis Guide and Reference*, LY19-6405
- *Tivoli Operations Planning and Control V2R3 Getting Started*, SH19-4481
- *Tivoli Operations Planning and Control V2R3 Tracker Agent*, SH19-4484
- *Tivoli Scheduling Agent for SAP R/3*, GC31-5147
- *Tivoli Workload Scheduler 7.0 Planning and Installation Guide*, GC32-0422
- *Tivoli Workload Scheduler 7.0 Reference Guide*, GC32-0424
- *TME 10 Framework 3.6 Planning and Installation Guide*, SC31-8432
- *TME 10 OPC Installation Guide*, SH19-4379
- *TME 10 OPC Messages and Codes*, SH19-4480
- *TME 10 OPC Quick Reference*, GH19-4374
- *TWS Extended Agent for MVS and OS390*, GC32-0642
- *TWS 7.0 Plus Module Users Guide*, GC31-8401

---

## I.4 Referenced Web sites

The following Web sites are also relevant as further information sources:

- <http://www.tivoli.com/>
- [www.ibm.com/java/jdk/download/index.html](http://www.ibm.com/java/jdk/download/index.html)
- <http://ftp.tivoli.com/support/patches>
- [ftp://vic.cc.purdue.edu/pub/tools/unix/lsof/binaries/](http://vic.cc.purdue.edu/pub/tools/unix/lsof/binaries/)

- <ftp://ftp.tivoli.com/support/patches>
- <http://www.asapuser.com/index.cfm>  
The ASAP user group is focusing on TWS and OPC. You need to get a membership to access this forum, and the ASAP user group can be accessed from this Web address.
- <http://www.egroups.com/group/maestro>  
The EGROUPS Maestro-L list forum is a very valuable resource to get answers and other information about TWS, and the EFGROUPS Maestro-L can be accessed from this Web address.
- <http://www.gse.nordic.org/>  
The Nordic GSE OPC site is a good OPC forum to get information, but it is a member-only zone. The Nordic GSE OPC Web site can be accessed from this Web address.
- [http://www.tivoli.com/support/faqs/Tivoli\\_OPC.html](http://www.tivoli.com/support/faqs/Tivoli_OPC.html)  
You can access the OPC Web site from this Web address.
- [http://www.tivoli.com/support/faqs/OPC\\_ESA.html](http://www.tivoli.com/support/faqs/OPC_ESA.html)  
The OPCESA Web site can be accessed from this Web address.
- <http://www.ibm.com/services/learning/>  
There are courses given for OPC and TWS in the IBM Learning Center, which you can visit at this Web address.
- <http://www.ibm.com/solutions/softwaremigration/tivmigteam.html>  
This is a WEB site that you can find information about Software Migration Project Office (SMPO) Service Offering.





---

## How to get IBM Redbooks

This section explains how both customers and IBM employees can find out about IBM Redbooks, redpieces, and CD-ROMs. A form for ordering books and CD-ROMs by fax or e-mail is also provided.

- **Redbooks Web Site** [ibm.com/redbooks](http://ibm.com/redbooks)

Search for, view, download, or order hardcopy/CD-ROM Redbooks from the Redbooks Web site. Also read redpieces and download additional materials (code samples or diskette/CD-ROM images) from this Redbooks site.

Redpieces are Redbooks in progress; not all Redbooks become redpieces and sometimes just a few chapters will be published this way. The intent is to get the information out much quicker than the formal publishing process allows.

- **E-mail Orders**

Send orders by e-mail including information from the IBM Redbooks fax order form to:

	<b>e-mail address</b>
In United States or Canada	<a href="mailto:pubscan@us.ibm.com">pubscan@us.ibm.com</a>
Outside North America	Contact information is in the "How to Order" section at this site: <a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a>

- **Telephone Orders**

United States (toll free)	1-800-879-2755
Canada (toll free)	1-800-IBM-4YOU
Outside North America	Country coordinator phone number is in the "How to Order" section at this site: <a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a>

- **Fax Orders**

United States (toll free)	1-800-445-9269
Canada	1-403-267-4455
Outside North America	Fax phone number is in the "How to Order" section at this site: <a href="http://www.elink.ibm.link.ibm.com/pbl/pbl">http://www.elink.ibm.link.ibm.com/pbl/pbl</a>

This information was current at the time of publication, but is continually subject to change. The latest information may be found at the Redbooks Web site.

### IBM Intranet for Employees

IBM employees may register for information on workshops, residencies, and Redbooks by accessing the IBM Intranet Web site at <http://w3.itso.ibm.com/> and clicking the ITSO Mailing List button. Look in the Materials repository for workshops, presentations, papers, and Web pages developed and written by the ITSO technical professionals; click the Additional Materials button. Employees may access MyNews at <http://w3.ibm.com/> for redbook, residency, and workshop announcements.

---

## IBM Redbooks fax order form

Please send me the following:

Title	Order Number	Quantity

---

First name	Last name
------------	-----------

---

Company
---------

---

Address
---------

---

City	Postal code	Country
------	-------------	---------

---

Telephone number	Telefax number	VAT number
------------------	----------------	------------

---

<input type="checkbox"/> Invoice to customer number	
---	--

---

<input type="checkbox"/> Credit card number	
---	--

---

Credit card expiration date	Card issued to	Signature
-----------------------------	----------------	-----------

**We accept American Express, Diners, Eurocard, Master Card, and Visa. Payment by credit card not available in all countries. Signature mandatory for credit card payment.**

---

## Abbreviations and acronyms

<b><i>API</i></b>	Application Program Interface
<b><i>BDC</i></b>	Batch Data Collector
<b><i>CORBA</i></b>	Common Object Request Broker Architecture
<b><i>DMTF</i></b>	Desktop Management Task Force
<b><i>GID</i></b>	Group Identification Definition
<b><i>IBM</i></b>	International Business Machines Corporation
<b><i>ITSO</i></b>	International Technical Support Organization
<b><i>JSC</i></b>	Job Scheduling Console
<b><i>MAS</i></b>	Multi Access Spool
<b><i>RFC</i></b>	Remote Function Call
<b><i>PSP</i></b>	preventive service planning
<b><i>PTF</i></b>	program temporary fix
<b><i>TMR</i></b>	Tivoli Management Region
<b><i>TWS</i></b>	Tivoli Workload Scheduler
<b><i>X-agent</i></b>	Extended Agent
<b><i>OMG</i></b>	Object Management Group
<b><i>OPC</i></b>	Operations, Planning and Control



---

## Index

### Symbols

.msg file size 329  
.msg files 325  
.toc file 285  
/etc/services 199

### Numerics

0130 patch set 381  
24/7 availability 1  
3480 tape cartridge 333

### A

ABAP/4 Modules 171  
abend 123  
abend code 123  
abend message 124  
Abended jobs 117  
Abendu 124  
abnormal termination 127  
access method 157, 158  
account name 333  
ad hoc jobs 326  
aged job output 324  
AIX/6000 4  
AIXconsole.sh 76  
altpass 153  
API 9  
APPC 11  
Application Builder 308  
Application Description dialog 281  
application program interface  
    *See API*  
AS/400 1  
ascending order 116  
ascii 232  
AT Time 358  
audit feature 364  
Audit Level 168  
Audit log header 368  
auditing option 364  
authorization profile 169  
Authorization Roles 27  
Autolink 340  
automatic job tailoring 277  
automatic recovery 277

Automatic Restart Manager 352  
auto-refresh cycle 328

### B

BAAN 161  
Baan 7  
Backup Master 329  
Basis Administrator 197  
Batch Data Collector  
    *See BDC*  
batch job execution 5  
Batch jobs 87  
BDC 163  
BDC sessions 163  
Best Practices  
    OPC 307  
    TWS 321  
Blocktime 342  
boot time 324  
build commands 324

### C

C runtime library 30  
CA7 160  
cache 38  
Catalog Management 12  
central point of control 273  
centralized database 6  
Checkinterval 342  
chown 298  
clock values 299  
Closing MS-DOS window 82  
CODEPAGE 31  
codepage 279, 281  
command translation 334  
common interface 7  
Common Object Request Broker Architecture  
    *See CORBA*  
compiled security template 70  
composer 41  
composer replace 239  
configuration profile 8  
conman resource 239  
console 134  
console dump 134  
control library 333

- controller port 282
- Copyofobject 362
- CORBA 8
- correction files 171
- corrupted data 150, 324
- Cpu 357
- cpu intensive 328
- create instance 67
- cross-system coupling 11

## D

- data center 2
- Data Router 128
- data set 15
- Data Space 334
- Data Store 11, 12
- Data Store options 355
- Database 358
- Database applications 324
- database errors 324
- database installation 47
- database object 362
- dbexpand 365
- Deadline Time 358
- DECUX 332
- Default Database lists 359
- Default Plan lists 359
- default time-out 280
- dependency object 5
- descending order 116
- Desktop Management Task Force  
    *See DMTF*
- destination name 280
- df command 325
- DGUX 332
- Diagnose Flags 142
- Digital OpenVMS 4
- Digital UNIX 277
- Disable BDC Wait 162
- disk activity 327
- disk I/O 328
- Disk Space 329
- Disk space 324
- disk usage 324
- DISP=MOD parameter 129
- Distributed Monitoring 204, 328
- DMTF 8
- DNS name 278

- Domain Manager 7
- downgrade 147
- dummy job 254
- dump dataset 124
- dumpsec 71

## E

- earliest time 358
- EBCDIC-ASCII data conversion 334
- e-commerce 1
- Education 320
- Endpoint 8
- endpoint communications 9
- Endpoint manager 9
- end-to-end scheduling 1
- EPILOG 129
- Event Triggered Tracking 212
- evtsize 325
- EX Command 318
- EXCEPT Date 358
- EXIT11 12
- EXIT7 13
- export 284
- Extended Agent 157
- extended agent 161
- Extended Agent Workstations 250
- external program 162

## F

- False timeout 326
- FAQ sites 387
- fast I/O 38
- fault-tolerant 9
- Fault-tolerant agent  
    *See FTA*
- file descriptor 324
- File locks 324
- File Manager 333
- file permissions 297
- filesystem 324
- filesystem size 39
- filter 328
- foreign platforms 157
- Forums 387
- Framework classes 68
- Framework User 367
- free days 4
- Free Inodes 329

FTA 37, 158, 278  
FTP command 230  
FTP server 230

## G

Gateway 8  
gateway methods 8  
gateway program 334  
gconman 358  
General workstation 317  
generic object database 8  
Generic subtask 297  
GID 24, 241  
global refresh rate 361  
globalopts 363  
gracefull shutdown 334  
Greenwich Mean Time 299

## H

Hot standby controller 11  
HP 1  
HP MPE 161  
HP ServiceGuard 329  
HP-UX 4

## I

I/O bottlenecks 328  
IBM 1  
IBM HACMP 329  
IBM Learning Center 388  
IBM Ultrastar 328  
idle time 5  
IEFU84 exit 334  
IFuser 197  
Incorrect output 124  
INCORROUT 125  
initialization statements 354  
Inodes 325  
Installation  
    Extended Agent for OS390 332  
    Job Scheduling Services 64  
    JSC (TWS) 72  
    JSC(OPC) 19  
    OPC connector 26  
    OPC Tracker Agent 282  
    OPC V2R3 11  
    TCP/IP server 30

Tivoli Framework 42  
Tivoli Patch 52  
TWS 37  
X-agent for SAP R/3 165  
installation exits 354  
InstallShield 329  
InstallShield options 330  
instance settings 28  
INTEL 332  
Interlink 3.1 331  
internetwork dependencies 343  
Interpreted script file 300  
inter-process communication 325  
IP address 278, 280, 281  
ISPF application 16  
ISPF command table 16  
ISPF panels 4  
ISPF user 351  
ISPF/PDF 16  
IT console 329

## J

Java 112  
Java Development Kit 19  
Java version 1.1.8 72  
JCC  
    *See job-completion checker*  
JCL procedures 15  
JCL variable substitution 354  
JES 12  
JES control blocks 13  
JES exits 13  
JES2 7, 12  
JES2 hotstart 14  
JES3 7  
Jnextday 41  
Job 3  
Job FOLLOW 153  
job ID 160  
Job identifier 88  
Job instance 89  
    Deadline time 115  
    File dependencies 115  
    Priority 115  
    Prompt dependencies 115  
    Repeat range 115  
    Resource dependencies 115  
job number 160

- Job scheduling 1
- Job Scheduling Console
  - See JSC*
- Job Scheduling Editor 99
- Job Scheduling Services 37, 38, 42
- Job setup 3
- Job stream
  - Carry forward 115
  - Deadline time 115
  - File dependencies 115
  - Limit 115
  - Priority 115
  - Prompt dependencies 115
  - Resource dependencies 115
  - Starting time 115
  - Time zone 115
- job-completion checker 277
- joblogretrieval exit 12
- jobman.exe 158
- jobmanrc 157
- jobmon.exe 158
- jobs waiting 220
- Jobstream 3
- JSC 1, 5, 79, 351
  - client 85
  - dataflow 18
  - Error List 89
  - initial window 83
  - Job status mapping 116
  - performance 112
  - pre-customized profile 85
  - preferences 85
  - Ready Lists 89
  - Starting 84
- JSC Installation
  - On AIX 19
  - On Sun Solaris 20
  - On Windows 19
- JTOPTS 281

## K

- keyword 123
- keyword string 123

## L

- lack of inodes 325
- latest time 358
- legacy GUI 110

- legacy systems 1
- library 162
- License Key 46
- listproc 150
- LJuser 197
- LMHOSTS 43
- load library 333
- LoadLeveler 297
- local port 282
- localization 358
- log format 365
- logfile adapter 329
- logical model 2
- logical processor 357
- Logical resources 89
- Long Interval 168
- long-term plan 3
- Loop 124
- LOOP procedure 130
- Isol binaries 324

## M

- machine hang out 322
- maestro\_database 363
- maestro\_engine 362
- maestro\_plan 363
- MaestroDatabase 68
- MaestroEngine 68
- MaestroPlan 68
- mailman 150
- Managed Node 37, 69
- mapping file 21
- MAS 12
- Mass Update utility 353
- Master 321
- master copy 196
- Master Domain Manager 7
- maximum return code 281
- MAXWAIT 336
- member name 284
- message file size 325
- method.opts 158
- Microsoft 325
- migration 381
- Migration checklist 381
- Migration from 5.X to 6.1
  - checklist 381
  - Post installation 385



- Removing TWS 386
- Scratch install 384
- Update an instance 384
- Migration from 5.X to 6.1 381
- migration group 307
- Migration to TWS 7.0 389, 391
  - Behavioral changes 390
  - Jnextday changes 389
  - Security 390
  - viewing data locally 389
  - Writer changes 389
- MIPS 332
- modify access rights 241
- modify commands 364
- Modify security file 71
- mozart database 324
- Mozart Database Files 358
- Multi Access Spool
  - See MAS
- multiple calendars 4
- multiple languages 358
- multiple servers 321
- MVS gateway 334
- MVS gateway software 332
- MVS JES2 7
- MVS JES3 7
- MVSCA7 339
- MVSJES 339
- MVSJES method 230
- MVSOPC 339

## N

- name server 43
- netstat 151
- NetView communication 3
- Network Agent 7
- Network connectivity 325
- network hang out 322
- Network status 329
- network traffic 327
- New terminology 371
- NFS connections 241
- NFS filesystems 243
- NFS restrictions 298
- NIS 43, 282
- NOLIMIT parameter. 334
- non-IDRC format 333
- non-MVS platforms 277

- non-MVS workload 277
- non-reporting workstation 317
- non-scheduled application 212
- non-working days 314
- NT 277
- NTFS partition 38

## O

- Object Management Group
  - See OMG
- offset 89
- offset based run cycle 105
- OMG/CORBA 8
- OMVS segment 31
- ON Date 358
- one way link 275
- OPC xvii, 1, 5, 7, 9, 79
  - address spaces 15
  - architecture 4
  - Business processing cycles 4
  - calendar 4
  - Calendar maintenance 314
  - commands 17
  - controller 4
  - Current plan length 314
  - database 4
  - Deadline time 313
  - Dependencies 3
  - dialogs 79
  - distribution tape 12
  - Input arrival time 312
  - installation 17
  - long term plan 9
  - Naming standards 309
  - Plans 3
  - Special resources 3
  - subsystems 14
  - tracker 4
  - trends 10
  - Workstations 3
- OPC Connector 21, 26, 42
  - Creating instance 26
  - installing 26
- OPC ID mapping 33
- OPC PLEX 129
- OPC Tracker Agent 277
  - AIX 277
- OPC Tracker Agent for AIX/6000

- creating a user group 283
- creating user group 283
- customizing 283
- group ID 282
- Hardware requirements 277
- installation 282
- Reinstalling 285
- Software requirements 277
- user ID 282
- OPCMMSGCLASS 336
- OPCSUBSYSTEM 336
- open API 161
- Open Connect 2.3 331
- Open files 323
- Open Group 8
- OPENS dependency 159
- OPENS file 153
- OpenVMS 353
- Operation number 88
- Operation sequence number 311
- operator console 124
- Oracle 9
- Oracle Applications 7, 161
- OS/2 4, 277
- OS/390 1, 4, 9
- OS/390 calls 30
- OS/390 Open Edition 284
- OS/390 UNIX System Services 353
- OS/390 USS 4, 7
- OS/400 4
- oserv 43
- Out-of-effect date 89

## P

- PEERADDRESS 336
- PeopleSoft 7, 161
- PERFM procedure 132
- physical disk 324
- PIF
  - See Program Interface*
- ping 278
- Plan 358
- Policy Region. 56
- PORT 336
- port definition 160
- port number 297
- POSIX shell 353
- postprocessing 3

- preprocessing 3
- preventive service planning
  - See PSP*
- Print operations 3
- Problem analysis 123, 127
- problem-type keywords 124
- process id 150
- Process Status 329
- Processes 323
- production run number 160
- profile distribution 8
- ProfileManager 58
- Program Directory 12
- Program Interface 4, 352
- program temporary fix
  - See PTF*
- PROLOG 129
- Prompts waiting reply 117
- PSP 12
- PTF 12
- put command 232
- PUTLINE 336, 337

## R

- R/3
  - Client Number 167
  - Instance 167
  - System Name 167
  - Userid 167
- r3batch 161
- R3batch 4.0 162
- r3batch.opts 168
- r3debug file 197
- r3gateway 199
- r3login name 196
- R3OPT.EXE 168
- r3options 161, 167, 168
- RACF 32
- RACF user ID 32
- RAID array 328
- record mode 330
- recovery jobs 161
- redbooks 387
- reliability 324
- Remote 158
- remote clients 285
- remote console 6, 328
- remote file system 45

- Remote Function Call
  - See RFC
- remote writer 326
- removing files 324
- Reply prompts 115
- Reserve ports 31
- resource availability 210
- Resource modelling 311
- Resource synchronization 209
- Retrycount 342
- RFC 158, 162
- RFC user ID 169
- RFC-User 168
- RISC Systems/6000 277
- rmstdlist 324
- RODM 312
- roles 28
- roll over 5
- root authority 282, 298
- root userid 158
- rule based run cycle 252
- Rule-based run cycle 89, 104
- Run Cycle 358
- run cycle 5, 312
- Running jobs 117

## S

- SAF 33
- same paradigms 10
- samples directories 304
- SAP certification 162
- SAP R/3 4, 7
  - Application version 195
  - Batch 161
  - configuration 169
  - database 171
  - environment 163
  - Hot Package 8 165
  - Job Class 198
  - Job states 162
  - Kernel version 195
  - RFC 162
  - Versions 165
  - X-agent 162
  - X-agent installation 165
- SAP R/3 Extended Agent 161
  - benefits 161
  - define job 177

- Define job stream 189
- define workstation 174
- SAP R/3 V 4.6B 203
- SAP.TAR 165
- SAPGUI 163
- scalability 9
- scheduling scenarios 203
- script writing 330
- SCSI 38
- Security file 70
- Security Server 34
- security template 70
- Server A/S 31
- Server Parameters file 31
- serverID 322
- Service Functions 87
- Service Pack 5 205
- Set Symphony 116
- Setup program 38
- SFinal Job Stream 41
- SGL IRIX 4
- Shared DASD 11
- shell script 157
- Short Interval 168
- silent mode 330
- Silicon Graphics IRIX 277
- Sinfonia 154
- SMF exits 12
- SMIT 284, 285
- SMP/E usermods 13
- snap data 129
- Software Distribution 204
- Software Migration Project Office 307
- Special Resource Monitor 312
- Special resources 86, 89
- Spool 12
- SRSTAT command 210, 212
- SSA 38
- stageman 365
- Started tasks 3
- Starting JSC
  - In AIX 77
  - In Windows 77
  - On Windows 77
- Starting time 115
- submit commands 326
- submittor instance 297
- SUBSYS 336
- subsystem 14

- Sun 1
- Sun Solaris 4
- SunOS 4
- SVCDUMP 337
- Swap Space 329
- Symphony File 358
- SYS1.PARMLIB 14
- SYSMDUMP 129
- SYSPLEX 205
- System abends 128
- system administrator 323
- System dump 129
- System Load Avg 329

## T

- tar files 332
- TCP Reader 298
- TCP/IP 3.3 31
- TCP/IP network 38
- TCP/IP Parms 278
- TCP/IP Server 17, 142
  - Diagnose Flags 142
  - installation 30
  - Trace 142
- TCP/IP stack 30
- TCPIPID 280
- TCPIPPOINT 280
- TCPIPSTACK 337
- TCPNAME 337
- TCPTIMEOUT 280
- TEC-Console 204
- TEC-Server 204
- Terminology changes
  - JSC 371
  - TWS 357
- Terminology translation 371
- third-party vendors 8
- three-tiered management 9
- threshold 254
- Time zones 363
- Timeout 326
- timeout expire 326
- timezone enable 363
- Tivoli Administrator 22
  - Creating 21
  - Setting Logins 22
- Tivoli API 8
- Tivoli Customer Support 325

- Tivoli Desktop 37
- Tivoli Enterprise 8
- Tivoli Framework 37, 112, 358
- Tivoli Maestro 5.x 381
- Tivoli Management Agent 8
- Tivoli Management Framework 8
- Tivoli Management Server
  - See TMR Server*
- Tivoli Object Dispatcher 48
- Tivoli Plus Module for TWS 204
- Tivoli Scheduling Agent for SAP R/3 163
- Tivoli Software Distribution 329
- Tivoli support database 124
- Tivoli Workload Scheduler
  - See TWS*
- TMR Server 9, 37
- Toolkits 9
- touch 301
- Trace
  - JSC 145
  - OPC Connector 143
  - TCP/IP Server 142
- TRACEDATA. 146
- TRACELEVEL 146
- tracker user ID 288
- transaction support 8
- transformation 10
- transport files 171
- trends and directions 10
- Troubleshooting
  - Job Scheduling Console 145
  - OPC 123
  - OPC connector 143
  - TCP/IP Server 142
- TSDSPACE 334
- TSO command 4, 16
- TTY consoles 303
- TWS 5, 6, 7, 9, 79
  - architecture 5, 7, 321
  - Backup Domain Manager 6
  - Calendars 5
  - Composer 10
  - Conductor 10
  - Connector 37
  - Connector instance 68
  - database 5
  - database files 7
  - Deployment 329
  - deployment 323

- Domain Manager 6, 39
- Engine 37
- Extended Agent 7
- Fault-tolerant Agent 6
- FTA 322
- Hardware 328
- High availability 329
- home directory 325
- installing 39
- Job Streams 5
- JSC 110
- JSC Client 6
- Master 39, 55, 158, 321
- Master Domain Manager 6
- network 6, 322
- overview 5
- plan 5
- Resources 237
- RFC user 170
- security 70, 116, 330
- Standard Agent 7, 322
- Terminology changes 357
- trends 10
- user account 38
- valid versions 322
- Workstations 6
- TWS 5.2 165
- TWS 6.1 381
- TWS Connector 42
  - installation 67
  - multiple instances 69
  - start up 70
  - stopping 70
  - verify installation 68
- TWS Extended Agent for MVS and OS/390 331
- TWS OS/390 Extended Agent
  - OS/390 installation 333
  - Unix installation 332
  - Windows NT installation 333
- TWS system requirements
  - on all operating systems 38
  - on Unix 38
  - on Windows NT 38
- TWS versus SAP Job states 162
- TWSHOME 332
- TWS-MVS methods 332

## U

- UID 22, 241
- Ultra2 328
- unattended install 329, 330
- uncompressed format 333
- Unison 386
- Unix command shell 116
- UNIX Local 160
- UNIX Remote Shell 160
- unlink 327
- unnecessary data 328
- UNTIL Time 358
- US keyboard mapping 48
- User abends 127
- user groups 387
- Usermap 31, 33
- utilities 304

## V

- VTAM 124
- VTAM links 11

## W

- Windows 2000 72
- Windows 98 351
- Windows NT 4, 351
- Windows NT 4.0 Service Pack 4 37
- Windows NT 4.0 Service Pack 5 37
- Windows NT Clusters 329
- Windows NT start menu 81
- wmaeutil 365
- work days 4
- Workload 1
- workload 321
- Workload Manager 352
- Workstation 357
  - Change fence 115
  - Change limit 115
  - Link/unlink 115
  - Start/stop 115
- wtwsconn 69
- wtwsconn.sh utility 67

## X

- X/Open 8
- X-agent
  - See Extended Agent

XCF Communication links 11  
XCF group 11  
XCF local mode 11  
XMI Function Modules 197

## IBM Redbooks review

Your feedback is valued by the Redbook authors. In particular we are interested in situations where a Redbook "made the difference" in a task or problem you encountered. Using one of the following methods, **please review the Redbook, addressing value, subject matter, structure, depth and quality as appropriate.**

- Use the online **Contact us** review redbook form found at [ibm.com/redbooks](http://ibm.com/redbooks)
- Fax this form to: USA International Access Code + 1 914 432 8264
- Send your comments in an Internet note to [redbook@us.ibm.com](mailto:redbook@us.ibm.com)

<b>Document Number</b>	SG24-6013-00
<b>Redbook Title</b>	End-to-End Scheduling with OPC and TWS
<b>Review</b>	<div></div> <div></div> <div></div> <div></div> <div></div> <div></div>
<b>What other subjects would you like to see IBM Redbooks address?</b>	<div></div> <div></div> <div></div>
<b>Please rate your overall satisfaction:</b>	<input type="radio"/> Very Good <input type="radio"/> Good <input type="radio"/> Average <input type="radio"/> Poor
<b>Please identify yourself as belonging to one of the following groups:</b>	<input type="radio"/> Customer <input type="radio"/> Business Partner <input type="radio"/> Solution Developer <input type="radio"/> IBM, Lotus or Tivoli Employee <input type="radio"/> None of the above
<b>Your email address:</b> The data you provide here may be used to provide you with information from IBM or our business partners about our products, services or activities.	<input type="checkbox"/> Please do not use the information collected here for future marketing or promotional contacts or other communications beyond the scope of this transaction.
<b>Questions about IBM's privacy policy?</b>	The following link explains how we protect your personal information. <a href="http://ibm.com/privacy/yourprivacy/">ibm.com/privacy/yourprivacy/</a>







## End-to-End Scheduling with OPC and TWS Mainframe and Distributed Environments







**Redbooks**

# End-to-End Scheduling with OPC and TWS

## Mainframe and Distributed Environments

**Use the Job  
Scheduling Console  
to integrate OPC and  
TWS**

**Model your  
environment using  
realistic scheduling  
scenarios**

**Implement SAP R/3  
workload scheduling**

Tivoli OPC and Tivoli Workload Scheduler (formerly Maestro) are the two scheduling engines that make up Tivoli's integrated solution for enterprise scheduling. Tivoli OPC has a firm share of the mainframe scheduling market, and TWS is the scheduler of choice for distributed environments. Tivoli has now made available the Job Scheduling Console, addressing the end-to-end scheduling requirements, that are central to multi-platform environments. Tivoli OPC 2.3 and Tivoli Workload Scheduler 7.0 can now be managed from this common graphical user interface, creating a best-of-breed cross-platform scheduling solution.

This IBM Redbook provides step-by-step setup instructions, as well as detailed troubleshooting guidelines for implementing OPC, TWS, and the Job Scheduling Console. It covers the most important features of both engines and provides illustrative practical usage scenarios, including exploitation of the new common GUI, installation and usage of Extended Agents, and advice on implementing high-availability environments. Users migrating from previous releases will find the migration checklists and the terminology translation tables extremely valuable.

This book is based on OPC 2.3 and TWS 7.0 and will be useful to those customers who are working to integrate OPC and TWS, as well as those working with OPC or TWS alone.

**INTERNATIONAL  
TECHNICAL  
SUPPORT  
ORGANIZATION**

**BUILDING TECHNICAL  
INFORMATION BASED ON  
PRACTICAL EXPERIENCE**

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)

SG24-6013-00

ISBN 0738418609