

Workload Scheduler for z/OS
Version 8.6

Planning and Installation



Workload Scheduler for z/OS
Version 8.6

Planning and Installation



Note

Before using this information and the product it supports, read the information in "Notices" on page 351.

This edition applies to version 8, release 6 of Tivoli Workload Scheduler for z/OS (program number 5698-A17) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC32-1264-05.

© **Copyright IBM Corporation 1991, 2011.**

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

Contents

Figures	ix
--------------------------	-----------

Tables	xi
-------------------------	-----------

About this publication	xiii
---	-------------

What is new in this release	xiii
What is new in this publication	xiii
Who should read this publication.	xiv
Publications	xiv
Using LookAt to look up message explanations	xiv
Accessibility	xv
Tivoli technical training	xv
Support information	xv
Conventions used in this publication	xvi

Part 1. Planning	1
-----------------------------------	----------

Chapter 1. Overview	3
--------------------------------------	----------

Hardware and software requirements	3
Hardware requirements.	3
Software requirements and optional software	4
Controlling system	4
Controlled z/OS systems	4
Optional software.	4
Related software	5
Software compatibility	5
Parts and their relationships	5
Tracker	6
Controller	6
Server	7
Graphical user interfaces	7
Data Store	8
Configurations.	9
Controlling system	9
Controlled systems	9
Integration with Tivoli Workload Scheduler	9
Subtasks	10
Relationship between the Scheduler and z/OS.	12
Using the Tivoli Workload Scheduler for z/OS	
Program Directory	13
Sample library	13
The installation process	13

Chapter 2. Planning your configuration 15
--

Planning considerations	15
Trackers	15
Initialization statements	15
Communication	16
How to connect Tivoli Workload Scheduler for z/OS	
systems.	16
Shared DASD	16
z/OS cross-system coupling facility	17
VTAM (network communication function)	17
TCP/IP.	17

Workstation destination	17
Workload restart.	18
JES considerations	18
Basic server configuration example	19
Basic configuration examples	20
DASD connected	20
VTAM connected	22
TCP/IP connected	24
XCF connected	25
Tracker and controller in a single address space	27
Basic data store configuration examples	28
SNA only connection	28
XCF only connection	30
TCP/IP only connection	33
Mixed SNA and XCF connection	34

Chapter 3. Planning your installation 39

Installation considerations	39
Configuring for availability	39
Hot standby	39
Starting an event writer with an event reader	
function	40
Using a Hierarchical File System cluster	40
Checklist for installing Tivoli Workload Scheduler	
for z/OS	40

Part 2. Tivoli Workload Scheduler	
for z/OS.	47

Chapter 4. Installing	51
--	-----------

Step 1. Loading tracker software	53
Step 2. Loading controller software	53
Step 3. Loading national language support software	54
Step 4. Using the EQQJOBS installation aid.	54
Setting up the EQQJOBS installation aid.	55
Creating the sample job JCL.	55
Generating batch-job skeletons	63
Generating Data Store samples	68
Step 5. Adding SMF and JES exits for event tracking	72
SMF only	73
JES2 only	74
JES3 only	74
Step 6. Updating SYS1.PARMLIB	75
Defining subsystems	75
Calculating MAXECSA values	76
Authorizing the load-module library	77
Updating SMF parameters	77
Updating z/OS dump options	79
Updating the z/OS link-library definition	80
Updating XCF initialization options	80
Modifying TSO parameters	81
Performance considerations	82
Defining the DLF exit for Hiperbatch support	82
Starting the product automatically.	83

Updating APPC options	83	Automatic-recovery-procedure library (EQQPRLIB).	110
Implementing support for data set triggering	83	Script library for end-to-end scheduling with fault tolerance capabilities (EQQSCLIB).	110
Step 7. Setting up the RACF environment	84	Started-task-submit data set (EQQSTC).	110
Controlling the user ID of the address space	84	Submit/release data set (EQQSUDS).	110
Controlling the user ID of submitted jobs	84	Centralized script data set for end-to-end scheduling with fault tolerance capabilities (EQQTWSCS)	111
Normal production jobs	85	Input and output events data sets for end-to-end scheduling with fault tolerance capabilities (EQQTWSIN and EQQTWSOU)	111
Stand-alone cleanup jobs	85	Allocating Data Store data sets	112
Dialog jobs	85	Allocating data sets for the Dynamic Workload Console reporting feature	113
Protecting data sets.	86	Allocating the files and directories	113
Controlling access to resources	86	Configuring for end-to-end scheduling with fault tolerance capabilities in a SYSPLEX environment.	115
Permitting access to the controller through the API	87	Step 10. Creating JCL procedures for address spaces	116
Controlling access to Tivoli Workload Scheduler for z/OS resources when using the Dynamic Workload Console	87	Implementing support for started-task operations	117
Permitting access to the controller through the Dynamic Workload Console	88	Required data sets.	118
Authorizing Tivoli Workload Scheduler for z/OS as a job submitter	88	Optional data sets	120
Authorizing Tivoli Workload Scheduler for z/OS to issue JES commands	89	Step 11. Defining the initialization statements.	122
Authorizing Tivoli Workload Scheduler for z/OS E2E server task to create USS processes	90	Step 12. Creating the DB2 database	122
Authorizing Tivoli Workload Scheduler for z/OS E2E and Dynamic Workload Console server tasks for security resource EZB.BINDDVIPARANGE	90	Sample to migrate the history database.	122
Authorizing Tivoli Workload Scheduler for z/OS Data Store to issue JES commands.	90	Step 13. Setting up the ISPF environment	123
Step 8. Securing communications	91	Setting up the CLIST library	124
Security for TCP/IP connections	91	Setting up the ISPF tables	124
Security for HTTP connections	93	Setting up the default dialog-controller connection table	124
Step 9. Allocating data sets	94	Setting up list tables and graphical attribute tables	126
Allocating the VSAM data sets	95	Allocating dialog data sets to your TSO session	126
Application description data set (EQQADDS)	98	Invoking the Tivoli Workload Scheduler for z/OS dialog.	127
Current plan data sets (EQQCPnDS)	98	Using the EQQOPCAC sample CLIST	127
Data sets for extended data (EQQXDnDS)	99	Modifying an existing ISPF selection menu	128
Current plan backup copy data set (EQQSCPDS)	99	Selecting the main menu directly from TSO	128
JCL repository data sets (EQQJSnDS)	99	Using the ISPF select service	128
Operator Instruction data set (EQQOIDS)	99	Switching to the advanced style for ISPF panels.	128
Allocating Restart and Cleanup VSAM data sets	100	Step 14. Using XCF for communication.	129
Restart and cleanup data sets (EQQPKIxx, EQQSKIxx, and EQQSDFxx)	100	XCF groups	129
Allocating non-VSAM data sets	100	XCF runtime options	130
Internal reader data set (EQQBRDS).	103	Initialization statements used for XCF	130
Checkpoint data set (EQQCKPT)	103	Step 15. Activating the network communication function	131
Diagnostic data sets (EQQDMSG, EQQDUMP, and SYSMDUMP)	103	Adding NCF to the VTAM network definitions	131
Event data sets (EQQEVDs, EQQEVDnn, and EQQHTTP0).	104	Adding NCF session parameters	132
Event-driven workload automation configuration file data set (EQQEVLIB).	106	COS table	133
Job library data set (EQQJBLIB)	106	Activating network resources	133
Job-completion-checker data sets	106	Diagnostic data set	134
Job-tracking data sets (EQQJTARC, EQQJTnn, EQQDLnn)	107	Step 16. Using TCP/IP for communication	134
Message log data set (EQQMLOG)	108	Initialization statements used for TCP/IP	134
Loop analysis log data set (EQQLOOP).	109	Step 17. Activating support for the API.	134
Parameter library (EQQPARM)	109	Defining VTAM resources	135
PIF parameter data set (EQQYPARM)	109	Defining a local LU	135
		Defining logon modes	135

Defining cross-domain resources	136
Updating APPC options	137
Activating support for APPC	137
Step 18. Activating support for the product dialog and programming interface using the server	137
Defining VTAM resources for the product dialog and program interface using the server	138
Defining VTAM resources for the server	139
Defining a local LU for the server	139
Defining logon modes for the server	139
Updating APPC options for the server	140
Defining VTAM resources in a parallel sysplex	140
Starting the server	141
Step 19. Activating support for the end-to-end scheduling with fault tolerance capabilities	141
Activating server support for the end-to-end scheduling with fault tolerance capabilities	142
Step 20. Activating support for the end-to-end scheduling with z-centric capabilities	142
Step 21. Activating support for Dynamic Workload Console	142
Prerequisites	143
Considerations	143
Activating server support for the Dynamic Workload Console	143
Step 22. Activating support for the Java utilities	144
Chapter 5. Verifying your installation	145
Overview of verification	145
Verifying installation of a tracker	145
Ensuring that all installation tasks are complete	146
Checking the message log (EQQMLOG)	146
Verifying tracking events	147
The event writer	147
The event data set	147
Performing problem determination for tracking events	148
Verifying installation of a controller and dialogs	151
Ensuring that all installation tasks are complete	151
Checking the message log (EQQMLOG)	152
Checking the server message log	152
Checking dialog functions	153
Performing problem determination	153
Dialog problems	153
Authority problems	154
Verifying installation of a standby controller	154
Ensuring that all installation tasks are complete	155
Checking the message log (EQQMLOG)	155
Verifying installation of the Restart and Cleanup function	156
Checking the message log (EQQMLOG)	156
Verifying configuration	158
Creating entries in the databases	158
Running batch jobs	158
Checking the message logs (EQQMLOG)	158
Controller message log	158
Tracker message log	165
Verifying workload submission	168
Controlling system	168
Controlled systems	168
Verifying job submission	169

Verifying takeover by a standby controller	170
--	-----

Chapter 6. Migrating 171

Planning for migration	171
Migration considerations	171
Customization considerations	173
Migration strategies	173
JES and SMF exits	173
Migrating to existing subsystem definitions	173
Migrating to new subsystem definitions	174
Getting the right software parts	174
Migration overview	175
Migration steps overview	176
Establishing the required environment	176
Program requirements	176
Installation and verification	176
Parallel testing	177
Migrating an end-to-end network	178
Migrating DB2	178
Changing a shared DASD tracker-to-controller connection to an NCF, XCF, or TCP/IP connection	178
Running on upgraded operating systems	180
Migrating actions	180
Migrating data sets	181
EQQICTOP VSAM data set conversion program	181
Data sets that you need to convert	183
Data sets that can be used	184
Empty data sets	184
Tracker and Data Store considerations	185
Switching into production mode	186
Closing down your production system	186
Converting VSAM files to the new system format	187
Starting the new system	187
Validating the new system	189
Migration steps for a system in a heavy workload environment	189
Performing fallback	192

Part 3. Tivoli Workload Scheduler for z/OS Connector 195

Chapter 7. Installing, Upgrading, and Uninstalling on the embedded WebSphere Application Server 197

Preparing	197
Authorization roles required for installing, upgrading, or uninstalling	197
Instances of Tivoli Workload Automation	197
Installing	198
Installation and uninstallation log files	199
Installing using the wizard in interactive mode	200
Installing using the wizard in silent mode	202
Installing using response file templates	202
Installing with an automatically generated response file	203
Installing from the launchpad	204

Upgrading	204
Upgrading with the wizard in interactive mode	204
Upgrading from version 8.3	204
Upgrading from version 8.5 or 8.5.1	205
Upgrading in silent mode	206
Upgrading from the launchpad	206
Uninstalling	207
Uninstalling using the wizard	207
Uninstalling in silent mode	207

Chapter 8. Installing and uninstalling on WebSphere Application Server for z/OS	209
Business Scenario	209
Authorization roles required for installing and uninstalling	209
Installing on WebSphere Application Server for z/OS	209
Installing using the Integrated Solutions Console	210
Installing using the zConnInstall.sh script	211
Installation and uninstallation log files	213
Enabling communications with Dynamic Workload Console	213
Applying maintenance	216
Uninstalling	217
Uninstalling using the Integrated Solutions Console	217
Uninstalling using the zConnUninstall.sh script	217

Chapter 9. Troubleshooting and maintaining the installation	219
Troubleshooting the installation	219
z/OS connector installation step hangs while installing on a TWA instance with an existing embedded WebSphere Application Server	219
On Windows the z/OS connector installation step fails because the user account does not belong to the Administrators group	219
Dynamic Workload Console creates wrong connection upon installation	220
Installation fails because Windows Workstation Service is not started	220
Failed installation of a dynamic domain manager in the same instance as the z/OS connector	220
Maintaining the installation of the z/OS connector	220
Updating the SOAP properties after changing the WebSphere Application Server user or its password	221
Updating the SOAP properties usage	221
updateWas.sh (.bat)	221

Part 4. Dynamic Workload Console 223

Chapter 10. Preparing	225
Overview of the Dynamic Workload Console	225
Installation overview	225
Installation considerations	226
Selecting your installation method	226

Instances of Tivoli Workload Automation	227
Installation media	228
Installation log files	228
Interactive wizard installation and uninstallation log files	228
Installation log files for the embedded WebSphere Application Server	229

Chapter 11. Installing 231

Installing the Dynamic Workload Console	231
Using the launchpad	231
Using the installation wizard	231
Installing a new instance of the Tivoli Integrated Portal	232
Installing on an existing instance of the embedded WebSphere Application Server	234
Installing on your existing instance of Tivoli Integrated Portal	235
Performing a silent installation	235
Installing the Tivoli Integrated Portal on an external WebSphere Application Server from the images	237
Post-installation steps to connect to Tivoli Workload Scheduler Version 8.3 Fix Pack 3	237
Post-installation steps to configure the use of Lightweight Third-Party Authentication (LDAP)	238
Accessing the Dynamic Workload Console	238
Quick steps to define a Tivoli Workload Scheduler engine connection	240
Quick steps to define a Dynamic Workload Broker connection	241
Starting and stopping the Dynamic Workload Console	242

Chapter 12. Configuring 245

Chapter 13. Getting started 247

Tivoli Workload Scheduler portfolio	247
Dynamic workload broker portfolio	249
First actions	250

Chapter 14. Upgrading 251

Updating authentication	251
Upgrading the console installed on an embedded WebSphere Application Server	252
Directory structure	252
Program directory	252
Directory for SSL files	253
Performing the upgrade	253

Chapter 15. Uninstalling 255

Uninstalling using the wizard	255
Uninstalling in silent mode	255

Chapter 16. Troubleshooting the installation, upgrade, and uninstallation 257

Installation and uninstallation log and trace files	257
Recovering a failed InstallShield wizard installation	257

Recovering a failed upgrade	257
Uninstalling the Dynamic Workload Console and the Tivoli Integrated Portal manually	258
Troubleshooting scenarios	259
Problems with the launchpad	259
Warning messages displayed when using the launchpad on Linux	259
Undefined error when using launchpad on Windows operating system.	260
Problems with the interactive wizard	260
The Dynamic Workload Console installation hangs	260
Installation hangs during stopWas command	260
Tivoli Integrated Portal installation fails even if into the logs you find successfully installed	261
Installation from a remote shared folder fails on Windows operating system.	262
Installation fails on a Linux 390 system with a hostname which is not a Fully Qualified Domain Name	262
Java Virtual Machine (JVM) failure when installing the Dynamic Workload Console on a Red Hat Enterprise Linux (RHEL) Version 5 or a Suse Linux system Version 11	263
The Dynamic Workload Console graphical installation and uninstallation fail to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64.	263
On Windows, the Dynamic Workload Console installation fails if you try to reinstall on a different profile of an external WebSphere Application Server.	264
Problems with the silent installation.	264
The silent uninstallation does not work and an error code is returned	264
Problems with the upgrade.	264
Upgrade fails with message AWSUI0085E	264
Problems with the uninstallation	265
Uninstall fails on Windows if the installation directory contains the @ character	265
The Dynamic Workload Console interactive uninstallation wizard fails to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64.	266
Installation fails when reinstalling the Dynamic Workload Console after having uninstalled it	266

Part 5. Tivoli Workload Scheduler for z/OS Agent 267

Chapter 17. Installing the Tivoli Workload Scheduler for z/OS Agent. . 269

User authorization requirements	270
Authorization roles for running the wizard and a silent installation	270
Authorization roles for running the twsinst script	270
Authorization roles for Software Distribution	270
Starting the launchpad	271

Installing with the installation wizard	271
Performing a silent installation	273
Silent installation using response file templates	274
Silent installation using an automatically generated response file	275
Installing using twsinst	276
twsinst	276
Installing using Software Distribution	280
Software packages and parameters	281
Installation procedure	283
Prerequisite: Installing the Common Inventory Technology (CIT)	283
Installing the Tivoli Workload Scheduler for z/OS Agent	284
Enabling dynamic capabilities	285
Adding the Java runtime to run job types with advanced options	285
Adding Java runtime environment after installation or upgrade	285
Enabling dynamic capabilities after installation or upgrade	286

Chapter 18. Upgrading the Tivoli Workload Scheduler for z/OS Agent. . 287

Coexistence with previous versions	287
User authorization requirements	287
Upgrading notes	287
Upgrading using the installation wizard	288
Upgrading using a silent installation	288
Upgrading using twsinst	289
Upgrading process	289
Examples.	291
Upgrading using Software Distribution.	291
Creating and installing the software package block	291
Upgrading procedure overview	292
Prerequisite: Install the Common Inventory Technology	292
Upgrading the agent	293
Upgrading the Java runtime to run job types with advanced options	294
Adding Java runtime or enabling dynamic capabilities after upgrade	295

Chapter 19. Uninstalling the Tivoli Workload Scheduler for z/OS Agent. . 297

User authorization requirements	297
Uninstalling using the wizard	297
Performing a silent uninstallation	298
Uninstalling the Tivoli Workload Scheduler for z/OS Agent using the twsinst script.	298
Uninstalling using the Software Distribution CLI	299

Appendix A. Sample library (SEQQSAMP) 301

Using the Visual Age compiler	305
SMP/E samples	307
Environment setup	307
RECEIVE processing	307
APPLY processing.	308

ACCEPT processing	308
SMF exits.	309
Exit installation.	309
Job step termination exit.	310
Initialization exit	311
Record write exits	311
JES exits	311
Exit installation.	311
JES2 QMOD phase change exit	312
JES2 JCT I/O exit	312
JES3 OSE modification exit	312
JES3 input service final-user exit	312
RACF samples	312
Class descriptor table.	312
Router table.	313
Sample library (SEQQSAMP)	313

Appendix B. Configuration examples 317

The controlling system	317
Automatic restart actions	317
Initialization statements	317
Multi-access spool systems connected through shared DASD	317
Individual systems connected via shared DASD	320
A z/OS Sysplex	321
A PLEX configuration	324
Controlling a z/OS system through a VTAM link	325
Controlling a z/OS system through a TCP/IP link	326
Controlling a JES2 MAS system through a VTAM link.	328

Appendix C. Invoking the EQQEXIT macro 331

Invoking EQQEXIT in SMF exits	331
Invoking EQQEXIT in JES exits	331
Macro invocation syntax for EQQEXIT	333

Appendix D. Invoking the EQQLSENT macro 335

Invoking EQQLSENT to create EQQDSLST	335
Macro invocation syntax for EQQLSENT	336

Appendix E. Using response files. . . 339

Appendix F. z/OS connector response file properties 341

Appendix G. The Dynamic Workload Console response file properties . . . 345

Notices 351

Trademarks	352
----------------------	-----

Index 355

Figures

1. A basic server configuration example	19	19. EQQJOBSA - Generate Tivoli Workload Scheduler for z/OS batch-job skeletons	66
2. A z/OS system connected through shared DASD	21	20. EQQJOBS5 - Create Data Store samples	69
3. A z/OS system with a VTAM connection	23	21. EQQJOBS6 - Create Data Store samples	70
4. A z/OS system with a TCP/IP connection	24	22. EQQJOBS7 - Create Data Store samples	71
5. A z/OS system with an XCF connection	26	23. Sample message log for a standby controller	156
6. A tracker and controller configured in a single address space	28	24. Sample Message Log for a controller	161
7. Controller and tracker in same address space with tracker connected through SNA	29	25. Sample message log for a tracker	167
8. Controller, tracker, and Data Store connected through XCF	31	26. Shows the keys to enable SSL between the z/OS connector and the Dynamic Workload Console	214
9. Controller and tracker in same address space with tracker connected through TCP/IP	33	27. Two z/OS JES2 MAS complexes connected through shared DASD	318
10. A mixed SNA and XCF connection.	35	28. Individual systems connected through shared DASD	320
11. EQQJOBS0 - EQQJOBS application menu	55	29. A z/OS Sysplex.	322
12. EQQJOBS3 - Create sample job JCL	56	30. A Tivoli Workload Scheduler for z/OS PLEX environment	324
13. EQQJOBS4 - Create sample job JCL	57	31. Controlling a z/OS system through a VTAM link	325
14. EQQJOBS8 - Create sample job JCL	58	32. Controlling a z/OS system through a TCP/IP link	327
15. EQQJOBS9 - Create sample job JCL	60	33. Controlling a JES2 MAS system through a VTAM link	328
16. EQQJOBSC - Create sample job JCL	61		
17. EQQJOBS1 - Generate Tivoli Workload Scheduler for z/OS batch-job skeletons	64		
18. EQQJOBS2 - Generate Tivoli Workload Scheduler for z/OS batch-job skeletons	65		

Tables

1. Tivoli Workload Scheduler for z/OS subtasks	10	33. Data sets that you need to convert	184
2. Stages summarizing the Tivoli Workload Scheduler for z/OS installation process	13	34. Data sets that Tivoli Workload Scheduler for z/OS can use	184
3. Example EQQSERP members for Figure 1	20	35. Installing into an existing instance of Tivoli Workload Automation	198
4. Example EQQPARM members for Figure 2	22	36. Installation log files	199
5. Example EQQPARM members for Figure 3	23	37. Default installation paths for Tivoli Workload Scheduler for z/OS connector.	201
6. Example EQQPARM members for Figure 4	25	38. Configuration page settings.	210
7. Example EQQPARM members for Figure 5	27	39. zConnInstall.properties properties and corresponding values	212
8. Example members for Figure 6	28	40. zConnUpdate.properties properties and corresponding values	216
9. Example members for Figure 7	30	41. zConnUninstall.properties properties and corresponding values	218
10. Example members for Figure 8	32	42. Installing into an existing instance of Tivoli Workload Automation	227
11. Example members for Figure 9	34	43. Installation log files	229
12. Example members for Figure 10.	36	44. Dynamic Workload Console response files	235
13. Checklist for installing Tivoli Workload Scheduler for z/OS	40	45. Required authorization roles for running the installation wizard.	270
14. Tivoli Workload Scheduler for z/OS installation tasks	51	46. Required authorization roles for running twsinst.	270
15. Tracker libraries loaded by SMP/E.	53	47. Required authorization roles for Software Distribution	271
16. Controller libraries loaded by SMP/E.	53	48. Response files	274
17. NLS libraries loaded by SMP/E.	54	49. SEQQSAMP library members	301
18. Sample JCL generated by the EQQJOBS dialog	61	50. Example EQQPARM members for the previous figure	319
19. Controller skeleton JCL generated by the EQQJOBS dialog	67	51. Example EQQPARM members for the previous figure	321
20. Data Store samples generated by the EQQJOBS dialog	72	52. Example EQQPARM members for the previous figure	323
21. Sample exits for Tivoli Workload Scheduler for z/OS.	73	53. Example EQQPARM Members for the previous figure	324
22. Examples of MAXECSA storage values	76	54. Example EQQPARM Members for the previous figure	326
23. Tivoli Workload Scheduler for z/OS VSAM data sets	95	55. Example EQQPARM Members for the previous figure	327
24. Calculations of VSAM data set size	97	56. Example EQQPARM members for the preceding figure	330
25. Restart and cleanup VSAM data sets	100	57. Tivoli Workload Scheduler for z/OS connector response file properties.	341
26. Tivoli Workload Scheduler for z/OS non-VSAM data sets	100	58. Dynamic Workload Console response file properties.	345
27. Data Store VSAM data sets	112		
28. Started task JCL samples for Tivoli Workload Scheduler for z/OS address spaces	116		
29. Tivoli Workload Scheduler for z/OS required data sets	118		
30. Tivoli Workload Scheduler for z/OS optional data sets	120		
31. ISPF and Tivoli Workload Scheduler for z/OS dialog data sets	126		
32. Problem determination for missing tracking events	148		

About this publication

IBM® Tivoli® Workload Scheduler for z/OS® Planning and Installation describes the configuration planning and installation tasks of IBM Tivoli Workload Scheduler for z/OS. Installation is the task of making a program ready to do useful work. This task includes adding the materials on the IBM distribution tape to your system, initializing the program, and applying PTFs to the program. When you install a product, you are carrying out decisions you made in the planning step. Customization, an optional step, gives you the opportunity to tailor the program to the desired behavior or special needs of your site.

Your workload can run on various platforms, but you control it from a central z/OS system that runs the Tivoli Workload Scheduler for z/OS controller.

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

The term *z/OS* is used in this publication to mean z/OS and OS/390® operating systems. Where the term *OS/390* appears, the related information applies only to OS/390 operating systems.

This publication complements the *Tivoli Workload Scheduler for z/OS Program Directory* that describes how to add the materials on the IBM distribution tape to your system.

The *Program Directory* is provided with the Tivoli Workload Scheduler for z/OS installation tape. It describes all of the installation materials and gives installation instructions specific to the product release level or feature number. If any differences exist between this publication and the *Program Directory*, use the information in the *Program Directory*.

What is new in this release

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

What is new in this publication

This section describes what has changed in this publication since version 8.5.1. Changed or added text is marked in the left margin with a vertical bar, except for:

- Editorial changes.
- Release-dependent suffix changed from I (version 8.5.1) to J (version 8.6) for the modules EQQINITJ, EQQSSCMJ, and EQQMINOJ.

Some highlights in this release:

Chapter 8, “Installing and uninstalling on WebSphere Application Server for z/OS,” on page 209

Describes the procedure to install and uninstall the Tivoli Workload Scheduler for z/OS connector on WebSphere Application Server for z/OS.

What is new in this publication

Part 5, “Tivoli Workload Scheduler for z/OS Agent,” on page 267

Describes the procedure to install, upgrade, and uninstall the Tivoli Workload Scheduler for z/OS Agent.

“Allocating the VSAM data sets” on page 95

New support added for extended format VSAM data set that can be allocated for JS data sets that exceed 4 GB.

Who should read this publication

This publication is intended for system programmers who are responsible for software on a z/OS system and plan on installing Tivoli Workload Scheduler for z/OS.

To use this publication effectively, you must be familiar with the following topics:

- Job control language (JCL)
- IBM System Modification Program Extended (SMP/E)
- z/OS
- JES concepts and facilities
- Writing small code fragments in the Assembler H language
- Interactive System Productivity Facility (ISPF)
- Interactive System Productivity Facility/Program Development Facility (ISPF/PDF)
- Time-Sharing Option (TSO)
- Virtual Storage Access Method (VSAM) (desirable but not essential)

The Tivoli Workload Scheduler for z/OS Application Programming Interface (API) uses advanced program-to-program communication (APPC) services. Defining and configuring the conversation partners requires some knowledge of APPC services.

Publications

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

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

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

Using LookAt to look up message explanations

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

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

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

- Your z/OS TSO/E host system. You can install code on your z/OS or z/OS.e systems to access IBM message explanations, using LookAt from a TSO/E command line (for example, TSO/E prompt, ISPF, or z/OS UNIX System Services running OMVS).
- Your Microsoft Windows workstation. You can install code to access IBM message explanations on the *IBM Online Library z/OS Software Products Collection Kit* (SK3T-4270), using LookAt from a Microsoft Windows DOS command line.
- Your wireless handheld device. You can use the LookAt Mobile Edition with a handheld device that has wireless access and an Internet browser (for example, Internet Explorer for Pocket PCs, Blazer, or Eudora for Palm OS, or Opera for Linux handheld devices). Link to the LookAt Mobile Edition from the LookAt Web site.

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

Accessibility

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

For full information with respect to the Dynamic Workload Console, see the Accessibility Appendix in the *Tivoli Workload Scheduler: User's Guide and Reference*, SC32-1274.

Tivoli technical training

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

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

Support information

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

Online

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

IBM Support Assistant

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

Support information

Troubleshooting Guide

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

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

Conventions used in this publication

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

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

Part 1. Planning

Chapter 1. Overview 3

Hardware and software requirements 3

Hardware requirements. 3

Software requirements and optional software . . . 4

Controlling system 4

Controlled z/OS systems 4

Optional software. 4

Related software 5

Software compatibility 5

Parts and their relationships 5

Tracker 6

Controller 6

Server 7

Graphical user interfaces 7

Data Store 8

Configurations. 9

Controlling system 9

Controlled systems 9

Integration with Tivoli Workload Scheduler . . . 9

Subtasks 10

Relationship between the Scheduler and z/OS. . . 12

Using the Tivoli Workload Scheduler for z/OS

Program Directory 13

Sample library 13

The installation process 13

Chapter 2. Planning your configuration 15

Planning considerations 15

Trackers 15

Initialization statements 15

Communication 16

How to connect Tivoli Workload Scheduler for z/OS

systems. 16

Shared DASD 16

z/OS cross-system coupling facility 17

VTAM (network communication function) . . . 17

TCP/IP. 17

Workstation destination 17

Workload restart. 18

JES considerations 18

Basic server configuration example 19

Basic configuration examples 20

DASD connected 20

VTAM connected 22

TCP/IP connected 24

XCF connected 25

Tracker and controller in a single address space 27

Basic data store configuration examples 28

SNA only connection 28

XCF only connection 30

TCP/IP only connection 33

Mixed SNA and XCF connection 34

Chapter 3. Planning your installation 39

Installation considerations 39

Configuring for availability 39

Hot standby 39

Starting an event writer with an event reader

function 40

Using a Hierarchical File System cluster 40

Checklist for installing Tivoli Workload Scheduler

for z/OS 40

Chapter 1. Overview

You can use IBM Tivoli Workload Scheduler for z/OS to plan, control, and automate the entire production workload in your complex, not just the z/OS batch subset. It automatically plans, controls, and monitors your production workload to maximize the throughput and optimize resource use, but lets you intervene manually when required. If you are currently using previously supported versions, follow the instructions in Chapter 6, “Migrating,” on page 171.

This chapter introduces Tivoli Workload Scheduler for z/OS and its implementation. Later chapters in Part 2, “Tivoli Workload Scheduler for z/OS,” on page 47 describe the installation and migration tasks in greater detail.

If you are not familiar with Tivoli Workload Scheduler for z/OS terminology or functions, read the *Tivoli Workload Automation: Overview*.

Hardware and software requirements

This section describes the hardware and software requirements of Tivoli Workload Scheduler for z/OS and includes optional and related software.

Hardware requirements

Tivoli Workload Scheduler for z/OS operates on any IBM hardware configuration supported by z/OS Version 1.10 (program number 5694-A01) or later.

Tivoli Workload Scheduler for z/OS needs a minimum region of 8 MB below the 16 MB line; at least 32 MB must be available above the 16 MB line. The region value depends strictly on the Tivoli Workload Scheduler for z/OS customization and workload. For Tivoli Workload Scheduler for z/OS to work correctly, you might need to specify a region value of 64 MB, which gives you all the available space below the 16 MB line plus 64 MB above the 16 MB line.

In particular, to avoid storage problems, the Tivoli Workload Scheduler for z/OS server must run with a region value of 64 MB when you use the end-to-end scheduling with fault tolerance capabilities. In addition, a region value of 64 MB is strongly recommended when the Tivoli Workload Scheduler for z/OS TCP/IP or APPC server is used for the remote interfaces, for example, the Dynamic Workload Console, PIF, and ISPF.

Consider to increase the region size specification if the server task will normally run for a long time (several weeks or months). Also make sure that the IEFUSI exit is not limiting the region size to a value less than the one coded in the JCL.

A Tivoli Workload Scheduler for z/OS dialog user needs a region of 3 MB below the 16 MB line; if you want to run EQQAUDIT interactively (option 9.10 of the main menu), this number then increases to 4 MB. Tivoli Workload Scheduler for z/OS report programs need a region of 3 MB below the 16 MB line.

Tivoli Workload Scheduler for z/OS uses less than 1 KB of 24-bit Common Service Area (CSA) storage. The amount of 31-bit Extended Common Service Area (ECSA) used is approximately 30 KB, plus 2 KB per active dialog user.

Hardware requirements

A display terminal supported by ISPF Version 4.2 or later is required to invoke and run the Tivoli Workload Scheduler for z/OS host dialogs.

The graphic function requires a terminal supporting Graphic Data Display Manager (GDDM/MVS) Version 2 Release 2 or later.

Software requirements and optional software

Installing and maintaining Tivoli Workload Scheduler for z/OS requires one of the following:

- z/OS (5694-A01) Version 1.10 or later.
- IBM SMP/E for z/OS Version 3 Release 4 (program number 5655-G44) or later.

Tivoli Workload Scheduler for z/OS requires the functions provided by a z/OS control program running on a z/OS system. The Job Entry Subsystem might be either JES2 or JES3.

Controlling system

The following IBM licensed programs are required on the Tivoli Workload Scheduler for z/OS controlling system:

- Tivoli Workload Scheduler for z/OS Version 8.6 (program number 5698-A17). Both the base product (the tracker) and the controller feature are required.

Controlled z/OS systems

On each z/OS system that is controlled by Tivoli Workload Scheduler for z/OS, one of the following IBM licensed programs is required:

- Tivoli Workload Scheduler for z/OS (program number 5698-A17). Only the base product (the tracker) is required.

Optional software

The following Tivoli Workload Scheduler for z/OS functions require specific IBM programs:

- Tracking resource availability requires the Resource Object Data Manager (RODM) in Tivoli NetView® (program number 5697-B82).
- Graphical view of jobs and their dependencies using ISPF panels requires GDDM®.
- z/OS Communication Server Version 1 Release 6 or later is required for TCP/IP Communications.
- Tivoli NetView (5697-B82) is required to enable Tivoli Workload Scheduler for z/OS to schedule generic alerts as defined by that NetView release and to specify an alert receiver ID other than the default receiver.
- User-authority-support functions require z/OS Security Server RACF® (5694-A01) Version 1.6 or later.
- z/OS DFSMS (5694-A01) with Hierarchical Storage Management component is required for the catalog management function to recall migrated data sets.
- IBM DB2 Universal Database Version 8 or later for Tivoli Workload Scheduler for z/OS history functions.
- The Tivoli Workload Scheduler for z/OS Control Language tool and the Dynamic Workload Console reporting feature require either the IBM Compiler Library for REXX/370 (5695-014) or the IBM Alternate Library for REXX on zSeries®, which can be downloaded from <http://www-01.ibm.com/support/docview.wss?rs=960&uid=swg24006107>.

- The Dynamic Workload Console reporting feature requires IBM DB2 WSE Version 9 Release 5 or later.
- For scheduling end-to-end in the distributed environment, see the *Tivoli Workload Scheduler Release Notes*.

Related software

These IBM licensed programs can be used with Tivoli Workload Scheduler for z/OS to provide comprehensive, integrated DP operations:

- Tivoli NetView Version 1 Release 4 (program number 5697-B82) or later
- Report Management and Distribution System (RMDS) Version 2 Release 3 (program number 5648-048) or later
- Tivoli Decision Support for z/OS Version 1 Release 8 (program number 5698-B06) or later
- System Automation for z/OS Version 2 Release 3 (program number 5645-006) or later
- IBM Tivoli Monitoring Version 6 Release 2

Software compatibility

- Tivoli Workload Scheduler for z/OS Version 8.6 uses only existing attachment interfaces to other IBM products.
- Event data sets containing events created by supported previous releases can be used as input to Tivoli Workload Scheduler for z/OS 8.6.
- A system with a current version of Tivoli Workload Scheduler for z/OS can be used to transmit work to a supported previous release using shared submit/release data set, Network Job Entry (NJE), Network Communications Facility (NCF), cross-system coupling facility (XCF), or by transmitting events on an established session between a current version of Tivoli Workload Scheduler for z/OS and a supported previous release.
- JES and SMF exits used to create events for a current version of Tivoli Workload Scheduler for z/OS can also be used to create events for a supported previous release after upgrading to Tivoli Workload Scheduler for z/OS 8.6.
- Existing PIF application programs that work with a supported previous release can be used unchanged with the current version. See *Tivoli Workload Automation: Developer's Guide: Driving Tivoli Workload Scheduler for z/OS*.

Parts and their relationships

Tivoli Workload Scheduler for z/OS consists of a base product, the *agent*, and a number of features. You need the base product to track your workload. Hereafter, you see the agent referred to as the *tracker* and the engine referred to as the *controller*. One z/OS system in your complex is designated the *controlling* system and runs the *controller* feature. Only one controller feature is required, even when you want to start standby controllers on other z/OS systems in a sysplex.

You can control work running in non-z/OS operating environments from a central controller.

To do this, you need the appropriate fault-tolerant workstation and end-to-end enabler feature on the controlling system of the scheduler. See the *Tivoli Workload Scheduler for z/OS: Planning and Installation Guide* for a list of supported operating environments.

Parts and their relationships

You can also control the workload in other operating environments using the end-to-end scheduling functions provided in IBM Tivoli Workload Scheduler for z/OS and in IBM Tivoli Workload Scheduler.

Additionally, national language features let you see the Tivoli Workload Scheduler for z/OS ISPF dialogs in the language of your choice. These languages are available:

- English
- German
- Japanese
- Spanish

The rest of this section describes the tracker and controller, their relationship, and how you can configure them.

Tracker

A tracker is required for every z/OS system in a Tivoli Workload Scheduler for z/OS configuration. The tracker handles the submission of jobs and tasks on the system, and keeps track of the status of the workload. In conjunction with standard interfaces to JES and SMF, Tivoli Workload Scheduler for z/OS records the relevant information about the workload by generating *event records*. The event records are captured and stored by the tracker. The tracker then communicates event information to the controller for further processing. The log where events are written by the tracker is called the *event data set*.

Tivoli Workload Scheduler for z/OS address spaces are defined as z/OS subsystems. The routines that run during subsystem initialization establish services that enable event information to be generated and stored in common storage (ECSA) even when an address space is not active.

You can optionally install a Data Store for each JES spool in a system. In a simple JES configuration this would mean one Data Store for each tracker. In systems with shared spools (for example, JES2 MAS), there is a Data Store for each spool, and there are fewer Data Stores than trackers.

Controller

The controller is the focal point of your Tivoli Workload Scheduler for z/OS configuration. It contains the controlling functions, ISPF dialogs, databases, and plans. The system that the controller is started on is called the Tivoli Workload Scheduler for z/OS controlling system. Tivoli Workload Scheduler for z/OS systems that communicate with the controlling system are called controlled or tracker systems. You need to install at least one controller for your production systems. This controls the entire Tivoli Workload Scheduler for z/OS configuration, the OPCplex, both local and remote.

You can use the controller to provide a single, consistent, control point for submitting and tracking the workload on any operating environment. Tivoli Workload Scheduler for z/OS provides distributed agents and open interfaces you use to integrate the planning, scheduling, and control of work units such as online transactions, file transfers, or batch processing in any operating environment that can communicate with z/OS.

Server

Tivoli Workload Scheduler for z/OS provides a server you use to access the controller remotely from ISPF dialogs, PIFs, and the Dynamic Workload Console interface. Connections with the server run through Advanced Program-to-Program Communications (APPC) sessions or Transmission Control Protocol/Internet Protocol (TCP/IP). The server runs in its own address space; however, it is optional if you do not access the controller remotely.

The server is also used to communicate with the distributed agents for the end-to-end scheduling with fault tolerance capabilities.

Using a Started Task JCL you start and stop one or more servers either individually, using the Start and Stop operator commands, or automatically with the controller, using a keyword in the OPCOPTS statement. A server must start on the same z/OS image as its controller. Only one server can be started with the end-to-end scheduling with fault tolerance capabilities active.

The PIF dialog connection to the controller, whether via server or subsystem interface, is only allowed when the code is at the same level on both sides of the interface.

Graphical user interfaces

One graphical user interface is packaged with the product. You can use it in addition to, or in place of, ISPF:

Dynamic Workload Console

The Web-based user interface for the entire Tivoli Workload Automation suite of products. It is the strategic user interface for the suite and includes support for the latest functions and enhancements featured in Tivoli Workload Scheduler for z/OS.

The console gets access to the controller by way of the Tivoli Workload Scheduler for z/OS connector component, which is also packaged with the product and is connected to Tivoli Workload Scheduler for z/OS by TCP/IP. For detailed information about how you install the z/OS connector, see Part 3, "Tivoli Workload Scheduler for z/OS Connector," on page 195.

The Dynamic Workload Console and Tivoli Workload Scheduler for z/OS connector are components that you install and run on distributed platforms, Windows, UNIX, and Linux.

One z/OS connector can be used to communicate with multiple Tivoli Workload Scheduler for z/OS controllers, and can serve multiple machines running Dynamic Workload Console . Both the graphical user interface and z/OS connector can be installed on the same computer.

The z/OS connector requires either the embedded WebSphere Application Server packaged with the product, which can be installed at the same time, or your own external version of WebSphere Application Server which satisfies the prerequisites listed in the System Requirements Document at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27020800>.

The Dynamic Workload Console documentation is part of the Tivoli Workload Scheduler library and is in the following Tivoli Workload Scheduler guides:

Tivoli Workload Scheduler: Administration Guide, SC23-9113

Documents configuration and miscellaneous administrative tasks for Dynamic Workload Console.

Tivoli Workload Scheduler: Troubleshooting Guide, SC32-1275

Documents logging and tracing Dynamic Workload Console and explains how to troubleshoot Dynamic Workload Console problems.

Tivoli Workload Automation: Messages and Codes, SC23-9114

Documents Dynamic Workload Console messages.

These manuals are available in the Tivoli Workload Scheduler information center:
<http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/index.jsp?toc=/com.ibm.tivoli.itws.doc/toc.xml>

Additional publications specific to Dynamic Workload Console are:

Dynamic Workload Console Download Document

Provides detailed information about downloading the product installation images.

Dynamic Workload Console Detailed System Requirements

A dynamically maintained document which provides detailed information about the supported platforms, the hardware and software prerequisites, and the supported client browsers.

Dynamic Workload Console Release Notes®

A dynamically maintained document which contains the following topics:

- What is new in the release
- Interoperability tables
- Software limitations and workarounds
- Installation limitations and workarounds
- Internationalization Notes
- Documentation updates
- APARS fixed in the release

You can access these documents from the Welcome page of the Tivoli Workload Scheduler information center on the Web.

The Dynamic Workload Console includes also exhaustive online help as well as user scenarios and viewlets.

Data Store

Data Store is a separate address space. Its function is to collect structured (steps and data sets) and, optionally, unstructured (SYSOUT) information for all submitted jobs.

Data Store is required if you want to use the Restart and Cleanup functions:

- Restart at the job or step level
- Data set clean up
- JOBLLOG retrieval

The controller can be connected to Data Store using XCF or SNA, or TCP/IP.

Configurations

You can configure Tivoli Workload Scheduler for z/OS to control virtually any combination of operating environments. Tivoli Workload Scheduler for z/OS can automatically schedule, submit, and track batch jobs, started tasks, and write-to-operator (WTO) messages. You can also use it to coordinate manual activities in your production workload.

Your configuration can include:

- A controlling system
- Controlled systems:
 - Local and remote controlled z/OS systems, including a parallel sysplex
 - Standby controller systems
 - Previous Operation Planning and Control (OPC) Releases
 - Controlled systems running on distributed agents
 - Other operating environments that do not support the distributed agents.

Chapter 2, “Planning your configuration,” on page 15 explains connections between Tivoli Workload Scheduler for z/OS systems and shows examples of configurations.

Controlling system

A controlling system requires both a tracker and a controller. If you install only one system, this is the controlling system. The controlling system can communicate with controlled z/OS systems using shared DASD, the cross-system coupling facility (XCF), network communication function (NCF), and Transmission Control Protocol/Internet Protocol (TCP/IP).

Controlled systems

A controlled z/OS system requires a tracker. Communication with the controlling system is through shared DASD, XCF, NCF, or TCP/IP. The tracker writes event records to an event data set, and transfers the records to the controlling system if connected using XCF, NCF or TCP/IP. NCF uses ACF/VTAM to link Tivoli Workload Scheduler for z/OS systems.

If you use XCF for communication, you can include a *standby controller* on one or more controlled systems. A standby controller is started in its own address space. It can take over the functions of the controller if z/OS fails or if the controller itself fails. It cannot perform the functions of a tracker while in standby mode.

The controller also controls the workload in distributed environments:

- In the end-to-end scheduling with z-centric capabilities network.
- In the end-to-end scheduling with fault tolerance capabilities network, through the end-to-end server.

Tivoli Workload Scheduler for z/OS: Customization and Tuning, SC32-1265 describes in detail how to control other operating environments.

Integration with Tivoli Workload Scheduler

Integration with Tivoli Workload Scheduler is provided by activating either of the following features:

End-to-end scheduling with z-centric capabilities

This feature is designed to let you schedule and control workload from the mainframe to distributed systems through z-centric agents, in a very simple architecture of the end-to-end scheduling framework. Tivoli Workload Scheduler for z/OS becomes the single point of control, providing you with all

Integration with Tivoli Workload Scheduler

the mainframe capabilities to manage distributed workload. Communication between the z-centric agents and Tivoli Workload Scheduler for z/OS controller is direct, through the HTTP or HTTPS protocol.

For detailed information about the end-to-end scheduling with z-centric capabilities, see the *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities* manual.

End-to-end scheduling with fault tolerance capabilities

This feature is based on the Common Agent Technology and it enables Tivoli Workload Scheduler for z/OS to be the master of a Tivoli Workload Scheduler distributed network. This configuration is implemented by connecting a Tivoli Workload Scheduler domain manager directly to Tivoli Workload Scheduler for z/OS.

Tivoli Workload Scheduler for z/OS receives events from the Tivoli Workload Scheduler distributed network and updates the current plan (CP) according to these events. Conversely, every time the current plan is updated, an event is sent to the distributed network to update local plans on the distributed agents.

Being fault-tolerant, the distributed agents can independently continue scheduling when communications with Tivoli Workload Scheduler are interrupted due to network problems. At the same time, the distributed agents are prevented from acting on Tivoli Workload Scheduler for z/OS jobs because these are viewed as running on the Master, the only node authorized to operate on those jobs.

A new type of CPU, named fault-tolerant workstation, has been added to Tivoli Workload Scheduler to logically define, both in the database and in the plan, the distributed agents that will be running jobs for Tivoli Workload Scheduler for z/OS. For detailed information about end-to-end scheduling with fault tolerance capabilities, see the *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with Fault Tolerance Capabilities* manual.

Subtasks

A Tivoli Workload Scheduler for z/OS address space consists of many z/OS subtasks. Some of these subtasks are always attached when the address space is started, others are conditionally attached according to initialization parameters specified for the scheduler options (OPCOPTS) statement in the Tivoli Workload Scheduler for z/OS parameter library. Table 1 describes the subtasks.

Table 1. Tivoli Workload Scheduler for z/OS subtasks

Subtask ID	Component code	Description	Component of FMID	Activated by OPCOPTS parameter	Function
APPC	PP	APPC functions	JWSZ302	APPCTASK(YES)	Starts APPC support
AR	AR	Automatic recovery	JWSZ302	RECOVERY(YES)	Manages failing operations
CPH	CPH	Critical path handler	JWSZ302	Always activated	Updates the critical job table
DRT	DX	Data router	HWSZ300	Always activated	Routes data to other subtasks or Tivoli Workload Scheduler for z/OS subsystems
EMGR	EM	Event manager	JWSZ302	OPCHOST(YES)	Processes job-tracking events

Table 1. Tivoli Workload Scheduler for z/OS subtasks (continued)

Subtask ID	Component code	Description	Component of FMID	Activated by OPCOPTS parameter	Function
ERDR	ER	Event reader	HWSZ300	ERDRTASK(<i>n</i>)	Reads events from an event data set
EWTR	EW	Event writer	HWSZ300	EWTRTASK(YES)	Writes events to an event data set
EXA	EX	External router	JWSZ302	OPCHOST(YES)	Calls EQQUX009 to route submit requests to a user-defined destination ID
FL	FL	Fetch joblog	JWSZ302	RCLEANUP(YES)	Retrieves JOBLOG information
GEN	GS	General service	JWSZ302	OPCHOST(YES)	Processes Tivoli Workload Scheduler for z/OS dialog requests
HTC	HTC	HTTP client	JWSZ302	HTTP keyword of ROUTOPTS	Manages communications with z-centric agents through the HTTP or HTTPS protocol
HTS	HTS	HTTP client	JWSZ302	HTTP keyword of ROUTOPTS	Listens for inbound requests from the z-centric agent
ID	ID	TCP/IP Data Store	HWSZ300	TCPDEST keyword of FLOPTS	Manages communications with TCP/IP-connected Data Stores
IP	IP	TCP/IP tracker	HWSZ300	TCPIP keyword of ROUTOPTS	Manages communications with TCP/IP-connected standard trackers
JCC	JC	Job completion checker	HWSZ300	JCCTASK(YES)	Scans SYSOUT data sets
JLA	JL	JT log archiver	JWSZ302	OPCHOST(YES)	Copies JT logs to the archive data set, EQQJTARC
NMM	NM	Normal mode manager	JWSZ302	OPCHOST(YES)	Maintains the current plan
PSU	PS	Pre-SUBMIT tailoring	JWSZ302	RCLEANUP(YES)	Tailors the JCL before submitting it by adding the EQQCLEAN pre-step
RODM	RM	RODM support	HWSZ300	RODMTASK (YES)	Starts RODM support
SUB	SU	Submit task	HWSZ300	Always activated	Initiates work (job submit, job release, and WTO and STC operations)
TWS	TWS	End-to-end with fault tolerance capabilities task	JWSZ302	TPLGYSRV keyword of OPCOPTS	Handles events to and from fault-tolerant workstations (using the Tivoli Workload Scheduler for z/OS server)
VTAM®	CB	Network communication function (NCF)	HWSZ300	NCFTASK(YES)	Transmits and receives Tivoli Workload Scheduler for z/OS data through a VTAM link
WSA	WA	Workstation analyzer	JWSZ302	OPCHOST(YES)	Schedules work for processing
Note: The subtask ID is the same identifier used to control the subtask using the z/OS MODIFY command.					

Subtasks

When a controller is started in standby mode, only the Tivoli Workload Scheduler for z/OS main task (EQQMAJOR) is started. The subtasks that comprise an active controller are attached when a takeover is performed.

Relationship between the Scheduler and z/OS

Tivoli Workload Scheduler for z/OS is a z/OS subsystem, initialized during IPL. Routines run during subsystem initialization establish basic services, such as an event queue in ECSA. Tivoli Workload Scheduler for z/OS uses standard interfaces to SMF and JES to gather relevant information about the workload on the z/OS system.

The functions of the controller are available when an address space has been created for it, and the required subtasks have been successfully initialized. The controller can run either as a started task or as a batch address space. Normally, the address space is started during the IPL process, that is by a z/OS start command in `COMMNDnn`, or by console automation. Alternatively, a z/OS operator can issue a `START` command from the operator console. The z/OS operator can also stop or modify the address space, using the `STOP` and `MODIFY` commands.

A TSO user accesses Tivoli Workload Scheduler for z/OS services using the dialogs. A dialog is a sequence of ISPF panels. Many of the functions supported by the dialogs pass service requests from the TSO user's address space to the controller address space for processing.

Before performing any function you request, the dialog function passes the request to the system authorization facility (SAF) router. If RACF, or a functionally equivalent security product, is installed and active on the z/OS system, the SAF router passes the verification request to RACF to perform this authority check.

A typical dialog service request is to access one or more records in VSAM files that are maintained and controlled by Tivoli Workload Scheduler for z/OS. Such a request is passed to Tivoli Workload Scheduler for z/OS through the z/OS subsystem interface (SSI). This interface invokes a routine that resides in common storage. This routine must be invoked in APF-authorized mode.

Consider that all long term plan (LTP) and CP batch planning jobs have to be excluded from SMARTBATCH DA (Data Accelerator) processing. When the SMARTBATCH DATA ACCELERATOR is used with the scheduler LTP and CP batch planning jobs, the normal I/O to EQQCKPT is delayed until END OF JOB (or at least END OF JOBSTEP). This interferes with the normal exchange of data between the batch job and the controller started task so that when the batch job signals the controller to check the EQQCKPT to determine whether a new current plan has been created, the required updates to the CKPT have not yet been made. This causes the controller to conclude that no NCP has been created, and no turnover processing is done. As a result, even if the plan jobs run successfully, the NCP is not taken into production by the controller unless a `CURRPLAN(NEW)` restart is performed.

The Data Store uses the MVS/JES SAPI functions to access sysout data sets, allowing concurrent access to multiple records from a single address space.

Batch optimizer utilities, such as BMC Batch Optimizer Data Optimizer and Mainview Batch Optimizer, prevent correct communication between the scheduler's controller and CP/LTP batch planning jobs. The scheduler's logic depends on an exchange of enqueues and real-time updates of several sequential data sets to pass

information back and forth between the controller's STC and the CP/LTP batch planning jobs. These optimizers hold I/O from the batch jobs until END OF STEP or END OF JOB, then preventing the required communication from taking place. When such utilities are allowed to "manage" I/O for the scheduler's CP or LTP batch planning jobs, communication between the jobs and the controller is disrupted. This causes numerous problems that are hard to diagnose. Most commonly, the CURRENT PLAN EXTEND or REPLAN jobs will run to normal completion, and an NCP data set will be successfully created, but the controller will fail to automatically take the new plan into production until it is forced to do so via a CURRPLAN(NEW) restart of the CONTROLLER. Use of BATCHPIPES with these batch planning jobs will result in the same sorts of problems.

Using the Tivoli Workload Scheduler for z/OS Program Directory

The *Tivoli Workload Scheduler for z/OS: Program Directory*, GI11-4248 provided with the product distribution tape might include technical information that is more recent than the information provided in this publication. In addition, the *Program Directory* describes the program temporary fix (PTF) level of the Tivoli Workload Scheduler for z/OS licensed program that you receive.

The *Program Directory* contains instructions for unloading the product and information about additional maintenance for your level of the distribution tape.

Before you start installing the product, check the *preventive service planning* (PSP) bucket for recommendations added by the service organizations after your *Program Directory* was produced. The PSP includes a Service Recommendations section that includes high impact or pervasive (HIPER) APARs. Ensure the corresponding PTFs are installed before you start a Tivoli Workload Scheduler for z/OS subsystem.

Sample library

SEQQSAMP is a library included on the distribution tape containing samples of exits, application programs, and the job control language (JCL). You can use the samples for specific installation tasks. Appendix A, "Sample library (SEQQSAMP)," on page 301 describes the members of the SEQQSAMP library. Familiarize yourself with the contents of the SEQQSAMP library before you begin installation.

The installation process

To understand the flow of the installation, migration, and customization processes, read through this guide before you install Tivoli Workload Scheduler for z/OS.

The following table shows the various stages in the installation process.

Table 2. Stages summarizing the Tivoli Workload Scheduler for z/OS installation process

Stage	Description	For more information ...
1	Plan your configuration. You might create a diagram of your own Tivoli Workload Scheduler for z/OS configuration to refer to during the installation process.	Chapter 2, "Planning your configuration" gives examples of common configurations.
2	Plan your product installation.	Chapter 3, "Planning your installation" describes considerations for installing Tivoli Workload Scheduler for z/OS and provides a checklist for the installation tasks.

The installation process

Table 2. Stages summarizing the Tivoli Workload Scheduler for z/OS installation process (continued)

Stage	Description	For more information ...
3	Install the product.	Chapter 4, “Installing” describes the installation tasks in detail.
4	Verify your installation.	Chapter 5, “Verifying your installation” describes how you can verify that Tivoli Workload Scheduler for z/OS is correctly installed.

When you have installed the product, you might want to include more functions. *Tivoli Workload Scheduler for z/OS: Customization and Tuning* explains how you do this.

Chapter 2. Planning your configuration

This chapter describes several areas to consider when planning the configuration for your installation. It explains connections between Tivoli Workload Scheduler for z/OS systems and provides some examples of basic configurations. See *Customization and Tuning* for details on how to configure end-to-end scheduling in a distributed environment.

Planning considerations

Tivoli Workload Scheduler for z/OS must recognize when events occur; for example, when a started task or job begins to run or terminates, or when a data set has been printed. It uses JES and SMF exits to obtain this information from z/OS and to create *event records* describing the changes in the system. The event records are stored in a sequential file called the *event data set* identified by the EQQEVDS DD name.

Tivoli Workload Scheduler for z/OS also uses the event data set to write checkpoint information for submission requests. The first record of the data set is used for this purpose, so the EQQEVDS DD name must be specified for all Tivoli Workload Scheduler for z/OS address spaces. The same data set can be used for both submit checkpointing and the event-writer subtask.

Trackers

A tracker must be installed on every z/OS system that you want Tivoli Workload Scheduler for z/OS to control. The tracker on each system writes events to the event data set. A subtask of the tracker, called the *event writer* performs this function. For the current plan to be updated, the event information must be communicated to, and processed by, the controller. The events are routed to the controller through the connection linking the tracker and the controller, either by an *event reader* subtask, or by requesting the event writer to queue the events immediately to the data router subtask, when the connected type is not shared DASD.

Initialization statements

Tivoli Workload Scheduler for z/OS initialization statements specified in the parameter library describe, among other things, the configuration of your installation. In a shared DASD environment, an event reader subtask started at the controller reads the events from the event data set. The events are then used to update the current plan. A sequence number, specified on the ERSEQNO of the ERDROPTS initialization statement, identifies each event reader subtask. This number is used to build a DD name in the JCL procedure of the address space where the event reader is started. This DD name identifies the event data set that the event reader should process. It has the format EQQEVDnn, where nn is the sequence number of the event reader that services this event data set.

When a tracker has a non-DASD connection with the controller (that is, an XCF, NCF, or TCP/IP connection) or the tracker and controller are running in the same address space, the event writer can be used to forward events directly to the controller.

Planning considerations

When an event writer is started with the EWSEQNO keyword on the EWTROPTS initialization statement, the event writer logs event information on the event data set and adds the event concurrently to the data router queue. The event is *not* read back from the event data set each time as it is by an event reader subtask. In this configuration, events are only read back from DASD if they need to be resent to the controller during restart processing, for example when the communication link to the controller becomes active after an outage.

See *Tivoli Workload Scheduler for z/OS: Customization and Tuning* for more information on the ERDROPTS and EWTROPTS initialization statements. Also see “Event data sets (EQQEVDS, EQQEVDDnn, and EQQHTTP0)” on page 104 which provides important information about allocating event data sets.

Communication

The data router subtask is responsible for communicating the event to the controller event manager subtask, either by XCF, NCF, TCP/IP or by adding directly to the queue when the tracker and controller are started in the same address space. This eliminates the need for a separate event-reader function, saves time, and saves I/O operations.

The EWSEQNO value is not used to build a DD name, as happens with the event reader subtask. The event writer uses the EQQEVDS DD name to identify the event data set.

If a connection is lost between a tracker and the controller, the event writer continues to record events. When the connection is restored, the event data set is processed from the last event received by the controller before the outage.

Note: Controllers scheduling work (for a given MVS[™] image) must have unique subsystem names.

How to connect Tivoli Workload Scheduler for z/OS systems

Tivoli Workload Scheduler for z/OS systems can be connected using any of these methods:

- Shared DASD
- XCF communication links
- VTAM link
- TCP/IP link

The controller uses any of these methods to transmit work to a tracker system. The tracker system uses the same connection to transmit events back to the controller.

Distributed agents communicate with the controller using TCP/IP services.

Shared DASD

When two Tivoli Workload Scheduler for z/OS systems are connected through shared DASD, they share two data sets for communication:

- Event data set
- Submit/release data set

The tracker writes the event information it collects to the event data set. An event reader, started in the controller, reads the data set and adds the events to the data router queue.

A submit/release data set is one method that the controller uses to pass work to a controlled system. When two Tivoli Workload Scheduler for z/OS systems share a submit/release data set, the data set can contain these records:

- Release commands
- Job JCL
- Started-task JCL procedures
- Data set cleanup requests
- WTO message text

Both the host and the controlled system must have access to the submit/release data set. The EQQSUDS DD name identifies the submit/release data set in the tracker address space. At the controller, the DD name is user-defined, but it must be the same name as that specified in the DASD keyword of the ROUTOPTS statement. The controller can write to any number of submit/release data sets.

z/OS cross-system coupling facility

Tivoli Workload Scheduler for z/OS uses the z/OS cross-system coupling facility (XCF) to connect Tivoli Workload Scheduler for z/OS systems using XCF communication links.

When one or more trackers are connected to the controller through XCF communication links, the Tivoli Workload Scheduler for z/OS systems form an XCF group. The systems use XCF group, monitoring, and signaling services to communicate. The controller submits work and control information to the trackers using XCF signaling services. The trackers use XCF services to transmit events back to the controller.

XCF connections let Tivoli Workload Scheduler for z/OS support a hot standby controller and automatic-workload-restart functions.

VTAM (network communication function)

Tivoli Workload Scheduler for z/OS uses the network communication function (NCF) to connect a tracker to the controller using a VTAM link. The controller transmits work to the tracker through NCF, and the same connection is used to pass back event information.

TCP/IP

Tivoli Workload Scheduler for z/OS uses Transmission Control Protocol/Internet Protocol (TCP/IP) to connect a tracker to the controller using a TCP/IP link. The controller transmits work to the tracker through TCP/IP, and the same connection is used to pass back event information. The scheduler uses TCP/IP also to connect a distributed agent to the server. The TCP/IP connection between the server and the agents is established by the server.

Workstation destination

The various physical and logical locations where tasks are performed at your installation are represented in Tivoli Workload Scheduler for z/OS by workstations. Each workstation groups related activities. Every operation in the application description database and the current plan is associated with a workstation. You define workstations in the workstation description database.

The destination field is one attribute of a workstation description. It identifies the system in your configuration that operations scheduled for this workstation should

Workstation destination

be submitted to. The field can contain the DD name of a submit/release data set, an XCF member name, the VTAM LU of a tracker, or a user-defined destination.

If the destination field is not blank, the same name must also be present in the APPC, DASD, SNA, TCP, TCPIP, USER, or XCF keywords of the ROUTOPTS statement, depending on the connection method.

The destination field can also remain blank. A blank destination field means that operations at this workstation will be submitted by the controller or by a fault-tolerant agent, whose workstation type is FTA.

The operation-initiation exit, EQQUX009, handles the routing of the workload to user-defined destinations.

Workload restart

You can use workload restart (WLR) to restart and reroute work in your Tivoli Workload Scheduler for z/OS configuration. WLR tracks the status of workstations. It can be invoked when a workstation becomes inactive; that is, when the controller cannot communicate with the tracker at the destination that the workstation represents. If an operation is *restartable*, it can be started again after a workstation failure. If an operation is *reroutable*, it can be moved to an alternative workstation for running when its workstation is no longer active.

For WLR purposes, the status of a workstation can be either active or inactive. An inactive workstation has a status of offline, failed, or unknown. The actions that WLR performs depend on the new status of the workstation and on the values you specify on the WSFAILURE and WSOFFLINE keywords of the JTOPTS initialization statement. The inactive status that a workstation can have depends on the type of connection between the tracker and the controller. The connection type and the new workstation status determine whether workload restart actions can be invoked automatically. You can use the full capabilities of WLR on systems that are connected by XCF.

Note: JES also has restart functions, which can be used when the system is restarted after a failure. JES can restart jobs that were active when the failure occurred. To prevent jobs from being started twice, ensure that both JES and WLR do not perform restart actions for jobs on the failing system.

JES considerations

The JES type and configuration in your installation has implications on your Tivoli Workload Scheduler for z/OS configuration. Consider these situations:

1. On systems where JES2 is installed, a Tivoli Workload Scheduler for z/OS Tracker must be installed on each system in the JES2 Multi-Access Spool (MAS) complex.
2. If you do not install Tivoli Workload Scheduler for z/OS on all systems in a JES3 complex, ensure that:
 - A tracker is installed on the global.
 - Jobs are submitted, whether by Tivoli Workload Scheduler for z/OS or outside Tivoli Workload Scheduler for z/OS, to a system where a tracker is installed. Use the `//*MAIN SYSTEM=sysid` statement in the JCL, or start job classes used by these jobs only on those systems where a tracker is installed.
 - If you track print operations, output is printed only on those systems where a tracker is installed.

Basic server configuration example

This section gives an example of a basic server configuration of Tivoli Workload Scheduler for z/OS. Tivoli Workload Scheduler for z/OS connects distributed agents to the server via TCP/IP. The controller transmits work to the fault-tolerant workstations via TCP/IP, and the same connection is used to pass event information back. The server is connected to the first level domain managers in the distributed network. TCP/IP is also used to connect the Dynamic Workload Console to the server through the connector. The server connects to the remote interfaces, either Programming Interfaces or remote ISPF interface users, using APPC. The following example shows a simple configuration using mixed protocols and minimal parameter customization.

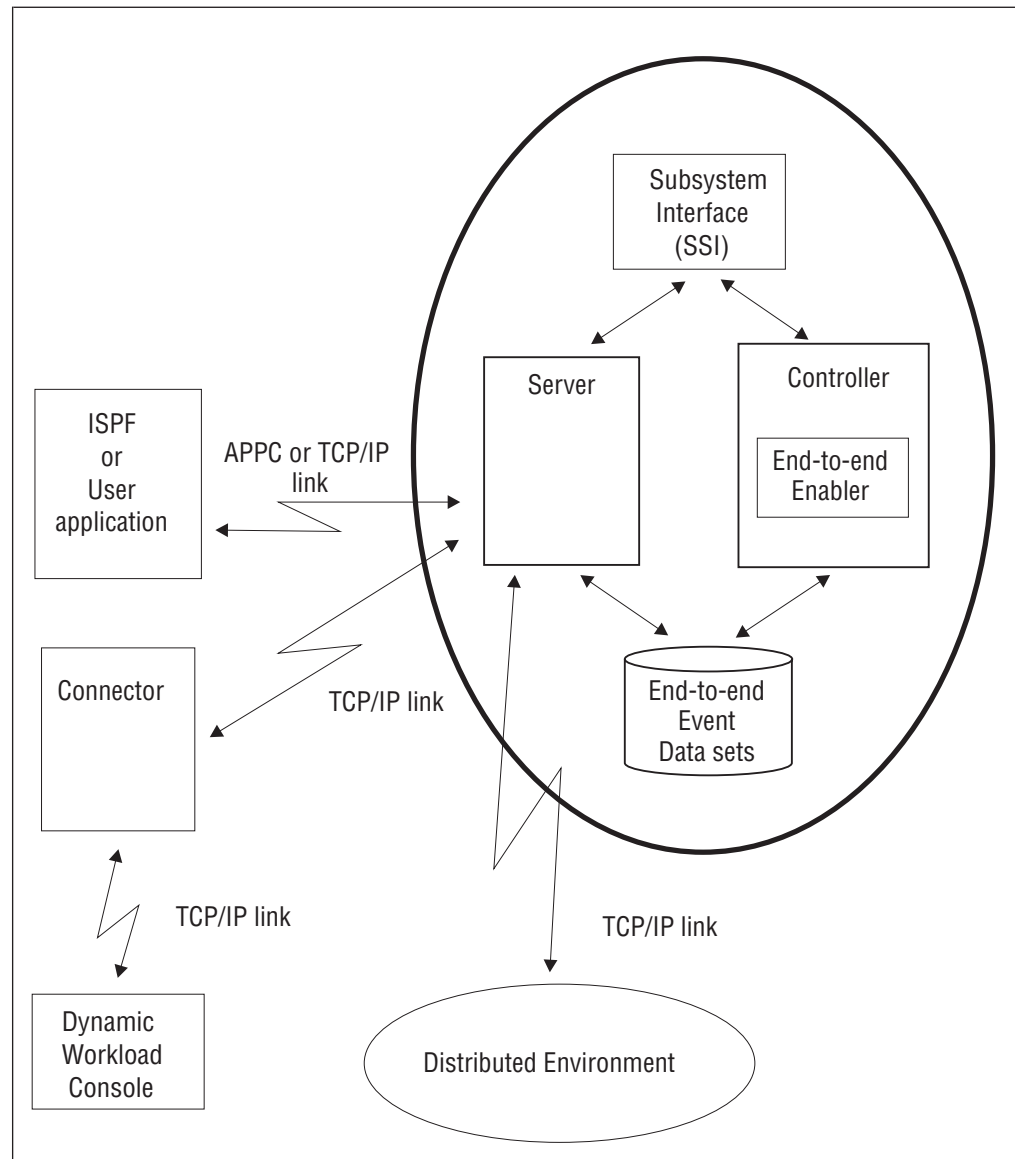


Figure 1. A basic server configuration example

Table 3 on page 20 shows the initialization statements you can use to create the configuration in Figure 1, using the TCP/IP link for the user application-server communication.

Basic server configuration example

Table 3. Example EQQSERP members for Figure 1

EQQSERP SERVOPTS SUBSYS(OPCA) USERMAP(USERS) PROTOCOL(E2E,TCP) TPLGYPRM(TPLGY) INIT CALENDAR(DEFAULT)	TPLGY TOPOLOGY TPLGYMEM(TPLGYDOM) BINDIR('/usr/lpp/TWS8.3.0') WRKDIR('/var/TWS/OPCA') USRMEM(TPLGYUSR) CODEPAGE(IBM-280)
TPLGYUSR USRREC USRCPU(FTW1) USRNAM('tw83') USRPSW('tw83')	TPLGYDOM DOMREC DOMAIN(DOM0) DOMMNGR(FTW1) DOMPARENT(MASTERDM) CPUREC CPUNAME(FTW1) CPUOS(WNT) CPUNODE('xxx.xx.xxx.x') CPUDOMAIN(DOM0) CPUTYPE(FTA) CPUTCPIP(31111) CPUFULLSTAT(ON) CPUAUTOLNK(ON) CPULIMIT(SYSTEM) FIREWALL(NO) CPUTZ('EUT')

Note: For USERS members, see the SERVOPTS USERMAP parameter, and for the TPLGY members, see the TOPOLOGY statement in *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

Basic configuration examples

This section gives examples of Tivoli Workload Scheduler for z/OS configurations using the various connection methods. The examples are based on a single-image z/OS environment. Appendix B, “Configuration examples,” on page 317 contains examples of more complex configurations.

The examples in this section show:

- All Tivoli Workload Scheduler for z/OS address spaces as Version 2 subsystems.
- Sample initialization statements that you can use to create the configuration. Only initialization statements that specifically relate to the configuration are included.
- The Tivoli Workload Scheduler for z/OS components that are required, the flow of automatic work submission, and event collection in various system combinations.

DASD connected

Figure 2 on page 21 shows two Tivoli Workload Scheduler for z/OS address spaces with a DASD connection on a z/OS system.

You represent this system to Tivoli Workload Scheduler for z/OS by defining a computer workstation with a destination field that specifies a submit/release DD name. The controller writes JCL, release commands, WTO messages, and cleanup requests into the submit/release data set. The tracker reads the submit/release data set and performs the following actions:

- Submits JCL for batch jobs to the JES internal reader
- Writes the JCL for started tasks into the EQQSTC data set and issues `START procname` z/OS commands
- Issues JES release commands for jobs in HOLD status
- Submits the cleanup job.

The event-tracking routines create event records to describe activities that occur on the system. These records are added to the tracker event writer queue in ECSA. The tracker processes the queue and writes the events into the event data set. An event-reader subtask started in the controller address space reads the event data set, and the current plan is updated.

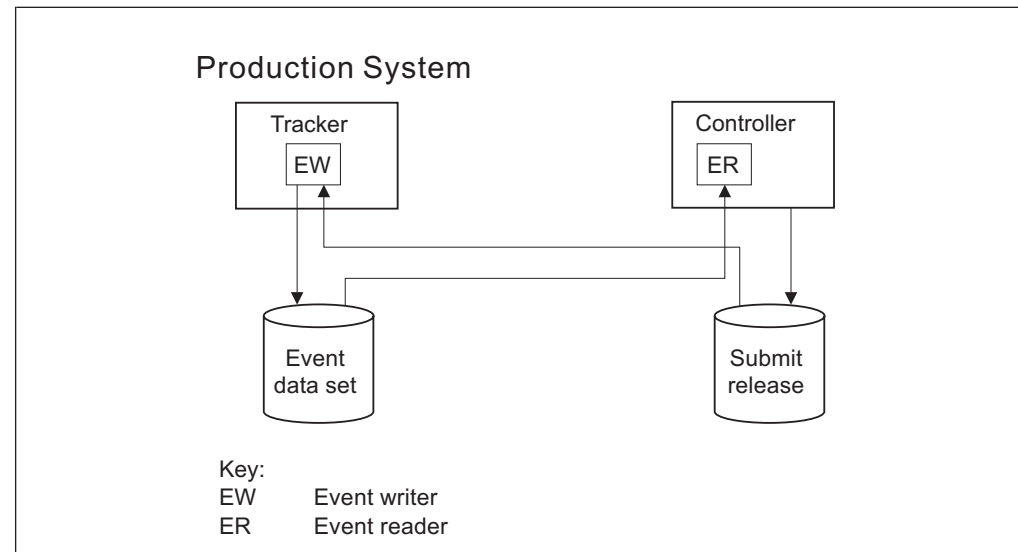


Figure 2. A z/OS system connected through shared DASD

You can also configure this system without a submit/release data set. When the workstation's destination is blank, batch jobs, started tasks, release commands, and WTO messages are processed by the submit subtask automatically started in the controller address space. The event-tracking process remains unchanged.

Table 4 on page 22 shows the initialization statements you can use to create the configuration in Figure 2.

Basic configuration examples

Table 4. Example EQQPARM members for Figure 2

Members for the controller		Members for the tracker	
OPCECNT		TRKA	
OPCOPTS	OPCHOST(YES) ERDRTASK(1) ERDRPARM(STDERDR)	OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(STDEWTR)
ROUTOPTS	DASD(EQQSYSA)	TRROPTS	HOSTCON(DASD)
STDERDR		STDEWTR	
ERDROPTS	ERSEQNO(01)	EWTROPTS	SUREL(YES)

Note: In this example, EQQSYSA is used for the user-defined DD name of the submit/release data set. This DD name appears in the JCL procedure of the controller and in the destination field of the workstation.

VTAM connected

Figure 3 on page 23 shows two Tivoli Workload Scheduler for z/OS address spaces with a VTAM connection on a z/OS system.

You represent this system to Tivoli Workload Scheduler for z/OS by defining a computer workstation with a destination field that specifies the LU name of the tracker. The controller transmits JCL, release commands, WTO messages, and cleanup requests across the LU-LU link using the NCF component. The tracker receives data across the VTAM link and performs the following actions:

- Submits JCL for batch jobs to the JES internal reader
- Writes the JCL for started tasks into the EQQSTC data set and issues *START procname* z/OS commands
- Issues JES release commands for jobs in HOLD status
- Submits the cleanup job.

The event-tracking routines create event records to describe activities that occur on the system. These records are added to the tracker event writer queue in ECSA. The tracker processes the queue, transmits the records to the controller across the VTAM link, and writes the events into the event data set. The VTAM subtask in the controller receives the event records, and the current plan is updated.

Note: You must specify EQQEVDS for a controller, even if an event writer is not started in the controller address space. The EQQEVDS data set is used for submit checkpointing. It can be the same data set that is used by an event-writer function. Use a unique EQQEVDS for each address space of the scheduler.

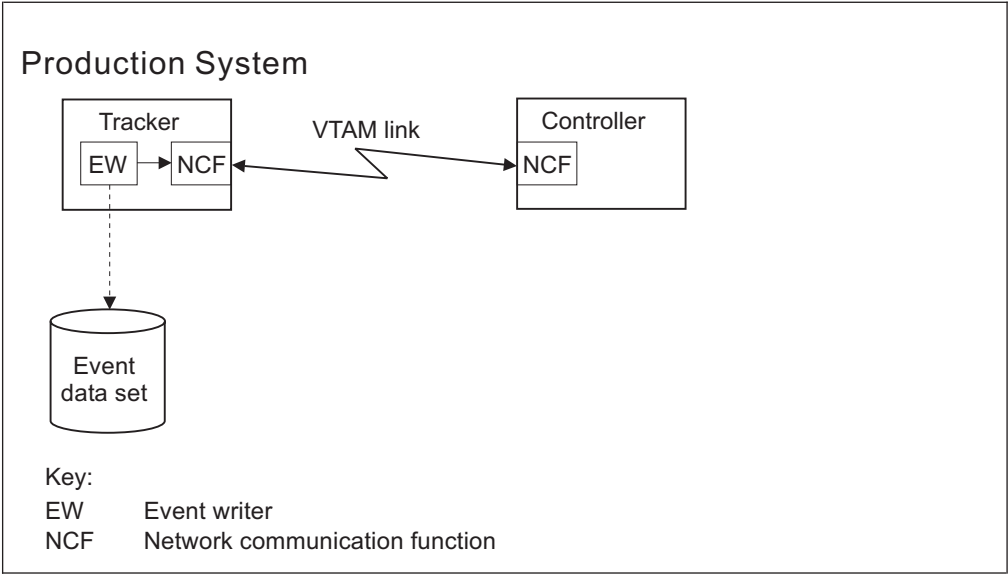


Figure 3. A z/OS system with a VTAM connection

Table 5 shows the initialization statements you can use to create the configuration in Figure 3.

Table 5. Example EQQPARM members for Figure 3

Members for the controller	Members for the tracker
<div>OPCECNT OPCOPTS OPCHOST(YES) ERDRTASK(0) NCFTASK(YES) NCFAPPL(CNTSYS) ROUTOPTS SNA(TRKSYS)</div>	<div>TRKA OPCOPTS OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(STDEWTR) NCFTASK(YES) NCFAPPL(TRKSYS) TRROPTS HOSTCON(SNA) SNAHOST(CNTSYS)</div> <div>STDEWTR EWTROPTS EWSEQNO(01)</div>

Note: In this example, the LU name of the controller is CNTSYS and the tracker uses TRKSYS. The tracker LU is defined in the destination field of the workstation.

TCP/IP connected

Figure 4 shows two Tivoli Workload Scheduler for z/OS address spaces with a TCP/IP connection on a z/OS system.

You represent this system to the scheduler by defining a computer workstation with a destination field that specifies the destination name of the tracker. The controller transmits JCL, release commands, WTO messages, and cleanup requests across the TCP/IP link. The tracker receives data across the TCP/IP link and performs the following actions:

- Submits JCL for batch jobs to the JES internal reader
- Writes the JCL for started tasks into the EQQSTC data set and issues `START procname z/OS` commands
- Issues JES release commands for jobs in HOLD status
- Submits the cleanup job.

The event-tracking routines create event records to describe activities that occur on the system. These records are added to the tracker event writer queue in ECSA. The tracker processes the queue, transmits the records to the controller across the TCP/IP link, and writes the events into the event data set. The IP task in the controller receives the event records, and the current plan is updated.

Note: You must specify EQQEVDS for a controller, even if an event writer is not started in the controller address space. The EQQEVDS data set is used for submit checkpointing. It can be the same data set that is used by an event-writer function. Use a unique EQQEVDS for each address space of the scheduler.

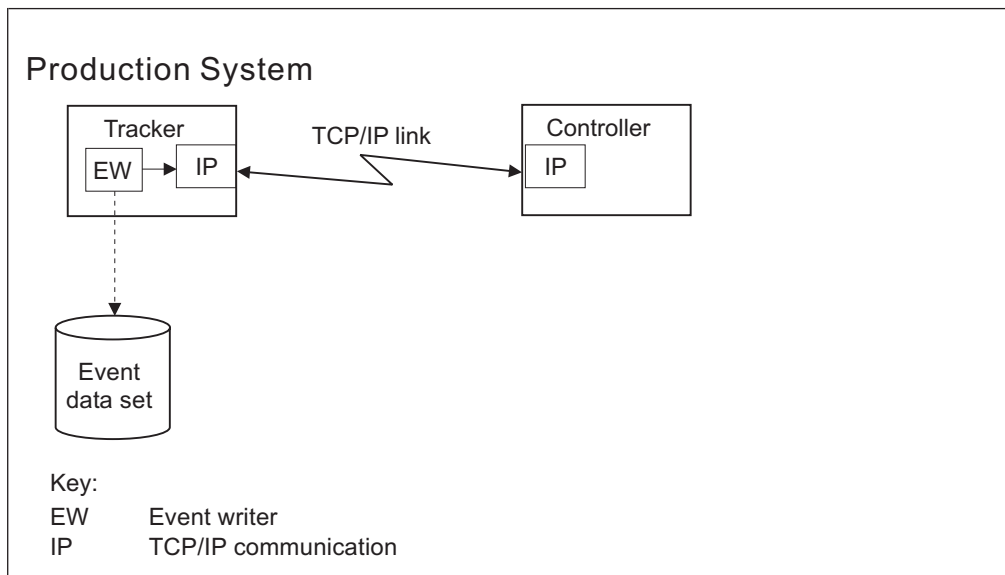


Figure 4. A z/OS system with a TCP/IP connection

Table 6 on page 25 shows the initialization statements you can use to create the configuration in Figure 4.

Table 6. Example EQQPARM members for Figure 4

Members for the controller	Members for the tracker
<div><div>OPCECNT</div><div>OPCOPTS OPCHOST(YES) ERDRTASK(0) ROUTOPTS TCPIP(DEST1:'1.111.111.111'/4444) TCPOPTS TCPIPJOBNAME('TCPIP') HOSTNAME('9.12.134.1') TRKPORTNUMBER(8888)</div></div>	<div><div>TRKA</div><div>OPCOPTS OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) TRROPTS HOSTCON(TCP) TCPHOSTNAME('9.12.134.1') TCPPORTNUMBER(8888) TCPOPTS TCPIPJOBNAME('TCPIP') HOSTNAME('1.111.111.111') TRKPORTNUMBER(4444)</div></div> <div><div>STDEWTR</div><div>EWTROPTS EWSEQNO(01)</div></div>

Note: In this example, the name of the destination is DEST1. The destination name is defined also in the destination field of the workstation.

XCF connected

Figure 5 on page 26 shows two Tivoli Workload Scheduler for z/OS address spaces with an XCF connection in a z/OS monoplex.

You represent this system to Tivoli Workload Scheduler for z/OS by defining a computer workstation with a destination field that specifies the XCF member name of the tracker. The controller uses XCF services to transport JCL, release commands, WTO messages, and cleanup requests to members in the sysplex. The tracker receives data from XCF and performs the following actions:

- Submits JCL for batch jobs to the JES internal reader
- Writes the JCL for started tasks into the EQQSTC data set and issues START *procname* z/OS commands
- Issues JES release commands for jobs in HOLD status
- Submits the cleanup job.

The event-tracking routines create event records to describe activities that occur on the system. These records are added to the tracker event writer queue in ECSA. The tracker processes the queue, transports the records to the controller across the XCF link, and writes the events into the event data set. The data router subtask in the controller receives the event records from XCF, and the current plan is updated.

Basic configuration examples

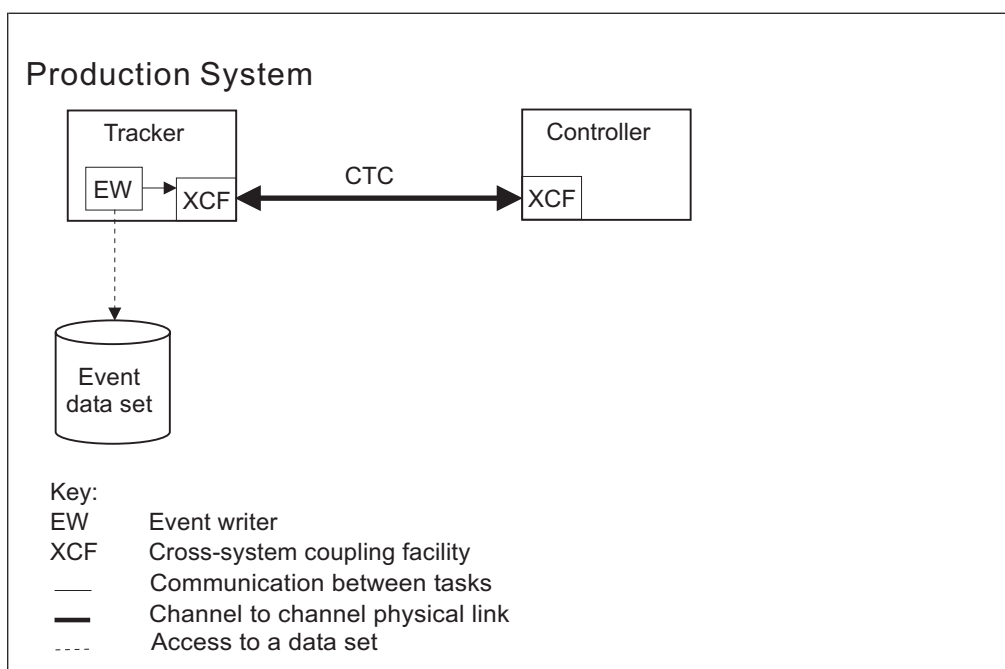


Figure 5. A z/OS system with an XCF connection

Table 7 on page 27 shows the initialization statements you can use to create the configuration in Figure 5.

Table 7. Example EQQPARM members for Figure 5

Members for the controller	Members for the tracker
<div><div>OPCECNT</div><div>OPCOPTS OPCHOST(YES) ERDRTASK(0) ROUTOPTS XCF(OPCTRK) XCFOPTS MEMBER(OPCCNT) GROUP(PLEXSYSA)</div></div>	<div><div>TRKA</div><div>OPCOPTS OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(STDEWTR) TRROPTS HOSTCON(XCF) XCFOPTS MEMBER(OPCTRK) GROUP(PLEXSYSA)</div></div> <div><div>STDEWTR</div><div>EWTROPTS EWSEQNO(01)</div></div>
<p>Note: In this example, the name of the monoplex is PLEXSYSA. The members in that group are:</p> <p>OPCCNT The controller</p> <p>OPCTRK The tracker</p> <p>The tracker member name is defined in the destination field of the workstation.</p>	

Tracker and controller in a single address space

Figure 6 on page 28 shows one Tivoli Workload Scheduler for z/OS address space performing the function of both the tracker and the controller. To optimize availability, do not use this configuration in your production environment. However, at least one of your Tivoli Workload Scheduler for z/OS test environments will probably use this configuration.

You represent this system to Tivoli Workload Scheduler for z/OS by defining a computer workstation with a blank destination field. The submit subtask performs the following actions:

- Submits JCL for batch jobs to the JES internal reader
- Writes the JCL for started tasks into the EQQSTC data set and issues START *procname* z/OS commands
- Issues JES release commands for jobs in HOLD status

The event-tracking routines create event records to describe activities that occur on the system. These records are added to the subsystem event writer queue in ECSA. The event writer subtask processes the events and:

- Adds the event to the data router queue, and the current plan is updated
- Writes the events into the event data set.

Basic configuration examples

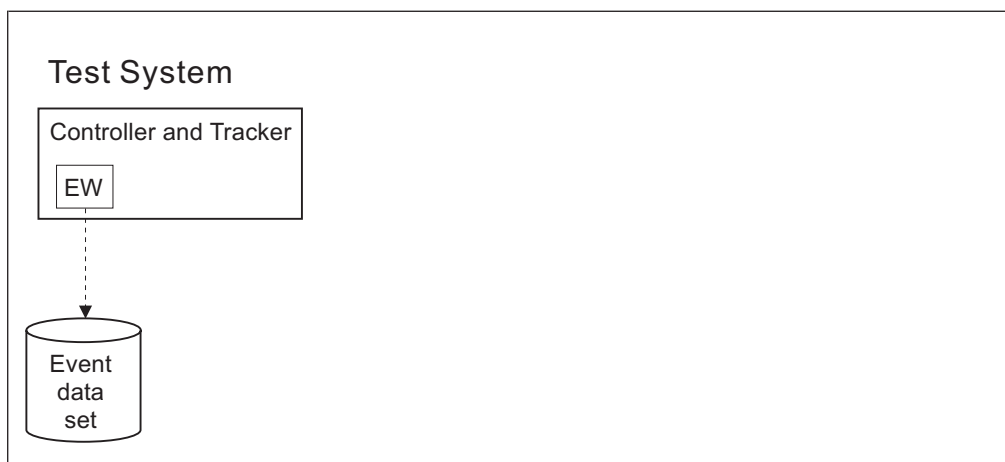


Figure 6. A tracker and controller configured in a single address space

Table 8 shows initialization statements to create the configuration in Figure 6.

Table 8. Example EQQPARM members for Figure 6

EQQPARM members for the address space	
OPCECNT	STDEWTR
OPCOPTS	OPCHOST(YES)
	ERDRTASK(0)
	EWTRTASK(YES)
	EWTRPARM(STDEWTR)
	EWTRROPTS EWSEQNO(01)

Appendix B, “Configuration examples,” on page 317 contains Tivoli Workload Scheduler for z/OS configuration examples for more complex environments.

Basic data store configuration examples

You need to install a Data Store for each spool tracked by Tivoli Workload Scheduler for z/OS in the configuration. If you have a shared spool, for example, JES2 MAS, you can have a single Data Store for multiple trackers. Two kinds of controller-Data Store connections are supported: SNA and XCF. The Data Store type must be defined either as SNA or XCF, but the same controller can connect to both XCF and SNA Data Stores. Note that you need separate LU and XCF values for controller-tracker and controller-Data Store connections. The controller is identified by two separate LU values: one for the Data Stores and one for the trackers. All Data Stores work on a reserved destination, which must always have the same name.

SNA only connection

Figure 7 on page 29 shows a JES2 with two images. In Image 1, the controller and tracker are in the same address space. Image 2 contains a tracker. The spool is not shared. Two Data Stores are required, one for Image 1 and one for Image 2. All connections are VTAM links.

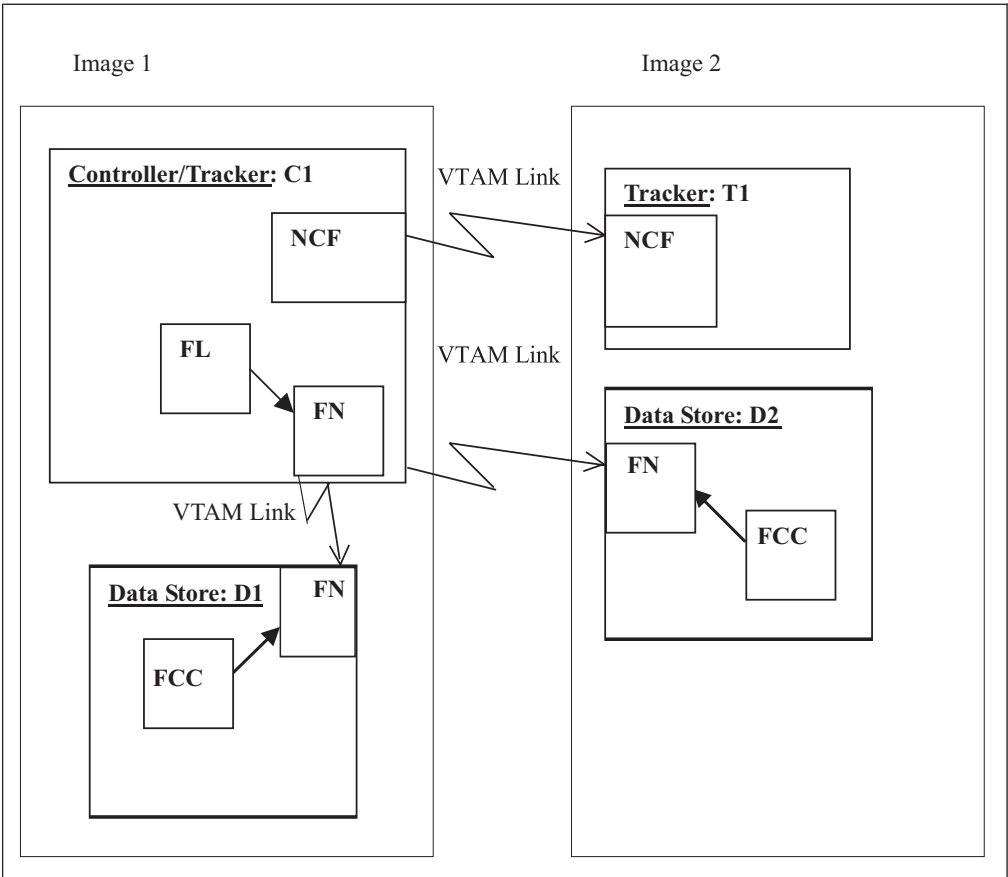


Figure 7. Controller and tracker in same address space with tracker connected through SNA

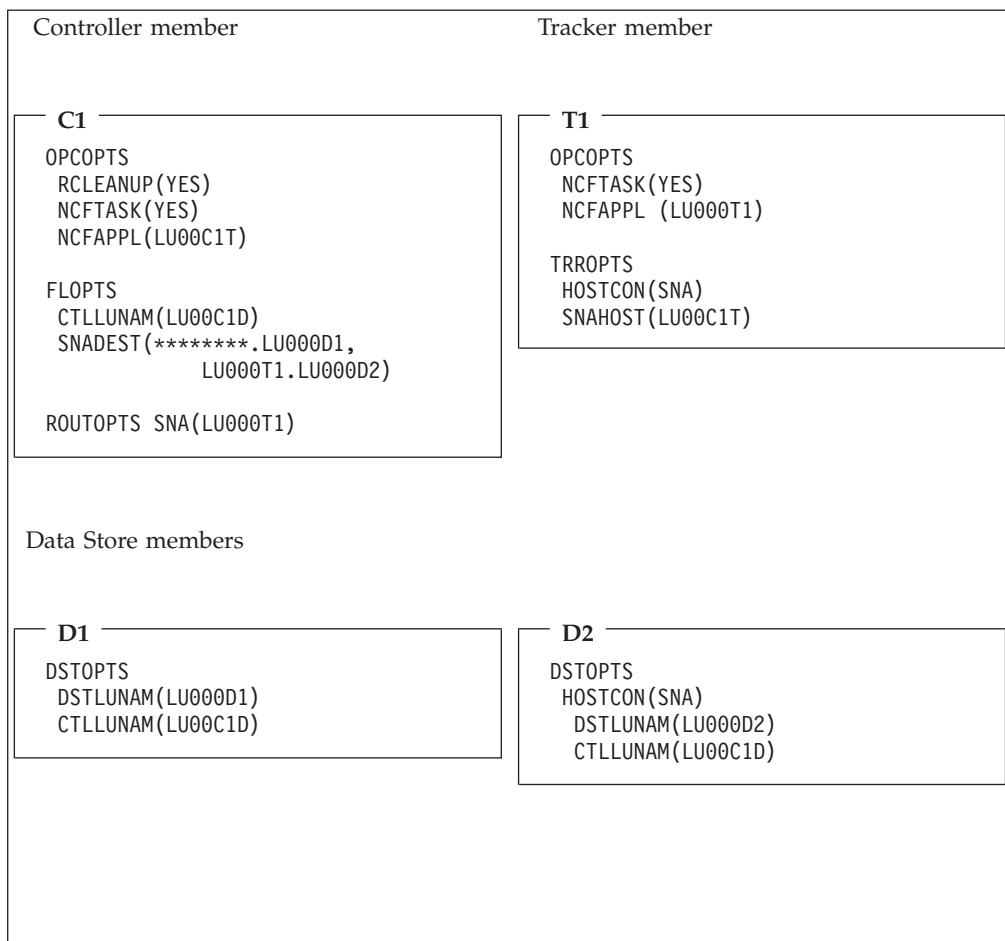
Key:

- FCC** Data Store Communication task
- FL** Fetch Job Log Task
- FN** Data Store SNA handler task
- NCF** Network Communication function

Table 9 on page 30 shows the initialization statements you can use to create the configuration in Figure 7.

Basic data store configuration examples

Table 9. Example members for Figure 7



Note: In this example, the LU names for the communication partners are the following:

LU00C1D

Controller C1, when communicating with a Data Store.

LU000D1

Data Store D1.

LU000D2

Data Store D2.

LU00C1T

Controller C1, when communicating with tracker T1.

LU000T1

Tracker T1.

XCF only connection

Figure 8 on page 31 shows a JES2 MAS (shared spool) with two images. In Image 1, the controller and a tracker are in the same address space and connected via XCF. Image 2 contains another tracker. You need only one Data Store, which is installed in Image 2. The controller will request the Job Log from the Data Store using the FL subtask.

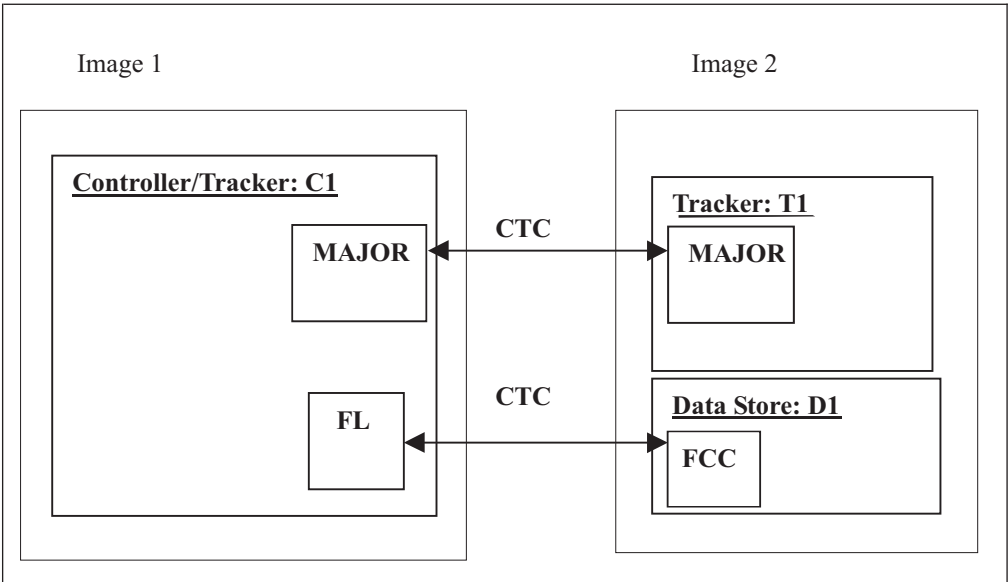


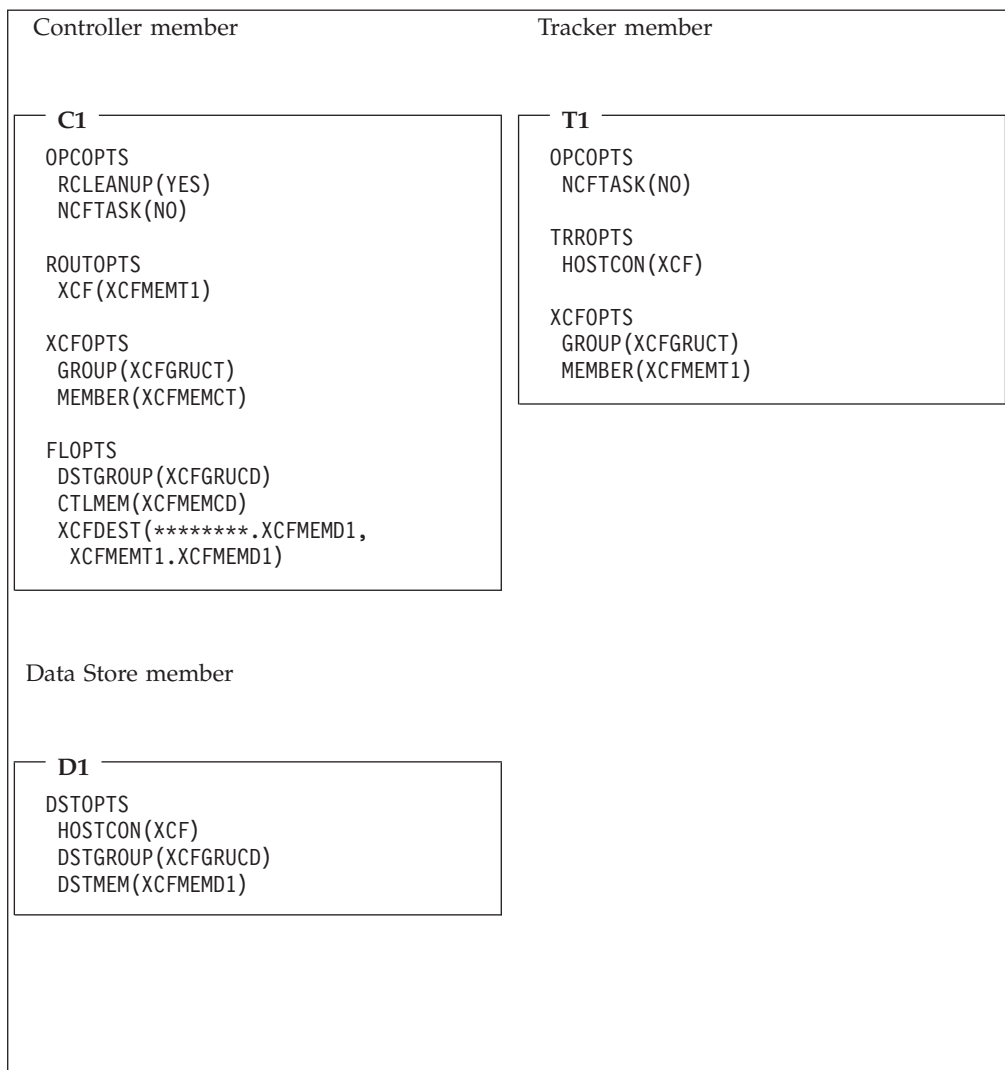
Figure 8. Controller, tracker, and Data Store connected through XCF

- Key:
- FL** Fetch Job Log task
 - FCC** Data Store Communication task
 - MAJOR**
 Controller/tracker main task

Table 10 on page 32 shows the initialization statements you can use to create the configuration in Figure 8.

Basic data store configuration examples

Table 10. Example members for Figure 8



Note: In this example, the XCF groups for the communication partners are the following:

XCFGRUCD

The XCF group for the communication between controller and Data Store. The members in the group are:

XCFMEMCD

The controller.

XCFMEMD1

The Data Store.

XCFGRUCT

The XCF group for the communication between controller and tracker. The members in the group are:

XCFMEMCT

The controller.

XCFMEMT1

The tracker.

TCP/IP only connection

Figure 9 shows a JES2 with two images. In Image 1, the controller and tracker are in the same address space. Image 2 contains a tracker. The spool is not shared. Two Data Stores are required, one for Image 1 and one for Image 2. All connections are TCP/IP links.

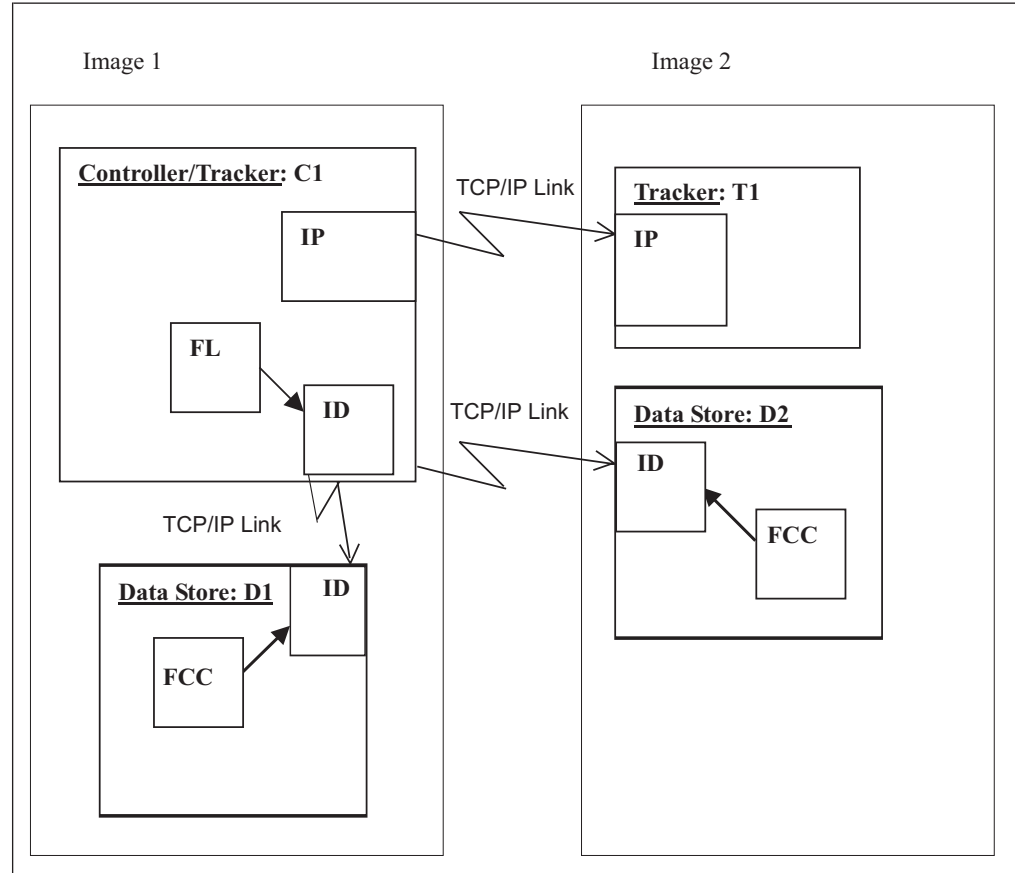


Figure 9. Controller and tracker in same address space with tracker connected through TCP/IP

Key:

- FCC** Data Store Communication task
- FL** Fetch Job Log Task
- ID** Task for Data Store-to-controller TCP/IP communication
- IP** Task for tracker-to-controller TCP/IP communication

Table 11 on page 34 shows the initialization statements you can use to create the configuration in Figure 9.

Basic data store configuration examples

Table 11. Example members for Figure 9

Controller member	Tracker member
C1 OPCOPTS RCLEANUP(YES) FLOPTS TCPDEST(*****.'9.12.134.1', '9.12.134.9') ROUTOPTS TCPIP(TRK1:'9.12.134.9')	T1 TRROPTS HOSTCON(TCP) TCPHOSTNAME('9.12.134.1')
Data Store members	
D1 DSTOPTS HOSTCON(TCP) CTLHOSTNAME('9.12.134.1')	D2 DSTOPTS HOSTCON(TCP) CTLHOSTNAME('9.12.134.1')

Note: In this example, the name of the tracker destination is TRK1. The destination name is defined also in the destination field of the workstation. The TCP/IP address of image 1 is 9.12.134.1 and the TCP/IP address of image 2 is 9.12.134.9.

Mixed SNA and XCF connection

Figure 10 shows a mixed SNA and XCF connection. In Image 1, the controller and tracker are in the same address space. In Image 2, the tracker is connected using XCF. In Image 3, the remote tracker is connected using SNA with a VTAM link. The spool is only shared between Image 1 and Image 2 (JES2 MAS). You must have two Data Stores, one installed in Image 2 and one in Image 3.

Note that the controller and tracker in Image 1 must have two different LU names. For each XCF connection, there must be a different XCF group name.

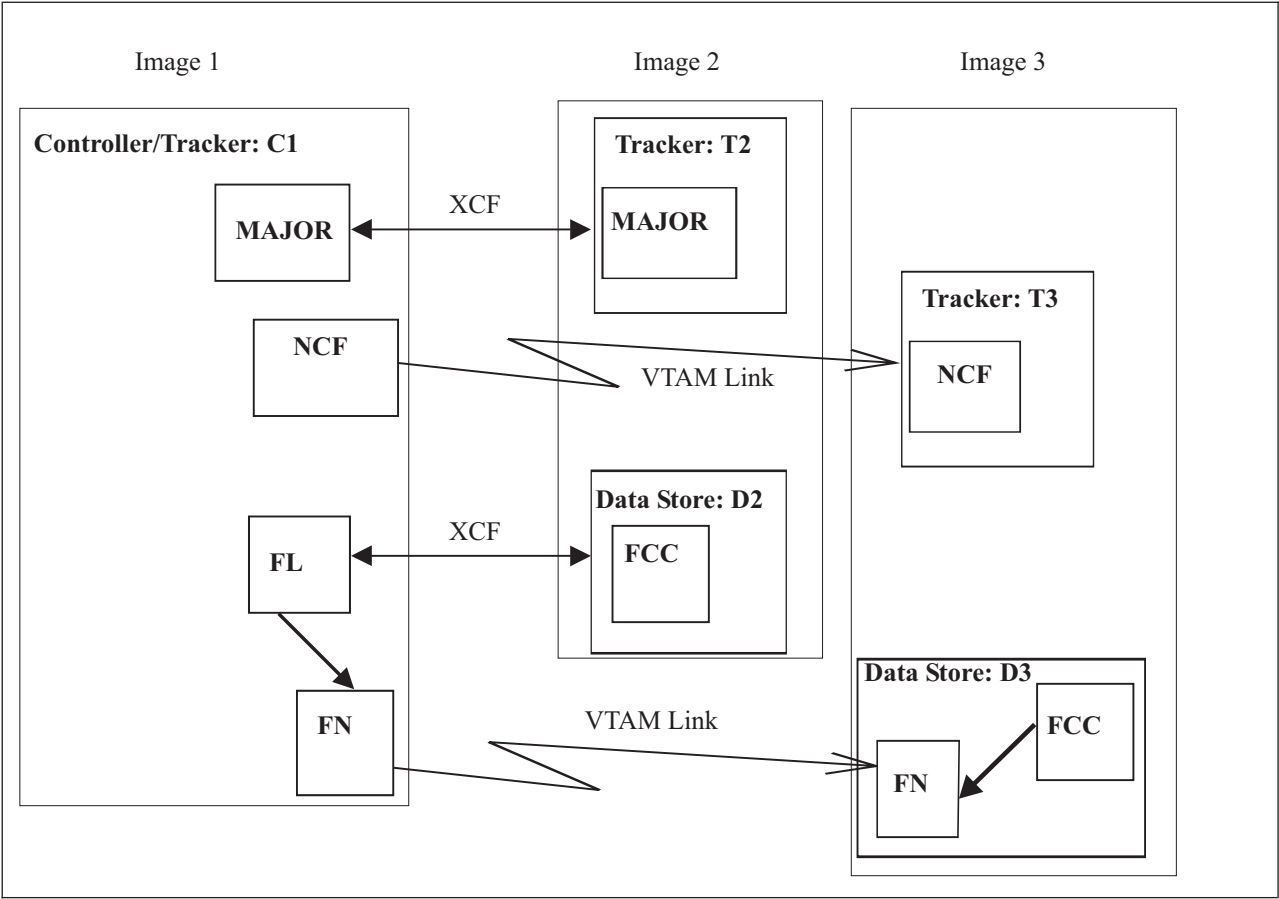


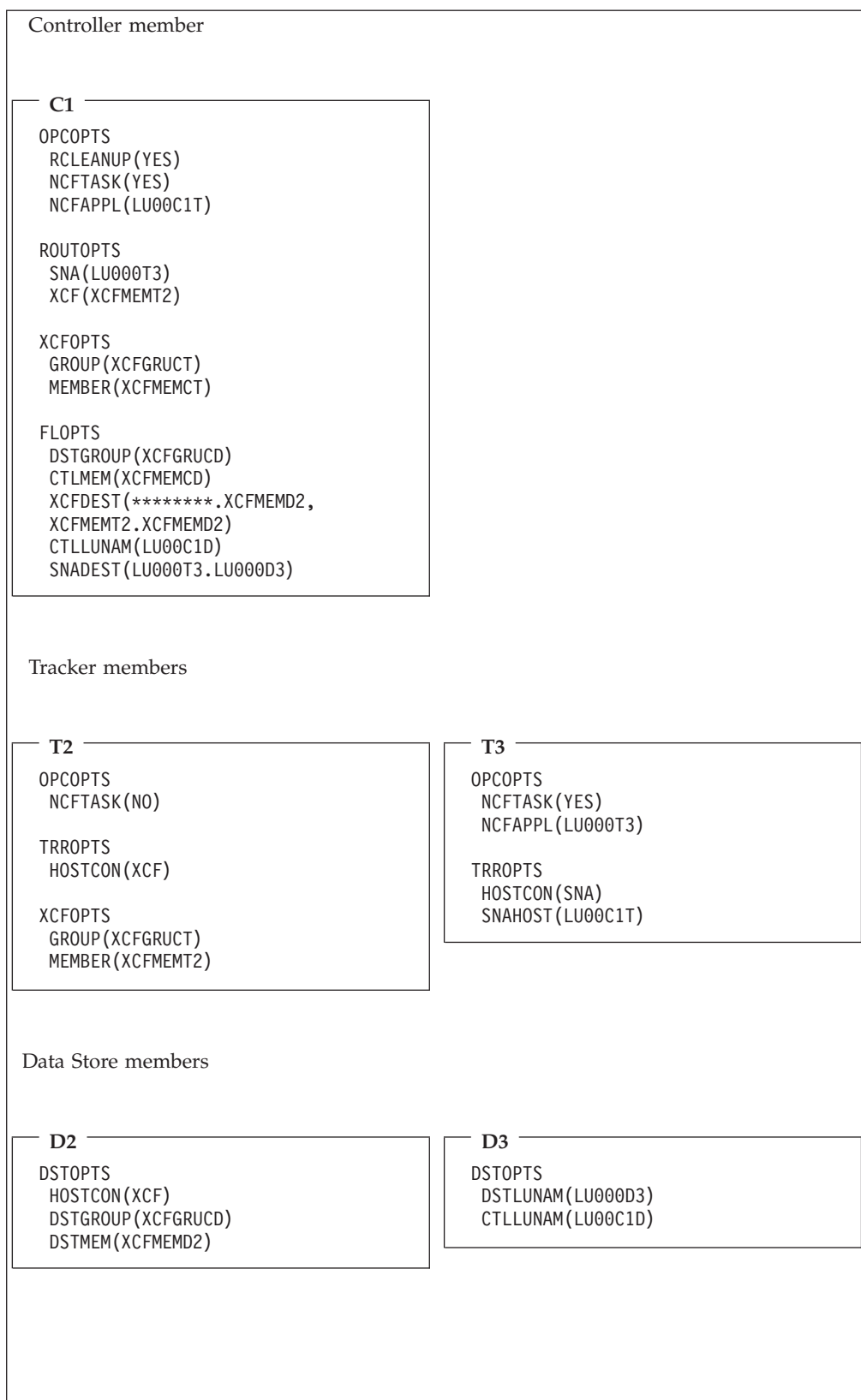
Figure 10. A mixed SNA and XCF connection

- Key:
- FCC** Data Store Communication task
 - FL** Fetch Job Log task
 - FN** Data Store SNA handler task
 - MAJOR** Controller/tracker main task
 - NCF** Network Communication function

Table 12 on page 36 shows the initialization statements you can use to create the configuration in Figure 10.

Basic data store configuration examples

Table 12. Example members for Figure 10



Note: In this example, the XCF groups or the LU names for the communication partners are the following:

XCFGRUCD

The XCF group for the communication between controller and Data Store. The members in the group are:

XCFMEMCD

Controller C1.

XCFMEMD2

Data Store D2.

XCFGRUCT

The XCF group for the communication between controller and tracker. The members in the group are:

XCFMEMCT

Controller C1.

XCFMEMT2

Tracker T2.

LU00C1D

Controller C1, when communicating with D3.

LU00D3

Data Store D3.

LU00C1T

Controller C1, when communicating with T3.

LU00T3

Tracker T3.

Basic data store configuration examples

Chapter 3. Planning your installation

This chapter offers considerations for installing Tivoli Workload Scheduler for z/OS and provides a checklist that you can use as you work through the installation process. Chapter 4, “Installing,” on page 51 describes the installation tasks in detail.

Installation considerations

During the planning stage of your Tivoli Workload Scheduler for z/OS project, consider carefully how you want to install the scheduler to control your production workload. The installation consists of installing the tracker and controller in combinations to suit your processing environment, and connecting them using one or more of the communication methods described in Chapter 2, “Planning your configuration,” on page 15. Later, you can customize your Tivoli Workload Scheduler for z/OS systems to include more functions.

Before you start the installation tasks, ensure that you have all the resources you need to complete your installation.

Configuring for availability

It is recommended that you install the tracker and controller as separate subsystems on a Tivoli Workload Scheduler for z/OS controlling system. The tracker can then continue to collect events even when the controller is stopped. Events are created by SMF and JES exits, and added to a queue in the z/OS extended common service area (ECSA). If the event writer is not active while work is running in the system, the queue might fill up, and new events might be lost. Tivoli Workload Scheduler for z/OS cannot recover these lost events.

You can improve Tivoli Workload Scheduler for z/OS availability using the z/OS Automatic Restart Manager (ARM). Automatic restart management can reduce the impact of an unexpected error on Tivoli Workload Scheduler for z/OS because it can restart it automatically, without operator intervention.

To use Automatic Restart Manager, set the ARM parameter in the OPCOPTS statement to YES. For details about the ARM parameter, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265.

Hot standby

If you connect your Tivoli Workload Scheduler for z/OS systems using the z/OS 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 z/OS system that it was active on fails. You can create a standby controller on one or more Tivoli Workload Scheduler for z/OS controlled systems within the XCF group. Each standby system must have access to the same resources as the controller. These resources include data sets and VTAM cross-domain resources. However, if EQQMLOG is allocated as a data set, it cannot be shared between the controller and standby controller. The standby system is started the same way as the other Tivoli Workload Scheduler for z/OS address spaces, but is not activated unless a failure occurs or unless it is directed to take over through a z/OS operator modify command. If you use one or more

Installation considerations

servers to remotely access the controller or to schedule using the end-to-end with fault tolerance capabilities feature, note that the server must always run on the same system as the controller.

Starting an event writer with an event reader function

In situations where a tracker does not have a DASD connection with the controller, use an event writer that is started with an event-reader function.

This can improve performance because events are not written to the event data set and then read back again, which requires an event-reader task to continually check the event data set for new events.

Instead, the event writer writes events to the event data set and forwards them directly to the controller through an XCF, NCF, or TCP/IP link.

Using a Hierarchical File System cluster

If you plan to install the end-to-end scheduling with fault tolerance capabilities feature, consider that the server starts multiple tasks and processes using the UNIX System Services (USS) on z/OS. The End-to-End server accesses USS in a Hierarchical File System cluster, that can be either HFS or zFS. For details, see Table 13 on page 40.

Checklist for installing Tivoli Workload Scheduler for z/OS

This section contains a checklist to guide you through the installation tasks for a tracker, a controller, a standby controller, or the ISPF dialogs.

Note: Always install the tracker first on the controlling system or on a system where a standby controller will be installed.

In the checklist, the task numbers are arranged in a recommended order but are not meant to imply a required order. You perform the tasks suited to your own configuration.

The **Applies to** column indicates for which Tivoli Workload Scheduler for z/OS address space you should perform that particular task. You might not need to perform every task outlined in the list. Skip those tasks or actions that do not apply to your installation.

A check mark (✓) in the **IPL** column means that an IPL of the z/OS system is needed for the change to take effect. It does not indicate how many IPLs are needed. You can install Tivoli Workload Scheduler for z/OS with only one IPL of the z/OS system by performing all the required steps before a scheduled IPL.

The **Page** column indicates the page in this guide where the task is described.

Table 13. Checklist for installing Tivoli Workload Scheduler for z/OS

Task	Description	Applies to	IPL	Page
1	<i>Load tracker software.</i> Perform these actions on each system in your Tivoli Workload Scheduler for z/OS configuration: <ul style="list-style-type: none">• Run SMP/E to receive tracker software.• Apply tracker maintenance.	Tracker		53

Table 13. Checklist for installing Tivoli Workload Scheduler for z/OS (continued)

Task	Description	Applies to	IPL	Page
2	Load controller software. Perform these actions on each system where you are installing a controller, standby controller, or dialogs: <ul style="list-style-type: none"> • Run SMP/E to receive controller software. • Apply controller maintenance. 	Controller Standby controller Dialogs		53
3	Load national language support (NLS) software for the controller. Perform these actions on each system where you are installing a controller standby controller, or dialogs: <ul style="list-style-type: none"> • Run SMP/E to receive NLS software. • Apply NLS maintenance. 	Controller Standby controller Dialogs		54
4	Run the EQQJOBS installation aid. You can run EQQJOBS as soon as the tracker software is loaded. It helps you install Tivoli Workload Scheduler for z/OS: <ul style="list-style-type: none"> • Set up EQQJOBS. • Create the sample job JCL. Do this to generate tailored samples from the EQQJOBS dialog. • Generate batch job skeletons. Use EQQJOBS to generate skeletons for the ISPF dialogs. • Optionally generate the Data Store samples if you want to install the Data Store. 	Tracker controller Standby controller Dialogs		55
5	Add SMF and JES event tracking exits. Perform this task on every z/OS system in your Tivoli Workload Scheduler for z/OS configuration. Note: If you place exits in a link-pack-area (LPA) library, you must perform an IPL of the z/OS system with the CLPA option.	Tracker	✓	72

Checklist for installing

Table 13. Checklist for installing Tivoli Workload Scheduler for z/OS (continued)

Task	Description	Applies to	IPL	Page
6	<p>Update SYS1.PARMLIB.</p> <p>On each system where you are installing the product, perform the actions that are applicable to your installation:</p> <ul style="list-style-type: none"> • Define Tivoli Workload Scheduler for z/OS subsystems (IEFSSNnn). This is required for each system where the product is installed. • Authorize the Tivoli Workload Scheduler for z/OS load module library (IEAAPFnn). Do this if you install the product in a separate load module library. • Update SMF parameters (SMFPRMnn). Do this when installing a tracker. • Update dump-content definitions. Consider this on each system where you are installing the product. • Update the z/OS link-library definition (LNKLSTnn) on each system where you are installing the product. • Update XCF initialization options (COUPLEnn). Review this section if you use XCF connections. • Modify TSO parameters (IKJTSONn). Do this when installing a controller, a standby controller, or the ISPF dialogs. • Update PPT for performance (SCHEDnn) on each system where you are installing the product. • Define the DLF exit for Hiperbatch support (COFDLFnn). Do this if you use Hiperbatch support. • Choose whether to Tivoli Workload Scheduler for z/OS automatically (COMMNDnn). Consider this on each system where you are installing the product. • Update APPC options (APPCPMnn). Consider this action if you intend to use the Tivoli Workload Scheduler for z/OS API or server. Define VTAM resources before you update SYS1.PARMLIB. Coordinate this action with task 18 or 19. 	Tracker controller Standby controller Dialogs	✓	75
7	<p>Set up the RACF environment.</p> <p>Perform these actions on each system in your Tivoli Workload Scheduler for z/OS configuration:</p> <ul style="list-style-type: none"> • Update RACF for Tivoli Workload Scheduler for z/OS started tasks (ICHRIN03) on all Tivoli Workload Scheduler for z/OS started tasks on each system. • Update RACF for a controller or standby controller. • Use functions of RACF 1.9 or later. Consider this action if you use RACF 1.9 or later. 	Tracker controller Standby controller Dialogs	✓	84
At this point, if you placed exit modules in LPA, you can IPL with CLPA. No other options for Tivoli Workload Scheduler for z/OS require an IPL.				
8	<p>Set up the SSL environment</p> <p>Perform these actions to activate a secure communication in a TCP/IP network:</p> <ul style="list-style-type: none"> • Create the SSL work directory. • Create as many private keys, certificates, and trusted certification authority (CA) chains as you plan to use in your network. • Configure the scheduler, by specifying the TCPOPTS statement for each component of your network. 	Tracker controller Standby controller Data Store server User address space		91

Table 13. Checklist for installing Tivoli Workload Scheduler for z/OS (continued)

Task	Description	Applies to	IPL	Page
9	Allocate data sets. Perform these actions if you are installing a tracker or a controller: <ul style="list-style-type: none"> • Review the section on allocating Tivoli Workload Scheduler for z/OS data sets. Do this before you allocate data sets. • Allocate VSAM data sets for a controller. Perform this action to create new data sets for a controller. • Allocate non-VSAM data sets. Perform this action for each Tivoli Workload Scheduler for z/OS address space. • Optionally allocate the VSAM Data Store data sets if you want to use the Data Store. • Optionally allocate the files and directory to use the end-to-end scheduling with fault tolerance capabilities. 	Tracker controller Data Store server		94
10	Update SYS1.PROCLIB. Perform these actions for each Tivoli Workload Scheduler for z/OS address space. <ul style="list-style-type: none"> • Create a JCL procedure for each address space on all z/OS systems where you are installing Tivoli Workload Scheduler for z/OS. • If you use Tivoli Workload Scheduler for z/OS to schedule started-task operations, ensure that the started-task-submit data set (EQQSTC) is in the JES PROCLIB concatenation and in the master scheduler start procedure. • If you use Restart and Cleanup, copy the EQQCLEAN sample procedure to a data set that is referenced in the JES PROCLIB concatenation. 	Tracker controller Standby controller		117
11	Define initialization statements. Perform this task for each Tivoli Workload Scheduler for z/OS address space: <ul style="list-style-type: none"> • Define initialization statements. Create members in the parameter library (EQQPARM) for each address space. 	Tracker controller Standby controller		122
12	Create a DB2 database. Perform this task if you need history support: <ul style="list-style-type: none"> • Update initialization statements. • Create a DB2 database. 	Controller Standby controller		122
If you are not using NCF, XCF, or TCP/IP connections you can now start a tracker and continue with the verification task.				
13	Set up the ISPF environment. Perform these actions on the system where you are installing the ISPF dialogs. <ul style="list-style-type: none"> • Set up the Tivoli Workload Scheduler for z/OS CLIST library. • Set up the ISPF tables. • Allocate ISPF and Tivoli Workload Scheduler for z/OS data sets to your TSO session. • Invoke the Tivoli Workload Scheduler for z/OS dialog. 	Dialogs		123
If you are not using NCF, XCF, or TCP/IP connections, the API or server, you can now start a controller and continue with the verification task.				

Checklist for installing

Table 13. Checklist for installing Tivoli Workload Scheduler for z/OS (continued)

Task	Description	Applies to	IPL	Page
14	<p>Using XCF for local communications.</p> <p>Even if you have already specified XCF initialization statements in Task 12 and updated the COUPLE$_{nn}$ member in Task 7, consider these actions if you use XCF:</p> <ul style="list-style-type: none"> • Update XCF initialization options. Ensure that XCF initialization options are suitable for your Tivoli Workload Scheduler for z/OS configuration. • Add initialization statement options for XCF. Specify XCF runtime options in the parameter library for all started tasks. 	Tracker controller Standby controller		129
15	<p>Activate NCF for VTAM connections.</p> <p>Perform these actions for each address space that is connected through NCF. Ensure that a standby controller has the same tracker connections as the controller and that each tracker can connect to all standby controllers:</p> <ul style="list-style-type: none"> • Add NCF network definitions. Define VTAM applications on each system where a started task uses NCF. • Add NCF session parameters on each z/OS system where Tivoli Workload Scheduler for z/OS is installed. • Update the COS table. Consider this action if you do not want to use the VTAM COS default entry. • Activate network resources. Check that all the required VTAM resources are active. • Add NCF initialization options. Include initialization statement options in the parameter library for all started tasks that use NCF. 	Tracker controller Standby controller		131
16	<p>Activate TCP/IP connections</p> <p>Perform these actions for each address space that is connected via TCP/IP. Ensure that a standby controller has the same tracker connections as the controller and that each tracker can connect to all standby controllers:</p> <ul style="list-style-type: none"> • Add TCP/IP network definitions. Define IP address for the controller and tracker. • Add TCP/IP initialization options. Include initialization statement options in the parameter library for all started tasks that use TCP/IP. • For TCP/IP, the Tivoli Workload Scheduler for z/OS server can manage up to 500 concurrent connection requests in a queue. In the PROFILE.TCPIP configuration file, set the SOMAXCONN statement to a value not greater than 500. 	Tracker controller Standby controller		134
17	<p>Activate support for the Tivoli Workload Scheduler for z/OS API.</p> <p>To use the API, perform these actions for each Tivoli Workload Scheduler for z/OS address space that you want to send requests to:</p> <ul style="list-style-type: none"> • Define VTAM resources. Define a local LU for Tivoli Workload Scheduler for z/OS, logon modes, and cross-domain resources, as required. • Update APPC options. Update the APPCPM$_{nn}$ member of SYS1.PARMLIB. • Activate Tivoli Workload Scheduler for z/OS support for APPC. In the parameter library, include APPCTASK(YES) on the OPCOPTS statement. 	Tracker controller Standby controller		134

Table 13. Checklist for installing Tivoli Workload Scheduler for z/OS (continued)

Task	Description	Applies to	IPL	Page
18	<p>Activate support for the Tivoli Workload Scheduler for z/OS Server to use APPC or TCP/IP communications.</p> <p>Note: Include this task when activating an Tivoli Workload Scheduler for z/OS server.</p> <p>To activate the server, perform these actions in the order shown:</p> <ol style="list-style-type: none"> 1. Define VTAM resources. Define a local LU for Tivoli Workload Scheduler for z/OS, logon modes, and cross-domain resources, as required. 2. Update APPC options. Update the APPCPMnn member of SYS1.PARMLIB. 3. Activate Tivoli Workload Scheduler for z/OS support for APPC. In the parameter library, include SERVERS on the OPCOPTS statement. 	Server controller Standby controller		137
19	<p>Activate support for end-to-end scheduling with fault tolerance capabilities</p> <p>Note: Include this task when you intend to use IBM Tivoli Workload Scheduler for z/OS to schedule jobs on distributed fault-tolerant agents.</p> <ul style="list-style-type: none"> • Ensure that you have loaded the fault-tolerant end-to-end enabler software on the system where you have installed the controller. • Verify that all the VSAM and non-VSAM data sets and the files used for the end-to-end scheduling with fault tolerance capabilities have been allocated (for details, see the task that describes how to allocate data sets). • To activate the server, include TPLGYPRM on the SERVOPTS statement in the IBM Tivoli Workload Scheduler for z/OS parameter library. • To activate the controller, include TPLGYSRV on the OPCOPTS statement in the IBM Tivoli Workload Scheduler for z/OS parameter library. • To activate the Daily Planning batch jobs, include TPLGYPRM in the BATCHOPTS statement in the IBM Tivoli Workload Scheduler for z/OS parameter library. 	Controller Standby controller server		141
20	<p>Activate support for end-to-end scheduling with z-centric capabilities</p> <p>Note: Include this task when you intend to use IBM Tivoli Workload Scheduler for z/OS to schedule jobs on distributed z-centric agents.</p> <ul style="list-style-type: none"> • Define the ROUTOPTS initialization parameters for the controller. • Define the HTTPOPTS initialization parameters for the tracker. 	Tracker controller Standby controller		142
21	<p>Activate Support for the Dynamic Workload Console</p> <p>Note: Include this task when activating a server and intending to use the Dynamic Workload Console.</p> <p>To activate the server, perform these actions:</p> <ul style="list-style-type: none"> • Install the Connector • In the parameter library, include SERVERS on the OPCOPTS statement. 	Server controller Standby controller		142

Checklist for installing

Table 13. Checklist for installing Tivoli Workload Scheduler for z/OS (continued)

Task	Description	Applies to	IPL	Page
22	Activate Support for the Java utilities Perform this task if you want to use one of the following features: <ul style="list-style-type: none">• Dynamic Workload Console reporting.• Event-driven workload automation for data set triggering, with centralized deploy process.	Controller		144
23	Verify your installation of Tivoli Workload Scheduler for z/OS In the Tivoli Workload Scheduler for z/OS address space, verify the following installations: <ul style="list-style-type: none">• Tracker installation.• Controller installation.• Standby controller installation.• End-to-end installation.• Data Store installation.• Server installation. When a current plan has been created, verify your Tivoli Workload Scheduler for z/OS configuration.	Tracker controller Standby controller		145

Part 2. Tivoli Workload Scheduler for z/OS

Chapter 4. Installing	51
Step 1. Loading tracker software	53
Step 2. Loading controller software	53
Step 3. Loading national language support software	54
Step 4. Using the EQQJOBS installation aid.	54
Setting up the EQQJOBS installation aid.	55
Creating the sample job JCL.	55
Generating batch-job skeletons	63
Generating Data Store samples	68
Step 5. Adding SMF and JES exits for event tracking	72
SMF only	73
JES2 only	74
JES3 only	74
Step 6. Updating SYS1.PARMLIB	75
Defining subsystems	75
Calculating MAXECSA values	76
Authorizing the load-module library	77
Updating SMF parameters	77
Updating z/OS dump options	79
Updating the z/OS link-library definition	80
Updating XCF initialization options	80
Modifying TSO parameters	81
Performance considerations	82
Defining the DLF exit for Hiperbatch support	82
Starting the product automatically.	83
Updating APPC options	83
Implementing support for data set triggering	83
Step 7. Setting up the RACF environment	84
Controlling the user ID of the address space	84
Controlling the user ID of submitted jobs	84
Normal production jobs	85
Stand-alone cleanup jobs	85
Dialog jobs	85
Protecting data sets.	86
Controlling access to resources	86
Permitting access to the controller through the API	87
Controlling access to Tivoli Workload Scheduler for z/OS resources when using the Dynamic Workload Console	87
Permitting access to the controller through the Dynamic Workload Console	88
Authorizing Tivoli Workload Scheduler for z/OS as a job submitter	88
Authorizing Tivoli Workload Scheduler for z/OS to issue JES commands	89
Authorizing Tivoli Workload Scheduler for z/OS E2E server task to create USS processes	90
Authorizing Tivoli Workload Scheduler for z/OS E2E and Dynamic Workload Console server tasks for security resource EZB.BINDDVIPARANGE	90
Authorizing Tivoli Workload Scheduler for z/OS Data Store to issue JES commands.	90
Step 8. Securing communications	91
Security for TCP/IP connections	91
Security for HTTP connections	93
Step 9. Allocating data sets	94
Allocating the VSAM data sets	95
Application description data set (EQQADDS)	98
Current plan data sets (EQQCPnDS)	98
Data sets for extended data (EQQXDnDS)	99
Current plan backup copy data set (EQQSCPDS)	99
JCL repository data sets (EQQJSnDS)	99
Operator Instruction data set (EQQOIDS)	99
Allocating Restart and Cleanup VSAM data sets	100
Restart and cleanup data sets (EQQPKIxx, EQQSKIxx, and EQQSDFxx)	100
Allocating non-VSAM data sets	100
Internal reader data set (EQQBRDS)	103
Checkpoint data set (EQQCKPT)	103
Diagnostic data sets (EQQDMSG, EQQDUMP, and SYSMDUMP)	103
Diagnostic message and trace data set (EQQDMSG)	103
Diagnostic data set (EQQDUMP)	103
Dump data set (SYSMDUMP)	103
Event data sets (EQQEVDs, EQQEVDnn, and EQQHTTP0)	104
Event-driven workload automation configuration file data set (EQQEVLIB)	106
Job library data set (EQQJBLIB)	106
Job-completion-checker data sets	106
JCC-message-table library (EQQJCLIB)	106
JCC-incident-log data set	106
JCC-incident work data set (EQQINCWK)	106
Job-tracking data sets (EQQJTARC, EQQJTnn, EQQDLnn)	107
Message log data set (EQQMLOG)	108
Loop analysis log data set (EQQLOOP)	109
Parameter library (EQQPARM)	109
PIF parameter data set (EQQYPARM)	109
Automatic-recovery-procedure library (EQQPRLIB)	110
Script library for end-to-end scheduling with fault tolerance capabilities (EQQSCLIB)	110
Started-task-submit data set (EQQSTC)	110
Submit/release data set (EQQSUDS)	110
Centralized script data set for end-to-end scheduling with fault tolerance capabilities (EQQTWSCS)	111
Input and output events data sets for end-to-end scheduling with fault tolerance capabilities (EQQTWSIN and EQQTWSOU)	111
Allocating Data Store data sets	112
Allocating data sets for the Dynamic Workload Console reporting feature	113
Allocating the files and directories	113
Configuring for end-to-end scheduling with fault tolerance capabilities in a SYSPLEX environment.	115

Step 10. Creating JCL procedures for address spaces	116
Implementing support for started-task operations	117
Required data sets	118
Optional data sets	120
Step 11. Defining the initialization statements	122
Step 12. Creating the DB2 database	122
Sample to migrate the history database	122
Step 13. Setting up the ISPF environment	123
Setting up the CLIST library	124
Setting up the ISPF tables	124
Setting up the default dialog-controller connection table	124
Setting up list tables and graphical attribute tables	126
Allocating dialog data sets to your TSO session	126
Invoking the Tivoli Workload Scheduler for z/OS dialog	127
Using the EQQOPCAC sample CLIST	127
Modifying an existing ISPF selection menu	128
Selecting the main menu directly from TSO	128
Using the ISPF select service	128
Switching to the advanced style for ISPF panels	128
Step 14. Using XCF for communication	129
XCF groups	129
XCF runtime options	130
Initialization statements used for XCF	130
Step 15. Activating the network communication function	131
Adding NCF to the VTAM network definitions	131
Adding NCF session parameters	132
COS table	133
Activating network resources	133
Diagnostic data set	134
Step 16. Using TCP/IP for communication	134
Initialization statements used for TCP/IP	134
Step 17. Activating support for the API	134
Defining VTAM resources	135
Defining a local LU	135
Defining logon modes	135
Defining cross-domain resources	136
Updating APPC options	137
Activating support for APPC	137
Step 18. Activating support for the product dialog and programming interface using the server	137
Defining VTAM resources for the product dialog and program interface using the server	138
Defining VTAM resources for the server	139
Defining a local LU for the server	139
Defining logon modes for the server	139
Updating APPC options for the server	140
Defining VTAM resources in a parallel sysplex	140
Starting the server	141
Step 19. Activating support for the end-to-end scheduling with fault tolerance capabilities	141
Activating server support for the end-to-end scheduling with fault tolerance capabilities	142
Step 20. Activating support for the end-to-end scheduling with z-centric capabilities	142

Step 21. Activating support for Dynamic Workload Console	142
Prerequisites	143
Considerations	143
Activating server support for the Dynamic Workload Console	143
Step 22. Activating support for the Java utilities	144

Chapter 5. Verifying your installation	145
Overview of verification	145
Verifying installation of a tracker	145
Ensuring that all installation tasks are complete	146
Checking the message log (EQQMLOG)	146
Verifying tracking events	147
The event writer	147
The event data set	147
Performing problem determination for tracking events	148
Verifying installation of a controller and dialogs	151
Ensuring that all installation tasks are complete	151
Checking the message log (EQQMLOG)	152
Checking the server message log	152
Checking dialog functions	153
Performing problem determination	153
Dialog problems	153
Authority problems	154
Verifying installation of a standby controller	154
Ensuring that all installation tasks are complete	155
Checking the message log (EQQMLOG)	155
Verifying installation of the Restart and Cleanup function	156
Checking the message log (EQQMLOG)	156
Verifying configuration	158
Creating entries in the databases	158
Running batch jobs	158
Checking the message logs (EQQMLOG)	158
Controller message log	158
Tracker message log	165
Verifying workload submission	168
Controlling system	168
Controlled systems	168
Verifying job submission	169
Verifying takeover by a standby controller	170

Chapter 6. Migrating	171
Planning for migration	171
Migration considerations	171
Customization considerations	173
Migration strategies	173
JES and SMF exits	173
Migrating to existing subsystem definitions	173
Migrating to new subsystem definitions	174
Getting the right software parts	174
Load modules	174
The ISPF environment	174
Migration overview	175
Migration steps overview	176
Establishing the required environment	176
Program requirements	176
Installation and verification	176
Parallel testing	177

Migrating an end-to-end network	178
Migrating DB2	178
Changing a shared DASD tracker-to-controller connection to an NCF, XCF, or TCP/IP connection	178
Running on upgraded operating systems	180
Migrating actions	180
Migrating data sets	181
EQQICTOP VSAM data set conversion program	181
Data sets that you need to convert	183
Data sets that can be used	184
Empty data sets	184
Tracker and Data Store considerations	185
Switching into production mode	186
Closing down your production system	186
Converting VSAM files to the new system format.	187
Starting the new system	187
Validating the new system	189
Migration steps for a system in a heavy workload environment	189
Close down your production system	189
Convert VSAM files to the new system format.	189
Initialize the new system	189
Produce a checkpoint data set containing data from the old production system	189
Start the new system	189
Validate the new system.	189
Performing fallback	192

Chapter 4. Installing

This chapter describes the tasks you perform to install Tivoli Workload Scheduler for z/OS. Before you begin these tasks, you must determine the Tivoli Workload Scheduler for z/OS configuration you want to create and the functions you want to use. See Chapter 2, “Planning your configuration,” on page 15. Table 14 summarizes the installation tasks and identifies the address space type for each task. Depending on your configuration, you might not need to perform every task indicated in the table. Skip those sections that are not relevant to your installation.

Table 14. Tivoli Workload Scheduler for z/OS installation tasks

Installation task	Perform for						Page
	Tracker	Controller	Server	Standby controller	Data Store	Dialogs	
Step 1. Loading tracker software	✓	✓		✓	✓	✓	53
Step 2. Loading controller software		✓		✓		✓	53
Step 3. Loading national language support software		✓		✓		✓	54
Step 4. Using the EQQJOBS installation aid		✓			✓	✓	54
Step 5. Adding SMF and JES exits for event tracking	✓						72
Step 6. Updating SYS1.PARMLIB	✓	✓		✓		✓	75
Step 7. Setting up the RACF environment	✓	✓		✓			84
Step 8. Securing communications	✓	✓	✓	✓	✓	✓	91
Step 9. Allocating data sets				✓	✓		94
• Allocating the VSAM data sets		✓					95
• Allocating non-VSAM data sets	✓	✓					100
• Allocating the files and directories			✓				113
Step 10. Creating JCL procedures for address spaces	✓	✓		✓	✓		116
Step 11. Defining the initialization statements	✓	✓		✓	✓		122
Step 12. Creating the DB2 database		✓		✓			122
Step 13. Setting up the ISPF environment						✓	123

Table 14. Tivoli Workload Scheduler for z/OS installation tasks (continued)

Installation task	Perform for						Page
	Tracker	Controller	Server	Standby controller	Data Store	Dialogs	
Step 14. Using XCF for communication	✓	✓		✓	✓		129
Step 15. Activating the network communication function	✓	✓		✓	✓		131
Step 16. Using TCP/IP for communication	✓	✓		✓			134
Step 17. Activating support for the API		✓		✓			134
Step 18. Activating support for the product dialog and programming interface using the server		✓		✓			137
Step 19. Activating support for the end-to-end scheduling with fault tolerance capabilities		✓	✓	✓			141
Step 20. Activating support for the end-to-end scheduling with z-centric capabilities	✓	✓		✓			142
Step 21. Activating support for Dynamic Workload Console		✓		✓			142
Step 22. Activating support for the Java utilities		✓					144
Step 23. Verifying Tivoli Workload Scheduler for z/OS installation							
• Verifying installation of a tracker	✓						145
• Verifying installation of a Controller		✓					151
• Verifying installation of a standby controller				✓			154
• Verifying installation of the Restart and Cleanup function		✓		✓	✓		156
• Verifying configuration	✓	✓		✓			158

Step 1. Loading tracker software

You must load tracker software on each z/OS system in your configuration. Process the software distribution tape using the facilities of System Modification Program Extended (SMP/E). This creates or updates the necessary software libraries on your system. Table 15 describes the distribution and target libraries that are created or updated by SMP/E.

Table 15. Tracker libraries loaded by SMP/E

SMP/E DD name		Description
Distribution	Target	
AEQQPNL0	SEQQPNL0	Panels for the EQQJOBS installation aid
AEQQMOD0 (object)	SEQQLMD0 (load)	Tracker modules
AEQQMSG0	SEQQMSG0	Messages
AEQQMAC0	SEQQMAC0	Assembler macros
AEQQCLIB	SEQQCLIB	Tivoli Workload Scheduler for z/OS CLISTs
AEQQSAMP	SEQQSAMP	Sample exits, programs, and JCL
AEQQSKL0	SEQQSKL0	JCL skeletons, input to EQQJOBS
AEQQTBL0	SEQQTBL0	ISPF tables
AEQQDATA	SEQQDATA	Sample Tivoli Workload Scheduler for z/OS databases
AEQQMISC	SEQQMISC	DBRM files

It is recommended that you place all the Tivoli Workload Scheduler for z/OS load modules in a separate library. Use the same library for both the tracker and the controller. Create the library before you run the SMP/E jobs.

Alternatively, you can place the Tivoli Workload Scheduler for z/OS load modules in one of your existing load-module libraries, for example SYS1.LINKLIB. The remaining data sets loaded by SMP/E are new data sets that you must create before running the SMP/E jobs. The *Tivoli Workload Scheduler for z/OS: Program Directory*, GI11-4248 contains the JCL and instructions for loading the software.

When you have loaded the tracker software, apply any recommended maintenance described in the PSP bucket.

Step 2. Loading controller software

To load controller software, process the software distribution tape using SMP/E. This creates or updates the necessary disk-resident libraries on your system. Table 16 describes the data set that is created or updated by SMP/E.

Table 16. Controller libraries loaded by SMP/E

SMP/E DD name		Description
Distribution	Target	
AEQQMOD0 (object)	SEQQLMD0 (load)	Controller modules

Step 2. Loading controller software

Table 16. Controller libraries loaded by SMP/E (continued)

SMP/E DD name		Description
Distribution	Target	
AEQQMISC	SEQQMISC	Batch command interface tool modules and Control Language tool modules

Recommendation: Install the controller in the same library that you are using for the tracker.

When you have loaded the controller software, apply any recommended maintenance described in the PSP bucket.

Step 3. Loading national language support software

To load national language support (NLS) software, process the software distribution tape using SMP/E. This creates or updates the necessary disk-resident libraries on your system. Table 17 describes the data sets that are created or updated by SMP/E.

Table 17. NLS libraries loaded by SMP/E

SMP/E DD name		Description
Distribution	Target	
AEQQPxxx	SEQQPxxx	Panels
AEQQMxxx	SEQQMxxx	Messages
AEQQQLxxx	SEQQLxxx	Advanced ISPF panel templates
AEQQGxxx	SEQQGxxx	Advanced ISPF panels
Note: The suffix xxx is the NLS identifier. It is recommended that you place NLS software in the same library that you are using for the tracker and controller.		

When you have loaded the national language support software, apply any recommended maintenance described in the PSP bucket.

Step 4. Using the EQQJOBS installation aid

EQQJOBS is a CLIST-driven ISPF dialog that can help you install Tivoli Workload Scheduler for z/OS. You can set up EQQJOBS as soon as the tracker software has been installed. EQQJOBS assists in the installation of the tracker and controller by building batch-job JCL that is tailored to your requirements. You can use this JCL to perform the following actions:

- Install the tracking exits
- Set up RACF security
- Create data sets
- Create started-task JCL
- Perform planning functions

Set up EQQJOBS now so that it is ready to use when needed.

Setting up the EQQJOBS installation aid

EQQJOBS reads skeleton JCL from the SEQQSAMP or SEQQSKL0 libraries, tailors the JCL, and then writes the tailored JCL to an output library that you specify. The output library must be a partitioned data set with record length 80 and record format FB, and must be allocated before you run EQQJOBS. The components of EQQJOBS reside in these libraries:

SEQQCLIB	CLIST to drive the dialog
SEQQPNL0	EQQJOBS panels
SEQQSAMP	Sample JCL
SEQQSKL0	Tivoli Workload Scheduler for z/OS batch-job skeletons.

To be able to run EQQJOBS, allocate these libraries to the DD statements in your TSO session:

- SEQQCLIB to SYSPROC
- SEQQPNL0 to ISPLIB
- SEQQSKL0 and SEQQSAMP to ISPSLIB.

To invoke EQQJOBS, enter the TSO command EQQJOBS from an ISPF environment. This panel is displayed:

```

EQQJOBS0 ----- EQQJOBS APPLICATION MENU -----
Select option ==>

1 - Create sample job JCL
2 - Generate batch-job skeletons
3 - Generate Data Store samples
X - Exit from the EQQJOBS dialog

Userid   - SYSPROC
Time     - 15:04
Terminal - 3278
  
```

Figure 11. EQQJOBS0 - EQQJOBS application menu

The following sections describe:

- Option 1, “Creating the sample job JCL”
- Option 2, “Generating batch-job skeletons” on page 63
- Option 3, “Generating Data Store samples” on page 68

Note: To ensure that all files are correctly allocated, perform first option 2, followed by option 1.

Creating the sample job JCL

To ensure that all files are correctly allocated, before creating the sample job JCL you must have generated the batch-job skeletons as described in “Generating batch-job skeletons” on page 63.

To create the sample job JCL:

1. Select option 1 from the EQQJOBS application menu. This panel is displayed:

Step 4. Using the EQQJOBS installation aid

```
EQQJOBS3 ----- CREATE SAMPLE JOB JCL -----
Command ==>

The data set names specified on this panel should be fully qualified
names without any enclosing apostrophes.

Enter the name of the output library:

Sample job JCL      ==> CCOPC.OPCA.INSTJCL_____

Job statement information:

==> //SYSPROG1 JOB (111111,2222),'OPCESA BATCH',CLASS=B,MSGCLASS=H,_____
==> //          MSGLEVEL=(1,1),NOTIFY=SYSPROG_____
==> //*_____
==> //*_____

The following data set names are used by one or more of the generated job

Message library name ==> OPC.SEQMSG0_____
Data library name    ==> TWSDEV.DATA_____
Parameter library     ==> CCOPC.OPCA.PARM_____
Checkpoint data set   ==> CCOPC.OPCA.CKPT_____
Press ENTER to continue
```

Figure 12. EQQJOBS3 - Create sample job JCL

The data set names that you specify on this panel must be fully-qualified and not be enclosed within apostrophes.

Sample job JCL

Required input. Enter the name of a library where you want the generated JCL samples written to. The library must be allocated before you generate the batch JCL samples. Ensure that the library that you specify has sufficient directory blocks to store all the sample members that are generated by EQQJOBS (see Table 18 on page 61.)

Job statement information

Required input. Enter a JOB statement that follows standard JCL syntax and your installation standards.

Message library name

Required input. Enter the name of the library that contains the Tivoli Workload Scheduler for z/OS messages (SMP/E target DDNAME SEQMSG0).

Data library name

Required input. Enter the name of the library that will contain the SSL certificates that are provided with Tivoli Workload Scheduler for z/OS. For detailed information about these certificates, see the *Tivoli Workload Scheduler for z/OS: Scheduling End-to-end with z-centric Capabilities* manual.

Parameter library

Required input. Enter the name of a library that will contain the initialization statements. This library will be allocated by the EQQPCS01 batch job.

Checkpoint data set

Required input. Enter the name of the checkpoint data set. This library will also be allocated by the EQQPCS01 batch job.

2. Press Enter, and this panel is displayed:

```

EQQJOBS4 ----- CREATE SAMPLE JOB JCL -----
Command ==>

Enter the following required job stream parameters:

non-VSAM dsn prefix  ==> CCOPC.OPCA
VSAM dsn prefix      ==> CCOPC.OPCAV
Unit name            ==> 3390           Default unit name
Primary volume serial ==> PROD01       Primary volume serial for VSAM
Backup volume serial  ==> PROD02       Secondary volume serial for VSAM
SYSOUT class          ==> *            SYSOUT class for reports

The following information is optional:

STEPLIB dsname       ==> OPC.SEQQLMD0
VSAMCAT dsname        ==>
VSAM password         ==>
Dsn prefix of old     ==>
VSAM files            ==> CCOPC.OPCAV
non-VSAM files        ==> CCOPC.OPCA
Samples with cloning support generated:  N  (Y/N)

Static symbol used    ==> SYSCLONE without enclosing '&' and period

Press ENTER to continue

```

Figure 13. EQQJOBS4 - Create sample job JCL

non-VSAM dsn prefix

Required input. Enter the qualifiers that prefix the non-VSAM data set names. Tivoli Workload Scheduler for z/OS adds a low-level qualifier to the prefix to uniquely identify the non-VSAM data sets. For example, it adds EV for the event data set. In this example, the full name is CCOPC.OPCA.EV.

Note: Tivoli Workload Scheduler for z/OS does not use the prefix for the parameter library or checkpoint data set. You entered the names of these data sets, fully qualified, on the previous panel.

VSAM dsn prefix

Required input. Enter the qualifiers that prefix the VSAM data set names. Tivoli Workload Scheduler for z/OS adds a low-level qualifier to the prefix to uniquely identify each Tivoli Workload Scheduler for z/OS VSAM data set. For example, it adds AD for the application description data set. In this example, the full name is CCOPC.OPCAV.AD .

Unit name

Required input. Enter a device name that is valid at your installation. This could be a device type, for example 3380, or a group name, for example PROD or TEST.

Primary volume serial

Required input. Enter a volume to be used by sample job EQQPCS01 to allocate the primary data sets. Some Tivoli Workload Scheduler for z/OS logical files are implemented as two physical data sets, a primary and an alternate; for example, the current plan data set. To minimize the potential impact of errors on a particular device, allocate the primary and alternate data sets on different physical devices.

SYSOUT Class

Required input. Enter the SYSOUT class that you want to use for the reports that are generated by the sample jobs.

Step 4. Using the EQQJOBS installation aid

STEPLIB dsname

Optional. Enter the name of the Tivoli Workload Scheduler for z/OS load module library if the load modules are not in a data set included in an active LNKLST member.

VSAMCAT dsname

Optional. Enter the name of a catalog in which VSAM data sets are to be defined if they are not to be defined in the master catalog.

VSAM password

Optional. Enter the password for the VSAM catalog if the catalog is password-protected.

VSAM files

Optional. Enter the qualifiers that prefix your existing Tivoli Workload Scheduler for z/OS VSAM data set names. These are used to create the data set-conversion sample JCL.

non-VSAM files

Optional. Enter the qualifiers that prefix your existing Tivoli Workload Scheduler for z/OS non-VSAM data set names. These are used to create the data set-conversion sample JCL.

Samples with cloning support generated

Optional. Enter Y if you want the SYSCLONE variable resolved.

Notes:

- a. Generated JCLs do not contain a period before &SYSCLONE.
- b. &SYSCLONE variable is intended to be substituted in the scheduler started tasks. It is not substituted in the generated JCLs that run as batch jobs. To obtain the variable substitution, run the JCL as cataloged procedure.

3. Press Enter. The following panel is displayed:

```
EQQJOBS8 ----- CREATE SAMPLE JOB JCL -----
Command ==>

END-TO-END WITH FAULT TOLERANCE:  Y      (Y= Yes ,N= No)
Installation Directory             ==> /usr/lpp/TWS/V8R5M0_____
                                   ==> _____
                                   ==> _____
Work Directory                    ==> /var/TWS/inst_____
                                   ==> _____
                                   ==> _____
User for OPC address space        ==> UID_____
Refresh CP group                  ==> GID_____
RESTART AND CLEANUP (Data Store)  Y      (Y= Yes ,N= No)
Reserved destination              ==> OPCX_____
Connection type                   ==> SNA      (SNA/XCF/TCP)
SNA Data Store luname             ==> I9PC45AA (only for SNA connection )
SNA FN task luname                ==> I9PC45RA (only for SNA connection )
Xcf Group                        ==> _____ (only for XCF connection )
Xcf Data Store member             ==> _____ (only for XCF connection )
Xcf FL task member                ==> _____ (only for XCF connection )
TCP Data store host name:         ==> _____ (only for TCP connection )
                                   ==> _____
TCP Data store port number        ==> _____ (only for TCP connection )
Press ENTER to continue
```

Figure 14. EQQJOBS8 - Create sample job JCL

END-TO-END WITH FAULT TOLERANCE

Specify Y if you want to schedule jobs on fault-tolerant agent workstations.

Step 4. Using the EQQJOBS installation aid

Only the Server requires a UID and a GID. Set the UID to a nonzero value, unless you plan to run the EQQPCS05 sample JCL.

Installation Directory

Specify the path where SMP/E has installed the Tivoli Workload Scheduler for z/OS files for UNIX system services that apply the end-to-end enabler feature. This directory is the one containing the bin directory. The default path is /usr/lpp/TWS/VvRrMm.

Work Directory

Specify where the subsystem-specific files are. Replace */inst* with a name that uniquely identifies your subsystem. Each subsystem that will use the fault-tolerant workstations must have its own work directory. Only the server and the daily planning batch jobs go in the work directory. See “Allocating the files and directories” on page 113.

User for OPC address space

This information is used to create the procedure to build the directory with the right ownership. To run the end-to-end scheduling with fault tolerance capabilities correctly, the ownership of the work directory, and the files it contains, must be assigned to the same user ID that RACF associates with the Server Started Task. In the User for OPC address space field, specify the RACF user ID used for the Server address space. This is the name specified in the started-procedure table.

Refresh CP group

This information is used to create the procedure to build the directory with the right ownership. To create the new Symphony file, the user ID used to run the daily planning batch job must belong to the group that you are specifying in this field. Also ensure that the user ID associated with the Server address space (the one specified in User for OPC address space field) belongs to this group or has this group as supplementary group.

RESTART AND CLEANUP (Data Store)

Specify Y if you want to use the Restart and Cleanup function.

Note: The panel shown in Figure 14 on page 58 creates only one sample entry, but you can define all kinds of connections, including a combination of mixed connections.

Reserved destination

Specify the destination reserved for Data Store output. It must be the same as that for controller and Data Store parameters.

Connection type

Specify the connection method used to handle communication between FN/FL tasks and Data Store. It can be SNA, XCF, or TCP. Required.

SNA Data Store luname

If you chose SNA in Connection type, specify the Data Store VTAM connection name.

SNA FN task luname

If you chose SNA in Connection type, specify the VTAM application name of the controller FN task.

Xcf Group

If you chose XCF in Connection type, specify the name of the XCF group.

Step 4. Using the EQQJOBS installation aid

Xcf Data Store member

If you chose XCF in Connection type, specify the name of the Data Store XCF member.

Xcf FL task member

If you chose XCF in Connection type, specify the name of the controller FL task XCF member.

TCP Data store host name

Specify the Data Store TCP/IP host name if you chose TCP in Connection type.

TCP Data store port number

Specify the Data Store TCP/IP port number if you chose TCP in Connection type.

4. Press Enter. The following panel is displayed:

```
EQQJOBS9 ----- CREATE SAMPLE JOB JCL -----
Command ==>

JAVA UTILITIES ENABLEMENT:  Y      (Y= Yes ,N= No)
  Installation Directory    ==> /usr/lpp/TWS/V8R5M0_____
                             ==> _____
                             ==> _____
  Java Directory            ==> /usr/lpp/java/J5.6_____
                             ==> _____
                             ==> _____
  Work Directory            ==> /var/TWS/inst_____
                             ==> _____
                             ==> _____
  User ID                   ==> UID_____
  Group ID                  ==> GID_____

  JZOS Batch Launcher
    PDSE Library            ==> _____
    Load Module Name       ==> JVMLDM66

Press ENTER to continue
```

Figure 15. EQQJOBS9 - Create sample job JCL

JAVA UTILITIES ENABLEMENT

Specify Y to enable one or both the following features:

- Dynamic Workload Console reporting
- Event-driven workload automation feature for data set triggering

Installation Directory

Specify the directory, with its complete path, where the product specific HFS or ZFS files are stored. This path corresponds to the path where the binary files are located, omitting the subdirectory /bin.

Java Directory

Specify the HFS or ZFS directory where the Java Software Development Kit (SDK) for z/OS is installed.

Work Directory

Specify the directory where the subsystem specific HFS or ZFS files are stored. Each subsystem supporting the JAVA utility must have its own work directory.

User ID

Specify the UNIX System Services user ID.

Step 4. Using the EQQJOBS installation aid

Group ID

Specify the UNIX System Services group ID.

JZOS Batch Launcher PDSE Library

Specify the PDSE that contains the JZOS Batch Launcher JVMLDM module.

JZOS Batch Launcher Load Module Name

Specify JVMLDM66, that is the JZOS Batch Launcher load module name used with 64-bit SDK for z/OS 6.0.

5. Press Enter. The following panel is displayed:

```
EQQJOBSC ----- CREATE SAMPLE JOB JCL -----
Command ==>

SSL FOR TCP/IP CONNECTION      Y      (Y= Yes ,N= No)
SSL Work Directory              ==> /var/TWS/inst/ssl_____
                                ==> _____
                                ==> _____
SSL User ID                     ==> UID_____
SSL Group ID                    ==> GID_____

Press ENTER to create sample job JCL
```

Figure 16. EQQJOBSC - Create sample job JCL

SSL FOR TCP/IP CONNECTION

Specify Y if you have trackers connected to the controller through TCP/IP to protect the communication with SSL.

SSL Work Directory

Specify the directory, with its complete path, where the HFS or ZFS files containing the SSL certificates are stored. The default value is the same work directory used for the fault-tolerant end-to-end scheduling, including the /ssl subdirectory.

SSL User ID

Specify the RACF user ID that takes the ownership of the SSL work directory. The default value is the same user ID specified for the fault-tolerant end-to-end scheduling.

SSL Group ID

Specify the RACF group that takes the ownership of the SSL work directory. The default value is the same group specified for the fault-tolerant end-to-end scheduling.

6. Press Enter when you have entered the information on panel EQQJOBSC. The dialog process now generates several members in the output library that you specified. These members, which are described in Table 18, will be used at various stages in the installation.

Table 18. Sample JCL generated by the EQQJOBS dialog

Member	Description of job
EQQ9SM01	Updates the RACF 1.9 router table (ICHRFR01)

Step 4. Using the EQQJOBS installation aid

Table 18. Sample JCL generated by the EQQJOBS dialog (continued)

Member	Description of job
EQQ9SMDE	Updates the RACF 1.9 class-descriptor table (ICHRRCDE)
EQQAUDIB	Sample to invoke EQQAUDIT in batch mode outside of the dialog Note: EQQAUDIB can be used successfully only if the EQQTROUT dsname and the EQQAUDIT output dsn fields in the EQQJOBSA panel are filled in
EQQBENCR	Sample JCL to run the utility that encrypts the Windows passwords set in the USRPSW parameter of the USRREC statements.
EQQBSCAN	Uses the batch loader to scan an application description
EQQBSUBS	Uses the batch loader to create the application descriptions and operator instructions
EQQBVSAM	Deletes and defines an application description data set and creates an application description and operator instructions, using the batch loader
EQQCHKEV	Utility that checks if all events in EQQTWSIN and EQQTWSOU have been correctly processed
EQQCONOP	Sample initial parameters for the controller only
EQQCONO	Sample started task procedure for the controller only
EQQCONP	Sample initial parameters for a controller and tracker in same address space
EQQCON	Sample started task procedure for a controller and tracker in same address space
EQQDBENC	Contains the JCL to encrypt the password in the DBOPT statement
EQQDBOPT	Sample DBOPT statement
EQQDPCOP	JCL and usage notes for copy VSAM function
EQQE2EP	Sample initial parameters for server and batch to define if the end-to-end scheduling with fault tolerance capabilities is active
EQQFLWAT	Sample JCL to call filewatch utility to monitor HFS or ZFS file changes
EQQICNVH	Sample job to migrate history DB2 tables
EQQICNV5	Migrates VSAM files
EQQJES2	Assembles and link-edits the JES2 EXIT7
EQQJER2U	Restores the EXIT7 as a JES2 usermod
EQQJER2V	Restores the EXIT5 as a JES2 usermod
EQQJER3U	Restores the EQQUX191 and EQQUX291 as JES3 usermods
EQQJES21	Assembles and link-edits the JES2 EXIT51
EQQJES2U	Installs the JES2 EXIT7 usermod
EQQJES2V	Installs the JES2 EXIT51 usermod
EQQJES3	Assembles and link-edits a JES3 exit
EQQJES3U	Installs the JES3 usermod
EQQMTWSO	Migrates TWSOU data set size for extended job names from lrecl 120 to lrecl 160
EQQORST	Resets the USS environment for the end-to-end scheduling with fault tolerance capabilities
EQQPCS01	Allocates unique data sets within the sysplex
EQQPCS02	Allocates non-unique data sets

Table 18. Sample JCL generated by the EQQJOBS dialog (continued)

Member	Description of job
EQQPCS03	Generates a job that allocates VSAM copy data set
EQQPCS05	Allocates files used by a controller to enable Fault Tolerant Workstations
EQQPCS06	Allocates VSAM data sets for integration with the end-to-end scheduling with fault tolerance capabilities
EQQPCS07	Allocates VSAM data sets for Restart and Cleanup
EQQPCS08	Allocates USS files for Java utilities enablement
EQQPCS09	Allocates the GDG root and VSAM data set used as input by the archiving process supporting the Dynamic Workload Console reporting feature
EQQPCS10	Creates the SSL work directory used for TCP/IP communication with the controller
EQQSAMPI	Copies sample databases from the sample library to VSAM data sets
EQQSERP	Sample initial parameters for a Server
EQQSER	Sample started task procedure for a Server
EQQSLCHK	JCL to perform a syntactic check on SCRIPT library members
EQQSMF	Updates SMF exits for Tivoli Workload Scheduler for z/OS
EQQTRA	Sample started task procedure for a tracker
EQQTRAP	Sample initial parameters for a tracker
EQQTROPT	Sample TRGOPT statement

Generating batch-job skeletons

To ensure that all files are correctly allocated, you must generate the batch-job skeletons before creating the sample job JCL.

Several controller functions, such as daily planning, are performed by batch jobs that are submitted from the Tivoli Workload Scheduler for z/OS dialog. To generate the skeleton JCL for these jobs:

1. Select option 2 from the EQQJOBS main menu. This panel is displayed:

Step 4. Using the EQQJOBS installation aid

```
EQQJOBS1 ----- GENERATE TWSz BATCH-JOB SKELETONS -----
Command ==>

Enter the name of the output library. This should be a fully qualified
data set name without any enclosing apostrophes. This library should be
allocated to ISPSLIB.

Batch-job skeletons   ==> CCOPC.OPCA.JCLSKELS_____

The following data set names are used by one or more of the generated job
You can specify an asterisk (*) to indicate the name of the subsystem.

Message library name  ==> OPC.SEQMSG0_____
Parameter library     ==> CCOPC.*.PARM_____
Member in parameter   ==> BATCHOPT_____
library
Checkpoint data set   ==> CCOPC.*.CKPT_____

Press ENTER to continue
```

Figure 17. EQQJOBS1 - Generate Tivoli Workload Scheduler for z/OS batch-job skeletons

Batch-job skeletons

Required input. Enter the name of the library where the JCL skeletons are to be stored. Before you use the Tivoli Workload Scheduler for z/OS dialog to submit batch jobs, allocate this library to the ISPSLIB DD statement in the TSO session of dialog users.

“Step 13. Setting up the ISPF environment” on page 123 explains how you set up the dialog. You can create a new library for the skeleton JCL members or put them in an existing skeleton-JCL library.

In the following fields, you can enter &XOPCNM. as one of the qualifiers for the data set names. This is an ISPF variable that is stored in the profile and is the same variable that you specify in option 0.1 (SUBSYSTEM NAME) in the Tivoli Workload Scheduler for z/OS dialogs. When a skeleton is then used by a dialog of the scheduler, &XOPCNM. is substituted with the name of the scheduler subsystem that is being used.

Ensure that &XOPCNM. ends with a period if it is not the low-level qualifier. For example, you could enter CCOPC.&XOPCNM..PARMLIB but CCOPC.&XOPCNM.PARMLIB results in a JCL error.

If you enter an asterisk (*) as a data set qualifier, the generated skeletons will contain &XOPCNM. in place of the asterisk.

Message library name

Required input. Enter the name of the library that contains the Tivoli Workload Scheduler for z/OS messages (SMP/E target DD name SEQMSG0).

Parameter library

Required input. Enter the name of the library that will contain the initialization statements.

Member in parameter library

Required input. Enter the name of a member in the parameter library that will contain the BATCHOPT initialization statement. The Tivoli Workload Scheduler for z/OS batch jobs will use this member. If you have not already created the BATCHOPT statement, you can still generate the batch skeletons, but remember to create a member with the same name when you create the initialization statements.

Checkpoint data set

Required input. Enter the name of the checkpoint data set.

2. Press Enter, and this panel is displayed:

```

EQQJOBS2 ----- GENERATE TWSz BATCH-JOB SKELETONS -----
Command ==>

Enter the following required job stream parameters:
Non-VSAM dsn prefix ==> CCOPC.*_____

VSAM dsn prefix      ==> CCOPC.*V_____
Unit name            ==> 3390_____      Default unit name
Unit name (temp ds) ==> SYSDA_____      Unit name for temporary data sets
Unit name (sort ds) ==> SYSDA_____      Unit name for sort work data sets
SYSOUT class         ==> *_____         SYSOUT class for reports

The following information is optional:

STEPLIB dsname       ==> OPC.SEQQLMD0_____
STEPCLAT dsname      ==> _____
EQQLMLOG dsname      ==> CCOPC.*.MLOGBAT_____

The following information is REQUIRED WITH DBCS support:

KJSRTBL dsname       ==> _____

Press ENTER to generate OPC batch-job skeletons

```

Figure 18. EQQJOBS2 - Generate Tivoli Workload Scheduler for z/OS batch-job skeletons

Non-VSAM dsn prefix

Required input. Enter the qualifiers that prefix the non-VSAM data set names. Tivoli Workload Scheduler for z/OS adds a low-level qualifier to the prefix to uniquely identify the non-VSAM data sets. For example, it adds JTARC for the job-tracking archive data set. If the subsystem name is OPCA, the data set name will be CCOPC.OPCA.JTARC when the skeleton is used by the dialogs.

VSAM dsn prefix

Required input. Enter the qualifiers that prefix the VSAM data set names. Tivoli Workload Scheduler for z/OS adds a low-level qualifier to the prefix to uniquely identify each Tivoli Workload Scheduler for z/OS VSAM data set. For example, it adds WS for the workstation description data set. If the subsystem name is OPCA, the data set name will be CCOPC.OPCAV.WS when the skeleton is used by the dialogs.

Unit name

Required input. Enter a device name that is valid at your installation. This can be a device type, for example 3380, or a group name, for example PROD or TEST.

Unit name (temp ds)

Required input. Enter a device name that can be used for temporary data sets.

Unit name (sort ds)

Required input. Enter a device name that can be used for sort-work data sets.

SYSOUT class

Required input. Specify the SYSOUT class that you want to use for the reports that are generated by the batch jobs.

Step 4. Using the EQQJOBS installation aid

STEPLIB dsname

Optional. Enter the name of the Tivoli Workload Scheduler for z/OS load module library if the load modules are not in a data set included in an active LNKLIST member.

STPCAT dsname

Optional. Enter the name of a private catalog if one or more data sets cannot be reached via the master catalog. To customize the EQQAUDNS skeleton clist with the appropriate loadlib that should be referenced when audit/debug is invoked, you must specify the dsname.

EQQMLOG dsname

Optional. Enter the name of a message log data set if messages are not sent to SYSOUT. This must not be the same data set that is used by a tracker, controller, or standby controller.

KJSRTBL dsname

Required if you use the Japanese language feature. Enter the name of a data set that will be used when sorting fields containing DBCS data.

3. Press Enter. The following panel is displayed:

```
EQQJOBSA ----- GENERATE TWSz BATCH-JOB SKELETONS -----
Command ==>
Specify if you want to use the following optional features:

END-TO-END WITH FAULT TOLERANCE:      N      (Y= Yes ,N= No)
(To schedule jobs on fault-tolerant workstations)

RESTART AND CLEAN UP (DATA STORE):     Y      (Y= Yes ,N= No)
(To be able to retrieve joblog,
execute data set clean up actions and step restart)

FORMATTED REPORT OF TRACKLOG EVENTS:   N      (Y= Yes ,N= No)
EQQTROUT dsname      ==> _____
EQQAUDIT output dsn   ==> _____

JAVA UTILITIES ENABLEMENT:             N      (Y= Yes ,N= No)
Work Directory        ==> /var/TWS/inst_____
                      ==> _____
                      ==> _____
JZOS PDSE Library     ==> _____
JZOS Load Module Name ==> JVMLDM66
REXX SYSEXEC dsname   ==> OPC.SEQMISC
Input XML dsname for  ==> TWS.EVLIB.XML($$$$$$)_____
data set triggering

Press ENTER to generate OPC batch-job skeletons
```

Figure 19. EQQJOBSA - Generate Tivoli Workload Scheduler for z/OS batch-job skeletons

END-TO-END WITH FAULT TOLERANCE

Specify Y if you want to work with Tivoli Workload Scheduler fault-tolerant workstations.

RESTART AND CLEAN UP (Data Store)

Specify Y if you want to use the Restart and Cleanup feature.

FORMATTED REPORT OF TRACKING EVENTS

Specify Y if you want to use the feature that produces a formatted report of the tracklog events.

EQQTROUT dsname

This entry is optional. Specify the name of the data set in which DP Extend and Replan writes tracklog events. Leave blank if you want the

Step 4. Using the EQQJOBS installation aid

corresponding DD card for these jobs to specify DUMMY as in previous releases. Fill out if you plan to use sample EQQAUDIB (see Table 18).

EQQAUDIT output dsn

Specify the name of a data set where the EQQAUDIT output is to be written. Required if FORMATTED REPORT OF TRACKLOG EVENTS is set to Y.

Work Directory

Specify the directory where the subsystem specific HFS or ZFS files are stored. Each subsystem supporting the JAVA utility must have its own work directory.

JZOS PDSE Library

JZOS PDSE Library Specify the PDSE containing the JZOS Batch Launcher JVMLDM module.

JZOS Load Module Name

Specify JVMLDM66, that is the JZOS Batch Launcher load module name used with 64-bit SDK for z/OS 6.0.

REXX SYSEXEC dsname

Specify the installation SEQQMISC library containing the REXX programs EQQRXARC and EQQRXTRG.

Note: In controller MLOG dsn, EQQTROUT dsname, and EQQAUDIT output dsn you can use an asterisk (*) for the subsystem name. It will be replaced with the current subsystem name when the dialog is invoked.

4. Press Enter when you have entered the information on panel EQQJOBSA. The dialog now generates the batch-job skeleton members.

After completing this procedure, you can proceed with the creation of the sample job JCL as described in “Creating the sample job JCL” on page 55.

If you are not sure at this stage what some of the values will be, it does not matter. You can rerun the dialog as many times as you want to regenerate the skeletons. You can also edit the generated skeletons manually.

This table shows the JCL skeleton members that EQQJOBS generates:

Table 19. Controller skeleton JCL generated by the EQQJOBS dialog

Member	Batch job description
EQQADCDS	Application cross-reference of conditional dependencies.
EQQADCOS	Calculate and print run dates of an application.
EQQADDES	Application cross-reference of external dependencies.
EQQADPRS	Application print program.
EQQADXRS	Application cross-reference program.
EQQADX1S	Application cross-reference of selected fields.
EQQAMUPS	Application description mass update.
EQQAPARS	Procedure to gather diagnostic information.
EQQAUDIS	Extract and format job tracking events (batch invocation).
EQQAUDNS	Extract and format job tracking events (interactive invocation) Note: Ensure to copy this member from the library where it was created by EQQJOBS into a procedure library. This step is required since this member must be invoked interactively.

Step 4. Using the EQQJOBS installation aid

Table 19. Controller skeleton JCL generated by the EQQJOBS dialog (continued)

Member	Batch job description
EQQDBARS	Daily Planning - Historical run data archiver for Dynamic Workload Console reporting feature
EQQDPEXS	Daily planning - plan next period.
EQQDPPRS	Daily planning - print current period results.
EQQDPRCS	Daily planning - replan current period.
EQQDPSJS	Daily planning -DBCS sort step.
EQQDPSTS	Daily planning - normal sort step.
EQQDPTRS	Daily planning - plan a trial period.
EQQJVPRS	Print JCL variable tables.
EQQLEXTS	Long-term planning - extend the long-term plan.
EQQLMOAS	Long-term planning - modify all occurrences.
EQQLMOOS	Long-term planning - modify one occurrence.
EQQLPRAS	Long-term planning - print all occurrences.
EQQLPRTS	Long-term planning - print one occurrence.
EQQLTRES	Long-term planning - create the long-term plan.
EQQLTRYs	Long-term planning - trial.
EQQOIBAS	Operator instructions - batch program.
EQQOIBLS	Operator instructions - batch input from a sequential data set.
EQQSYRES	Daily Planning - Symphony Renew.
EQQTPRPS	Print periods.
EQQTPRTS	Print calendars.
EQQTRBLS	Event driven workload automation - Create configuration files for data set triggering
EQQWPRTS	Print workstation descriptions.

Generating Data Store samples

To create the Data Store samples:

1. Select option 3 from the EQQJOBS application menu. This panel is displayed:

Step 4. Using the EQQJOBS installation aid

```
EQQJOBS5 ----- CREATE DATA STORE SAMPLES -----
Command ==>

The data set names specified on this panel should be fully qualified
names without any enclosing apostrophes.

Enter the name of the output library:

Sample job JCL      ==> CCOPC.OPCA.JCLDS_____

Job statement information:

==> //SYSPROG1 JOB (111111,2222),'OPCESA BATCH',CLASS=B,MSGCLASS=H,___
==> //          MSGLEVEL=(1,1),NOTIFY=SYSPROG_____
==> _____
==> _____

The following data set names are used by one or more of the generated jobs.
Message library name ==> OPC.SEQQMSG0_____
Parameter library    ==> CCOPC.OPCEDS.PARM_____

Press ENTER to continue
```

Figure 20. EQQJOBS5 - Create Data Store samples

Sample job JCL

Required input. Enter the name of a library where you want the generated Data Store samples written to. The library must be allocated before you generate the Data Store samples. Ensure that the library that you specify has sufficient directory blocks to store all the sample members that are generated by EQQJOBS (see Table 20 on page 72).

Job statement information

Required input. Enter a JOB statement that follows standard JCL syntax and your installation standards.

Message library name

Required input. Enter the name of the library that contains the scheduler messages (SMP/E target DDNAME SEQQMSG0).

Parameter library

Required input. Enter the name of a library that will contain the initialization statements. This library must be allocated by the user. Use a name different from that of the controller and tracker parameter library.

2. Press Enter, and this panel is displayed:

Step 4. Using the EQQJOBS installation aid

```
EQQJOBS6 ----- CREATE DATA STORE SAMPLES -----
Command ==>

Enter the following required job stream parameters:
Non-VSAM dsn prefix ==> CCOPC.OPCA_____
VSAM dsn prefix      ==> CCOPC.OPCAV_____
Unit name           ==> 3390_____      Default unit name
Primary volume serial ==> PROD01_____   Primary volume serial for VSAM

The following information is optional:
STEPLIB dsname      ==> OPC.SEQQLMD0_____
VSAMCAT dsname      ==> _____
VSAM password       ==> _____

Press ENTER to continue
```

Figure 21. EQQJOBS6 - Create Data Store samples

Non-VSAM dsn prefix

Required input. Enter the qualifiers that prefix the non-VSAM data set names. The scheduler adds a low-level qualifier to the prefix to uniquely identify each Data Store non-VSAM data set.

VSAM dsn prefix

Required input. Enter the qualifiers that prefix the VSAM data set names. The scheduler adds a low-level qualifier to the prefix to uniquely identify each Data Store VSAM data set.

Unit name

Required input. Enter a device name that is valid at your installation. This could be a device type, for example 3390, or a group name, for example PROD or TEST.

Primary volume serial

Required input. Enter a volume that will be used by sample job EQQPCS04 to allocate the primary data sets.

STEPLIB dsname

Optional. Enter the name of the scheduler load module library if the load modules are not in a data set included in an active LNKLIST member.

VSAMCAT dsname

Optional. Enter the name of a catalog in which VSAM data sets are to be defined if they are not to be defined in the master catalog.

VSAM password

Optional. Enter the password for the VSAM catalog if the catalog is password-protected.

3. Press Enter, and this panel is displayed:

```

EQQJOBS7 ----- CREATE DATA STORE SAMPLES -----
Command ==>

Enter the parameters to build DSTOPTS and DSTUTIL options samples:

Reserved destination      ==> OPCX
Connection type           ==> SNA (SNA/XCF/TCP)
  SNA Data Store luname   ==> I9PC45AA (only for SNA connection)
  SNA Controller luname   ==> I9PC45RA (only for SNA connection)
  Xcf Group               ==> _____ (only for XCF connection)
  Xcf Data Store member   ==> _____ (only for XCF connection)
  Xcf FL task member      ==> _____ (only for XCF connection)
  TCP Controller host name: _____ (only for TCP connection)
  ==>
  TCP Controller port number ==> _____ (only for TCP connection)
Jobdata ret. period       ==> 2_ (number of days)
Joblog retrieval          ==> Y (Y/N)
  Max n. lines to store   ==> 0
  Joblog ret. period      ==> 5_ (number of days)

Press ENTER to create sample job JCL

```

Figure 22. EQQJOBS7 - Create Data Store samples

Reserved destination

Required input. Enter the Data Store reserved output destination. It must correspond to DSTDEST parameter in RCLOPTS option.

Connection Type

Required input. Enter the connection method used to handle communication between FN/FL tasks and Data Store. It can be SNA or XCF.

SNA Data Store luname

Enter Data Store VTAM application name if SNA connection type has been chosen.

SNA Controller luname

Enter controller FN task VTAM application name if SNA connection type has been chosen.

Xcf Group

Enter the name of XCF group if XCF connection type has been chosen.

Xcf Data Store member

Enter the name of Data Store XCF member if XCF connection type has been chosen.

Xcf FL task member

Enter the name of FL task XCF member if XCF connection type has been chosen.

Job data retention period

Enter the Data Store structured information retention period. It consists of the interval in days used by the online cleanup and is necessary to be able to use the Restart and Cleanup feature.

Joblog retrieval

Specify if the joblog retrieval must be enabled. This means that the Data Store will save the unstructured data in the joblog.

Max n. of lines to store

Enter the maximum number of user sysout lines to be stored. The range is 0 to 10000.

Step 4. Using the EQQJOBS installation aid

Joblog retention period

Enter the Data Store unstructured information retention period. It consists of the interval in days used by the online cleanup and is necessary to enable the Joblog Browse function.

TCP Controller host name

Specify the Controller TCP/IP host name if you chose TCP in Connection type.

TCP Controller port number

Specify the Controller TCP/IP port number if you chose TCP in Connection type.

4. Press Enter when you have entered the information on panel EQQJOBS7. The dialog now generates several members in the output library that you specified. These members, which are described in Table 20 is used at various stages in the installation:

Table 20. Data Store samples generated by the EQQJOBS dialog

Member	Sample Description
EQQCLEAN	Sample procedure invoking EQQCLEAN program
EQQDSCL	Batch Clean Up sample
EQQDSCLP	Batch Clean up sample parameters
EQQDSEX	Batch Export sample
EQQDSEXP	Batch Export sample parameters
EQQDSIM	Batch Import sample
EQQDSIMP	Batch Import sample parameters
EQQDSRG	Batch sample reorg
EQQDSRI	Batch Recovery index
EQQDSRIP	Batch Recovery index parameters
EQQDST	Sample procedure to start Data Store
EQQDSTP	Parameters for sample procedure to start Data Store
EQQPCS04	Allocate VSAM data sets for Data Store

Step 5. Adding SMF and JES exits for event tracking

Perform this task if you are installing a tracker.

Tivoli Workload Scheduler for z/OS tracks the progress of jobs and started tasks through the z/OS system by using JES and SMF exit points. Add all these exits on each z/OS system where you will start Tivoli Workload Scheduler for z/OS.

To simplify the installation of Tivoli Workload Scheduler for z/OS event tracking, several sample event-tracking exits can be found in your sample library, SEQQSAMP. To assemble and install exits, you can use the sample JCL provided 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 that Tivoli Workload Scheduler for z/OS is dependent on.

Step 5. Adding SMF and JES exits for event tracking

If you install a new release of Tivoli Workload Scheduler for z/OS in a new CSI, and the JES usermod is already installed in the same CSI as a previous release, follow these steps:

1. Apply any necessary tolerance PTFs so that the previous release can run with the new exit code.
2. Change the DDDEFs for JES so that they point to the SEQQSAMP and SEQQMAC0 libraries of the *new* release.
3. APPLY REDO the JES usermod. This reassembles the exits with the new code.

The sample exits all use the EQQEXIT macro to create event-generating code. For more information on the EQQEXIT macro, see Appendix C, “Invoking the EQQEXIT macro,” on page 331.

Table 21 describes the samples that you can use to generate and install the exits. The sample exit, skeleton JCL, and *usermod* entries identify members of the SEQQSAMP library. The event types in the table are prefixed with A for JES2 or B for JES3, when they are created by the exit. (See “Verifying tracking events” on page 147 for more information about event types.)

Table 21. Sample exits for Tivoli Workload Scheduler for z/OS

Exit name	Exit type	Sample exit	Sample JCL/ usermod	Event supported	Event type
IEFACTRT	SMF	EQQACTR1	EQQSMF	Job and step completion	3J,3S
IEFUJI	SMF	EQQUJI1	EQQSMF	Job start	2
IEFU83	SMF	EQQU831	EQQSMF	End of print group, purge (JES3 only), data set triggering support, and automatic change support	4,5,S,T
EXIT7	JES2	EQQXIT74	EQQJES2/ EQQJES2U	JCT I/O exit for JES2, purge	1,3P,5
EXIT51	JES2	EQQXIT51	EQQJES21/ EQQJES2V	JES2 QMOD phase change exit, z/OS 1.7 and later	1
IATUX19	JES3	EQQUX191	EQQJES3/ EQQJES3U	Output processing complete	3P
IATUX29	JES3	EQQUX291	EQQJES3/ EQQJES3U	On job queue	1

SMF only

The EQQU831 sample generates type 4 and type 5 events and also generates resource availability events when a data set is closed after read or update processing. See “Implementing support for data set triggering” on page 83 for more information.

You must tailor the sample JCL to the requirements of your installation. If you have already run EQQJOBS, tailored versions of the JCL will already exist in the EQQJOBS output library. Alternatively, you can copy any of the members from the SEQQSAMP library to one of your own libraries and manually tailor the JCL.

If you are unfamiliar with how to activate SMF exits, see “Updating SMF parameters” on page 77 and the documentation for SMF.

Step 5. Adding SMF and JES2 exits for event tracking

JES2 only

The load module names are the same as the exit names, except for JES2. The load module of the JES2 exits, which are EXIT7 and EXIT51, are called OPCAXIT7 and TWSXIT51, and their entry points are called OPCAENT7 and TWSSENT51, respectively.

If your z/OS system is a JES2 system, include these records in the JES2 initialization member:

JES2 Initialization Statements

```
LOAD(OPCAXIT7) /*
Load Tivoli Workload Scheduler for z/OS exit mod */
EXIT(7) ROUTINES=OPCAENT7,STATUS=ENABLED /*
Define EXIT7 entry point */
```

If your z/OS system is version 1.7 or later, add the following records to the JES2 initialization member:

```
LOAD(TWSXIT51) /*
Load Tivoli Workload Scheduler for z/OS exit mod */
EXIT(51) ROUTINES=TWSSENT51,STATUS=ENABLED /*
Define EXIT51 entry point */
```

To dynamically install the JES2 exits for Tivoli Workload Scheduler for z/OS, use these commands once the modules are available in the LNKLIST:

```
$ADD LOADMOD(OPCAXIT7),STORAGE=PVT
$T EXIT(7),ROUTINES=OPCAENT7,
  STATUS=ENABLED
$ADD LOADMOD(TWSXIT51),STORAGE=PVT
$T EXIT(51),ROUTINES=TWSSENT51,
  STATUS=ENABLED
```

To put a new version of an exit (that was previously installed) in place, use these commands once the modules are available in the LNKLIST:

```
$TLOADMOD(OPCAXIT7),REFRESH
$TLOADMOD(TWSXIT51),REFRESH
```

For more information on JES2 initialization statements, see *JES2 Initialization and Tuning Reference*.

JES3 only

To activate the exits for a JES3 system, you can link them to a library that is concatenated ahead of SYS1.JES3LIB. Alternatively, you can replace the existing exits in SYS1.JES3LIB with the Tivoli Workload Scheduler for z/OS-supplied IATUX19 and IATUX29 exits. For more information, see *JES3 Initialization and Tuning*. If you get RC=4 and the warning ASMA303W Multiple address resolutions may result when you assemble IATUX19 running the EQQJES3/EQQJES3U sample, you can ignore the message. If version ASMA90 of the compiler reports errors, and the RMODE=ANY statement is defined, remove the RMODE=ANY statement from the sample exit.

Step 6. Updating SYS1.PARMLIB

The following sections describe the updates to SYS1.PARMLIB for your environment:

Defining subsystems

When you define the subsystem names of the Tivoli Workload Scheduler for z/OS controllers and trackers, consider the following:

- The Subsystem/STC name of Tivoli Workload Scheduler for z/OS controllers is unique within the PLEX. If two different controllers (regardless of their location) are configured to track work on the same z/OS system, they must have different Subsystem/STC names.
- Because subsystem names on a given LPAR must be unique, and because all Tivoli Workload Scheduler for z/OS trackers and controllers started tasks must have the same name as their associated subsystems, all started tasks on any given LPAR must have unique names. That is, inside an MVS image, controllers and trackers must have unique Subsystem/STC names.
- Trackers running on different LPARs but connected to the same controller can have the same Subsystem/STC name. In this case, system variables like &SYSNAME can be used with the condition that each tracker uses different Tivoli Workload Scheduler for z/OS data sets. The tracker name cannot be the same as the name of a controller.

You must define the name of every new Tivoli Workload Scheduler for z/OS subsystem in the active subsystem-name-table member of SYS1.PARMLIB. Install at least two Tivoli Workload Scheduler for z/OS controlling systems, one for testing and one for your production environment.

Note: It is recommended that you install the tracker and the controller in separate address spaces on the controlling system.

To define the subsystems, update the active IEFSSNnn member in SYS1.PARMLIB. Include records as in the following example:

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

subsystem name

The name assigned to a Tivoli Workload Scheduler for z/OS subsystem. The name must be from 2 to 4 characters. All the subsystem names, as defined in the SYS1.PARMLIB member IEFSSNnn, must be unique within a GRS complex with the exception of a standby controller. Also, the subsystem names of the controllers must be unique within your OPCplex/OPC network, both local and remote systems. Tivoli Workload Scheduler for z/OS requires the started task name or jobname used for a Tivoli Workload Scheduler for z/OS address space to exactly match the name of the associated subsystem.

module name

The name of the subsystem initialization module, EQQINITJ.

maxecsa

Defines the maximum amount of extended common service area (ECSA) that is used to queue 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 job-tracking events. The maximum value allowed for MAXECSA is 2816.

Step 6. Updating SYS1.PARMLIB

suffix The module name suffix for the EQQSSCM module that EQQINITJ 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 Workload Scheduler for z/OS is EQQINITJ, specify a suffix value of J. If you do not provide a suffix, EQQINITJ attempts to load module name EQQSSCMJ. You can also specify a subsystem communication module name in the SSCMNAME keyword of the OPCOPTS initialization statement to load an updated version of the module before a scheduled IPL. For details, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

“Updating the z/OS link-library definition” on page 80 provides more information about EQQSSCM modules.

This example illustrates a record you can include in the SYS1.PARMLIB IEFSSNnn member:

```
/*Subsystem definition example*/  
SUBSYS SUBNAME(OPCA) INITRTN(EQQINITJ) INITPARM ('100,J')
```

The record defines an Tivoli Workload Scheduler for z/OS subsystem called OPCA. This represents a tracker. Its initialization module is EQQINITJ. The amount of ECSA that is allocated, 101104 bytes, is enough for 1136 job-tracking events. Because suffix value J is specified, EQQINITJ loads module EQQSSCMJ.

Calculating MAXECSA values

Tivoli Workload Scheduler for z/OS allocates ECSA storage for job-tracking events in blocks of 1424 bytes. Each block is equivalent to 16 events. Table 22 gives examples of the storage needed for, the storage actually allocated, and the events accommodated for several **maxecs** values. The number of events created for each job or started task in your environment is influenced by the definitions in the EWTROPTS initialization statement. Every job or started task creates a minimum of six events. If the job or started task generates output and PRINTEVENTS(ALL) or PRINTEVENTS(END) is specified, an event is created when each output group is purged. If STEPEVENTS(ALL) is specified, an event is created for every step in the job or started task.

If you want to calculate values that are not shown in Table 22 for a given MAXECSA value, use this method:

- Space requested = MAXECSA * 1024
- Blocks = space requested / 1424 (round down to a whole number)
- Space allocated = blocks * 1424
- Events accommodated = blocks * 16

Table 22. Examples of MAXECSA storage values

MAXECSA value	Amount of MAXECSA space requested	Blocks of ECSA space allocated (bytes)	Number of events accommodated
0	0	0 (0)	0
4	4096	2 (2848)	32
8	8192	5 (7120)	80
16	16384	11 (15664)	176
36	36864	25 (35600)	400
72	73728	51 (72624)	816
100	102400	71 (101104)	1136

Table 22. Examples of MAXECSA storage values (continued)

MAXECSA value	Amount of MAXECSA space requested	Blocks of ECSA space allocated (bytes)	Number of events accommodated
200	204800	143 (203632)	2288
400	409600	287 (408688)	4592
500	512000	359 (511216)	5744

Notes:

1. Allocate enough ECSA storage so that job-tracking events are not lost when the Tivoli Workload Scheduler for z/OS event-writer subtask is not active. When the event writer is active, the number of queued events in ECSA is almost always 0. Allocate enough ECSA for the maximum amount of time you expect the event writer to be inactive.

For example, after the IPL of a z/OS system, job-tracking events can occur before the tracker address space has become active. If you expect a maximum of 50 events to occur during this time, you should set a MAXECSA value of 8, as shown in Table 22 on page 76. When the event writer becomes active, the queued events are processed and removed from ECSA.

If events are lost, message EQQZ035E is stored to the message log. For a description of this message, see *Tivoli Workload Automation: Messages and Codes*, SC23-9114.

2. ECSA storage for job-tracking events is required only if the started task includes an event-writer subtask. On a controlling system, you can have one address space running only an event writer subtask, and another one running the controller functions and the remaining tracker functions. In this situation, you must specify a MAXECSA value of 0 for the subsystem that contains the controller functions.
3. All ECSA storage is allocated above the 16 MB line.

Authorizing the load-module library

You must update the active authorized-program-facility member (IEAAPFnn, or PROGnn) to authorize the load-module library. Each record, except the last, ends with a comma. For the following example, assume that you have installed Tivoli Workload Scheduler for z/OS load modules in the data set OPC.SEQQLMD0 and that this data set is on volume ABC123. To authorize this library, insert this record before the last entry in the IEAAPFnn:

```
OPC.SEQQLMD0    ABC123,
```

or update the PROGnn member.

Note: Libraries that are defined in the IEAAPFnn or PROGnn member are authorized only if they remain on the volume specified. If DFHSM is used in your system, change DFHSM parameters so that the new authorized library is not migrated by DFHSM.

Updating SMF parameters

The SMFPRMnn member defines parameters for the System Management Facilities (SMF). You must verify that the active SMF parameter member, SMFPRMnn, specifies that all SMF exits used by Tivoli Workload Scheduler for z/OS event tracking are activated, and that the required SMF records are being collected. If this is not the case, you must update the active SMF parameter member. Event tracking requires these SMF exits:

Step 6. Updating SYS1.PARMLIB

IEFUJI	Job initiation exit
IEFACTRT	Job-end and step-end exits
IEFU83	Record write exit. It is optional, and required only for data set triggering, automatic time change, and print event functions.

Tivoli Workload Scheduler for z/OS uses the following SMF record types:

- 6 For PRINT (A4 and B4) events, used only for tracking work on PRINT workstations
- 14 Only for data set triggering with SRREAD=YES
- 15 For data set triggering with SRREAD=YES or SRREAD=NO
- 18 Only if you want to monitor renaming data sets.
- 26 For all job tracking
- 30 For all job tracking
- 64 Only for data set triggering with VSAM data sets
- 90 Only if you want automatic daylight savings time change

Tivoli Workload Scheduler for z/OS requires more SMF records to be collected if you install the SMF IEFU83 exit with SRREAD set to YES on the EQQEXIT invocation. Specify this if you want special resource availability events automatically generated when a data set is closed after being opened for:

- Read processing
- Output processing
- Either read or output processing

These SMF records are needed:

- Type 14 records are required for non-VSAM data sets opened for INPUT or RDRBACK processing.
- Type 15 records are required for non-VSAM data sets opened for output.
- Type 64 records are required for VSAM data sets.
- Type 90 records support daylight savings time automatically (optional).

You can specify that the SMF records used by the exit are not written to the SMF log. If your installation does not currently collect SMF records 14, 15, or 64, but you want resource availability events automatically generated, change the EQQU831 sample so that these records are not written to the SMF log.

To avoid data set triggering, and thus to improve performance, specify SRREAD=NO in the IEFU83 SMF exit on invocation of the EQQEXIT macro. The SRREAD=NO parameter prevents data set triggering for only SMF record type 14.

Active exits are defined by the EXITS parameter of the SYS and SUBSYS keywords. An example of these keywords is:

```
/*SYS and SUBSYS keywords*/  
SYS(TYPE(6,14,15,18,26,30,60,62,64,90),EXITS(IEFU83,IEFACTRT,IEFUJI))  
SUBSYS(STC,EXITS(IEFUJI,IEFACTRT,IEFU83))  
SUBSYS(JESn,EXITS(IEFUJI,IEFACTRT,IEFU83))
```

Notes:

1. JESn is either JES2 or JES3. This parameter does not refer to JES itself, but to batch jobs handled by JES. So do not suppress exit invocation. Ensure that you do not specify TYPE6=NO and TYPE26=NO on the JOBCLASS and STCCCLASS statements of the JES2 initialization parameters.

2. You might find it useful during installation to code two SMFPRMnn members, one with the exits active and the other with the exits inactive. You can then use the SET SMF=nn z/OS command to switch your current SMF parameters to the new member. By switching back, using the SET SMF=nn command, you avoid the need to re-IPL, if you encounter a problem.
3. Exits for SUBSYS STC are required by Tivoli Workload Scheduler for z/OS.

Use the PROGnn parmlib member to specify installation exits and control their use. Using PROGnn, you can associate multiple exit routines with installation exits at IPL, or while the system is running. IBM recommends that you use PROGnn in addition to SMFPRMnn to specify exits, whether or not you want to take advantage of these functions.

The following example shows how you can specify SMF exits in a PROGxx parmlib member. If you specify this in SMFPRMnn:

```
SYS(...EXITS(IEFU83,IEFACTRT,IEFUJI))
```

you would add this to get the equivalent processing in PROGnn:

```
EXIT ADD EXITNAME(SYS.IEFU83) MODNAME(IEFU83)
EXIT ADD EXITNAME(SYS.IEFACTRT) MODNAME(IEFACTRT)
EXIT ADD EXITNAME(SYS.IEFUJI) MODNAME(IEFUJI)
```

When you associate new exit routines with SMF exits through PROGnn or the SETPROG command, you must use the following naming conventions:

- For exits listed on the EXITS keyword of the SYS statement in SMFPRMnn, each exit will have the name SYS.xxxx (where xxxx is one of the exits listed).
- For exits listed on the EXITS keyword of the SUBSYS statement of SMFPRMnn, each exit will have the name SYSzzzzz.xxxx (where zzzzz is the name of the subsystem and xxxx is one of the exits listed).

If you define two members in SYS1.PARMLIB with two different names, for example, PROG03 in which there is the statement EXIT ADD EXITNAME(SYS.IEFACTRT) MODNAME(EQQACTR1), you can switch to the version EQQACTR1 without re-ipling by issuing the command:/SET PROG=03

If you are using FTP, you must add the following statement to the SMFPRMxx member:

```
SUBSYS(OMVS,EXITS(IEFUJI,IEFU83))
```

Also, these statements must be added to the PROGnn member, making sure that you replace MODNAME with the module name that was used when the exits were link-edited:

```
EXIT ADD EXITNAME(SYSOMVS.IEFU83) MODNAME(EQU831)
EXIT ADD EXITNAME(SYSOMVS.IEFUJI) MODNAME(EQUJ11)
```

For information on using PROGnn to control the use of exits and exit routines, see *z/OS Initialization and Tuning Reference*

Updating z/OS dump options

The sample JCL procedure for a Tivoli Workload Scheduler for z/OS address space includes a DD statement and a dump data set is allocated by the EQQPCS02 JCL created by EQQJOBS. SYSDUMP is the dump format preferred by the service organization.

Step 6. Updating SYS1.PARMLIB

Ensure that the dump options for SYSMDUMP include RGN, LSQA, TRT, CSA, and GRSQ on systems where a Tivoli Workload Scheduler for z/OS address space will run. To display the current SYSMDUMP options, issue the z/OS 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 is performed.

To dump a Tivoli Workload Scheduler for z/OS address space using the z/OS `DUMP` command, the `SDUMP` options should specify RGN, LSQA, TRT, CSA, and GRSQ. Consider defining these options as your system default.

Updating the z/OS link-library definition

If you installed Tivoli Workload Scheduler for z/OS in a separate load-module library, it is recommended that you define this library in the active `LNKLSTnn` member. Alternatively, you can define the load-module library on the `STEPLIB DD` statement of the started-task JCL and TSO logon procedures of Tivoli Workload Scheduler for z/OS dialog users.

If you installed load modules in the data set `OPC.SEQQLMD0` and this data set is cataloged in the master catalog, insert this record before the last entry in the `LNKLSTnn` member to add this library to the link library concatenation:

```
Adding LINKLIB
OPC.SEQQLMD0
```

If you choose not to define the Tivoli Workload Scheduler for z/OS load-module library in the `LNKLSTnn` member, you *must*:

- Copy the tracker modules, `EQQINITJ` and `EQQSSCMJ`, to a library in the z/OS link-library concatenation. `EQQINITJ` is used by the master-scheduler-initialization function when the z/OS system is being IPLed. `EQQINITJ` then loads `EQQSSCMJ` into common storage. `EQQSSCMJ` is about 23KB and is placed above the 16MB line. Remember to copy the modules again whenever they are updated by Tivoli Workload Scheduler for z/OS maintenance. This is especially important for the `EQQSSCMJ` module, which must be at the same update level as the rest of the Tivoli Workload Scheduler for z/OS code.
- Define the Tivoli Workload Scheduler for z/OS load-module library on a `STEPLIB DD` statement in the started-task JCL.
- Define the Tivoli Workload Scheduler for z/OS load-module library on a `STEPLIB DD` statement in the TSO logon procedure of all Tivoli Workload Scheduler for z/OS dialog users.
- Load the dialog module, `EQQMINOJ`, from an APF-authorized library. If you define the Tivoli Workload Scheduler for z/OS load-module library on a TSO `STEPLIB DD` statement, and any of the other libraries defined on this `DD` statement are not authorized, you must copy `EQQMINOJ` to another library in the `LNKLST` concatenation so that it is loaded APF authorized. You must also remember to copy the module again whenever it is updated by Tivoli Workload Scheduler for z/OS maintenance.

Updating XCF initialization options

This section is useful if you use XCF for communication.

XCF initialization options are specified in the `COUPLEnn` member of `SYS1.PARMLIB`. If you have not specified your own `COUPLEnn` member, the system uses the default member, `COUPLE00`. The IBM-supplied `COUPLE00` member causes the system to be IPLed in XCF-LOCAL mode. This mode is *not*

supported by Tivoli Workload Scheduler for z/OS. So ensure that your system uses a COUPLEnn member that does not IPL the system in XCF-LOCAL mode. The COUPLEnn member must include the PCOUPLE keyword of the COUPLE statement. If this is omitted, XCF is initialized in XCF-LOCAL mode. For Tivoli Workload Scheduler for z/OS purposes, you can use the default values for the remaining XCF options.

```
COUPLEnn example
COUPLE  SYSPLEX(PLEX1)           /* SYSPLEX name          */
        PCOUPLE(PLEX1.COUPLE1)  /* Primary couple data set */
        ACOUPLE(PLEX2.COUPLE2)  /* Alternate couple data set*/
        MAXMSG(2000)            /* No of 1k message buffers */
CLASSDEF CLASS(DEFAULT)         /* Default transport class */
        CLASSLEN(956)           /* Message length          */
        GROUP(OPCGRP,OPCDS)     /* OPC Group names          */
PATHIN  DEVICE(cccc,dddd)
PATHOUT DEVICE(aaaa,bbbb)
PATHIN  STRNAME(str1,str2) CLASS(DEFAULT)
PATHOUT STRNAME(str1,str2) CLASS(DEFAULT)
```

Issue the console command "D XCF,CLASSDEF,CLASS=ALL" to see if you already have a DEFAULT class (the name of this class might be something other than DEFAULT) having CLASSLEN(956), which is the default value. If there is such a class, you just need to add the TWS specific GROUP names (OPCGRP,OPCDS) to the CLASSDEF statement for that CLASS, as shown in the example above.

Note: By specifying MAXMSG(2000) on the COUPLE statement, as shown above, all transport classes will use this value unless a different value is specified at the CLASSDEF level. MAXMSG(2000) is the default value.

If XCF is used to connect the Data Store to the controller, a specific XCF group must be defined, and it must be different from the one used to connect the controller to the z/OS tracker. These two separate XCF groups can use the same XCF transport class.

Note: You can change XCF options while the system is active by using the SETXCF operator command.

For more information about XCF, see *z/OS MVS Setting up a Sysplex*.

Modifying TSO parameters

You must define the EQQMINOJ module to TSO on each system where you install the scheduler dialogs. You must also authorize the Tivoli Workload Scheduler for z/OS TSO commands on every system where you install Tivoli Workload Scheduler for z/OS. If you do not authorize the TSO commands, the commands will only work on the system where the controller is installed.

To request services from the subsystem for a TSO user, the Tivoli Workload Scheduler for z/OS dialog invokes the EQQMINOJ module using the TSO service facility. EQQMINOJ is the dialog interface module. It must run as an APF-authorized program. To achieve this, define EQQMINOJ to TSO. If you are installing the scheduler dialogs, include EQQMINOJ in the list of programs defined by the AUTHTSF statement in the IKJTSOnn member of SYS1.PARMLIB. This statement defines programs to be authorized when invoked using the TSO service facility. Here is an example of such a statement:

Step 6. Updating SYS1.PARMLIB

```
IKJTSONn AUTHTSF example
AUTHTSF NAMES(IKJEFF76 +
               IEBCOPY +
               EQQMINOJ)
```

If you prefer, you can put EQQMINOJ in CSECT IKJEFTAP instead of IKJTSONn. For more information about using IKJEFTAPR, see *TSO/E Customization*.

Tivoli Workload Scheduler for z/OS supports the BACKUP, BULKDISC, OPINFO, OPSTAT, SRSTAT, and WSSTAT TSO commands. Update the IKJTSONn member on each system where you are installing Tivoli Workload Scheduler for z/OS to define these commands as authorized commands. To do this, add them to the list of commands defined by the NAMES keyword of the AUTHCMD statement. Here is an example of such a statement:

```
IKJTSONn AUTHCMD example
AUTHCMD NAMES(BACKUP +
               BULKDISC +
               JSUACT +
               OPINFO +
               OPSTAT +
               SRSTAT +
               WSSTAT)
```

If the default entry in the ISPF TSO command table ISPTCM is set for unauthorized TSO commands, then ISPTCM must be updated. The ISPTCM can be updated using the ISPMTCM macro. Define the BACKUP, BULKDISC, OPINFO, OPSTAT, SRSTAT, and WSSTAT commands like this:

```
ISPTCM example
ISPMTCM FLAG=62,ENTNAME=BACKUP
ISPMTCM FLAG=62,ENTNAME=BULKDISC
ISPMTCM FLAG=62,ENTNAME=JSUACT
ISPMTCM FLAG=62,ENTNAME=OPINFO
ISPMTCM FLAG=62,ENTNAME=OPSTAT
ISPMTCM FLAG=62,ENTNAME=SRSTAT
ISPMTCM FLAG=62,ENTNAME=WSSTAT
```

No update is needed to ISPTCM if the default entry is set up for authorized TSO commands. For more information about the ISPMTCM macro statements, see *ISPF Planning and Customization*.

Performance considerations

The tracker and the controller address spaces must be nonswappable. To do this, include the definition of their top load module, EQQMAJOR, in the program properties table (PPT). This PPT entry example is defined in a SCHEDnn member of SYS1.PARMLIB:

```
SCHEDnn example
PPT PGMNAME(EQQMAJOR) NOSWAP
```

The EQQMAJOR program must run in storage key 8, the default value.

To ensure prompt processing by Tivoli Workload Scheduler for z/OS and to avoid delays in the handling of event records, the tracker subsystem performance rating (that is, its dispatching priority) should match that of the JES subsystem.

Defining the DLF exit for Hiperbatch support

If you want to include Hiperbatch support for Tivoli Workload Scheduler for z/OS controlled jobs, specify the DLF exit name in the COFDLFnn member of SYS1.PARMLIB. A DLF exit sample is supplied with the SEQQSAMP library. The

exit must reside in an authorized library in the LNKLIST concatenation. This example of a COFDLFnn member defines a DLF exit called OPCDLF:

```
COFDLFnn example  
CLASS MAXEXPB(nnnn) PCTRETB(nnn) CONEXIT(OPCDLF)
```

For more information on including Hiperbatch support in Tivoli Workload Scheduler for z/OS, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

Starting the product automatically

The COMMNDnn member of SYS1.PARMLIB list z/OS commands automatically issued during system initialization. To avoid delays in starting Tivoli Workload Scheduler for z/OS when the z/OS system is started, consider including the names of your Tivoli Workload Scheduler for z/OS started tasks in this member. For information on how to include start commands for your Tivoli Workload Scheduler for z/OS address spaces, see *MVS Initialization and Tuning Reference*.

Updating APPC options

If you want to use the API, or the server, to communicate with Tivoli Workload Scheduler for z/OS, you must update APPC options. See “Step 17. Activating support for the API” on page 134, or “Step 18. Activating support for the product dialog and programming interface using the server” on page 137, for a detailed description of what you need to do.

Implementing support for data set triggering

Use the Tivoli Workload Scheduler for z/OS data set triggering function to start dependent processing or schedule unplannable work by automatically generating special resource availability events when a data set is closed after being opened for:

- Read processing
- Output processing
- Either read or output processing.

Tivoli Workload Scheduler for z/OS uses the SMF exit IEFU83 to generate a resource availability event when IEFU83 is called for SMF record types 14, 15, or 64. The data set activity SMF records are generated when a data set is closed or processed by EOVS. Tivoli Workload Scheduler for z/OS will generate resource availability events only when the data set is closed. When a VSAM data set is closed, two SMF 64 records are created, one each for the DATA and INDEX components. When resource availability events are requested for VSAM data sets, the event will be created when the DATA component is closed, Tivoli Workload Scheduler for z/OS will not generate an event when the INDEX component is closed.

SMF data set activity records are written when the data set is closed, regardless of whether the JOB/STEP/TASK/USER completed successfully. For more information about the data sets that generate SMF record types 14, 15, or 64, see the documentation for MVS SMF.

To define the data sets for which you want events to be generated, you can perform either of the following:

- Use the EQQRXTRG program to centralize and automate the population of the data set to which the EQQJCLLIB DD name refers. For detailed information about running event-driven workload automation, see *Managing the Workload*.

Step 6. Updating SYS1.PARMLIB

- Build a selection table, as described in Appendix D, “Invoking the EQQLSENT macro,” on page 335. The selection table is located in ECSA. It is automatically loaded from the data set referred to by the EQQJCLIB DD name when the event writer is started in a tracker if a table has not previously been loaded since IPL. To reload the table at any time, issue the z/OS modify command:

`F procname,NEWDSLST`

Note: No support is available for the data set triggering function before the event writer is started immediately after a z/OS IPL. When the event writer has started after IPL, data set triggering functions are available if the event writer is subsequently stopped. To stop data set triggering at any time issue the NEWDSLST modify command to load a table that contains only the end-of-table indicator.

To implement support for the data set triggering function, perform these actions:

- Update SYS1.PARMLIB member SMFPRMnn as described in “Updating SMF parameters” on page 77.
- Install SMF exit IEFU83 using the EQQU831 sample. See “Macro invocation syntax for EQQEXIT” on page 333 on how to specify the SRREAD parameter.
- Define the data set selection criteria as described by the event-driven resource handling section in *Managing the Workload*.

The procedure described in Appendix D, “Invoking the EQQLSENT macro,” on page 335 is supported for backward compatibility only.

Step 7. Setting up the RACF environment

If your installation protects data and resources from unauthorized use, you must define Tivoli Workload Scheduler for z/OS to your security system. This section assumes that the Resource Access Control Facility (RACF) is installed and active on your z/OS system. It describes the activities you must perform to define and enable the security environment for Tivoli Workload Scheduler for z/OS.

Tivoli Workload Scheduler for z/OS: Customization and Tuning contains detailed plans and instructions for establishing a security strategy for your Tivoli Workload Scheduler for z/OS resources.

Controlling the user ID of the address space

If you run Tivoli Workload Scheduler for z/OS as a started task, you must associate the cataloged procedure name with a suitably authorized RACF user. The user ID must be defined in the STARTED resource class.

If you use any of the following definitions in your initialization statements:

- TPLGYSRV parameter in the OPCOPTS statement
- TCPIP parameter in the ROUTOPTS statement
- MONOPTS statement

you must also define an OMVS segment for the controller user ID.

Controlling the user ID of submitted jobs

Tivoli Workload Scheduler for z/OS can submit three kinds of jobs:

- Normal production jobs, which are submitted when their prerequisites in the current plan are fulfilled.

- Stand-alone cleanup jobs, which are submitted to run cleanup actions separately from the original job.
- Dialog jobs, which you can submit directly from a panel in the Tivoli Workload Scheduler for z/OS dialog.

Normal production jobs

Tivoli Workload Scheduler for z/OS submits production jobs to the internal reader, or starts started tasks, when all prerequisites are fulfilled. The JCL comes from the JS file (EQQJSnDS), the JCL job library (EQQJBLIB), or the job-library-read exit (EQQUX002). You can determine the authority given to a job or started task in several ways:

- You can submit work with the authority of the Tivoli Workload Scheduler for z/OS address space. The job or started task is given the same authority as the controller or tracker whose submit subtask actually submits the work. For example, work that is transmitted from the controller and then submitted by the tracker is given the authority of the tracker.
- Another method is to use the job submit exit, EQQUX001. This exit is called when Tivoli Workload Scheduler for z/OS is about to submit work.
 - You can use the RUSER parameter of the EQQUX001 exit to cause the job or started task to be submitted with a specified user ID. The RUSER name is supported even if the job or started task is first sent to a tracker before being started.
 - In certain circumstances you might need to include a password in the JCL to propagate the authority of a particular user. You can use the job-submit exit (EQQUX001) to modify the JCL and include a password. The JCL is saved in the JCL repository (JSn) data set *before* the exit is called, thus avoiding the need to store JCL with specific passwords. This method prevents the password from being visible externally. For more information about the job-submit exit, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265.

Stand-alone cleanup jobs

Their purpose is to run data set cleanup actions and can be submitted when:

- An automatic internal process takes place (for example, when cleanup type immediate is used and an operation ends in error)
- A Start Cleanup command is issued by a Tivoli Workload Scheduler for z/OS dialog or the Dynamic Workload Console.

Activate exit EQQUX001 to make sure that the submitter of the stand-alone cleanup job is the same as the submitter of the original job, otherwise the stand-alone cleanup job will run with the same authority as the controller or the tracker that submits it. The current EQQUX001 sample contains a procedure to set the RUSER value according to the value of the USER= keyword in the jobcard of the original job.

Dialog jobs

When you submit Tivoli Workload Scheduler for z/OS batch jobs from your TSO address space, they go through normal TSO functions. This means that you can submit any job allowed by TSO/E. Tivoli Workload Scheduler for z/OS makes no authority checks when the job is submitted.

For the Tivoli Workload Scheduler for z/OS batch job to run successfully, it must be authorized to reference the data sets it uses. The submitting TSO user might

Step 7. Setting up RACF environment

also need authorization to use a specific function. For example, a user could have update authority to the AD file but not have the authority to use the AD mass update function.

Protecting data sets

For basic security of Tivoli Workload Scheduler for z/OS data, you should restrict access to all the product data sets.

Two categories of users need different levels of access to the product data sets:

- Software support people must be able to debug problems and reorganize VSAM files. You might give them alter access to all the product data sets.
- Administrators and operators must be able to use the product dialogs. They need read access to ISPF-related data sets (such as the panel and message libraries), but they do not access the databases (such as the workstation database) directly: these files are accessed by the Tivoli Workload Scheduler for z/OS subsystem, not by any code in the TSO user's address space. Authority to access the data for a dialog user is given using the authorization functions provided by the product.

The Tivoli Workload Scheduler for z/OS started task needs:

- Alter access to VSAM data sets
- Read access to input data sets, such as the message library (EQQMLIB) and parameter library (EQQPARM)
- Update access to all other Tivoli Workload Scheduler for z/OS data sets
- Update access to catalogs and alter access to data sets for all work that Tivoli Workload Scheduler for z/OS tracks, if you use the Restart and Cleanup function.

Controlling access to resources

Before Tivoli Workload Scheduler for z/OS performs any request initiated by a user, a security verification check is passed to the system authorization facility (SAF) to ensure that the user is authorized to access all resources needed to run the request. A user can request Tivoli Workload Scheduler for z/OS services from:

- An ISPF dialog session
- TSO commands
- The program interface (PIF)
- The application programming interface (API)
- Dynamic Workload Console

Any security software that interfaces with SAF also works with Tivoli Workload Scheduler for z/OS. For this section, the security product is assumed to be RACF.

The z/OS router service calls RACF to perform authority checks. It provides an installation exit that you can use instead of, or in addition to, RACF to perform resource control functions.

Use the Tivoli Workload Scheduler for z/OS reserved resource class IBMOPC.

The default class for Tivoli Workload Scheduler for z/OS is OPCCLASS. If you use a different class name, you must specify it in the AUTHDEF statement. Generally, this means specifying CLASS(IBMOPC) in the AUTHDEF statement. If you are running more than one Tivoli Workload Scheduler for z/OS system, for example a test system and production system, you might want to define more than one RACF

Step 7. Setting up RACF environment

class. By using different CLASS parameters in each AUTHDEF statement, you can specify a different authorization scheme for each system.

To control access to Tivoli Workload Scheduler for z/OS functions, give at least one TSO user-class authority to the resource class. This TSO user can then allow other Tivoli Workload Scheduler for z/OS users to access resources as needed.

Tivoli Workload Scheduler for z/OS also uses the APPL resource class. Define the subsystem name as a resource in the APPL class. The easiest way to do this is to have the RACF administrator give class authority to the APPL resource class to one TSO user. This TSO user defines the subsystem name (for example, OPCC) to the APPL resource class by entering:

```
/*Define subsystem resource*/  
RDEFINE APPL OPCC UACC(NONE)
```

See *RACF Command Reference* and *RACF Administrator's Guide* if you are unfamiliar with this process.

When the subsystem name is defined to RACF, you can give other TSO users access to Tivoli Workload Scheduler for z/OS. For example, to allow the TSO user OPCUGRP to access OPCC with an update access authority by default, enter:

```
/*Permit access to Tivoli Workload Scheduler for z/OS*/  
PERMIT OPCC ID(OPCUGRP) ACCESS(UPDATE) CLASS(APPL)
```

For remote dialog users and remotely run PIF applications, the server will do the authority checking; it will check both the APPL class subsystem name resource and the scheduler fixed resources. The user for which the server does authority checking is:

- For dialog users, the TSO user ID.
- For PIF applications, the user ID defined in the security environment of the PIF job.

Permitting access to the controller through the API

If you use the API, you can control access to the controller through the security functions of both APPC/MVS and Tivoli Workload Scheduler for z/OS. Ensure that you consider both these environments when you update RACF. For more information about controlling access to Tivoli Workload Scheduler for z/OS through the API, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265.

Controlling access to Tivoli Workload Scheduler for z/OS resources when using the Dynamic Workload Console

The WebSphere® Application Server performs a security check when a user tries to use Dynamic Workload Console, checking the user ID and password. The WebSphere Application Server associates each user ID and password to an administrator.

The scheduler resources are currently protected by RACF.

The Dynamic Workload Console user should only have to enter a single user ID and password combination, and not provide two levels of security checking (at the WebSphere Application Server level and then again at the Tivoli Workload Scheduler for z/OS level).

Step 7. Setting up RACF environment

The security model is based on having the WebSphere Application Server 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 z/OS.

z/OS security is based on a table mapping the administrator to a RACF user ID. When a WebSphere Application Server user tries to initiate an action on z/OS, the 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 Workload Scheduler for z/OS services, so the administrator must relate, or map, to a corresponding RACF user ID.

For information on how to get the RACF user ID, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

Permitting access to the controller through the Dynamic Workload Console

If you use the Dynamic Workload Console, you can control access to the controller through the security functions of both WebSphere Application Server and Tivoli Workload Scheduler for z/OS. Ensure that you consider both these environments when you update RACF. For more information about controlling access to Tivoli Workload Scheduler for z/OS through the Dynamic Workload Console, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265.

Authorizing Tivoli Workload Scheduler for z/OS as a job submitter

Consider the following resource classes when implementing security for Tivoli Workload Scheduler for z/OS. The examples assume that the RACF user for the Tivoli Workload Scheduler for z/OS address space is OPCAPPL, which is the name specified in the started-procedure table.

JESJOBS

If your installation has activated the JESJOBS class, you must permit Tivoli Workload Scheduler for z/OS to submit all jobs that are defined in the current plan. One way of doing this is to permit Tivoli Workload Scheduler for z/OS to submit all jobs. You can do this by:

1. Defining the submit resource:

```
RDEFINE JESJOBS SUBMIT.*.*.* UACC(NONE) OWNER(OPCAPPL)
```

2. Authorizing Tivoli Workload Scheduler for z/OS:

```
PERMIT SUBMIT.*.*.* CLASS(JESJOBS) ID(OPCAPPL) ACC(READ)
```

SURROGAT

A *surrogate job submission* occurs when all the following conditions are met:

1. USER=xxxx is specified on the job card of the submitted job.
2. The xxxx is not the same as the submitting (RACF) user.
3. No password is specified on the job card.

You might use the job-submit exit (EQQUX001) to return a submitting user in the RUSER field. This is required if you want stand-alone cleanup jobs to be submitted with the same authority as the original job, otherwise you can replace it with surrogate job submission.

To permit Tivoli Workload Scheduler for z/OS to submit this job, perform the following steps:

1. Activate the surrogate class:
`SETRPTS CLASSACT(SURROGAT)`
2. Define the submit resource:
`RDEFINE SURROGAT APLUSER.SUBMIT UACC(NONE) OWNER(APLUSER)`
3. Authorize Tivoli Workload Scheduler for z/OS:
`PERMIT APLUSER.SUBMIT CLASS(SURROGAT) ID(OPCAPPL) ACC(READ)`

If the PRIVILEGED or TRUSTED attribute is set in the Started Procedure Table (SPT) entry, the Tivoli Workload Scheduler for z/OS is authorized to submit jobs under any user regardless of what is defined in the resource rules.

For further information, see the *RACF Administrator's Guide*.

Authorizing Tivoli Workload Scheduler for z/OS to issue JES commands

Consider the following resource classes when implementing security for Tivoli Workload Scheduler for z/OS. The examples assume that the RACF user for the Tivoli Workload Scheduler for z/OS address space is OPCAPPL, which is the name specified in the started-procedure table.

OPERCMDS If the OPERCMDs class is active and you have specified HOLDJOB(YES) or HOLDJOB(USER) for an event writer, the Tivoli Workload Scheduler for z/OS address space where the event writer is started must be authorized to issue the JES release command. One method is to permit Tivoli Workload Scheduler for z/OS to issue all JES commands. To permit Tivoli Workload Scheduler for z/OS to issue JES commands on a JES2 system, perform the following steps:

1. Define the resource:
`RDEFINE OPERCMDs JES2.* UACC(NONE)`
2. Authorize Tivoli Workload Scheduler for z/OS:
`PERMIT JES2.* CLASS(OPERCMDs) ID(OPCAPPL) ACC(UPDATE)`

On a JES3 system, replace JES2.* with JES3.* in the example. Alternatively, you could specify the JES%.* resource name for either a JES2 or JES3 system.

If you use Tivoli Workload Scheduler for z/OS to schedule started tasks, the address space must be authorized to issue the z/OS start command. One way of doing this is to permit Tivoli Workload Scheduler for z/OS to issue all z/OS commands. To do this, perform the following steps:

1. Define the resource:
`RDEFINE OPERCMDs MVS.* UACC(NONE)`
2. Authorize Tivoli Workload Scheduler for z/OS:
`PERMIT MVS.* CLASS(OPERCMDs) ID(OPCAPPL) ACC(UPDATE)`

Authority to use the z/OS start command is also required if you use Hiperbatch support for Tivoli Workload Scheduler for z/OS operations.

JESSPOOL If the JESSPOOL class is active and you use the Tivoli Workload Scheduler for z/OS JCC function, you must authorize Tivoli Workload Scheduler for z/OS to access SYSOUT data sets for all jobs in the current plan. One way of doing this is to permit Tivoli

Step 7. Setting up RACF environment

Workload Scheduler for z/OS to access all SYSOUT data sets. To permit Tivoli Workload Scheduler for z/OS to access all SYSOUT data sets, perform these steps on each system where the JCC is started:

1. Define the resource:
`RDEFINE JESSPOOL *.* UACC(NONE)`
2. Authorize Tivoli Workload Scheduler for z/OS:
`PERMIT *.* CLASS(JESSPOOL) ID(OPCAPPL) ACC(ALTER)`

If the `PRIVILEGED` or `TRUSTED` attribute is set in the Started Procedure Table (SPT) entry for Tivoli Workload Scheduler for z/OS, then the address space is authorized to issue any commands and to process spool data sets regardless of what is defined in the resource rules.

For further information, see the *RACF Security Administrator's Quick Reference*.

Authorizing Tivoli Workload Scheduler for z/OS E2E server task to create USS processes

In a RACF environment you can define profiles in the `UNIXPRIV` class to grant RACF authorization for certain z/OS UNIX privileges. If the `UNIXPRIV` class is active, the user ID of the E2E server task (eqqUID, as specified in the `EQQPCS05` job) must have at least `READ` authorization for the `SUPERUSER.FILESYS` and `SUPERUSER.PROCESS.*` profiles, otherwise the user ID cannot create the USS processes.

Make sure that you use a unique UID with a nonzero value; for additional information about this requirement, see `INFO APAR II1423`.

Authorizing Tivoli Workload Scheduler for z/OS E2E and Dynamic Workload Console server tasks for security resource EZB.BINDDVIPARANGE

You must give `UPDATE` authorization for the `EZB.BINDDVIPARANGE` resource to the user ID of the end-to-end server when using DVIPA host names. Specifically, this authorization is always needed when the `TOPOLOGY HOSTNAME` value represents a DVIPA address.

If you use the Dynamic Workload Console, you must give `UPDATE` authorization for the `EZB.BINDDVIPARANGE` to the user ID of the Dynamic Workload Console server when using DVIPA hostnames. Specifically, this authorization is always needed when the `SERVOPTS JSCHOSTNAME` value represents a DVIPA address.

Authorizing Tivoli Workload Scheduler for z/OS Data Store to issue JES commands

If your system has RACF Version 1 Release 9 or later, and JES2 or JES3 Version 3 Release 1 Modification level 3 or later, consider the following resource classes when implementing security for Data Store. The examples assume that the RACF user for the Data Store address space is `OPCDS`, which is the name specified in the started-procedure table.

OPERCMDS If the `OPERCMDS` class is active, the Data Store must be authorized to issue the JES command. One method is to allow the Data Store to issue all JES commands. To do this on a JES system, perform the following steps:

Step 7. Setting up RACF environment

1. Define the resource:

```
RDEFINE OPERCMDS JES2.* UACC(NONE)
```

2. Authorize Data Store:

```
PERMIT JES2.* CLASS(OPERCMDS) ID(OPCDS) ACC(UPDATE)
```

On a JES3 system, replace JES2.* with JES3.* in the example. Alternatively, you could specify the JES%.* resource name for either a JES2 or JES3 system.

Authority to use the z/OS start command is also required if you use Hiperbatch support for Tivoli Workload Scheduler for z/OS operations.

JESSPOOL

If the JESSPOOL class is active, you must authorize the Data Store to access SYSOUT data sets for all jobs managed by the Data Store itself. One way of doing this is to permit the Data Store user to access all SYSOUT data sets. To do this, perform these steps on each system where the Data Store is started:

1. Define the resource:

```
RDEFINE JESSPOOL *.* UACC(NONE)
```

2. Authorize Data Store:

```
PERMIT *.* CLASS(JESSPOOL) ID(OPCDS) ACC(ALTER)
```

If the privileged or trusted attribute is set to in the Started Procedure Table (SPT) entry for the Data Store, then the address space is authorized to issue any command and process spool data sets regardless of what is defined in the resource rules.

See the *RACF Administrator's Guide* for detailed information.

Step 8. Securing communications

Tivoli Workload Scheduler for z/OS supports authentication and cryptography by activating the Secure Sockets Layer (SSL) transport protocol for transmitting and accepting secure information.

You can configure Tivoli Workload Scheduler for z/OS to enable SSL communication in a TCP/IP network or, you can implement SSL security for HTTP connections as required.

Security for TCP/IP connections

The scheduler authentication mechanism uses the SSL services of z/OS. For further details, see *z/OS Cryptographic Services System Secure Sockets Layer Programming*.

To enable SSL authentication for your network, perform the following actions:

1. Create the SSL work directory by using the EQQPCS10 sample JCL. You can use the same directory as the one used for SSL in E2E. In the following examples, the directory is: /u/tws/ssl

2. From /u/tws/ssl/ as current directory, open a shell prompt, start the **gskkyman** utility of z/OS Cryptographic Services System SSL, and do the following:

- a. Create the keystore database and consider protecting it from unauthorized access, because it has to contain private key and trusted certificates. For example, consider the following database: /u/tws/ssl/TWS1.kdb.

Step 8. Setting up the SSL environment

- b. Generate a password file and store it in the SSL directory defined in the previous step, for example /u/tws/ssl/TWS1.sth.
- c. At this point you can:
 - Create a certificate request and send it to the Certificate Authority.
 - Store the signed certificate in the database.
 - Import the certificate of the Certification Authority which signed your certificate.

In this way you have a database containing both your certificate and Certification Authority's one.

The scheduler uses a default name to identify your certificate; therefore you are not required to set up a multiple database handling. If you need different certificates in order to partition your network from a security point of view, you need different databases. The advantage of this solution is that you can store each database in a different directory, with its own access list.

3. Configure IBM Tivoli Workload Scheduler for z/OS, by specifying the TCPOPTS statement for each component of your network. Consider each component according a client-server model. Typically, a client-server group is composed by the trackers and data stores communicating with the corresponding controller, or by a remote interface communicating with the corresponding server.

When the controller or the server started task communicates with a partner component, the communication is always started by the partner component; therefore the partner acts as a client. Differently from the end-to-end communication, the communicating partners use the same port numbers for both non-SSL and SSL communications.

Specify the same TCPOPTS parameters for all the components in a client-server group.

For a detailed description of the TCPOPTS statement, see *Customization and Tuning*. The following example shows a TCPOPTS definition to activate the SSL support.

```
TCPOPTS
. . .
SSLLEVEL(FORCE)
SSLKEYSTORE('/u/tws/ssl/TWS1.kdb')
SSLKEYSTOREPSW('/u/tws/ssl/TWS1.sth')
SSLAUTHSTRING('OPCMaster')
SSLAUTHMODE(String)
```

1
2
3
4
5

In this example:

- 1** The FORCE keyword enables the SSL communication.
- 2** TWS1.kdb is the database containing the certificate.
- 3** TWS1.sth is the password file to access the database.
- 4** OPCMASTER is the string defined as Common Name (CN) in the certificate.
- 5** The STRING keyword enables the check on the CN string.

When designing your configuration from a security point of view, consider that:

- To enforce your security, you can use the SSLAUTHMODE(String), that requires to:

- Create an SSL certificate for each controller started task. This certificate will be used by the controller and its remote partners. Define the certificate using as Common Name a unique string corresponding to the controller.
- Create an SSL certificate for each server started task. This certificate will be used by the server and its remote partners. Define the certificate using as Common Name a unique string corresponding to the server.

The SSLAUTHSTRING must match the information contained in the certificate sent by the partner. To verify it, you can use the **gskkyman** utility that allows displaying the keys database and SSL certificate content. The certificate CN is returned by **gskkyman** as the first line of the “Subject”.

- If you prefer to use SSLAUTHMODE(CAONLY), then you can use a single SSL certificate for all your network.

Security for HTTP connections

You can provide security for an HTTP connection between the following components:

- The z/OS controller and the Tivoli Workload Scheduler for z/OS Agent.
- The z/OS controller and another z/OS controller (z/OS remote engine).
- The z/OS controller and the dynamic domain manager.
- The z/OS controller and the Tivoli Workload Scheduler master domain manager (distributed remote engine).

SSL-secure connections are implemented using specific settings in the HTTPOPTS initialization statement, and the HTTPS keyword in the ROUTOPTS initialization. For more information about these statements, see *Customization and Tuning*.

If you use the secure connection with the SSL protocol, you must import the security certificates into your security system.

Note: If you imported the default security certificates during the installation of the previous version of the product, you must remove them and run the EQQRCERT job to import the new certificates. If you already imported the new default security certificates during the installation of the Tivoli Workload Scheduler agent for z/OS, then you must not perform this procedure again. Complete the procedure for creating a secure connection by configuring the SSLKEYRING keyword with the value used for installation of the Tivoli Workload Scheduler agent for z/OS.

At installation time, the default security certificates are automatically stored into the SEQQDATA library:

EQQCERCL

The security certificate for the client.

EQQCERSR

The security certificate for the sever.

You can decide to use these default certificates or create your own. In both cases, you must import them into your security system. If you are using RACF, you are provided with the sample job EQQRCERT to import the certificates. To run this job, ensure that you use the same user ID that RACF associates with the controller started task.

If you create your own certificates for an HTTP connection with the master domain manager or with the dynamic domain manager, you must run the customizing

Step 8. Setting up the SSL environment

steps described in the section about customizing SSL connection to the master domain manager and dynamic domain manager in *Tivoli Workload Scheduler: Administration Guide*.

If you are using SSL to communicate with a master domain manager, backup master domain manager, or dynamic domain manager, then the prefix of the common name of the controller certificate must be defined in the **Broker.AuthorizedCNs** option in the `BrokerWorkstation.properties` file located in the `TWA_home/TDWB/config` directory of the distributed engine.

The EQQRCERT job performs the following actions:

- Copies the EQQCERCL certificate to a temporary sequential data set
- Copies the EQQCERSR certificate to a temporary sequential data set
- Imports EQQCERCL to RACF
- Imports EQQCERSR to RACF
- Deletes the temporary sequential data sets
- Creates the SAF key ring that is used to connect the imported certificates
- Updates the RACF database with the new certificates and key ring

Step 9. Allocating data sets

Note: A standby controller uses the same data sets as the controller.

At this stage of the installation of your Tivoli Workload Scheduler for z/OS system, you allocate the data sets that your JCL procedures refer to. You can create the data sets by using the jobs created by the EQQJOBS installation aid.

If you are using the EQQJOBS installation aid, you will already have generated several members in the output library that you specified.

Consider carefully where Tivoli Workload Scheduler for z/OS data sets are allocated in your production environment. Some data sets can be highly active. Avoid placing these data sets on DASD volumes with high activity because this can result in poor performance due to contention. Also consider the ability to recover data sets if a DASD volume becomes unusable. If you place all your data sets on the same volume, you must recover many data sets before you can continue your Tivoli Workload Scheduler for z/OS service. *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265 describes recovery of Tivoli Workload Scheduler for z/OS data sets in detail.

The space to allocate for your data sets 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 space to at least get started. These amounts will be enough for the Tivoli Workload Scheduler for z/OS service for many installations. Use Table 24 on page 97 as a guide to allocate space for VSAM data sets.

The following sections describe the Tivoli Workload Scheduler for z/OS data sets and include examples of the JCL needed to create them.

Allocating the VSAM data sets

Perform this task if you are installing a controller.

Table 23 shows the VSAM data sets and their characteristics. The JCL procedure for the controller uses all of these data sets except for EQQLDDS and EQQLTBKP, which are used only in the planning batch jobs. Allocate all these VSAM data sets for a controller.

Table 23. Tivoli Workload Scheduler for z/OS VSAM data sets

Sample	DD name	Record type	Attributes	Share option	Keys	Record size	Data set
EQQPCS01	EQQADD5	KSDS	UNIQUE SPANNED	3	25 0	1000 131072*	Application description
EQQPCS01	EQQCP1DS	KSDS	REUSE NSPND	3	19 0	200 32000	Current plan 1
EQQPCS01	EQQCP2DS	KSDS	REUSE NSPND	3	19 0	200 32000	Current plan 2
EQQPCS01	EQQCXDS	KSDS	REUSE NSPND	3	64 0	500 32000	Current plan extension
EQQPCS01	EQQXD1DS	KSDS	REUSE NSPND	3	68 0	500 32000	Extended data 1
EQQPCS01	EQQXD2DS	KSDS	REUSE NSPND	3	68 0	500 32000	Extended data 2
EQQPCS01	EQQNXDDS	KSDS	REUSE NSPND	3	68 0	500 32000	New extended data
EQQPCS01	EQQJS1DS	KSDS	REUSE SPANNED	3	28 0	804 180004	JCL repository 1
EQQPCS01	EQQJS2DS	KSDS	REUSE SPANNED	3	28 0	804 180004	JCL repository 2
EQQPCS01	EQQLDDS	KSDS	REUSE SPANNED	2	28 0	440 131072	Long-term-plan work
EQQPCS01	EQQLTBKP	KSDS	REUSE SPANNED	3	28 0	200 131072	Long-term-plan backup
EQQPCS01	EQQLTDS	KSDS	REUSE SPANNED	3	28 0	200 131072	Long-term plan
EQQPCS01	EQQNCPDS	KSDS	REUSE NSPND	3	19 0	200 32000	New current plan
EQQPCS01	EQQNCXDS	KSDS	REUSE NSPND	3	64 0	500 32000	New current plan extension
EQQPCS01	EQQOIDS	KSDS	UNIQUE NSPND	3	28 0	800 32000	Operator instruction
EQQPCS07	EQQPKIxx	KSDS	UNIQUE INDEXED	1,3	34 0	77 77	Primary Index
EQQPCS01	EQQRDDS	KSDS	UNIQUE NSPND	3	64 0	400 32000	Special resource descriptions
EQQPCS07	EQQSDFxx	LINEAR	N/A	2,3	N/A	N/A	Data files
EQQPCS01	EQQSIDS	KSDS	UNIQUE NSPND	3	64 0	110 220	Side information file: ETT and configuration information

Step 9. Allocating data sets

Table 23. Tivoli Workload Scheduler for z/OS VSAM data sets (continued)

Sample	DD name	Record type	Attributes	Share option	Keys	Record size	Data set
EQQPCS07	EQQSKIxx	KSDS	UNIQUE INDEXED	1,3	38 0	76 32000	Secondary Index
EQQPCS01	EQQWSDS	KSDS	UNIQUE NSPND	3	10 0	100 32000	Workstation, calendar, and period descriptions.
EQQPCS01	EQQSCPDS	KSDS	REUSE NSPND	3	19 0	200 32000	Current plan backup copy for Symphony creation and for IBM Tivoli Monitoring integration
Note: <ul style="list-style-type: none"> *The maximum record size for EQQPCS01 is the default maximum value. This can be increased as in the example that follows. In specific situations where the size of the CP files (CP1, CP2, NCP, SCP) are large and the batch daily planning jobs cause a considerable number of updates to the NCP, it is possible for the NCP to become very large. This might require the allocation of additional extents (not additional volumes, since ADDVOL support is not available for the NCP file). Consider freespace allocation for the current plan, included NCP, (EQQCP1DS, EQQCP2DS, EQQCXDS, EQQNCPDS and EQQSCPDS), application descriptions (EQQADDs), resource descriptions (EQQRDDS), and operator instructions (EQQOIDS) data sets. 							

You can allocate the VSAM data sets by submitting the sample listed in Table 23 on page 95. Alternatively, you can allocate one or more of the VSAM data sets by running a job like this:

```
Allocating a VSAM data set
//ALOCVSAM JOB STATEMENT PARAMETERS
/*-----*
/* ALLOCATE AN OPC VSAM DATA SET *
/*-----*
//ALLOC EXEC PGM=IDCAMS,REGION=512K
//SYSPRINT DD SYSOUT=Q
//EQQVOL1 DD DISP=OLD,VOL=SER=volser,UNIT=3390
//SYSIN DD *
DEFINE +
  CLUSTER ( +
    NAME('OPC.INST.AD') UNIQUE +
    SPANNED +
    SHR(3) VOL(volser) CYLINDERS(2 2) +
  ) +
  DATA ( +
    NAME('OPC.INST.ADDATA') +
    KEYS(25 0) RECORDSIZE(1000 132072) +
  ) +
  INDEX ( +
    NAME('OPC.INST.ADINDEX') +
  )
/*
```

This example allocates the application description database.

You can allocate VSAM data sets on different device types.

Allocate enough space for your data sets, depending upon the amount of work Tivoli Workload Scheduler for z/OS processes at your installation. You can use Table 24 on page 97 as a guide to allocate space for VSAM data sets.

Table 24. Calculations of VSAM data set size

Data set	Size in bytes is total of:	
	Number of	Multiplied by
Application description (EQQADDs)	Application and group definitions	208
	Run cycles	120
	Positive run days	3
	Negative run days	3
	Operations	110
	Internal dependencies	16
	External dependencies	84
	Special resources	64
	Operation Extended Information	200
	Variable tables	98
	Variables	476
	Variable dependencies	88
	Extended Name	
Current plan (EQQCPnDS)	Header record (one only)	188
	Workstations	212
	Workstation open intervals	48
	Workstation access method data	72
	Occurrences	302
	Operations	356
	Dependencies	14
	Special resource references	64
	Operation Extended Information	200
	Jobs	116
	Executed steps	20
	Print operations	20
	Unique application names	64
	Operations currently in error	264
	Reruns of an operation	264
	Potential predecessor occurrences	32
	Potential successor occurrences	24
	Operations for which job log information has been collected	111
	Stand-alone clean up	70
	Restart and clean up operinfo retrieved	44
	Number of occurrences	43
Extended data (EQQXDnDS and EQQNXDDS)	Header record (one only)	244
	Bind requests	565
JCL repository (EQQJSnDS)	Number of jobs and started tasks	80
	Total lines of JCL	80
	Operations for which job log information has been collected	107
	Total lines of job log information	143
Note: As a base, calculate a figure for all your jobs and started tasks controlled by Tivoli Workload Scheduler for z/OS. Add to this figure the expected space required for jobs and started tasks in the current plan.		
Long-term plan (EQQLTDS)	Header record (one only)	92
	Occurrences	160
	External dependencies	35
	Operations changed in the LTP dialog	58
Operator instruction (EQQOIDS)	Instructions	78
	Instruction lines	72
Special resource database (EQQRDDS)	Resource definitions	216
	Defined intervals	48
	Entries in the WS connect table	8

Step 9. Allocating data sets

Table 24. Calculations of VSAM data set size (continued)

Data set	Size in bytes is total of:	
	Number of	Multiplied by
Side information file (EQQSIDS)	ETT requests	128
Workstation/calendar (EQQWSDS)	Calendars	96
	Calendar dates	52
	Periods	94
	Period origin dates	6
	Workstation closed dates	80
	Workstations	124
	Workstation access method data	72
	Interval dates	52
	Intervals	32
Note:		
1. Use the current plan data set calculation (EQQCPnDS) for the new current plan data sets (EQQNCPDS and EQQSCPDS).		
2. Use the long-term-plan data set calculation (EQQLTDS) for the long-term-plan work data set (EQQLDDS) and the long-term-plan backup (EQQLTBKP).		
3. Use the special resource database calculation (EQQRDDS) for the current plan extension data set (EQQCXDS) and the new current plan extension (EQQNCXDS).		

Consider the following information when allocating VSAM data sets.

Application description data set (EQQADDs)

The application description data set contains application descriptions and JCL variable tables. This data set is allocated as a spanned data set by EQQPCS01. It has a default maximum record size of 131 072. This allocation limits the variable definitions in a variable table to 275 (131 072/476 = 275), provided there are no variable dependencies. If you also use variable dependencies, the number of variables in a JCL variable table is less than 275.

If you will use a greater number of variable definitions in a variable table, allocate the application description data set with a greater record size. To calculate how great the record size should be, use this method:

$$\text{LRECL} = 86 + (\text{maximum number of variables in one table} * 476) + (\text{number of variable dependencies} * 88)$$

where 86 is the length of the header record, 476 is the length of each variable record and 88 is the length of each variable dependency record.

This VSAM data set must be allocated with share option set to 3 SHR(3). Do not use share option 2 or 1.

Current plan data sets (EQQCPnDS)

The current plan VSAM files are opened and closed many times by Tivoli Workload Scheduler for z/OS during normal processing. If Tivoli Workload Scheduler for z/OS is unable to open one of the files, for example if the file is already opened by another job or TSO user, the normal mode manager (NMM), is terminated. The NMM issues message EQQN027E which reports the reason for the unexpected termination. You can issue a MODIFY command to restart the NMM subtask.

It is recommended that you do not access the current plan files from outside the Tivoli Workload Scheduler for z/OS address space. Backups of current plan information should be taken from the new current plan (EQQNCPDS). Shut down the controller address space if full-pack backups are taken of the volumes where the data sets reside.

Data sets for extended data (EQQXDnDS)

The extended data VSAM files are opened and closed by Tivoli Workload Scheduler for z/OS together with the current plan VSAM files. For this reason, the same considerations for the current plan data sets apply to the data sets for the extended data.

Current plan backup copy data set (EQQSCPDS)

During the creation of the current plan, the SCP data set is used as a CP backup copy for the production of the Symphony file and for the integration with IBM Tivoli Monitoring.

This VSAM data set must be allocated with share option set to 3 SHR(3). Do not use share option 2 or 1.

JCL repository data sets (EQQJSnDS)

Take special care when allocating the JCL repository data sets. The following information describes the allocation and use of these data sets.

Tivoli Workload Scheduler for z/OS maintains its own copy of JCL in the JCL repository data set for every job that it submits in the current plan. It uses a primary and alternate data set for the JCL repository, EQQJS1DS and EQQJS2DS. It reorganizes the JCL repository data set that is in use by copying it to the alternate data set and then switching over to use the newly copied data set. The value you specify on the MAXJSFILE keyword defines whether the JCL repository should be automatically copied and determines how often the automatic copy process should occur.

Use the EQQPCS01 sample job created by the EQQJOBS installation aid to allocate the JS data sets. This job allocates the JS data sets with the SPANNED attribute and maximum record size 180 000. This limits the maximum number of JCL statements to 2 249 for any one job. If you run jobs with a greater number of JCL statements, increase the record size. Calculate the required record size, in bytes, by multiplying the number of JCL statements in your largest job by 80, and add an extra 80 bytes for the header record. If you define your JS file without SPANNED, the greatest maximum record size that you can specify is 32 760 bytes. This lets you store a job with up to 408 JCL records. If you define the JS data sets with SPANNED, the maximum record size you can specify is slightly less than a control area (CA). If you use the EQQUX002 exit, the largest job that can be returned by this exit is 7 599 JCL records. Consider this, when you define the maximum record size of the JS data sets.

Note:

An extended-format data set for VSAM can be allocated for JS data sets that exceed 4 GB.

Operator Instruction data set (EQQOIDS)

The operator instruction (OI) database contains operator instructions, each of which corresponds to an operation in the AD database and provides specific instructions about how this operation has to be handled.

Step 9. Allocating data sets

This VSAM data set must be allocated with share option set to 3 SHR(3). Do not use share option 2 or 1.

Allocating Restart and Cleanup VSAM data sets

Use the EQQPCS07 member generated by the EQQJOBS installation aid. It is contained in the output library specified on the CREATE SAMPLE JOB JCL panel (EQQJOBS8). Submit the EQQPCS07 job to define and initialize the Restart and Clean Up VSAM files.

Note: You can omit this step if you are migrating from a previous Tivoli Workload Scheduler for z/OS version.

The files are described in the following table:

Table 25. Restart and cleanup VSAM data sets

Sample	DD Name	Rec. Type	Attributes	Share Option	Keys	Record Size	data set
EQQPCS07	EQQPKIxx	KSDS	UNIQUE INDEXED	1, 3	34 0	77 77	Primary Index
EQQPCS07	EQQSDFxx	LINEAR	N/A	2, 3	N/A	N/A	Data files
EQQPCS07	EQQSKIxx	KSDS	UNIQUE INDEXED	1, 3	38 0	76 32000	Secondary Index

Restart and cleanup data sets (EQQPKIxx, EQQSKIxx, and EQQSDFxx)

Every Tivoli Workload Scheduler for z/OS address space that uses the Restart and Cleanup function requires the allocation of a local VSAM repository for the structured information related to each job run.

These data sets have the same structure as the Data Store VSAM files and can be allocated by running the EQQPCS07 sample. Keep in mind that every Tivoli Workload Scheduler for z/OS requires the allocation of a unique local VSAM repository.

Allocating non-VSAM data sets

This section describes the physical sequential (PS) and partitioned (PDS) data sets. Table 26 shows the non-VSAM data sets and their characteristics. Before you allocate the non-VSAM data sets, review the following sections, which contain important information about each of these data sets.

Table 26. Tivoli Workload Scheduler for z/OS non-VSAM data sets

Sample	DD Name	RECFM	LRECL	BLKSIZE	DSORG	Data set
EQQPCS02	AUDITPRT	FBA	133	13300	PS	Input to EQQAUDIT
EQQPCS01	–	U	–	6300	PS	CLIST library (optional)
EQQPCS01	EQQCKPT	U	–	8200	PS	Checkpoint
	EQQDLnn	U	–	6300	PS	Dual job-tracking-log
EQQPCS01	EQQDMSG	VBA	84	3120	PS	Tivoli Workload Scheduler for z/OS diagnostic message and trace
EQQPCS02	EQQDUMP	FB	80	3120	PS	Tivoli Workload Scheduler for z/OS diagnostic

Table 26. Tivoli Workload Scheduler for z/OS non-VSAM data sets (continued)

Sample	DD Name	RECFM	LRECL	BLKSIZE	DSORG	Data set
EQQPCS02	EQQEVD _S / EQQEVD _{nn} / EQQHTTP0	F	100	100	PSU	Event
EQQPCS01	EQQEVLIB	FB	80	3120	PDS	Event-driven workload automation (EDWA) configuration file repository
EQQPCS02	EQQINCWK	FB	80	3120	PS	JCC incident work
EQQPCS01	EQQJBLIB	FB	80	3120	PDS	Job library
EQQPCS01	EQQJCLIB	FB	80	3120	PDS	JCC message table
EQQPCS01	EQQJTABL	F	240	240	PS	Critical job table log file
EQQPCS01	EQQJTARC	U	–	6300	PS	Job-tracking archive
EQQPCS01	EQQJT _{nn}	U	–	6300	PS	Job-tracking-log
EQQPCS01	EQQLOGRC	F	128	128	PS	Joblog and Restart Information pending requests Log data set
EQQPCS02	EQQLOOP	VBA	125	1632	PS	Loop analysis message log
EQQPCS02	EQQMLOG	VBA	125	1632	PS	Message log
EQQPCS01	EQQMONDS	F	160	160	PSU	Monitoring task data set used to store events for IBM Tivoli Monitoring
EQQPCS09	EQQOCPBK	–	–	–	–	Data set to allocate the GDG root. The GDG entry is allocated during DP batch run and contains a backup of the old current plan.
EQQPCS01	EQQPARM	FB	80	3120	PDS	Initialization-statement library
EQQPCS01	EQQPRLIB	FB	80	3120	PDS	Automatic-recovery-procedure library
EQQPCS06	EQQSCLIB	FB	80	3120	PDS	Script library for end-to-end scheduling with fault tolerance capabilities
EQQPCS01	EQQSTC	FB	80	3120	PDS	Started-task submit
EQQPCS01	EQQSUDS/ user-defined	F	820	820	PSU	Submit/release
EQQPCS02	EQQTROUT	VB	32756	32760	PS	Input to EQQAUDIT
EQQPCS06	EQQTWSCS	FB	80	3120	PDSE	Data set for centralized script support in end-to-end with fault tolerance capabilities
EQQPCS06	EQQTWSIN and EQQTWSOU	F	160, 160	160, 160	PSU	Event data sets for end-to-end with fault tolerance capabilities
–	EQQYPARM				PDS/PS	PIF
EQQPCS01 EQQPCS02	SYSMDUMP	F	4160	4160	PS	System dump data set
–	–	FB	80	3120	PS	Job-completion-checker incident log

Step 9. Allocating data sets

You can allocate these non-VSAM data sets using the samples listed in Table 26 on page 100 that are generated by the EQQJOBS installation aid.

Note: The data sets cannot be defined as compressed SMS data sets. If you have not tailored the members as described on page 94, you can allocate a partitioned data set by running a job like this:

```
Allocating a Tivoli Workload Scheduler for z/OS partitioned data set
//ALLOCPDS JOB STATEMENT PARAMETERS
//*-----*
/* ALLOCATE A PARTITIONED DATA SET *
/*-----*
//ALLOC EXEC PGM=IEFBR14
//SYSUT1 DD DSN=OPCESA.INST.EQQSTC,
// DISP=(,CATLG),
// VOL=SER=volser,
// SPACE=(TRK,(5,0,1)),
// UNIT=3390,
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
```

This example allocates a started-task-submit data set (EQQSTC).

To allocate a Tivoli Workload Scheduler for z/OS sequential data set, you can run a job like the this:

```
Allocating a Tivoli Workload Scheduler for z/OS sequential data set
//ALLOCPDS JOB STATEMENT PARAMETERS
//*-----*
/* ALLOCATE A SEQUENTIAL DATA SET *
/*-----*
//ALLOC EXEC PGM=IEBGENER
//SYSPRINT DD DUMMY
//SYSUT1 DD DUMMY,DCB=(RECFM=F,BLKSIZE=100,LRECL=100)
//SYSUT2 DD DSN=OPCESA.INST.EVENTS,
// DISP=(NEW,CATLG),
// UNIT=3390,
// VOL=SER=volser,
// SPACE=(CYL,3,,CONTIG),
// DCB=(RECFM=F,BLKSIZE=100,LRECL=100,DSORG=PS)
//SYSIN DD DUMMY
```

This example allocates an event data set (EQQEVDS). The IEBGENER utility ensures that the allocated data set has an end-of-file marker in it.

Note: If you allocate Tivoli Workload Scheduler for z/OS data sets using your own jobs, ensure that they have an end-of-file marker in them.

To allocate a Tivoli Workload Scheduler for z/OS partitioned extended data set, you can run a job such as the following one:

```
Allocating an extended partitioned data set
//ALLOPDSE JOB STATEMENT PARAMETERS
//*-----*
/*ALLOCATE A PDSE DATA SET *
/*-----*
//ALLOC EXEC PGM=IEBR14
//SYSUT1 DD DSN=OPCESA.INST.CS,
// DSNTYPE=LIBRARY,
// DISP=(NEW,CATLG),
// UNIT=3390,
// VOL=SER=volser,
// SPACE=(CYL,(1,1,10)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
```


This example allocates a data set for centralized script support (EQQTWSCS) in an end-to-end with fault tolerance capabilities environment.

The following sections describe the Tivoli Workload Scheduler for z/OS non-VSAM data sets. They contain important information to consider when allocating your data sets.

Internal reader data set (EQQBRDS)

When a Tivoli Workload Scheduler for z/OS subsystem is used to submit work, specify the internal reader data set, EQQBRDS, in your started-task procedures. The DD statement must contain the external-writer-data set name, INTRDR, and the class of the internal reader. The class you specify is used as a default message class for jobs that do not have a MSGCLASS parameter specified on their job cards.

Example internal reader DD statement
//EQQBRDS DD SYSOUT=(A,INTRDR)

Checkpoint data set (EQQCKPT)

Tivoli Workload Scheduler for z/OS uses the checkpoint data set to save the current status of the Tivoli Workload Scheduler for z/OS system. If the controller is stopped and then restarted, Tivoli Workload Scheduler for z/OS uses the checkpoint data set to return the system to the same state as when it was stopped, ready to continue processing.

Tivoli Workload Scheduler for z/OS automatically formats the checkpoint data set the first time it is used. In its initial state, the checkpoint data set specifies that a new current plan exists. The new current plan is defined by DD name EQQNCPDS. Tivoli Workload Scheduler for z/OS attempts to copy the new plan and make it the current plan. If the copy is successful, Tivoli Workload Scheduler for z/OS is fully operational. If the copy is not successful, Tivoli Workload Scheduler for z/OS has become active without a current plan.

Notes:

1. A strong relationship exists between the Tivoli Workload Scheduler for z/OS checkpoint data set and the current plan data set. There is also a strong relationship between the event positioning record (EPR) in the checkpoint data set, EQQCKPT, and the tracker event data set, EQQEVDX, referenced in the controller started task procedure, when a DASD connectivity is used. In fact, the EPR is associated with a specific destination and, therefore, also to a specific event data set. If this relationship is broken, the results of the synchronization processing at controller restart can be unpredictable. This is because events could be lost or reprocessed. Ensure that you do not accidentally delete or overwrite the checkpoint data set.
2. To initialize the checkpoint data set, the OPCHOST keyword of the OPCOPTS initialization statement must be set to its default value, that is, OPCHOST(YES), the first time the scheduler is started. With OPCHOST(YES), the NMM initializes the checkpoint data set with FMID and LEVEL corresponding to SSX. The OPCHOST value can then be changed. For example, you can change the value to OPCHOST(PLEX) when the subsystem is used as the controlling system in XCF.

The space allocation for the data set must be at least 15 tracks. This allocation can accommodate 1000 workstation destinations.

Diagnostic data sets (EQQDMSG, EQQDUMP, and SYSMDUMP)

Allocate diagnostic data sets for Tivoli Workload Scheduler for z/OS address spaces, dialog users, batch jobs, and server.

Step 9. Allocating data sets

Diagnostic message and trace data set (EQQDMSG): You should allocate EQQDMSG for each dialog user. You can allocate EQQDMSG either as a SYSOUT data set or as a DASD data set. Usually only a small volume of diagnostic information exists, so an initial allocation of two tracks of DASD should be enough. If EQQDMSG is not defined, output is written to EQQDUMP.

Diagnostic data set (EQQDUMP): The tracker, controller, and server write debugging information to diagnostic data sets when validity checking discovers internal error conditions. When diagnostic information is logged, a 3999 user abend normally accompanies it. For service purposes, always include an EQQDUMP DD statement for every Tivoli Workload Scheduler for z/OS address space, dialog user, batch job, and server.

Diagnostic data sets are usually allocated as DASD data sets, but they can also be allocated to SYSOUT. Usually only a small volume of diagnostic information exists, so an initial allocation of two tracks on DASD should be enough.

Dump data set (SYSMDUMP): EQQPCS02 contains two allocations for the SYSMDUMP data set. For a Tivoli Workload Scheduler for z/OS address space, the data set is allocated with the low-level qualifier SYSDUMP. Allocate a unique SYSMDUMP data set for every Tivoli Workload Scheduler for z/OS address space. For the scheduler server jobs, SYSMDUMP is allocated with the low-level qualifier SYSDUMPS. EQQPCS01 contains the allocation for the SYSMDUMP data set for Tivoli Workload Scheduler for z/OS batch jobs; this data set is allocated with the low-level qualifier SYSDUMPB. The Tivoli Workload Scheduler for z/OS batch jobs can use the same data set. It is allocated with a disposition of MOD in the JCL tailored by EQQJOBS.

Furthermore, SYSMDUMP data sets should be defined with a UACC of UPDATE, that is, WRITE-ENABLED to all user IDs under which a job scheduled by Tivoli Workload Scheduler for z/OS might possibly be submitted. This is because the SUBMIT SUBTASK of the controller or of the tracker which is submitting a given job might abend while running under the user exit EQQUX001 supplied user ID (RUSER user ID) rather than under the user ID associated with the started task. If this occurs, DUMPTASK fails with an ABEND913 if the user ID in control does not have WRITE access to the SYSMDUMP data set.

The UACC of UPDATE access should be defined to all PIF, dialog, and Dynamic Workload Console servers. If a user is not authorized to update the SYSMDUMP data set, and a server failure occurs while running a request for that user, DUMPTASK fails with an ABEND 912. No diagnostic data will be captured.

Event data sets (EQQEVDS, EQQEVDnn, and EQQHTTP0)

Every Tivoli Workload Scheduler for z/OS address space requires a unique event data set. The data set is device-dependent and must have only a primary space allocation. Do *NOT* allocate any secondary space. The data set is formatted the first time it is used. Each time you use the data set, Tivoli Workload Scheduler for z/OS keeps a record of where to start. When the last track of the data set is written, Tivoli Workload Scheduler for z/OS starts writing on the first track again.

Note: The first time Tivoli Workload Scheduler for z/OS is started with a newly allocated event data set, an SD37 error occurs when Tivoli Workload Scheduler for z/OS formats the event data set. Do not treat this as an error.

The data set contains records that describe events created by Tivoli Workload Scheduler for z/OS job-tracking functions. An event-writer task writes to this data

set; an event-reader task reads from it. The job-submit task also uses the event data set to checkpoint its activities, using the first record in the data set (the header record). The submit task in a controller address space takes these checkpoints when the computer workstation is the same system (the workstation destination is blank), so the address space needs the EQQEVDs event data set allocated even if there is no event writer task. When an event writer task is started in the controller address space, it shares the data set with the submit task.

The header record contains checkpoint information for up to 13 workstations per destination. If you plan to have more than 13 workstations defined to use a single destination, you can allocate the event data set with a large logical record length to accommodate the required number. To calculate the record length required, use this formula:

$$\text{LRECL} = (\text{No-of-WS-with-this-destination} * 6) + 22$$

Because the event data set provides a record of each event, events will not be lost if an event-processing component of Tivoli Workload Scheduler for z/OS must be restarted. The submit checkpointing process ensures that submit requests are synchronized with the controller, thereby preventing lost requests caused by communication failures.

Define enough space for a single extent data set so that it does not wrap around and write over itself before an event is processed. Two cylinders are enough at most installations. The space allocation must be at least 2 tracks when the record length is 100. There must be sufficient space in the event data set to accommodate 100 records. Consider this requirement if you will define the event data set with a record length greater than 100. For example if you define an LRECL of 15 000, the minimum space allocation is 34 tracks, which equates to 102 records and an event data set that would wrap around very quickly in most installations.

To aid performance, place the event data set on a device that has low activity. If you run programs that use the RESERVE macro, try to allocate the event data set on a device that is not reserved or where only short reserves are taken. The reserve period must be less than 5 minutes.

If you use the job log retrieval function, consider allocating the event data set with a greater LRECL value than that in Table 26 on page 100. This improves performance because input/output (I/O) operations will be reduced because fewer continuation (type NN) events will be created. You can specify 0, or a value from 100 to 32 000 bytes for LRECL. Any other value will cause the event writer to end, and message EQQW053E will be written to the message log. If you do not specify a value for LRECL or specify 0, the data set will be forced to have an LRECL of 100 when it is opened by Tivoli Workload Scheduler for z/OS. However, the data set must be **unblocked**: the block size must be equal to the logical record length. If you intend to activate job log retrieval function, use one of the these formulas to estimate the LRECL that you should specify:

$$\begin{aligned} &\text{Calculating the optimum LRECL} \\ &\text{LRECL} = ((\text{NN}/\text{EV}) * 20) + 100 \quad \text{OR} \quad \text{LRECL} = (4 * \text{N}) + 100 \end{aligned}$$

In the first formula, NN is the number of continuation events, and EV is the number of all other events. Event types are found in position 21 of the event records. In the second formula, N is the average number of NN events per job. If your calculation yields a value of less than 110, there will be little or no improvement in performance. In this case, you should specify an LRECL value of 100.

Step 9. Allocating data sets

You will probably need to test your system first to get an idea of the number and event types that are created. You can then reallocate the event data set when you have gathered information about the events created at your installation. But, before you reallocate an event data set, ensure that the current plan is completely up-to-date. You must also stop the event writer, and any event reader, that uses the data set.

Note: Do not move Tivoli Workload Scheduler for z/OS event data sets once they are allocated. They contain device-dependent information and cannot be copied from one device type to another, or moved on the same volume. An event data set that is moved will be reinitialized. This causes all events in the data set to be lost. If you have DFHSM or a similar product installed, you should specify that Tivoli Workload Scheduler for z/OS event data sets are not migrated or moved.

Event-driven workload automation configuration file data set (EQQEVLIB)

This data set contains the configuration files required by the event-driven workload automation (EDWA) process. The configuration files, which are created by the EQQRXTRG program, are used by the trackers to monitor the event conditions. The event-driven workload automation configuration file data set is accessed by the controller, which, when configuration files are created or modified, deploy them to the trackers by storing the files into the data set identified by the EQQJCLIB DD card. This is the same data set to which the trackers' JCLs refer.

By using the event-driven workload automation configuration file data set, you can automate and centralize the deployment of configuration files to the trackers without having to use the EQQLSENT macro for each tracker.

Job library data set (EQQJBLIB)

The job library data set contains the JCL for the jobs and started tasks that Tivoli Workload Scheduler for z/OS will submit. It is required by a controller. If you already have a job library that you will use for Tivoli Workload Scheduler for z/OS purposes, specify this data set on the EQQJBLIB statement. If not, allocate one before you start the controller.

Note: Allocate the job library data set with a only primary space allocation. If a secondary allocation is defined and the library goes into an extent when Tivoli Workload Scheduler for z/OS is active, you must stop and restart the controller. Also, do not compress members in this PDS. For example, do not use the ISPF PACK ON command, because Tivoli Workload Scheduler for z/OS does not use ISPF services to read it.

The limitation of allocating the job library data set with only a primary space allocation is not applicable for PDSE data sets.

Note: Each member in the EQQJBLIB must contain one job stream (only one job card), and the job name on the job card must match the job name in the Tivoli Workload Scheduler for z/OS scheduled operation.

Job-completion-checker data sets

You can optionally use the job completion checker (JCC) to scan SYSOUT for jobs and started tasks. Depending on the JCC functions you want to use, allocate at least one of the three data sets associated with the JCC:

JCC-message-table library (EQQJCLIB): If the success or failure of a job or started task cannot be determined by system completion codes, the JCC function can be used to scan the SYSOUT created and set an appropriate error code. You determine how the SYSOUT data is scanned by creating JCC message tables. A general message table (EQQGJCCT) must be defined. Job-specific message tables can be created to search for specific data strings in particular jobs. These tables are stored in the PDS with a member name that matches the job name.

Every Tivoli Workload Scheduler for z/OS subsystem where you start the JCC task must have access to a message table library. If you want, you can use the same message table library for all Tivoli Workload Scheduler for z/OS systems.

If you use the data set-triggering function, the data set-selection table (EQQEVLSST or EQQDSLST) must be stored in EQQJCLIB.

Note: Allocate the JCC message table data set with only primary space allocation. The limitation is not applicable for PDSE data sets.

JCC-incident-log data set: You can optionally use the JCC to write records to an incident log data set. This data set is defined by the INCDSN keyword of the JCCOPTS statement.

When scanning SYSOUT data sets, the JCC recognizes events that you define as unusual. If the EQQUX006 exit is loaded by Tivoli Workload Scheduler for z/OS, the JCC records these events in the incident log data set. The incident log data set can be shared by several JCC tasks running on the same system or on different systems. The data set can also be updated manually or even reallocated while the JCC is active. If the JCC is unable to write to the incident log, the incident work data set is used instead.

JCC-incident work data set (EQQINCWK): Occasionally, the JCC cannot allocate the incident log data set. This can happen if another subsystem or a Tivoli Workload Scheduler for z/OS user has already accessed the data set. In this case, the JCC writes to the incident work file, EQQINCWK, instead. If it is not empty, the work file is copied and emptied each time the incident log data set is allocated.

Job-tracking data sets (EQQJTARC, EQQJTnn, EQQDLnn)

Job-tracking data sets are a log of updates to the current plan. They optionally contain audit trail records. Job-tracking data sets comprise:

- Job-tracking logs (EQQJTnn)
- Dual job-tracking logs (EQQDLnn)
- Job-tracking archive (EQQJTARC)

You must allocate EQQJTARC and at least two job-tracking logs (EQQJT01 and EQQJT02) for a controller. The actual number of JT logs that you should allocate is determined by the value that you specify on the JTLOGS keyword of the JTOPTS initialization statement. If you decide to allocate three job-tracking logs, specify the DD names EQQJT01, EQQJT02, and EQQJT03. If you specified EQQJT01, EQQJT02, and EQQJT04, an error occurs and Tivoli Workload Scheduler for z/OS terminates. Tivoli Workload Scheduler for z/OS uses the job-tracking logs in turn. When a current plan backup is performed, the active log is appended to EQQJTARC data set.

The size of the CP files, JT and JTARC, can become large, but with appropriate tuning of the size and of the DP frequency, they will not allocate additional extents. If necessary, use the allocation of additional extents (not additional

Step 9. Allocating data sets

volumes, since just extent allocation is supported in the shipped JT allocation samples. The JTLOG keyword default defines five job-tracking logs. It is recommended that you specify at least three job-tracking logs. Job-tracking logs are switched at every current plan backup. If the interval between backups is very low and JTLOGS(2) is specified, the previously used job-tracking log might not have been archived before Tivoli Workload Scheduler for z/OS must switch again. If it cannot switch successfully, the normal-mode-manager (NMM) subtask is automatically shut down, preventing further updates to the current plan.

You can optionally allocate dual JT logs. These logs are identified by the EQQDL nn DD names in the controller started-task JCL. Allocate the same number of dual JT logs as JT logs. The numeric suffixes, nn , must be the same as for the JT logs, because Tivoli Workload Scheduler for z/OS uses the logs with the same number: EQQJT01 and EQQDL01, EQQJT02 and EQQDL02, and so on. Tivoli Workload Scheduler for z/OS writes job-tracking information to both logs, so that if the active JT log is lost it can be restored from the dual log, and Tivoli Workload Scheduler for z/OS can be restarted without losing any events. To achieve the maximum benefit from dual JT logs, you should allocate them:

- With the same attributes as the JT logs
- With at least the same amount of space as the JT logs
- On alternate I/O paths and physical volumes than their corresponding JT logs

Tivoli Workload Scheduler for z/OS tries to use dual JT logs if you specify DUAL(YES) on the JTOPTS initialization statement of a controller.

The job-tracking-archive data set accumulates all job-tracking data between successive creations of a new current plan (NCP). So allocate EQQJTARC with enough space for all job-tracking records that are created between daily planning jobs; that is, extend or replan of the current plan. In other words, be sure that you allocate for EQQJTARC an equal or greater amount of space than the total of the space you allocate for the JT files, or you will get a system error. When the daily planning batch job is run, the active job-tracking log is appended to EQQJTARC, and the JT log is switched. The archive log, EQQJTARC, is then copied to the track log data set referenced by the EQQTROUT DD name during the daily planning process. When Tivoli Workload Scheduler for z/OS takes over the NCP, the archive data set is emptied.

Tivoli Workload Scheduler for z/OS recovery procedures that use the job-tracking data sets are described in *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

Message log data set (EQQMLOG)

The message log data set can be written to SYSOUT or a data set. The data control block (DCB) for this data set is defined by Tivoli Workload Scheduler for z/OS as follows:

```
EQQMLOG DCB attributes  
DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
```

If the message log data set becomes full during initialization, or when a subtask is restarted, Tivoli Workload Scheduler for z/OS will abend with error code SD37. In either case, you must stop Tivoli Workload Scheduler for z/OS and reallocate the message log data set with more space. In all other circumstances, if the data set fills up, Tivoli Workload Scheduler for z/OS redirects messages to the system log instead.

Note: The scheduler ABENDs with error code sb37 or sd37 if the message log data set becomes full under any of the following circumstances:

- During initialization
- When a subtask is restarted
- While processing any modify command which requires parsing of initialization parameters or specifies the newnoerr, noerrmem(member), or lstnoerr options

In the last case, the ABEND also occurs if the EQQMLOG is already full when any such command is issued. In all these cases you must reallocate more space to the message log data set. In all the other cases, if the data set fills up, the scheduler redirects messages to the system log instead.

EQQPCS02 contains two allocations for the EQQMLOG data set. For a Tivoli Workload Scheduler for z/OS address space, the data set is allocated with the low-level qualifier MLOG. For the scheduler server jobs, the data set is allocated with the low-level qualifier MLOGS.

Note: If you allocate the message log data set on DASD, define a different data set for Tivoli Workload Scheduler for z/OS batch program. The data set must also be different from the one used by each IBM Tivoli Workload Scheduler for z/OS address space (controller, standby controller, tracker, and server). The data set cannot be shared.

Loop analysis log data set (EQQLOOP)

The loop analysis log data set can be written to SYSOUT or a data set. The data control block (DCB) for this data set is defined by Tivoli Workload Scheduler for z/OS as follows:

```
EQQLOOP DCB attributes  
DCB=(RECFM=VBA,LRECL=125,BLKSIZE=1632)
```

This data set is defined the same way as for EQQMLOG, but it is specific for loop analysis and is populated only if a loop condition occurs. It is required by daily planning batch programs (extend, replan, and trial).

Parameter library (EQQPARM)

Each Tivoli Workload Scheduler for z/OS subsystem reads members of a parameter library when it is started. Parameter library members (residing in library extent), that have been created, cannot be accessed after they have been opened. To avoid this problem, the data set that defines the EQQPARM library should be allocated without any secondary extents. The limitation is not applicable for PDSE data sets. The library contains initialization statements that define runtime options for the subsystem. Allocate at least one parameter library for your Tivoli Workload Scheduler for z/OS systems. You can keep the parameters for all your subsystems in one library, as long as it resides on a DASD volume that is accessible by all systems.

PIF parameter data set (EQQYPARM)

Allocate the PIF parameter data set if you intend to use a programming interface to Tivoli Workload Scheduler for z/OS. The data set can be sequential or partitioned. In the PIF parameter file you specify how requests from the programming interface should be processed by Tivoli Workload Scheduler for z/OS. By defining an INIT initialization statement in the PIF parameter data set, you override the global settings of the INTFOPTS statement.

The initialization statements are described in *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265.

Step 9. Allocating data sets

Automatic-recovery-procedure library (EQQPRLIB)

Allocate a data set for the automatic-recovery-procedure library if you intend to use the Tivoli Workload Scheduler for z/OS automatic-recovery function. The library is used by the ADDPROC JCL rebuild parameter of the JCL recovery statement. This parameter lets you include JCL procedures in a failing job or started task before it is restarted.

Script library for end-to-end scheduling with fault tolerance capabilities (EQQSCLIB)

This script library data set includes members containing the commands or the job definitions for fault-tolerant workstations. It is required in the controller if you want to use the end-to-end scheduling with fault tolerance capabilities. See *Customization and Tuning* for details on the JOBREC, RECOVERY, and VARSUB statements.

Note: Do not compress members in this PDS. For example, do not use the ISPF PACK ON command, because Tivoli Workload Scheduler for z/OS does not use ISPF services to read it.

Started-task-submit data set (EQQSTC)

The started-task-submit data set is used by Tivoli Workload Scheduler for z/OS to temporarily store JCL when a started task is to be started. Use these attributes for this data set:

```
EQQSTC attributes  
SPACE=(TRK,(5,0,1)),  
DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
```

Include an EQQSTC in the JES PROCLIB concatenation on each system where Tivoli Workload Scheduler for z/OS schedules started-task operations. The data set is used as a temporary staging area for the started-task JCL procedure. When the start command has been issued for the task and control for the task has passed to JES, Tivoli Workload Scheduler for z/OS deletes the JCL by resetting the PDS. This means that you never need to compress the data set. For more information, see “Implementing support for started-task operations” on page 117.

Note: Tivoli Workload Scheduler for z/OS does not support partitioned data set extended (PDSE) libraries for a started-task-submit data set.

Submit/release data set (EQQSUDS)

The submit/release data set is device dependent and must have only a primary space allocation. Do **not** allocate any secondary space. The data set is formatted the first time it is used. Each time you use the data set, Tivoli Workload Scheduler for z/OS keeps a record of where to start. When the last track of the data set is written, Tivoli Workload Scheduler for z/OS starts writing on the first track again.

Two cylinders are enough at most installations.

Notes:

1. The first time Tivoli Workload Scheduler for z/OS is started with a newly allocated submit/release data set, an SD37 error occurs when it formats the data set. Expect this, do not treat it as an error.
2. Do not move Tivoli Workload Scheduler for z/OS submit/release data sets once they are allocated. They contain device-dependent information and cannot be copied from one device type to another, or moved on the same volume. A submit/release data set that is moved will be re-initialized. This causes all information in the data set to be lost. If you have DFHSM or a similar product

installed, define Tivoli Workload Scheduler for z/OS submit/release data sets so that they are not migrated or moved.

Centralized script data set for end-to-end scheduling with fault tolerance capabilities (EQQTWSCS)

In an end-to-end with fault tolerance capabilities environment, Tivoli Workload Scheduler for z/OS uses the centralized script data set to temporarily store a script when it is downloaded from the JOBLIB data set to the agent for its submission. Set the following attributes for EQQTWSCS:

```
DSNTYPE=LIBRARY,  
SPACE=(CYL,(1,1,10)),  
DCB=(RECFM=FB,LRECL=80,BLKSIZE=3120)
```

If you want to use centralized script support when scheduling end-to-end with fault tolerance capabilities, you need to use the EQQTWSCS DD statement in the controller and server started tasks. The data set must be a partitioned extended data set.

Input and output events data sets for end-to-end scheduling with fault tolerance capabilities (EQQTWSIN and EQQTWSOU)

These data sets are required by every Tivoli Workload Scheduler for z/OS address space that uses the end-to-end scheduling with fault tolerance capabilities. They record the descriptions of events related with operations running on fault-tolerant workstations and are used by both the End-to-end enabler task and the translator process in the scheduler's server.

The data sets are device-dependent and can have only primary space allocation. Do not allocate any secondary space. They are automatically formatted by Tivoli Workload Scheduler for z/OS the first time they are used.

Note: An SD37 abend code is produced when Tivoli Workload Scheduler for z/OS formats a newly allocated data set. Ignore this error.

EQQTWSIN and EQQTWSOU are wrap-around data sets. In each data set, the header record is used to track the amount of *read* and *write* records. To avoid the loss of event records, a writer task does not write any new records until more space is available when all the existing records have been read.

The quantity of space that you need to define for each data set requires some attention. Because the two data sets are also used for joblog retrieval, the limit for the joblog length is half the maximum number of records that can be stored in the input events data set. Two cylinders are sufficient for most installations.

The maximum length of the events logged in these two data sets, including the joblogs, is 160 bytes. Anyway, it is possible to allocate the data sets with a longer logical record length. Using record lengths greater than 160 bytes does not produce either advantages or problems. The maximum allowed value is 32000 bytes; greater values will cause the E2E task to terminate. In both data sets there must be enough space for at least 1000 events (the maximum number of joblog events is 500). Use this as a reference, if you plan to define a record length greater than 160 bytes. When the record length of 160 bytes is used, the space allocation must be at least 1 cylinder. The data sets must be unblocked and the block size must be the same as the logical record length. A minimum record length of 160 bytes is necessary for the EQQTWSOU data set in order to be able to decide how to build the job name in the symphony file (for details about the TWSJOBNAME parameter in the JTOPTS statement, see *Customization and Tuning*).

Step 9. Allocating data sets

For good performance, define the data sets on a device with plenty of availability. If you run programs that use the RESERVE macro, try to allocate the data sets on a device that is not, or slightly, reserved.

Initially, you might need to test your system to estimate the number and type of events that are created at your installation. When you have gathered enough information, you can then reallocate the data sets. Before you reallocate a data set, ensure that the current plan is entirely up-to-date. You must also stop the end-to-end sender and receiver task on the controller and the translator thread on the server that use this data set. EQQTWSIN and EQQTWSOU must not be allocated multivolume.

Note: Do not move these data sets once they have been allocated. They contain device-dependent information and cannot be copied from one type of device to another, or moved around on the same volume. An end-to-end event data set that is moved will be re-initialized. This causes all events in the data set to be lost. If you have DFHSM or a similar product installed, you should specify that E2E event data sets are not migrated or moved.

Allocating Data Store data sets

At this stage of your installation, use the EQQPCS04 member generated by the EQQJOBS installation aid. It is contained in the output library specified on the CREATE DATA STORE SAMPLES panel (EQQJOBS5). Submit the EQQPCS04 job to define and initialize the Data Store VSAM files.

Note: You can omit this step if you are migrating from a previous Tivoli Workload Scheduler for z/OS version.

The Data Store VSAM files can be of three types:

Unstructured

The type associated to EQQUDFxx; used to save joblogs. These files are allocated only when the Joblog Retrieval option in panel EQQJOBS7 is set to Y.

Structured

The type associated to EQQSDFxx; used to save structured joblog information. These files are required.

KSDS The type used for EQQPKIxx and EQQSKIxx.

They are listed and described in Table 27:

Table 27. Data Store VSAM data sets

Sample	DD Name	Rec. Type	Attributes	Share Option	Keys	Record Size	data set
EQQPCS04	EQQPKIxx	KSDS	UNIQUE INDEXED	1, 3	34 0	77 77	Primary Index
EQQPCS04	EQQSDFxx	LINEAR	N/A	2, 3	N/A	N/A	Data files
EQQPCS04	EQQSKIxx	KSDS	UNIQUE INDEXED	1, 3	38 0	76 32000	Secondary Index
EQQPCS04	EQQUDFxx	LINEAR	N/A	2, 3	N/A	N/A	Data files

For information about how to estimate the size of the Data Store VSAM files, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265.

Allocating data sets for the Dynamic Workload Console reporting feature

Use the EQQPCS09 member generated by the EQQJOBS installation aid and contained in the output library specified on the CREATE SAMPLE JOB JCL panel (EQQJOBS3) to define and allocate:

- The GDG base entry used for old current plan backup which is created during the daily plan batch process, when you specify the BATCHOPTS statement with the JRUNHISTORY parameter set to YES. The GDG data set is identified in the daily planning EXTEND or REPLAN batch job by the EQQOCPBK ddname.
- The VSAM data set where the archiving process copies each generation data set. Allocate the VSAM data set with the same characteristics as the current plan VSAM data set, because it is used to store the old current plan.

For detailed information about the archiving process, see *Tivoli Workload Scheduler for z/OS: Managing the Workload*, SC32-1263.

Allocating the files and directories

The following features use files on UNIX System Services (USS):

- End-to-end scheduling with fault tolerance capabilities.
- End-to-end scheduling with z-centric capabilities, if SSLKEYRINGTYPE is set to USS in the HTTPPTS statement.
- Features running Java utilities:
 - Historical run data archiving for Dynamic Workload Console reporting
 - Event-driven workload automation for data set triggering

By default, the EQQJOBS installation aid sets the following paths for the following directories:

End-to-end with fault tolerance work directory (EQQJOBS8)

/var/TWS/inst

JAVA utilities enablement work directory (EQQJOBS9)

/var/TWS/inst

SSL for TCP/IP connection work directory (EQQJOBSC)

/var/TWS/inst/ssl

By keeping the default directories, if the end-to-end work directory is deleted, the Java and SSL work directories are also deleted. To avoid this problem, set different paths for the different work directories. For example:

End-to-end with fault tolerance work directory (EQQJOBS8)

/var/TWS/E2E

JAVA utilities enablement work directory (EQQJOBS9)

/var/TWS/JAVAUTL

SSL for TCP/IP connection work directory (EQQJOBSC)

/var/TWS/SSL

To create the correct directories and files, run the following sample jobs for each controller that supports the specific feature:

- The EQQPCS05 sample for the end-to-end scheduling with fault tolerance capabilities

Step 9. Allocating data sets

- The EQQPCS08 sample for the historical run data archiving and event-driven workload automation

To run the previous samples, you must have one of the following permissions:

- UNIX System Services (USS) user ID (UID) equal to 0
- BPX.SUPERUSER FACILITY class profile in RACF
- UID specified in the JCL in eqqUID and belonging to the group (GID) specified in the JCL in eqqGID

For the EQQPCS05 sample, if the GID or the UID were not specified in EQQJOBS, you can specify them in the STDENV DD before running the sample. Make sure that you specify a unique UID with a nonzero value; for additional information about this requirement, see INFO APAR II1423.

The user must also have the /bin/sh login shell defined in his OMVS section of the RACF profile. Make sure that the login shell is set as a system default or use the following TSO command to define it:

```
ALTUSER username OMVS(PROGRAM('/bin/sh'))
```

To check the current settings:

1. Run the following TSO command:

```
LISTUSER username OMVS
```
2. Look in the PROGRAM line of the OMVS section.

After running EQQPCS05, you find the following files in the work directory:

localopts

Defines the attributes of the local workstation (OPCMaster) for batchman, mailman, netman and writer processes and for SSL. The parameters that have no effect in an end-to-end environment are indicated and commented out. For information about customizing this file, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

mozart/globalopts

Defines the attributes of the Tivoli Workload Scheduler network (OPCMaster ignores them).

Netconf

Netman configuration files

TWSCCLog.properties

Defines attributes for the trace function.

You will also find the following directories in the work directory:

- mozart
- pobox
- stdlist
- stdlist/logs contains the USS processes logs files

After running EQQPCS08, you find the following file in the work directory:

java/env.profile

Defines the environmental variable required by the Java utilities.

Configuring for end-to-end scheduling with fault tolerance capabilities in a SYSPLEX environment

In a configuration with a controller and no stand-by controllers, define the end-to-end server work directory in a file system mounted under either a system-specific HFS or a system-specific zFS.

Then configure the Byte Range Lock Manager (BRLM) server in a distributed form (see following considerations about BRLM). In this way the server will not be affected by the failure of other systems in the sysplex.

Having a shared HFS or zFS in a sysplex configuration means that all file systems are available to all systems participating in the shared HFS or zFS support. With the shared HFS or zFS support there is no I/O performance reduction for an HFS or zFS read-only (R/O). However, the intersystem communication (XCF) required for shared HFS or zFS might affect the response time on read/write (R/W) file systems being shared in a sysplex. For example, assume that a user on system SYS1 issued a read request to a file system owned R/W on system SYS2. Using shared HFS or zFS support, the read request message is sent via an XCF messaging function. After SYS2 receives the message, it gathers the requested data from the file and returns the data using the same request message.

In many cases, when accessing data on a system which owns a file system, the file I/O time is only the path length to the buffer manager to retrieve the data from the cache. On the contrary, file I/O to a shared HFS or zFS from a client which does not own the mount, requires additional path length to be considered, plus the time involved in the XCF messaging function. Increased XCF message traffic is a factor which can contribute to performance degradation. For this reason, it is recommended for system files to be owned by the system where the end-to end server runs.

In a configuration with an active controller and several stand-by controllers, make sure that all the related end-to-end servers running on the different systems in the Sysplex have access to the same work directory.

On z/OS systems, the shared ZFS capability is available: all file systems that are mounted by a system participating in shared ZFS are available to all participating systems. When allocating the work directory in a shared ZFS you can decide to define it in a file system mounted under the system-specific ZFS or in a file system mounted under the sysplex root. A system-specific file system becomes unreachable if the system is not active. To make good use of the takeover process, define the work directory in a file system mounted under the sysplex root and defined as automove.

The Byte Range Lock Manager (BRLM) locks some files in the work directory. The BRLM can be implemented:

- With a central BRLM server running on one member of the sysplex and managing locks for all processes running in the sysplex.
- In a distributed form, where each system in the sysplex has its own BRLM server responsible for handling lock requests for all regular files in a file system which is mounted and owned locally (see APARs OW48204 and OW52293).

If the system where the BRLM runs experiences a scheduled or unscheduled outage, all locks held under the old BRLM are lost. To preserve data integrity, further locking and I/O on any opened files is prevented until files are closed and reopened. Moreover, any process locking a file is terminated.

Step 9. Allocating data sets

To avoid these kinds of error in the end-to-end server, before starting a scheduled shutdown procedure for a system, you must stop the end-to-end server if either or both of the following conditions occurs:

- The work directory is owned by the system to be closed
 - The **df -v** command on OMVS displays the owners of the mounted file systems
- The system hosts the central BRLM server
 - The console command **DISPLAY OMVS,O** can be used to display the name of the system where the BRLM runs. If the BRLM Server becomes unavailable, then the distributed BRLM is implemented. In this case the E2E server needs to be stopped only if the system which owns the work directory is stopped.

The server can be restarted after a new system in the sharing has taken the ownership of the file system and/or a new BRLM is established by one of the surviving systems.

To minimize the risk of filling up the Tivoli Workload Scheduler internal queues while the server is down, schedule the closure of the system when the workload is low.

A separate file system data set is recommended for each stdlist directory mounted in R/W on `/var/TWS/inst/stdlist`, where *inst* varies depending on your configuration.

When you calculate the size of a file, consider that you need 10 MG for each of the following files: `Intercom.msg`, `Mailbox.msg`, `pobox/tomaster.msg`, and `pobox/CPUDOMAIN.msg`.

You need 512 bytes for each record in the Symphony, Symold, Sinfonia, and Sinfold files. Consider a record for each CPU, schedule, and job/recovery job.

You can specify the number of days that the trace files are kept on the file system using the parameter `TRCDAYS` in the `TOPOLOGY` statement.

Step 10. Creating JCL procedures for address spaces

Perform this task for a tracker, Data Store, controller, or standby controller.

You must define a JCL procedure or batch job for each Tivoli Workload Scheduler for z/OS address space.

See “Defining subsystems” on page 75 for details.

The `EQQJOBS` dialog generates several members in the output library that you specified. The following table lists the members that provide samples for the scheduler’s address spaces:

Table 28. Started task JCL samples for Tivoli Workload Scheduler for z/OS address spaces

Address Space for:	Member
Controller and tracker	EQQCON (sample started task) EQQCONP (sample started task parameters)
Controller	EQQCONO (sample started task) EQQCONOP (sample started task parameters)

Step 10. Creating JCL procedures for address spaces

Table 28. Started task JCL samples for Tivoli Workload Scheduler for z/OS address spaces (continued)

Address Space for:	Member
Tracker	EQQTRA (sample started task) EQQTRAP (sample started task parameters)
Server	EQQSER (sample started task) EQQSERP (sample started task parameters)
Data Store	EQQDST (sample started task) EQQDSTP (sample started task parameters)

These members contain started task JCL that is tailored with the values you entered in the dialog. Tailor these members further, according to the data sets you require. Alternatively, you can copy a member from the SEQQSAMP library to one of your own libraries, and tailor it manually.

If you create a new library for your Tivoli Workload Scheduler for z/OS started-task procedures, remember to specify the library in the JES PROCLIB concatenation. Then you must restart JES to include the new library.

If you prefer, you can run Tivoli Workload Scheduler for z/OS as a batch job rather than as a started task. Here, the JCL can reside in any library and will require a job card, besides the JCL requirements in Table 29 on page 118.

Implementing support for started-task operations

The JCL procedures for started-task operations started by Tivoli Workload Scheduler for z/OS must be stored in a PDS concatenated on the EQQJBLIB DD name. You can include existing data sets, such as SYS1.PROCLIB, if you prefer. Preparation, tailoring, and variable substitution are handled the same way as for batch job operations. When a started-task operation is started by Tivoli Workload Scheduler for z/OS, the JCL procedure is written to the started-task-submit data set (EQQSTC) on the system where the operation is to be run. Tivoli Workload Scheduler for z/OS issues a START command for this procedure and then removes the JCL procedure from the EQQSTC data set.

JES2 users should specify the started-task-submit data set on the PROC nn DD statement of the JES2 JCL procedure on each z/OS system. The suffix nn is the value specified for the PROCLIB parameter of the STCCLASS statement in JES2PARM. To ensure that the correct version of the JCL procedure is started, place the EQQSTC data set first in the concatenation.

JES3 users should specify the started-task-submit data set on the IATPLB nn DD statement of the JES3 global system. The suffix nn is the value specified in the JES3 standards parameter STCPROC. To ensure that the correct JCL procedure will be started, place the EQQSTC data set first in the concatenation. For each submit task that is running on a JES3 local system in the JES3 complex, also include that data set in the JES3 global concatenation.

If you do not use the Restart and Cleanup function, you must follow the previous instructions to work with started-task operations. Otherwise, because the Restart and Cleanup function adds a job card to the procedures for scheduled STC workstation operations at the same time that it adds the //TIVDSTxx output JCL statements, there are some exceptions to the previous instructions if you want to use the Restart and Cleanup function. The JCL for a started task can contain a job

Step 10. Creating JCL procedures for address spaces

card *only* if the JCL is in a data set in the IEFPSI or IEFJOBS concatenations of MSTJCLxx when the start command is issued.

You must add the EQQSTC data set to the IEFPSI DD statement in MSTJCLxx instead of to the JES2 PROCnn or the JES3 global IATPLBnn DD statement as mentioned above.

In addition, all data sets listed in IEFPSI must be included in the system master catalog.

Notes:

1. To include EQQSTC, you must restart JES.
2. Do not use the BLDL parameter of the JES3 PROC statement to specify the procedure name of a started task that is to be scheduled by Tivoli Workload Scheduler for z/OS.

The EQQSTC data set can be shared by Tivoli Workload Scheduler for z/OS subsystems that run on the same z/OS image. If you use *global resource serialization* (GRS), the EQQSTC data set can be shared by all z/OS systems defined in the GRS ring if you propagate requests for the resource. To propagate the resource requests to all systems in the ring, define the resource SYSZDRK.data data set name in the SYSTEM inclusion RNL of the GRSRNLnn member of SYS1.PARMLIB. For more information about defining the GRS resource name list, see *z/OS Initialization and Tuning Reference*.

Required data sets

Table 29 shows the data sets required by a Tivoli Workload Scheduler for z/OS started task. Include the data sets in your JCL procedures as indicated in this table.

Table 29. Tivoli Workload Scheduler for z/OS required data sets

DD Name	Required by				Defines
	Controller	Tracker	Server	Data Store	
EQQADDS	✓				Application descriptions and JCL variable tables
EQQBRDS	✓	✓			A JES internal-reader
EQQCKPT	✓				Checkpoint data set
EQQCP1DS	✓				Primary current plan
EQQCP2DS	✓				Alternate current plan
EQQCXDS	✓				Current plan extension
EQQEVDS	✓	✓			Event data set for the submit checkpointing function and for the event-writer task
EQQEVLIB	✓				Configuration file repository for event-triggered resource handling
EQQJBLIB	✓				JCL PDS libraries
EQQLOGRC	✓				Joblog and Restart Information pending requests log data set
EQQJS1DS	✓				Primary JCL repository
EQQJS2DS	✓				Alternate JCL repository

Step 10. Creating JCL procedures for address spaces

Table 29. Tivoli Workload Scheduler for z/OS required data sets (continued)

DD Name	Required by				Defines
	Controller	Tracker	Server	Data Store	
EQQJTABL	✓				Job table log file. The scheduler considers this data set as required only if you defined at least one critical job. Allocate it with the same size as EQQJTARC.
EQQJTARC	✓				Job-tracking archive
EQQJTnn	✓				Job-tracking logs
EQQLTDS	✓				Long-term plan
EQQMLIB	✓	✓	✓	✓	Message library
EQQMLOG	✓	✓	✓	✓	Output message log
EQQNCPDS	✓				New current plan
EQQNCXDS	✓				New current plan extension
EQQOIDS	✓				Operator instructions
EQQPARM	✓	✓	✓	✓	Parameter library
EQQRDDS	✓				Special resource descriptions
EQQSCPDS	✓				Current plan backup copy data set for the creation of Symphony. Needed for integration with IBM Tivoli Monitoring.
EQQSIDS	✓				Side information; ETT criteria and configuration data
EQQWSDS	✓				Workstation, calendar and period descriptions

Notes:

1. The data sets that are required for a controller are also required for a standby controller.
2. The number of job-tracking-log data sets to include depends on the value that you specify in the JTLOGS keyword of the JTOPTS initialization statement. Specify at least 3 job-tracking logs. The default value is 5.
3. You must specify EQQEVDS for a controller even if an event writer is not started in the controller address space. The EQQEVDS data set is used for submit checkpointing. It can be the same data set that is used by an event-writer function. Use a unique EQQEVDS for each address space.
4. In order to set the TCP/IP task up correctly, you need to change the scheduler start procedure to include the C runtime libraries (CEE.SCEERUN in the STEPLIB DD statement).

If you have multiple TCP/IP stacks, or if the name you used for the procedure that started up the TCPIP address space was not the default (TCPIP), then you need to change the start procedure to include the SYSTCPD DD card to point to a data set containing the TCPIPJOBNAME parameter.

The standard method to determine the connecting TCP/IP image is:

- Connect the TCP/IP specified by TCPIPJOBNAME in the active TCPIP.DATA
- Locate TCPIP.DATA using the SYSTCPD DD card

Step 10. Creating JCL procedures for address spaces

Optional data sets

Table 30 shows the data sets that you can optionally include in your JCL procedures. Specify these data sets only if you want to use the function with which they are associated.

Table 30. Tivoli Workload Scheduler for z/OS optional data sets

DD Name	Can be used by				Defines
	Controller	Tracker	Server	Data Store	
AUDITPRT	✓				Input to EQQAUDIT
EQQDLnn	✓				Dual job-tracking logs
EQQDUMP	✓	✓	✓		Diagnostic dump output
EQQEVDnn	✓	✓			Event data set for an event-reader task
EQQINCWK		✓			JCC incident work file
EQQJCLIB		✓			JCC library for message tables and for data set triggering selection table
EQQMONDS	✓				Data set used by monitoring task to store events for IBM Tivoli Monitoring.
EQQPKIxx	✓				Primary index
EQQPRLIB	✓	✓			Automatic-recovery procedures
EQQSCLIB	✓				Scriptlib
EQQSDFnn	✓				Structured data files
EQQSKIxx	✓				Secondary index
EQQSTC	✓	✓			Started-task-submit data set
EQQSUDS		✓			Submit/release data set for an event-writer task
EQQTROUT	✓				Input to EQQAUDIT
EQQTWSCS	✓		✓		Data set for centralized script support in end-to-end scheduling with fault tolerance capabilities
EQQTWSIN	✓		✓		Input event data set in end-to-end scheduling with fault tolerance capabilities
EQQTWSOU	✓		✓		Output event data set in end-to-end scheduling with fault tolerance capabilities
EQQUDFnn				✓	Unstructured data files
STDENV			✓		This data set/member contains the environment variables of the end-to-end with fault tolerance capabilities processes
STEPLIB	✓	✓	✓		Load-module library
SYSMDUMP	✓	✓	✓		Dump data set
<i>user-defined</i>	✓				Submit/release data set for the controller submit task

Step 10. Creating JCL procedures for address spaces

Table 30. Tivoli Workload Scheduler for z/OS optional data sets (continued)

DD Name	Can be used by				Defines
	Controller	Tracker	Server	Data Store	
Note:					
<ol style="list-style-type: none">1. The optional data sets that you specify for a controller must also be specified for a standby controller.2. If you use dual job-tracking, the number of dual job-tracking logs (EQQDLnn) must be the same as the number of job-tracking logs (EQQJTnn).3. Include EQQDUMP and SYSMDUMP for diagnostic purposes.4. The EQQEVDnn DD name identifies the event data set for an event-reader task. The nn value is the sequence number specified in the ERSEQNO keyword of the event reader that will process this data set. It is always a 2-digit number. That is, if the sequence number is less than 10, a leading 0 must be added.5. Specify the EQQSTC data set if you use Tivoli Workload Scheduler for z/OS to schedule started-task operations.6. Use the standard JCL naming conventions for each user-defined DD name; that is, 1-8 alphanumeric or national characters, of which the first character must be alphabetic or national.7. The submit/release data set is identified by a controller, with a user-defined DD name. The same name must appear in the procedure JCL, the DASD keyword of the ROUTOPTS statement, and the destination field of the workstation representing the system that work is to be sent to. The same data set is identified in a tracker, by the EQQSUDS DD name.8. When using end-to-end functions, the same EQQTWSIN, EQQTWSOU, and EQQTWSCS data sets must be allocated to the controller and the end-to-end server.9. The STDENV DD name can point to a sequential DS or a PDS member (for example, a member of the PARMLIB) in which the user can define environment variables to initialize Language Environment®. STDENV must have a F or FB format with a record length equal or greater than 80. In this data set/member you can put your environment variables specifying VARNAME=value. On each row you can specify only 1 variable, characters after column 71 are ignored. If you need more than 71 characters, you can add any character in column 72 and continue on the next row (the character in column 72 is ignored).10. THE EQQTROUT DD card must point to a dataset or be dummy, but it cannot be removed from the daily plan jcl. In particular if a cp extend or replan job is submitted with the EQQTROUT DD deleted or commented out, the DRTOP/DNTOP JOBSTEP can end with RC08, even if a new plan is created and taken over, because EQQTROUT could not be opened. Also the dataset pointed by the EQQTROUT DD must be allocated by using the DCB values provided in the allocation JCL sample: RECFM=VB,LRECL=32756,BLKSIZE=32760 otherwise the contents of the dataset will be unreadable.					

Step 11. Defining the initialization statements

In this step of your installation, you define the initialization statements.

When Tivoli Workload Scheduler for z/OS starts running, it reads the parameter library to determine initialization options and parameters. The parameter library is specified by the EQQPARM DD statement in the Tivoli Workload Scheduler for z/OS started-task procedure.

The initialization statements that you should define depend on the functions of Tivoli Workload Scheduler for z/OS that you want to use. For details about how to define initialization statements, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

Step 12. Creating the DB2 database

In this optional step, required only if you need the history function, you create the DB2 database, tables, and indexes. If you need to migrate the history data that you have already defined, then run only the EQQICNVH sample. You can use the history function to rerun operations after they have completed and are no longer in the current plan. When the history function is active, details about completed operations are copied to the DB2 database when you extend the current plan, and remain in the database for a period that you specify. See *Tivoli Workload Scheduler for z/OS: Managing the Workload*, SC32-1263 for a description of how to use the history function, and *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265 for a description of the necessary initialization parameters.

Note: Do not stop DB2 with the quiesce option if the scheduler history function is implemented. A quiesce of DB2 could cause any dialog user to hang until DB2 termination is complete. Use the +stop db2,force command, instead.

Edit and run the supplied sample job EQQINIDB. This job:

1. Binds the DB2 plan.
2. Grants authorities.
3. Creates the database.
4. Creates the tablespace.
5. Creates the tables and indexes.

Save the output of this job, because this lists the objects that were created, with their parameters. Ensure that you bind the plan and grant the necessary authorities after applying service to Tivoli Workload Scheduler for z/OS.

Sample to migrate the history database

Use the EQQICNVH sample to migrate the Tivoli Workload Scheduler for z/OS Operation History data from one release to another.

EQQICNVH is a job with the following steps:

1. **IDCAMS**

Makes it possible to rerun the job by deleting the data sets created in the previous run.

2. **UNLOAD**

Uses the DB2 utility named DSNTIAUL to unload the four tables with the Operation History data, the main table OPCMAIN, the occurrence table OPCOCC, the operation table OPCOPR and the joblog table OPCJL into the

UNLOAD data sets. DSNTIAUL also creates the control statements required to later load the unloaded tables in the x.HISTMIG.CNTL data set.

3. EDIT

Modifies the LOAD control statements created in the preceding step to make the LOAD step add table entries to the existing tables instead of replacing the tables. The RESUME YES parameter is added.

4. CONVERT

Reads the entries of the unloaded tables. You specify the input release and the input subsystem name, and the output release and the output subsystem name. The entries in the UNLOAD data sets for the input release and input subsystem are converted to the record layouts of the output release and given the output subsystem name. The converted table entries are written to the LOAD data sets. The input and output releases can be the same.

5. LOAD

Uses the DB2 LOAD utility to add the new table entries to the Operation History tables; that is, the entries created by the CONVERT step.

6. RESET

Uses the DB2 REPAIR utility to clean the Operation History tables after the processing of the LOAD step. This step is necessary because LOG NO is specified in the DB2 LOAD utility control statements.

After the job successfully run, the following data sets are created:

- x.UNLOAD.EQQHIMN
- x.UNLOAD.EQQHI3P
- x.UNLOAD.EQQHI3C
- x.UNLOAD.EQQHI14
- x.LOAD.EQQHIMN
- x.LOAD.EQQHI3P
- x.LOAD.EQQHI3C
- x.LOAD.EQQHI14
- x.HISTMIG.CNTL

After a successful run, the UNLOAD data sets contain a backup of the Operation History tables.

The statements in data set x.HISTMIG.CNTL can be used to recreate the original contents of the Operation History tables. Remove the RESUME YES parameters.

At the end of the process, you must make a BIND of the new version. You can use the BIND step provided with the EQQINIDB sample.

Step 13. Setting up the ISPF environment

Perform this task if you are installing the scheduler dialogs.

Because Tivoli Workload Scheduler for z/OS dialogs run under ISPF, you must set up an ISPF environment. If you are not familiar with ISPF dialogs, see *ISPF Guide and Reference* and *ISPF Examples*.

To set up your ISPF environment, perform these steps:

1. Set up the Tivoli Workload Scheduler for z/OS CLIST library.
2. Set up the ISPF tables.

Step 13. Setting up ISPF environment

3. Allocate ISPF and Tivoli Workload Scheduler for z/OS data sets to the TSO session.
4. Invoke the Tivoli Workload Scheduler for z/OS dialog.

These steps are described in the following sections.

Setting up the CLIST library

When you ran the SMP/E apply job, the scheduler CLIST library was copied to a data set allocated to DD name SEQQCLIB. Allocate this data set to the SYSPROC DD name of the TSO logon procedure JCL. This library includes the EQQXSUBC CLIST, which is used by the Tivoli Workload Scheduler for z/OS dialog when a user requests a Tivoli Workload Scheduler for z/OS background batch job to be submitted.

For the online EQQAUDIT to work, either copy EQQAUDNS into a library that is part of the TSO SYSPROC concatenation or add the batch-job skeleton library, which is created by EQQJOBS, into the SYSPROC concatenation.

Setting up the ISPF tables

These are the tables in the SEQQTBL0 library that you must allocate to the ISPF table library (ISPTLIB):

EQQACMDS	ISPF command table
EQQAEDIT	Default ISPF edit profile
EQQELDEF	Default ended-in-error-list layouts
EQQEVERT	Ended-in-error-list variable-entity read table
EQQLUDEF	Default dialog connect table
EQQRLDEF	Default ready-list layouts
EQQXVART	Dialog field definitions

If you use the ISPF command table EQQACMDS, invoke Tivoli Workload Scheduler for z/OS as a separate ISPF application with the name EQQA. “Invoking the Tivoli Workload Scheduler for z/OS dialog” on page 127 describes this in more detail. If you want to use a different ISPF application name, for example EQQB, create a command table with the name EQQBCMDS.

The customization of the ISPF Dialog is affected and depends on the ISPF application names. This makes necessary that you create copies of the EQQACMDS and EQQAEDIT members of SEQQBTLO for each ISPF application and locate these copies in ISPTLIB. For example, for the ISPF application names EQQX and EQQY you need to create the ISPTLIB members EQQXCMDs, EQQYCMDs, EQQXEDIT, and EQQYEDIT.

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 data set allocated to the ISPTABL.

Setting up the default dialog-controller connection table

Table EQQLUDEF contains values used when establishing the connection between the scheduler dialog user and the controller. These are default values set initially for your installation by the system programmer. Individual users can then modify the values to suit their requirements. Modify the table, adding the following information:

- The names of the controllers in your installation
- When a controller is accessed remotely, the combination of the controller name and the LU name of a server set up to communicate with it

Step 13. Setting up ISPF environment

- The set of dialog-controller connections that are to be available to all dialog users

When a user opens the scheduler dialog 0.1, the scheduler first tries to read the connection table `EQQALTCP` in the ISPF profile library `ISPPROF`. The connection table name begins with the `NEWAPPL` ID specified when invoking the scheduler dialog. For example, if the ISPF application name is `EQQB`, the connection table name is `EQQBLCPT`. If you used a different ISPF application name `xxxx`, the connection table name is `xxxxLTCP` (if the application name is shorter than four characters, it is filled with `x` up to length 4). If it cannot find the table, it reads the default connection table `EQQLUDEF` from the `ISPTLIB` allocation.

When a user modifies the connection table (through the scheduler dialog option 0.1), the changes are written to the `EQQALTCP` (or `xxxxLTCP`) table of `ISPPROF`.

To change the distributed `EQQLUDEF` table:

1. Choose the scheduler dialog option 0.1.
2. Set up the dialog-controller connections for the installation.
3. Copy the connection table `EQQALTCP` (or `xxxxLTCP`) from your ISPF profile library to the scheduler table library allocated to `ISPTLIB`, renaming the copy to the default connection table name `EQQLUDEF`.

You can access and work with different controllers from the same TSO session, using ISPF `SPLIT` to start different Tivoli Workload Scheduler for z/OS instances with different ISPF application names. In this case you might want to add more than one option to invoke Tivoli Workload Scheduler for z/OS from the ISPF master application menu, as in the following example:

```
BODY
.
.
.
  1 ..... - .....
  2 ..... - .....
  ..... - .....
OA OPC - Operations Planning and Control A <===
OB OPC - Operations Planning and Control B <===
  ..... - .....
PROC
.
.
.
  1, ....
  2, ....
  .., ....
OA, 'PANEL(EQQOPCAP) NEWAPPL(EQQA)
OB, 'PANEL(EQQOPCAP) NEWAPPL(EQQB)
.
.
.
END
```

Note: Because the value of the ISPF variable `&XOPCNM`. (displayed in the `EQQOPCAP` dialog as "You are communicating with xxxx") and the default controller selected in the 0.1 dialog (`EQQXLUSL`) are stored, respectively, in members `xxxxPROF` and `xxxxLOUT`, make sure that any changes you make to these ISPF profile members are made consistently. For example, if you modify or delete `xxxxPROF`, you must also modify or delete `xxxLOUT`.

Step 13. Setting up ISPF environment

Setting up list tables and graphical attribute tables

The ISPF tables for list layouts, EQQRLDEF and EQQELDEF, are the default tables displayed for all Tivoli Workload Scheduler for z/OS dialog users in your installation. They can be modified to suit an individual user's requirements or you can create new defaults for all users in your installation. Modified tables are stored in the user's ISPF profile library under another member name. *Tivoli Workload Scheduler for z/OS: Customization and Tuning* describes how to modify the default tables for your installation.

GDDM default values are used for graphical attributes. The defaults can be modified to suit the requirements of an individual user or you can create default values for all users. Modified defaults are stored in the EQQAXGRC member of the ISPF profile data set.

When setting up these tables for dialog users, keep the following points in mind:

- When a user requests a graphical display using the GRAPH command, Tivoli Workload Scheduler for z/OS first searches through the ISPPROF library for the EQQAXGRC ISPF table. If it cannot find the table there, the product searches the ISPTLIB library for the table.
- When a user modifies the graphical display attributes (using the ATTR command from within a Tivoli Workload Scheduler for z/OS dialog), the EQQAXGRC ISPF table is written to the ISPPROF library.
- When a user displays an ended-in-error list, Tivoli Workload Scheduler for z/OS first searches for the layout in the EQQELOUT table on ISPPROF. If it cannot find the layout there, the product uses the layout from the EQQELDEF table on ISPTLIB.
- When a user modifies an ended-in-error list layout, the changes are written to the EQQELOUT table.
- When a user displays a ready list, Tivoli Workload Scheduler for z/OS first searches for the layout in the EQQRLOUT table of ISPPROF. If it cannot find the layout there, the product uses the layout from the EQQRLDEF table on ISPTLIB.
- When a user modifies a ready list layout, the changes are written to the EQQRLOUT table.

Allocating dialog data sets to your TSO session

Table 31 describes the ISPF and Tivoli Workload Scheduler for z/OS data sets that you must allocate to the TSO session to run the Tivoli Workload Scheduler for z/OS dialog.

Table 31. ISPF and Tivoli Workload Scheduler for z/OS dialog data sets

DD Name	Tivoli Workload Scheduler for z/OS use	Created by
SYSPROC	CLIST library	SMP/E run (SEQQCLIB)
ISPPROF	User-session defaults, read/write tables	Your existing ISPPROF data set
ISPLIB	Panel library	SMP/E run (SEQQPxxx, SEQQGxxx)
ISPLIB	Message library	SMP/E run (SEQQMxxx)
ISPSLIB	Skeleton JCL library	EQQJOBS option 2
ISPTLIB	Read tables (default)	SMP/E run (SEQQTBL0)
EQQMLIB	Message library	SMP/E run (SEQQMxxx)

Table 31. ISPF and Tivoli Workload Scheduler for z/OS dialog data sets (continued)

DD Name	Tivoli Workload Scheduler for z/OS use	Created by
EQQMLOG	Message log	TSO logon procedure
EQQTmpl	Advanced ISPF panel templates	SMP/E run (SEQQLxxx)

Notes:

1. The xxx suffix represents the national language version supplied with your distribution tape.
2. If you did not install the Tivoli Workload Scheduler for z/OS load modules in a library defined in the LNKSTnn member of SYS1.PARMLIB, also allocate the load-module library to either the STEPLIB or ISPLLIB DD statements. Except for the EQQMNOJ module, the Tivoli Workload Scheduler for z/OS dialog modules need not run APF-authorized. So if EQQMNOJ is not in the LNKSTnn concatenation, you must copy it to another library so that it can be loaded APF-authorized. The product dialog loads EQQMNOJ through IKJEFTSR, therefore you cannot use LIBDEF to add the library containing EQQMNOJ to your STEPLIB or ISPLLIB concatenations.
3. Consider allocating EQQDMSG and EQQDUMP to the TSO session for diagnostic purposes.
4. Ensure that the library containing Tivoli Workload Scheduler for z/OS batch job skeletons, generated by EQQJOBS, is allocated to the ISPSLIB DD statement.
5. You need the EQQMLIB library to run the Tivoli Workload Scheduler for z/OS TSO commands or to use a TCP/IP connected dialog server..
6. You need the EQQMLOG data set to use a TCP/IP connected dialog server.
7. For the online EQQAUDIT to work, either copy EQQAUDNS into a library that is part of the TSO SYSPROC concatenation or add the batch-job skeleton library, which is created by EQQJOBS, into the SYSPROC concatenation.
8. EQQTmpl identifies the libraries where the Advanced ISPF panel templates are loaded. The templates are the predefined layouts available for the advanced ISPF panels.
More views are provided for the same panel, for example, for the EQQMOPRV panel (list of operations in the plan), the templates provided are:

EQQMOPRT	Compact view
EQQMOPLT	Full view
EQQMOPJT	Job Detail view

Invoking the Tivoli Workload Scheduler for z/OS dialog

The following section outlines ways of invoking the Tivoli Workload Scheduler for z/OS dialog.

Using the EQQOPCAC sample CLIST

You can invoke the Tivoli Workload Scheduler for z/OS dialog by using the sample CLIST EQQOPCAC. When you run the sample CLIST in TSO READY mode, EQQOPCAC allocates the dialog data sets and invokes ISPF with the initial master panel EQQ@MSTR. The EQQ@MSTR panel, which is in the Tivoli Workload Scheduler for z/OS panel library, lets you select the applications ISPF/PDF or Tivoli Workload Scheduler for z/OS.

Step 13. Setting up ISPF environment

Modifying an existing ISPF selection menu

You can invoke the Tivoli Workload Scheduler for z/OS dialog by including Tivoli Workload Scheduler for z/OS as an option on your existing ISPF master application menu, or on any other selection menu. The following example shows how to do this. The statements that you insert are marked on the right with an arrow (<====).

```
ISPF-selection-menu modification for Tivoli Workload Scheduler for z/OS
)BODY
:
1 ..... - .....
2 ..... - .....
. .... - .....
0 OPC - Operations Planning and Control <====
. .... - .....
)PROC
:
REQCLEANUP - Created by ActiveSystems 12/14/99 Entity not
defined. = TRANS(TRUNC(REQCLEANUP - Created by ActiveSystems 12/14/99 Entity not
defined.,'.')
1 , ....
2 , ....
. ....
0 , 'PANEL(EQQOPCAP) NEWAPPL(EQQA)' <====
. , ....
:
)END
```

Before you can invoke the Tivoli Workload Scheduler for z/OS dialog, allocate the data sets. You can allocate these data sets through the TSO logon procedure, or by running a CLIST after TSO logon.

Although you can use any name that follows the guidelines already established at your installation, the sample ISPF command table, EQQACMDS, is valid only if you use the ISPF application name EQQA. If you change the application name on the ISPSTART command, remember to create the corresponding ISPF command table in the table library.

Selecting the main menu directly from TSO

You can invoke the Tivoli Workload Scheduler for z/OS dialog by selecting the main menu directly from TSO. You do this from TSO by entering this TSO command:

```
/*Invoking the Tivoli Workload Scheduler for z/OS dialog directly from TSO*/
ISPSTART PANEL(EQQOPCAP) NEWAPPL(EQQA)
```

Using this method to invoke the dialog means that the main menu, panel EQQOPCAP, is the first ISPF panel displayed. If you enter the ISPF command SPLIT, EQQOPCAP is displayed on the alternate screen. With this method, you cannot use ISPF/PDF and Tivoli Workload Scheduler for z/OS dialogs at the same time. This method is therefore suitable for users who require *only* Tivoli Workload Scheduler for z/OS.

Using the ISPF select service

You can invoke the Tivoli Workload Scheduler for z/OS dialog by using the SELECT command from a CLIST or from a program. See your ISPF publications to review these procedures.

Switching to the advanced style for ISPF panels

To use the advanced style for ISPF panels, you need to specify Y in the 0.8 option, SETTING PANEL STYLE. The advanced ISPF panels enable you to get a quick,

at-a-glance scrollable view of the AD and CP operations, with color-coded fields that represent application and operation status, as well as the addition of an Action menu from where you can select administrative tasks to perform. They are provided for the AD application to enable you to list and browse a single AD and also for the CP operation to list and browse a single operation in the plan. All of the commands available for an operation in the current plan are concentrated in the new operation list panel (EQQSOPRV, EQQMOPRV).

Step 14. Using XCF for communication

Include this task when installing a tracker, controller, standby controller, or Data Store that will use XCF for communication.

If you want to use the cross-system coupling facility (XCF) for communication between Tivoli Workload Scheduler for z/OS systems, you must:

- Ensure that XCF startup options are suitable for your Tivoli Workload Scheduler for z/OS configuration
- Include the necessary initialization-statement options for each Tivoli Workload Scheduler for z/OS started task.

XCF groups

A Tivoli Workload Scheduler for z/OS XCF system consists of one controller and one or more trackers defined as members in the XCF group. You can include one or more standby controllers in the group. If you want to connect the Data Store to the controller via XCF, you need to define a specific XCF group for them, different to the one defined to connect the controller to the z/OS tracker. You can also specify more than one Tivoli Workload Scheduler for z/OS group in a sysplex. For example, you might want to have a test and production Tivoli Workload Scheduler for z/OS group in your sysplex.

Tivoli Workload Scheduler for z/OS supports these sysplex configurations:

MULTISYSTEM

XCF services are available to Tivoli Workload Scheduler for z/OS started tasks residing on different z/OS systems.

MONOPLEX XCF services are available only to Tivoli Workload Scheduler for z/OS started tasks residing on a single z/OS system.

Note: Because Tivoli Workload Scheduler for z/OS uses XCF signaling services, group services, and status monitoring services with permanent status recording, a *couple* data set is required. Tivoli Workload Scheduler for z/OS does not support a *local sysplex*.

For more information about setting up and running a sysplex, see *Sysplex Management*

With XCF communication links, the controller can submit workload and control information to trackers that use XCF signaling services. The trackers use XCF services to transmit events to the controller. Tivoli Workload Scheduler for z/OS systems are either ACTIVE, FAILED, or NOT-DEFINED for the Tivoli Workload Scheduler for z/OS XCF complex.

Step 14. Using XCF for communication

Each active member tracks the state of all other members in the group. If a Tivoli Workload Scheduler for z/OS group member becomes active, stops, or terminates abnormally, the other active members are notified. This list describes the actions taken by each started task in the group:

controller	When the controller detects that a tracker member changes to failed state, it stops sending work to the tracker. When it detects that a tracker has become active, it sends work to the tracker system and instructs the tracker to start transmitting event information.
Standby	When a standby controller that is enabled for takeover detects that the controller has changed to failed state, it attempts to become the new controller. If there is more than one standby controller in the group, the first one to detect failure of the controller attempts to take over the controller functions.
Tracker	When a tracker member detects that the controller or standby controller has failed, it stops sending event information. The tracker member continues to collect events and writes them to the event data set. When the controller or standby controller becomes active again it informs the tracker that it is ready to receive events.

XCF runtime options

You specify XCF runtime options in the `COUPLEnn` member of `SYS1.PARMLIB` and change them using `SETXCF` operator commands. “Updating XCF initialization options” on page 80 describes how to change the options in the `COUPLEnn` member.

Initialization statements used for XCF

Tivoli Workload Scheduler for z/OS started tasks use these initialization statements for XCF for controller/tracker connections:

XCFOPTS	Identifies the XCF group and member name for the started task. Include XCFOPTS for each started task that should join an XCF group.
ROUTOPTS	Identifies all XCF destinations to the controller or standby controller. Specify ROUTOPTS for each controller and standby controller.
TRROPTS	Identifies the controller for a tracker. TRROPTS is required for each tracker on a controlled system. On a controlling system, TRROPTS is not required if the tracker and the controller are started in the same address space, or if they use shared DASD for event communication. Otherwise, specify TRROPTS.

Tivoli Workload Scheduler for z/OS started tasks use these initialization statements for XCF for controller/Data Store connections:

CTLMEM	Defines the XCF member name identifying the controller in the XCF connection between controller and Data Store.
DSTGROUP	It defines the XCF group name identifying the Data Store in the XCF connection with the controller.
DSTMEM	XCF member name, identifying the Data Store in the XCF connection between controller and Data Store.
DSTOPTS	Defines the runtime options for the Data Store.

FLOPTS	Defines the options for Fetch Job Log (FL) task.
XCFDEST	It is used by the FL (Fetch Job Log) task to decide from which Data Store the Job Log will be retrieved.

If you did not include these runtime options when you defined the initialization statements, do this now. “Step 11. Defining the initialization statements” on page 122 and *Tivoli Workload Scheduler for z/OS: Customization and Tuning* describe the initialization statements.

Step 15. Activating the network communication function

Include this task when installing a tracker, controller, or standby controller that will use NCF for communication.

If you want to use a VTAM link to connect a tracker to the controller, activate NCF. The controller can then send work to the tracker and receive event information back, using the VTAM link. To achieve this connection, activate NCF in both the controller and the tracker. To do this:

- Add NCF to the VTAM network definitions.
- Add NCF session parameters.
- Activate network resources.

If you want to connect a controller and Data Store using SNA you need different VTAM definitions. NCF is involved only in the tracker connection; the equivalent task in the Data Store connection is the FN task.

Adding NCF to the VTAM network definitions

You must define NCF as a VTAM application on both the controlling system and each controlled system. Before defining NCF, select names for the NCF applications that are unique within the VTAM network.

To define NCF as an application to VTAM:

1. Add the NCF applications to the application node definitions, using APPL statements.
2. Add the application names that NCF is known by, in any partner systems, to the cross-domain resource definitions. Use cross-domain resource (CDRSC) statements to do this.

You must do this for all systems that are linked by NCF.

The application node and the cross-domain resource definitions are stored in the SYS1.VTAMLST data set, or in members of a data set that is in the same concatenation as SYS1.VTAMLST. For a detailed description of defining application program major nodes and cross-domain resources, see *VTAM Resource Definition Reference*.

The following example illustrates the definitions needed for a cross-domain setup between a controller and a tracker.

Notes:

1. Tivoli Workload Scheduler for z/OS requires that the application name and the ACBNAME are the same.
2. IS1MVS1 and IS1MVS2 are only sample names.

At the *controller*:

Step 15. Activating the network communication function

1. Define the NCF controller application. Add a VTAM APPL statement like this to the application node definitions:

```
controller VTAM applications
VBUILD TYPE=APPL
  OPCCONTR APPL  VPACING=10,                      C
                  ACBNAME=OPCCONTR
```

2. Define the NCF tracker application. Add a definition like this to the cross-domain resource definitions:

```
controller VTAM cross-domain resources
VBUILD TYPE=CDRSC
  OPCTRK1  CDRSC  CDRM=IS1MVS2
```

At the *tracker*:

1. Define the NCF tracker application. Add a VTAM APPL statement like this to the application node definitions:

```
Tracker VTAM applications
VBUILD TYPE=APPL
  OPCTRK1  APPL  ACBNAME=OPCTRK1,                  C
                  MODETAB=EQQLMTAB,                C
                  DLOGMOD=NCFSPARM
```

2. Define the NCF controller application. Add a CDRSC statement like this to the cross-domain resource definitions:

```
Tracker VTAM cross-domain resources
VBUILD TYPE=CDRSC
  OPCCONTR CDRSC  CDRM=IS1MVS1
```

IS1MVS1 and IS1MVS2 are the cross-domain resource managers for the controller and the tracker, respectively.

At the *Datastore*:

1. Define the NCF Datastore application. Add a VTAM APPL statement like this to the application node definitions:

```
Datastore VTAM applications
VBUILD TYPE=APPL
  OPCDST1 APPL ACBNAME=OPCDST1,                    C
                  MODETAB=EQQLMTAB,                C
                  DLOGMOD=NCFSPARM
```

2. Define the NCF controller application. Add a CDRSC statement like this to the cross-domain resource definitions:

```
Datastore VTAM cross-domain resources
VBUILD TYPE=CDRSC
  OPCCONTR CDRSC  CDRM=IS1MVS1
```

Adding NCF session parameters

You can define the session parameters for NCF either by adding the sample EQQLMTAB logon-mode table or by using your own table. If you use the sample table, assemble and link-edit the EQQLMTAB table into the SYS1.VTAMLIB library concatenation for all trackers where an NCF transmitter application is defined.

Note that the APPL statement that defines an NCF application at a tracker must contain the logon-mode-table information in the MODETAB and DLOGMOD parameters.

The EQQLMTAB member of the SEQQSKL0 library contains this logon table definition plus the JCL necessary to assemble and link-edit the table:

Step 15. Activating the network communication function

```
EQQLMTAB
//LOGON JOB STATEMENT PARAMETERS
//ASM EXEC PGM=ASMA90,PARM='OBJ,NODECK'
//SYSLIB DD DSN=SYS1.MACLIB,DISP=SHR
//      DD DSN=SYS1.SISTMAC1,DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSLIN DD DSN=&LOADSET,UNIT=SYSDA,SPACE=(80,(250,50)),
// DISP=(,PASS)
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
EQQLMTAB MODETAB
        MODEENT LOGMODE=NCFSPARM,                                C
                FMPPROF=X'04',                                    C
                TSPROF=X'04',                                    C
                PRIPROT=X'F3',                                    C
                SECPRROT=X'F3',                                   C
                COMPROT=X'0000',                                  C
                PSERVIC=X'0000000000000000000000000000',         C
                RUSIZES=X'8787'
        MODEEND
END
//LINK EXEC PGM=IEWL,PARM='XREF,LIST,LET,CALL'
//SYSPRINT DD SYSOUT=*
//SYSLMOD DD DSN=SYS1.VTAMLIB(EQQLMTAB),DISP=SHR
//SYSUT1 DD UNIT=SYSDA,SPACE=(1700,(400,50))
//SYSLIN DD DSN=&LOADSET,DISP=(OLD,DELETE)
```

If you choose to provide session parameters in another table or entry, modify the APPL definitions for the transmitter applications accordingly. Note that NCF uses an LU-type 0 protocol with a recommended minimum RU-size of 500 bytes. Do *not* specify an RU-size smaller than 32 bytes. NCF does not modify the session parameter specified in the LOGMODE table entry in any way.

For a complete description of logon mode tables and the macros that define them, see *VTAM Customization*.

COS table

No class of service (COS) table entry is specified for EQQLMTAB in the sample. Specify a COS entry that is valid in your VTAM environment unless you intend to use the default provided by VTAM.

The routing you specify in the COS entry should be fast and reliable so that unnecessary delays are not introduced in the Tivoli Workload Scheduler for z/OS remote job-tracking function.

Activating network resources

The VTAM network must be active when the NCF application is started so that network resources are available for the NCF sessions. All participating NCF-application minor nodes must be activated before the NCF application is started by the tracker.

You activate VTAM resources by entering the VARY NET command or by specifying automatic activation in the VTAM network-definition procedure used during VTAM startup. You can activate NCF-application minor nodes and CDRSC minor nodes directly, using the VARY ACT command. You can also activate them indirectly by activating their major nodes. See *VTAM Operation* for further information.

Step 15. Activating the network communication function

Diagnostic data set

If you have not already allocated the EQQDUMP diagnostic data set for the tracker or controller, do this now. NCF writes debugging information to this diagnostic data set when internal error conditions are detected. When diagnostic information is logged, the information is normally accompanied by a user abend.

Note: Update the Tivoli Workload Scheduler for z/OS started-task procedure with DD name EQQDUMP if this DD name is not already defined.

Step 16. Using TCP/IP for communication

Include this task when installing a scheduler component that will use TCP/IP for communication.

To use the Transmission Control Protocol/Internet Protocol (TCP/IP) for communication among IBM Tivoli Workload Scheduler for z/OS systems, do the following:

- Ensure that TCP/IP protocol is available and the relative started task is started on your z/OS configuration.
- Include the necessary initialization statement options for each product component.

Initialization statements used for TCP/IP

IBM Tivoli Workload Scheduler for z/OS started tasks use the following initialization statements for connecting the scheduler started tasks through TCP/IP:

ROUTOPTS

To identify all the TCP/IP remote destinations for the controller or standby controller. A ROUTOPTS statement is required for each controller and standby controller.

TRROPTS

To identify the controller for a tracker. A TRROPTS statement is required for each tracker on a controlled system.

FLOPTS

To identify all the TCP/IP data store remote destinations for the controller.

DSOPTS

To identify the controller for a Data Store.

TCPOPTS

An optional statement to specify the TCP/IP options for the local component. To identify the remote partner, use one of the previous statements.

Step 17. Activating support for the API

Include this task when installing a controller, or standby controller that you want to communicate with through the Tivoli Workload Scheduler for z/OS API.

Tivoli Workload Scheduler for z/OS uses LU to LU communication to pass data between an ATP and a subsystem through the API. To use API requests GET, PUT, and DELETE, the LU that the ATP sends requests to (the target LU) must be owned by the controller. For CREATE requests, if the target LU is not owned by a Tivoli Workload Scheduler for z/OS address space where an event-writer task is started, the ATP must send requests so that the events are broadcast on the target

z/OS system. *Tivoli Workload Automation: Developer's Guide: Driving Tivoli Workload Scheduler for z/OS*, SC32-1266 and *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265 describe when a request is broadcast.

To activate support for the API, perform these actions in the order shown:

1. Define VTAM resources.
2. Update APPC options.
3. Activate Tivoli Workload Scheduler for z/OS support for APPC.

If you are installing a standby controller, perform corresponding actions on the standby system.

You might need to refer to one or more of these publications:

- *VTAM Resource Definition Reference*
- *APPC Management*
- *z/OS Initialization and Tuning Reference*
- *Tivoli Workload Automation: Developer's Guide: Driving Tivoli Workload Scheduler for z/OS*, SC32-1266, which documents the API

The actions described here are based on z/OS systems. If you use a later z/OS release, check for enhancements that might make some actions unnecessary.

Defining VTAM resources

Start by defining the associated VTAM resources.

Defining a local LU

Define a local LU in a member in the SYS1.VTAMLST concatenation on the system where you are installing Tivoli Workload Scheduler for z/OS. This example shows how a VTAM APPL statement might be defined:

Local LU definition

```
VBUILD TYPE=APPL
  IS4MEOP4 APPL ACBNAME=IS4MEOP4,          C
              APPC=YES,                     C
              AUTOSES=5,                     C
              DMINWNL=3,                     C
              DMINWNR=6,                     C
              DSESLIM=9,                     C
              MODETAB=APPCMODE,              C
              SECACPT=CONV,                  C
              SRBEXIT=YES,                   C
              VERIFY=OPTIONAL,               C
              VPACING=2
```

The LU is called IS4MEOP4 and uses the logon-mode table APPCMODE.

Before you can establish a session with v, a partner LU must be defined. If a partner TP is run at a different node, ensure that an LU is defined at that node.

The controller subsystem currently have tasks utilizing APPC. The subsystem is defined as one LU node to APPC and VTAM.

Defining logon modes

The logon-mode table, which you specify in the LU APPL definition statement, must be in the SYS1.VTAMLIB concatenation. To enable LU 6.2 communication for z/OS, you need the VTAM logon-mode SNASVCMG. For applications, APPC also requires at least one logon-mode entry other than SNASVCMG. You can create a new logon-mode table or add logon modes to an existing table. The name of the logon-mode table that is used by the LU and the partner LU need not be the same,

Step 17. Activating support for the API

but both LUs must use the same logon-mode names. That is, the logon modes used by these LUs must appear in each table, and they must have the same names. This example of an uncompiled logon-mode table contains three logon modes:

Example logon-mode table

```
APPCMODE  MODETAB
```

EJECT

```

*-----*
* Logmode table entry for resources capable of acting as LU 6.2 *
* devices required for LU management. *
*-----*
SNASVCMG MODEENT
    LOGMODE=SNASVCMG,
    FMPPROF=X'13',
    TSPROF=X'07',
    PRIPROT=X'B0',
    SECPROT=X'B0',
    COMPROT=X'D0B1',
    RUSIZES=X'8585',
    ENCR=B'0000',
    PSERVIC=X'0602000000000000000000000300'
*-----*
* Logmode table entry for resources capable of acting as LU 6.2 *
* devices for PC target. *
*-----*
LU62SYS1 MODEENT
    LOGMODE=LU62SYS1,
    RUSIZES=X'8989',
    SRCVPAC=X'00',
    SSNDPAC=X'01'
*-----*
* Logmode table entry for resources capable of acting as LU 6.2 *
* devices for host target. *
*-----*
APPCHOST MODEENT
    LOGMODE=APPCHOST,
    RUSIZES=X'8F8F',
    SRCVPAC=X'00',
    SSNDPAC=X'01'
MODEEND
END

```

Defining cross-domain resources

If the Tivoli Workload Scheduler for z/OS TP and the partner TP are not running in the same VTAM domain, ensure that their respective LUs can communicate by defining cross-domain resources. In this example, LU name IS1MVS1 is used for the system where the controller is activated, and IS1MVS2 for the system that the partner TP is running on.

On the Tivoli Workload Scheduler for z/OS controlling system:

Partner LU cross-domain resources

VBUILD TYPE=CDRSC

LUMVS2 CDRSC CDRM=IS1MVS2

On the partner system:

```
&opc LU cross-domain resources
```

VBUILD TYPE=CDRSC

LUOPC CDRSC CDRM=IS1MVS1

Updating APPC options

You must update APPC options to associate the Tivoli Workload Scheduler for z/OS scheduler (the subsystem) with the local LU that you defined earlier. Do this by updating the APPCPMnn member of SYS1.PARMLIB. Here is an example of an APPCPMnn member:

```
APPCPMnn example
LUADD                /* Add local LU to APPC config. */
  ACBNAME(IS4MEOP4)  /* Name of LU */
  SCHED(EOP4)        /* Scheduler name/OPC subsys name */
  TPDATA(SYS1.APPCTP) /* Profile data set for this LU */
  TPLEVEL(SYSTEM)    /* TP level for which LU searches */
```

The scheduler name must be the same as the Tivoli Workload Scheduler for z/OS subsystem name. In this example, the subsystem name is EOP4. A side information file is not used by Tivoli Workload Scheduler for z/OS. However, the LU must be associated with a TP profile data set; you need not specify a profile for Tivoli Workload Scheduler for z/OS in the data set because Tivoli Workload Scheduler for z/OS does not use TP profiles.

If you must allocate a TP profile data set, you can run a job such as:

```
//ALTPDSET JOB STATEMENT PARAMETERS
//TPSAMPLE EXEC PGM=IDCAMS
//VOLOUT DD DISP=OLD,UNIT=3380,VOL=SER=volser
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
      DEFINE CLUSTER (NAME(SYS1.APPCTP) -
        VOLUMES(volser) -
        INDEXED REUSE -
        SHAREOPTIONS(3 3) -
        RECORDSIZE(3824 7024) -
        KEYS(112 0) -
        RECORDS(300 150)) -
      DATA -
        (NAME(SYS1.APPCTP.DATA)) -
      INDEX -
        (NAME(SYS1.APPCTP.INDEX))
```

TP profile data sets are VSAM KSDS data sets.

Activating support for APPC

When you have defined the necessary VTAM resources and updated APPC options, you can activate Tivoli Workload Scheduler for z/OS support for APPC. Do this by specifying APPCTASK(YES) on the OPCOPTS statement. Perform this action when you have completed all other actions and before you start to use the Tivoli Workload Scheduler for z/OS API.

Step 18. Activating support for the product dialog and programming interface using the server

Include this task when activating a Tivoli Workload Scheduler for z/OS server. To use the Dynamic Workload Console, see “Step 21. Activating support for Dynamic Workload Console” on page 142.

The Tivoli Workload Scheduler for z/OS dialogs and programming interface can be used on a z/OS system other than the system where the controller is running. A server is required, running on the same z/OS system as the controller.

Step 18. Activating support for product dialog and programming interface

The dialogs and the programming interface on the remote z/OS system communicate with the server using APPC or TCP/IP. The EQQXLUSL help panels flow describes how to activate the communication with the server using the dialog.

The APPC communication requires also the VTAM and APPC definitions that are described in the following sections.

For further information, see the following publications:

VTAM Resource Definition Reference

APPC Management

MVS Initialization and Tuning Reference

See also member EQQVTAMS in the EQQJOBS output library or SEQQSAMP library.

To activate the APPC communication, perform the following steps:

1. The server tasks run on the same system as the controller.
2. On the system where the servers and the controller run, you must define the following LUs:
 - One LU for the server 'without' the BASE keyword, and specifying the server started task as SCHEDULER.
3. On the system where you want to enable TSO users to communicate with Tivoli Workload Scheduler for z/OS via the server interface, you must define one LU with the keywords BASE and SCHED; that is:
 - An APPC MASTER LU 'with' the BASE keyword, and specifying SCHED(ASCH). This LU is not for Tivoli Workload Scheduler for z/OS; it is an APPC requirement that there be a BASE LU with SCHED(ASCH) on every system where APPC is used.
4. When the servers and the APPC address space are all started, you will see messages in the SYSLOG and server EQQMLOGs, stating that communication has been established between the server and APPC. These messages are displayed during the first start and after an IPL.
5. The dialog user then selects option 0.1 and specifies the name of the controller subsystem with which he wants to communicate, and the LUNAME of the server via which that communication is to be routed. For more information on specifying these values, press the PF1 (help) on panel EQQXLUSL.

The Tivoli Workload Scheduler for z/OS dialog code in the TSO logon address space then sends an APPC request which is picked up by the APPC BASE LU on that system and routed to the (server) LU specified in the request. The server then passes the dialog data to the controller across the z/OS subsystem interface, serving as a local proxy for the dialog user. The controller cannot tell whether it is talking to a local ISPF dialog user, or to a remote user via a server.

Defining VTAM resources for the product dialog and program interface using the server

If you intend to use the Tivoli Workload Scheduler for z/OS programming interface or dialog from a remote system, you need to activate the APPC support on the remote system.

Assure that there is a LU defined as default LU for the APPC communication (BASE LU) in the APPCPMnn parmlib member. If none is defined, add it as follows:

Step 18. Activating support for product dialog and programming interface

The Tivoli Workload Scheduler for z/OS dialog and programming interface use the default APPC support defined on the system on which the functions are used. To activate this support:

1. Define a default VTAM APPL supporting APPC:

```
VBUILD TYPE=APPL
APPCOUT APPL APPC=YES
        ACBNAME=APPCOUT
        ...
```

2. Update the APPCPMnn member of SYS1.PARMLIB for the default VTAM APPL defined above:

```
LUADD                                /* Add local LU to APPC config. */
ACBNAME(APPCOUT) /* Name of LU */
SCHED(ASCH)      /* No scheduler associated */
BASE             /* default LU for the system */
TPDATA(SYS1.APPCTP) /* Profile data set for this LU */
TPLEVEL(SYSTEM)  /* TP level for which LU search */
```

3. Add any cross-domain resource definition needed to resolve VTAM addressing.

Defining VTAM resources for the server

Start by defining the associated VTAM resources.

Defining a local LU for the server

Define a local LU in a member in the SYS1.VTAMLST concatenation on the system where you are installing Tivoli Workload Scheduler for z/OS. This example shows how a VTAM APPL statement might be defined:

Local LU for the server definition

```
VBUILD TYPE=APPL
IS4MEOP5 APPL APPC=YES,
        AUTOSES=5,
        DMINWNL=3,
        DMINWNR=6,
        DSESLIM=20,
        MODETAB=APPCMODE,
        SECACPT=ALREADYV,
        SRBEXIT=YES,
        VERIFY=OPTIONAL,
        VPACING=2
```

The LU is called IS4MEOP5 and uses the logon-mode table APPCMODE.

The maximum number of TSO dialog users and PIF programs that can simultaneously access a Tivoli Workload Scheduler for z/OS controller via a single server depends on the DSESLIM parameter of the VTAM LU for that server. Once the specified number of sessions has been established, all subsequent users and PIF programs that try to use that server will hang until one of the existing sessions ends.

The number of servers required by an installation depends on how extensive PIF applications are used. While it can be sufficient with one server for the dialogs, a number of servers can be required for the PIF applications. PIF applications that are frequently used and with long execution time might need separate servers.

Defining logon modes for the server

The logon-mode table, which you specify in the LU APPL definition statement, must be in the SYS1.VTAMLIB concatenation.

The server support requires logon-mode table entries as specified in the following uncompiled example:

Step 18. Activating support for product dialog and programming interface

```
APPCDIA logon-mode table for server
*-----*
* Logmode table entry for the dialogs and the programming interface *
*-----*
APPCDIA MODEENT C
          LOGMODE=APPCDIA, C
          RUSIZE=X'8888', C
          SRCVPAC=X'00', C
          SSNDPAC=X'01', C
          MODEENT C
APPCFIF MODEENT C
          LOGMODE=APPCFIF, C
          RUSIZE=X'8888', C
          SRCVPAC=X'00', C
          SSNDPAC=X'01', C
          MODEENT C
```

The RUSIZE gives a user size for sending buffer of 2048 bytes and a receiving buffer of 4096 bytes.

Updating APPC options for the server

You must update APPC options to associate the server (and controller) with the scheduler that you defined earlier. Do this by updating the a LUADD statement in the APPCPMnn member of SYS1.PARMLIB. Here is an example of an APPCPMnn member:

```
APPCPMnn example
LUADD          /* Add local LU to APPC config. */
  ACBNAME(IS4MEOP5) /* Name of LU */
  SCHED(EOP5)      /* Scheduler name/OPC subsys name */
  TPDATA(SYS1.APPCTP) /* Profile data set for this LU */
  TPLEVEL(SYSTEM)   /* TP level for which LU searches */
```

The scheduler name in LUADD must be the same as the scheduler name of the scheduler server. In this example it is EOP5.

Each server identifies itself to APPC as an APPC scheduler with the same name as the started task name. If the SCHEDULER keyword in the SERVOPTS statement is specified, this name is used instead of the started task name.

Defining VTAM resources in a parallel sysplex

In an installation with a Parallel Sysplex® where the scheduler can start on any of a number of z/OS images, each z/OS image within the parallel sysplex should have the same local LU name for a given server. The same LU name must not exist in any other network interconnected to the parallel sysplex network; identical LU names within network, unique LU names across networks.

For details on the parallel sysplex installation, see “Step 14. Using XCF for communication” on page 129.

For installations with VTAM Version 4 Release 3 the LU name (the APPL statement name) should be given with a wildcard character, in case the scheduler works in a parallel sysplex and is not set up to run on a specific z/OS image. The APPL statement will then become a Model Application Program Definition, for the identically named LUs on the z/OS images where the scheduler might start. The wildcard character should be chosen such that one model definition is set up for the controller and one model definition for each of the servers. The optional ACBNAME parameter must be omitted, the name of the APPL statement is then used as the ACBNAME.

Step 18. Activating support for product dialog and programming interface

For example, say the scheduler can start on z/OS images MVS1 and MVS2 in a parallel sysplex. The LU name for controller OPCB is IS4MOPCB, and there are three servers to handle the communication to OPCB, OPCBCOM1, OPCBCOM2 and OPCBCOM3, with LU names IS4MSV1B, IS4MSV2B and IS4MSV3B. VTAM Version 4 Release 3 is available. The following model definitions could then be used (a '?' in the APPL statement name represents a single unspecified character):

```
IS4MOP?B  APPL APPC=YES,...  
IS4MS?1B  APPL APPC=YES,...  
IS4MS?2B  APPL APPC=YES,...  
IS4MS?3B  APPL APPC=YES,...
```

Note that the wildcard character must be chosen such that the no other VTAM LU name than the intended LU name matches the model definition.

Starting the server

You can start the server by using the z/OS START command, or you can have the controller start and stop the server automatically. In the latter case, include the servers (srv1, srv2, ...) on the OPCOPTS statement in the Tivoli Workload Scheduler for z/OS parameter library.

A SERVOPTS statement is required in the parameters file. All SERVOPTS keywords can be left out and defaulted.

Step 19. Activating support for the end-to-end scheduling with fault tolerance capabilities

To schedule jobs on Tivoli Workload Scheduler distributed fault-tolerant agents, activate the end-to-end scheduling with fault tolerance capabilities. Follow these steps:

1. Run EQQJOBS and specify Y for the END-TO-END WITH FAULT TOLERANCE feature.
2. Allocate the data set running the generated EQQPCS06 sample.
3. Create and customize the work directory by running the generated EQQPCS05 sample.
4. Define CPU configuration and domain organization by using the CPUREC and DOMREC statements in a PARMLIB member (the default member name is TPLGINFO).
5. Define Windows user IDs and passwords by using the USRREC statement in a PARMLIB member (the default member name is USRINFO). To encrypt the passwords, run the EQQE2EPW JCL contained in the sample EQQBENCR JCL generated by EQQJOBS.

If you do not want to set the password through the USRREC statement (either in plaintext or encrypted), define the user and password locally on the Windows workstation by using the users utility, and set LOCALPSW=YES in the TOPOLOGY statement. For detailed information about the users script, see the *Tivoli Workload Scheduler for z/OS: End-to-end Scheduling with Fault Tolerance Capabilities* manual.

6. Define the end-to-end configuration by using the TOPOLOGY statement in a PARMLIB member (the default member name is TPLGPARM). In this statement, specify the following:
 - For the TPLGYMEM keyword, write the name of the member used in step 4.
 - For the USRMEM keyword, write the name of the member used in step 5.

Step 19. Activating support for end-to-end scheduling with fault tolerance capabilities

7. Add the TPLGYSRV keyword to the OPCOPTS statement to specify the server name that will be used for end-to-end communication.
8. Add the TPLGYPRM keyword to the SERVOPTS statement to specify the member name used in step 6. This step activates end-to-end communication in the Server.
9. Add the TPLGYPRM keyword to the BATCHOPT statement to specify the member name used in step 6. This step activates the end-to-end scheduling with fault tolerance capabilities feature in the Daily Planning batch programs.

Activating server support for the end-to-end scheduling with fault tolerance capabilities

Customize the INIT and SERVOPTS initialization parameters to set up the correct server environment. For example:

```
SERVOPTS SUBSYS (OPCX)
          PROTOCOL (E2E)
          TPLGYPRM(TPLGY)
```

For more information, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*.

You can start the server by using the z/OS START command, or you can have the controller start and stop the server automatically. In the latter case, include the server (srv1) in the OPCOPTS statement in the Tivoli Workload Scheduler for z/OS parameter library. The server with TCP/IP support requires access to the C language runtime library (either as STEPLIB or as LINKLIST). If you have multiple TCP/IP stacks, or a TCP/IP started task with a name different from 'TCPIP', then use the TCPIPJOBNAME parameter of the TOPOLOGY statement.

You always have to define OMVS segments for server started tasks.

Step 20. Activating support for the end-to-end scheduling with z-centric capabilities

To schedule jobs on Tivoli Workload Scheduler distributed z-centric agents, activate the end-to-end scheduling with z-centric capabilities. Follow these steps:

1. Define the z-centric agent destinations in the ROUTOPTS initialization statements.
2. Customize the connection parameters in the HTTPOPTS initialization statements.

Note: Use this statement to activate or disable the SSL connection protocol . If you want to disable the SSL connection, you can either:

- Specify neither SSLKEYRING nor SSLPORT keywords.
- Specify SSLPORT(0).

For details about the configuration steps, see *Scheduling End-to-end with z-centric Capabilities*.

Step 21. Activating support for Dynamic Workload Console

Perform this step if you want to use the Dynamic Workload Console to design and run your workload.

Prerequisites

Before using the Dynamic Workload Console, you need to install the console and the Tivoli Workload Scheduler for z/OS connector. The z/OS connector forms the bridge between the console and Tivoli Workload Scheduler for z/OS.

The console communicates with the product through the z/OS connector and the scheduler server by using the TCP/IP protocol. The console needs the server to run as a started task in a separate address space. The server communicates with Tivoli Workload Scheduler for z/OS and passes the data and return codes back to the z/OS connector.

Perform the following tasks:

- Install and configure the Tivoli Workload Scheduler for z/OS connector as described in Part 3, “Tivoli Workload Scheduler for z/OS Connector,” on page 195.
- Install and configure the Dynamic Workload Console as described in Part 4, “Dynamic Workload Console,” on page 223.

Considerations

The security model implemented for the Dynamic Workload Console is similar to that already implemented by other Tivoli products that have been ported to z/OS (namely Tivoli User Administration and Tivoli Security Management).

All versions of the Dynamic Workload Console use WebSphere Application Server to handle the initial user verification. In all cases, however, it is necessary to obtain a valid corresponding RACF user ID to be able to work with the security environment in z/OS.

Note: You cannot control the port from which the Dynamic Workload Console server started task replies to a request from the z/OS connector. The response ports are randomly selected. Therefore, if there is a firewall between the Dynamic Workload Console server and the z/OS Connector, that firewall must permit outgoing traffic from all ports above 1023.

To optimize the thread handling between z/OS connector and the scheduler server, you can group console users by RACF user ID. To define this grouping, associate a list of console users to the same RACF user ID, by editing the **TWSZOSConnConfig.properties** file in the `TWSInstallationPath\ewas\profiles\TIPProfile\properties` directory and setting the last two properties as follows:

```
com.ibm.tws.zconn.usr.mapping.enable=true
com.ibm.tws.zconn.usr.mapping.file=mapping_file_path\mapping_file
```

where *mapping_file* is the name of the file that contains the mapping between console user and RACF user ID, as in the following example:

```
engine=zos1919 user=twuser1,twuser2 zosuser=zos1919user1
user=twuser3,twuser4 zosuser=zos1919user2
```

Activating server support for the Dynamic Workload Console

Customize the INIT and SERVOPTS initialization parameters to set up the correct server environment. For example:

Step 21. Activating support for Dynamic Workload Console

```
SERVOPTS SUBSYS (OPCX)
          USERMAP (USERS)
          PROTOCOL (TCP)
          PORTNUMBER (425)
          CODEPAGE (IBM-037)
INIT      CALENDAR (DEFAULT)
```

For more information, see *Tivoli Workload Scheduler for z/OS: Customization and Tuning*, SC32-1265

You can start the server by using the z/OS START command, or you can have the controller start and stop the server automatically. In the latter case, include the servers (srv1, srv2, ...) on the OPCOPTS statement in the Tivoli Workload Scheduler for z/OS parameter library. The server with TCP/IP support requires access to the C language runtime library (either as STEPLIB or as LINKLIST). If you have multiple TCP/IP stacks, or a TCP/IP started task with a name different from 'TCPIP', then a SYSTCPD DD card is required pointing to a TCP/IP data set containing the TCPIPJOBNAME parameter.

You always have to define OMVS segments for Tivoli Workload Scheduler for z/OS server started tasks.

Step 22. Activating support for the Java utilities

This section describes actions that are required if you want to use one of the following features:

- Dynamic Workload Console reporting.
- Event-driven workload automation for data set triggering, with centralized deploy process.

For details about these features, see *Tivoli Workload Scheduler for z/OS: Managing the Workload*.

As installation actions, perform the following steps:

1. Install IBM Java SDK for z/OS platforms. For information about how to install it, see *IBM SDK for z/OS platforms, Java Technology Edition*.
2. Copy the JZOS Java Launcher load module (JVMLDM66) from the JAVA_HOME directory to the SYS1.SIEALNKE system dataset. For details about customizing the JZOS Java Launcher, see *JZOS Batch Launcher and Toolkit function in IBM SDK for z/OS*.
3. Make sure you applied FMID JWSZ603.
4. Run EQQJOBS with the option to enable JAVA utilities, to create the EQQPCS08 sample JCL.
5. Customize EQQPCS08 and submit it.
6. Define the TRGOPT initialization statement in a member of the EQQPARM library.
7. Define the event rule in XML format. You can use a partitioned data set member to be used as input for the following step. The SEQQSAMP library contains EQQXML01 member as sample of event rule definition.
8. Select option 1.7.3 from the main menu, edit and submit the job to produce the configuration files.

Chapter 5. Verifying your installation

Perform this task for a tracker controller or standby controller.

Use the following procedures to verify your installation of a single Tivoli Workload Scheduler for z/OS address space, or your configuration.

Overview of verification

When you have installed a tracker, controller, standby controller, or server, start it and perform initial verification procedures. To fully verify Tivoli Workload Scheduler for z/OS, start all the 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, and requires some knowledge of the product. Therefore, verification is divided into two parts:

- Initial verification of individual Tivoli Workload Scheduler for z/OS address spaces.
- Verification of your configuration.

You can therefore perform some verification tasks without needing to know more detailed aspects of Tivoli Workload Scheduler for z/OS. When you are more familiar with the product components and functions, you can perform further testing.

The following topics are described:

- “Verifying installation of a tracker”
- “Verifying installation of a controller and dialogs” on page 151
- “Verifying installation of a standby controller” on page 154
- “Verifying installation of the Restart and Cleanup function” on page 156
- “Verifying configuration” on page 158

If you are installing a tracker and controller in the same address space, review the initial verification procedures for both a tracker and a controller.

Verifying installation of a tracker

When you have completed the installation tasks for a tracker, perform initial verification of the tracker. Because connections and the submission of work cannot be verified in isolation, you can perform further verification of the tracker when you have installed the controlling system, established connections between Tivoli Workload Scheduler for z/OS systems, and created a current plan. These verification tasks are described in “Verifying configuration” on page 158.

To initially verify the tracker, perform these tasks:

1. Follow the appropriate procedures for the Tivoli Workload Scheduler for z/OS subsystem that you are installing.
2. Ensure that you have completed all the necessary installation tasks.
3. Start the tracker and check the message log (EQQMLOG).
4. Verify that tracking events are created in the event data set (EQQEVDs).
5. Perform problem determination for tracking events if events are missing from the event data set.

Verifying installation of tracker

For TCP/IP connections only, ensure that a valid current plan exists before verifying the tracker.

Ensuring that all installation tasks are complete

Ensure that you have performed all the installation tasks that are needed for your Tivoli Workload Scheduler for z/OS service. That is, you should have:

- Followed the appropriate procedures for the Tivoli Workload Scheduler for z/OS subsystem that you are installing
- Installed the required JES and SMF exits, and verified that they are active
- Created a JCL procedure for the tracker
- Allocated required data sets
- Given the security access for the subsystem to access the data sets
- Specified the initialization statements in the parameter library (EQQPARM)
- Included the tracker in the same XCF group as the controller, if the tracker uses an XCF connection
- Defined a VTAM LU name for the tracker and activated the VTAM resources, if the tracker uses an NCF connection.

Checking the message log (EQQMLOG)

Start the tracker.

When the tracker is started, check the message log:

- Check that the return code for all initialization options is 0 (message EQQZ016I).
- Ensure that all required subtasks are active.
 - The data-router and submit tasks are always started. You should see these messages:
EQQZ005I OPC SUBTASK DATA ROUTER TASK IS BEING STARTED
EQQF001I DATA ROUTER TASK INITIALIZATION IS COMPLETE

EQQZ005I OPC SUBTASK JOB SUBMIT TASK IS BEING STARTED
EQQS001I THE SUBMIT TASK HAS STARTED
 - Also, verify that the tracker has started an event writer. You should see these messages:
EQQZ005I OPC SUBTASK EVENT WRITER IS BEING STARTED
EQQW065I EVENT WRITER STARTED
- Examine error messages.

Note: The first time the event writer is started, it formats the event data set. Ignore the SD37 abend code that is issued during the formatting process.

If you see error messages in the message log for an event reader or an NCF connection, this is because you cannot fully verify an event-reader function or NCF connection until the controller is active and a current plan exists. Active tracker-connection messages for XCF connections are written to the controller message log when the controller is started. If you have specified any of these functions, follow the procedures in “Verifying configuration” on page 158 when you have completed initial verification procedures.

- Check that your log is complete.

Figure 25 shows an example of the MLOG for a tracker. If your log seems to be incomplete, information might be in a buffer. If you are unsure whether the log

is complete, issue a dummy modify command like this: F ssname,xx. Message EQQZ049E is written to the log when the command is processed. This message will be the last entry in the log.

Verifying tracking events

The next verification phase is to check that the tracker is collecting tracking-event information and writing it to the event data set (EQQEVDS).

Tivoli Workload Scheduler for z/OS job tracking works correctly only if it receives information about status changes for all jobs or started tasks to be tracked. Job tracking gets this information from SMF and JES exits. These exits gather the necessary information, and an exit record is added to the Tivoli Workload Scheduler for z/OS event-writer queue via ECSA buffers.

The event writer

The event writer removes the event from its queue and creates an event record that is written to an event data set. The event writer also forwards the event if it has been started with an event-reader function. If a separate event reader is used, the event is read from the event data set. In either case, the reader task uses the connection with the controller to transfer the event to a queue at the controller. The event-manager subtask then processes the event and the current plan is updated.

The event data set

The event data set is needed to even out any difference in the rate that events are being generated and processed, and to prevent events from being lost if the Tivoli Workload Scheduler for z/OS address space or a subtask must be restarted. The first byte in an exit record is A if the event is created on a JES2 system, or B if the event is created on a JES3 system. This byte is found in position 21 of a standard event record, or position 47 of a continuation (type N) event. Bytes 2 and 3 in the exit record define the event type. These event types are generated by Tivoli Workload Scheduler for z/OS for jobs and started tasks:

- | | |
|----|--|
| 1 | Reader event. A job has entered the JES system. |
| 2 | Job-start event. A job has started to execute. |
| 3S | Step-end event. A job step has finished executing. |
| 3J | Job-end event. A job has finished executing. |
| 3P | Job-termination event. A job has been added to the JES output queues. |
| 4 | Print event. An output group has been printed. |
| 5 | Purge event. All output for a job has been purged from the JES system. |

If any of these event types are not being created in the event data set (EQQEVDS), a problem must be corrected before Tivoli Workload Scheduler for z/OS is started in production mode.

Notes:

1. 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.
2. 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.

Perform these actions to verify that events are being created on your system:

1. Run a job:
 - a. Submit a job like the following, ensuring that the output is written to a non-held output class:

Verifying tracking events

Test job

```
//VERIFY1 JOB STATEMENT PARAMETERS
```

```
//VERIFY EXEC PGM=IEBGENER
//*
//SYSPRINT DD DUMMY
//SYSUT2 DD SYSOUT=A
//SYSIN DD DUMMY
//SYSUT1 DD *
        SAMPLE TEST OUTPUT STATEMENT 1
//*
```

- b. Verify that the job has executed, printed, and purged.
- c. Browse the EQQEVDs data set using the ISPF/PDF browse facility. You should find these events on the event data set:
 - Type 1 event
 - Type 2 event
 - Type 3J event
 - Type 3P event
 - Type 4 event
 - Type 5 event.

The events are prefixed with A for JES2 or B for JES3. You might also find type 3S as events, depending on the value specified on the STEPEVENTS keyword of the EWTROPTS initialization statement.

2. Repeat step 1 for a started task.

Performing problem determination for tracking events

Problem determination depends on which event is missing and whether the events are created on a JES2 or JES3 system. In Table 32, the first column refers to the event type that is missing, and the second column tells you what action to perform. Events created on a JES2 system are prefixed with A, and events created on a JES3 system with B. The first entry in the table applies when all event types are missing (when the event data set does not contain any tracking events).

Table 32. Problem determination for missing tracking events

Type	Problem determination actions
All	<ol style="list-style-type: none">1. Verify in the EQQMLOG data set that the event writer has started successfully.2. Verify that the definition of the EQQEVDs ddname in the Tivoli Workload Scheduler for z/OS started-task procedure is correct (that is, events are written to the correct data set).3. Verify that the required exits have been installed.4. Verify that the IEFSSN_{mm} member of SYS1.PARMLIB has been updated correctly, and that an IPL of the z/OS system has been performed since the update.

Performing problem determination for tracking events

Table 32. Problem determination for missing tracking events (continued)

Type	Problem determination actions
A1	<p>If both A3P and A5 events are also missing:</p> <ol style="list-style-type: none"> 1. Verify that the Tivoli Workload Scheduler for z/OS version of the JES2 exits 7 and 51 routines have been correctly installed. Use the JES commands \$T EXIT(7) and \$T EXIT(51) or \$DMODULE(OPCAXIT7) and \$DMODULE(TWSXIT51). 2. Verify that the JES2 initialization data set contains a LOAD statement and an EXIT7 statement for the Tivoli Workload Scheduler for z/OS version of JES2 exit 7 (OPCAXIT7). For z/OS version 1.7 or later, verify also that the JES2 initialization data set contains a LOAD statement and an EXIT51 statement for the version of JES2 exit 51 (TWSXIT51). 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. <p>If either A3P or A5 events are present in the event data set, call an IBM service representative for programming assistance.</p>
B1	<ol style="list-style-type: none"> 1. Verify that the Tivoli Workload Scheduler for z/OS version of the JES3 exit IATUX29 routine has been correctly installed. 2. Verify that the exit has been added to a load-module library that JES3 can access. 3. Verify that JES3 has been restarted.
A2/B2	<ol style="list-style-type: none"> 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. 2. Verify that the IEFUJI exit has been correctly installed: <ol style="list-style-type: none"> a. Verify that the SMF parameter member SMFPRM_{nnn} in the SYS1.PARMLIB data set specifies that the IEFUJI exit should be called. b. Verify that the IEFUJI exit has not been disabled by an operator command. c. Verify that the correct version of IEFUJI is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, z/OS uses the first IEFUJI module found. d. Verify that the library containing this module was updated by the Tivoli Workload Scheduler for z/OS version of IEFUJI and that z/OS has been IPLed since the change was made.

Performing problem determination for tracking events

Table 32. Problem determination for missing tracking events (continued)

Type	Problem determination actions
A3S/B3S	<p>If type 3J events are also missing:</p> <ol style="list-style-type: none"> 1. Verify that the IEFACRT exit has been correctly installed. 2. Verify that the SMF parameter member SMFPRM_{mm} in the SYS1.PARMLIB data set specifies that the IEFACRT exit should be called. 3. Verify that the IEFACRT exit has not been disabled by an operator command. 4. Verify that the correct version of IEFACRT is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, z/OS uses the first IEFACRT module found. 5. Verify that this library was updated by the Tivoli Workload Scheduler for z/OS version of IEFACRT and that z/OS has been IPLed since the change was made. <p>If type 3J events are not missing, verify, in the EQQMLOG data set, that the event writer has been requested to generate step-end events. Step-end events are created only 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"> 1. Verify that the Tivoli Workload Scheduler for z/OS version of the JES3 exit IATUX19 routine has been correctly installed. 2. Verify that the exit has been added to a load-module library that JES3 can access. 3. Verify that JES3 has been restarted.
A4/B4	<ol style="list-style-type: none"> 1. If you have specified PRINTEVENTS(NO) on the EWTROPTS initialization statement, no type 4 events are created. 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 data sets. 3. Verify that the IEFU83 exit has been correctly installed: <ol style="list-style-type: none"> a. Verify that the SMF parameter member SMFPRM_{mm} in the SYS1.PARMLIB data set specifies that the IEFU83 exit should be called. b. Verify that the IEFU83 exit has not been disabled by an operator command. c. Verify that the correct version of IEFU83 is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, z/OS uses the first IEFU83 module found. d. Verify that the library containing this module was updated by the Tivoli Workload Scheduler for z/OS version of IEFU83 and that z/OS has been IPLed since the change was made. 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.

Table 32. Problem determination for missing tracking events (continued)

Type	Problem determination actions
A5	<ol style="list-style-type: none"> 1. Verify that JES2 has purged the job for which no A5 event was created. 2. Ensure that you have not specified TYPE26=NO on the JOBCLASS and STCCCLASS statements of the JES2 initialization parameters. 3. If A1 events are also missing, follow the procedures described for A1 events. 4. If A1 events are not missing, call an IBM service representative for programming assistance.
B5	<ol style="list-style-type: none"> 1. Verify that JES3 has purged the job for which no B5 event was created. 2. If B4 events are also missing, follow the procedures described for B4 events. 3. If B4 events are not missing, call an IBM service representative for programming assistance.

Verifying installation of a controller and dialogs

When you have completed the installation tasks for a controller, perform initial verification of the controller. Because connections and the submission of work cannot be verified in isolation, you can perform further verification of the controller when you have installed the controlling system, established connections between Tivoli Workload Scheduler for z/OS systems, and created a current plan. “Verifying configuration” on page 158 describes these verification tasks.

To initially verify the controller, perform the following steps:

1. Ensure that you have completed the installation tasks.
2. Start the controller, and check the message log (EQQMLOG).
3. Check that you can access Tivoli Workload Scheduler for z/OS data via the dialogs, and that authority checking is functioning as required.

If you encounter an error during verification, see “Performing problem determination” on page 153.

Ensuring that all installation tasks are complete

Check that you have:

- Created a started-task procedure for the controller
- Allocated data sets
- Given security authority to the started task to access its data sets
- Specified the initialization statements in the parameter library (EQQPARM)
- Included the controller in an XCF group, if it uses an XCF connection
- Defined a VTAM application ID for the controller and activated the VTAM resources, if it uses an NCF connection
- Updated SYS1.PARMLIB and defined VTAM resources, if users communicate with the controller through the Tivoli Workload Scheduler for z/OS API or if the Tivoli Workload Scheduler for z/OS server is used.
- Set up the ISPF environment for Tivoli Workload Scheduler for z/OS dialog users.

Checking the message log (EQQMLOG)

Start the controller. See *Tivoli Workload Scheduler for z/OS: Managing the Workload*, SC32-1263 for information on starting and stopping Tivoli Workload Scheduler for z/OS.

When the controller is started, check the message log:

- Ensure that the return code for all initialization options is 0 (message EQQZ016I).

- Check that all required subtasks are active.

Look for these messages when the controller is started:

Active general-service messages

```
EQQZ005I OPC SUBTASK GENERAL SERVICE IS BEING STARTED
EQQZ085I OPC SUBTASK GS EXECUTOR 01 IS BEING STARTED
EQQG001I SUBTASK GS EXECUTOR 01 HAS STARTED
:
EQQG001I SUBTASK GENERAL SERVICE HAS STARTED
```

Note: The preceding messages, EQQZ085I and EQQG001I, are repeated for each general service executor that is started. The number of executors started depends on the value you specified on the GSTASK keyword of the OPCOPTS initialization statement. The default is to start all five executors.

Active data-router-task messages

```
EQQZ005I OPC SUBTASK DATA ROUTER TASK IS BEING STARTED
EQQF001I DATA ROUTER TASK INITIALIZATION IS COMPLETE
```

When you start a controller and no current plan exists, you will still see a number of EQQZ005I messages each indicating that a subtask is being started. But these subtasks will not start until a current plan is created. You will also see this message:

Current plan message

```
EQQN105W NO VALID CURRENT PLAN EXISTS. CURRENT PLAN VSAM I/O IS NOT
POSSIBLE
```

If you have specified an event-reader function or NCF connections, these tasks will end if no current plan exists. You can verify the remaining tasks when you have created a current plan and connections can be established. “Verifying configuration” on page 158 describes these tasks.

- Check that the log is complete.

Figure 24 shows an example of the MLOG for a controller.

Note: If your log seems to be incomplete, information might be in a buffer. If you are unsure whether the log is complete, issue a dummy modify command like this: F sname,xx. Message EQQZ049E is written to the log when the command is processed. This message will be the last entry in the log.

Checking the server message log

After the controller is started, it starts the servers automatically if you specified the SERVERS keyword on the OPCOPTS statement. Otherwise, you must start them using the z/OS START command. When the server is started, check the message log:

- Ensure that the return code for all the initialization options is 0 (message EQQZ016I)
- Look for these messages when the server is started:

Active server messages

```
EQQZ0051 OPC SUBTASK SERVER IS BEING STARTED
EQQPH001 SERVER TASK HAS STARTED
```

Checking dialog functions

Before invoking the Tivoli Workload Scheduler for z/OS dialog, ensure that you have set up the ISPF environment as described in “Step 13. Setting up the ISPF environment” on page 123. Then invoke the Tivoli Workload Scheduler for z/OS dialog, and select an option from 0 to 10 on the main menu. If a new panel appears, you have established communication with the Tivoli Workload Scheduler for z/OS subsystem. You can further test the dialogs by performing functions, such as creating an application description. See *Tivoli Workload Scheduler for z/OS: Managing the Workload*, SC32-1263 for more information on specific dialog functions.

If you have used RACF to protect controller resources from unauthorized access, verify that the protection mechanism works as expected.

Performing problem determination

If you encounter problems during your verification of the controller, correct the errors and verify that the problem has been fixed. For more information on problem determination, see *Tivoli Workload Scheduler for z/OS: Diagnosis Guide and Reference*, SC32-1261.

Dialog problems

Various errors can occur when you are running Tivoli Workload Scheduler for z/OS dialogs. These errors cause the terminal alarm to sound and a short message to appear in the upper-right corner of your terminal screen. The message text for errors that cause the terminal alarm to sound usually contains the ALARM=YES flag. If this happens when you are trying to verify that Tivoli Workload Scheduler for z/OS dialogs are correctly installed, press the Help key (usually PF1) in ISPF. ISPF then displays a more complete error message in the long message area on your terminal. The following examples show two dialog error messages. The message number in each example is followed by the long message text and an explanation of the error. The examples highlight two errors. They are related to the dialog interface module, EQQMINOx.

EQQX115 EQQX115E TSO Service Facility RC: 20, RSNC: 40

The EQQMINOx load module is not installed in a library that can be reached by TSO. EQQMINOx must be present either in the STEPLIB library of the current TSO session or in a library in the current LINKLIB LNKSTnn concatenation. If EQQMINOx has been installed in a LINKLIB library, either an LLA refresh process or an IPL is required to make the module accessible by z/OS users.

EQQX120 The EQQMINOx program can only be called by an APF-authorized task

The EQQMINOx load module must be APF authorized. It must reside in a data set that is defined in SYS1.PARMLIB as being an APF-authorized library. Also, EQQMINOx must be defined to TSO as being an APF-authorized program. Ensure that you have followed the instructions in “Modifying TSO parameters” on page 81.

Authority problems

It is easiest to verify that the controller has been correctly installed without activating authority checking. Then, when authority checking is activated, some TSO users should no longer be able to do what they could do before. A Tivoli Workload Scheduler for z/OS dialog message should be issued, specifying that they are not authorized to perform a particular dialog function, or that they are not authorized to use any Tivoli Workload Scheduler for z/OS dialog.

If the controller authority functions have not been correctly installed, this will usually enable TSO users to use dialog functions that they are not authorized to use. The symptom of this problem is the absence of an expected error message. If this happens, follow this procedure which assumes the security monitor being used is RACF.

1. Verify the APPL class. If the user should not be able to use any Tivoli Workload Scheduler for z/OS dialog, verify that the APPL class is active and that the controller is defined as a resource in the APPL class. Also, verify that the user is not present in the access list to any of these resources and that universal access NONE has been specified. Use the SETR LIST command to display active classes, and use the RLIST command to display access lists in the APPL resource class.
2. Verify that the name of the Tivoli Workload Scheduler for z/OS RACF resource class has been defined to the Tivoli Workload Scheduler for z/OS started task in the AUTHDEF statement. You can check this by browsing the controller message log (EQQMLOG).
3. Verify the definition of the Tivoli Workload Scheduler for z/OS resource class. Check the source of the RACF class descriptor table, and compare this with the definition supplied by the ICHRRCDE sample in the SEQQSAMP library (see Appendix A, "Sample library (SEQQSAMP)," on page 301).
4. Verify fixed resources. If the user should not be able to use a specific dialog, such as the Calendar dialog, verify that the Tivoli Workload Scheduler for z/OS RACF resource class is active and that CL is defined as a resource in this class. Also, verify that the user is not present in the access list to the CL resource and that universal access NONE has been specified.
5. Verify subresources. If, for example, the user should be able to update only a subset of all applications in the Application Description dialog, but is instead able to update all applications, verify that the SUBRESOURCES keyword has been correctly specified for the controller in the AUTHDEF statement. Also verify that the controller has been restarted since the AUTHDEF statement was changed, and that Tivoli Workload Scheduler for z/OS RACF profiles have been refreshed since the Tivoli Workload Scheduler for z/OS subresource profiles were updated.

Verifying installation of a standby controller

When you have completed the installation tasks for a standby controller, perform initial verification. Because connections cannot be verified in isolation, you can perform further verification of the standby controller when you have installed the controlling system, established connections between Tivoli Workload Scheduler for z/OS systems, and created a current plan. "Verifying configuration" on page 158 describes these verification tasks.

To initially verify the standby controller, perform these tasks:

1. Ensure that you have completed all the necessary installation tasks.
2. Start the standby controller, and check the message log (EQQMLOG).

Ensuring that all installation tasks are complete

Check the installation tasks to make sure that you have performed the following actions:

- Created a JCL procedure for the standby controller
- Given the security authority for the address space to access the same data sets as the controller
- Specified the initialization statements in the parameter library (EQQPARM)
- Included the standby controller in the same XCF group as the controller
- Defined a VTAM application ID for the standby controller, and activated the VTAM resources, if the standby controller uses NCF connections
- Assigned an IP address to the tracker, if the tracker uses a TCP/IP connection.
- Updated SYS1.PARMLIB and defined VTAM resources, if the Tivoli Workload Scheduler for z/OS API or the Tivoli Workload Scheduler for z/OS server is used.

Checking the message log (EQQMLOG)

Start the standby controller.

When the controller has started, you should check the message log.

When you browse the message log:

- Ensure that the return code for all initialization options is 0 (message EQQZ016I).
- Check that this message appears:
Standby controller message
EQQZ128I OPC ACTIVE IN STANDBY MODE

Figure 23 shows an example of the MLOG for a standby controller.

Verifying installation of Restart and Cleanup function

```
04/08 13.42.33 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER STANDBY
04/08 13.42.33 EQQZ015I INIT STATEMENT: OPCOPTS  OPCHOST(STANDBY)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          APPCTASK(YES)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          RECOVERY(YES)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          ERDRTASK(0)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          EWTRTASK(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          GTABLE(JOBCARD)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          NCFTASK(YES)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          NCFAPPL(NCFFN002)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          SERVERS(OSR1,OSR2)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          BUILDSSX(REBUILD)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          VARSUB(SCAN)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          SSCMNAME(EQQSSCMJ,TEMPORARY)
04/08 13.42.33 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 13.42.33 EQQZ015I INIT STATEMENT: ALERTS    MLOG(DURATION,ERROROPER,OPCERROR,QLIMEXCEED)
04/08 13.42.33 EQQZ015I INIT STATEMENT:
04/08 13.42.33 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 13.42.33 EQQZ015I INIT STATEMENT: AUTHDEF   CLASS(IBMOPC) SUBRESOURCES(CP.CPGDDEF LT.LTGDDEF)
04/08 13.42.33 EQQZ015I INIT STATEMENT:
04/08 13.42.33 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 13.42.33 EQQZ015I INIT STATEMENT: EXITS     CALL00(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL01(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL02(YES)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL03(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL04(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL05(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL06(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL07(NO)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          CALL09(YES)
04/08 13.42.33 EQQZ015I INIT STATEMENT:
04/08 13.42.33 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 13.42.33 EQQZ015I INIT STATEMENT: ROUTOPTS  DASD(SUBCPU1)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          SNA(NCFFN003)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          USER(OS2LAN1,OS2LAN2)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          XCF(OPC)
04/08 13.42.33 EQQZ015I INIT STATEMENT:
04/08 13.42.33 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 13.42.33 EQQZ015I INIT STATEMENT: XCFOPTS   MEMBER(SMOPC)
04/08 13.42.33 EQQZ015I INIT STATEMENT:          GROUP(PLEXM101)
04/08 13.42.33 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 13.42.34 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER STANDBY  IS: 0000
04/08 13.42.35 EQQZ172I SSX BLOCK OF VERSION 09HWSZ200  SUCCESSFULLY BUILT
04/08 13.42.35 EQQZ073I OPC HAS RECOGNIZED THAT THIS IS A JES2 SYSTEM WITH
04/08 13.42.35 EQQZ073I COMMAND CHARACTER $ AND THAT THE NJE NODE NAME IS ROMEMVS
04/08 13.42.35 EQQZ128I OPC ACTIVE IN STANDBY MODE
```

Figure 23. Sample message log for a standby controller

Verifying installation of the Restart and Cleanup function

To verify that the Restart and Cleanup function was installed and configured correctly, perform these tasks:

- Verify that for each spool a Data Store was installed and started correctly (verify the message log EQQMLOG).
- Verify that the controller was started with the correct parameters (see “SNA only connection” on page 28 for a sample configuration).

Checking the message log (EQQMLOG)

After the controller has been started, ensure that the following messages appear in the message log (this example shows messages for an SNA connection):


```
02/07 12.11.39 EQQZ015I INIT STATEMENT: RCLOPTS CLNJOBPX(EQQCL)
02/07 12.11.39 EQQZ015I INIT STATEMENT: DSTDEST(TWSFDEST)
02/07 12.11.43 EQQPS01I PRE SUBMITTER TASK INITIALIZATION COMPLETE
02/07 12.11.46 EQQFSF1I DATA FILE EQQSDF01 INITIALIZATION COMPLETED
02/07 12.11.46 EQQFSF1I DATA FILE EQQSDF02 INITIALIZATION COMPLETED
02/07 12.11.46 EQQFSF1I DATA FILE EQQSDF03 INITIALIZATION COMPLETED
02/07 12.11.46 EQQFSI1I SECONDARY KEY FILE INITIALIZATION COMPLETED
02/07 12.11.46 EQQFSD5I SYSOUT DATABASE INITIALIZATION COMPLETE
02/07 12.11.46 EQQFL01I JOBLOG FETCH TASK INITIALIZATION COMPLETE
02/07 12.11.46 EQQFSD1I SYSOUT DATABASE ERROR HANDLER TASK STARTED
02/07 12.11.46 EQQFV36I SESSION I9PC33A3-I9PC33Z3 ESTABLISHED
```

Notes:

1. There should be an EQQFSF1I message for each EQQSDFxx file specified in the startup procedure.
2. There should be an EQQFV36I message for each SNA connection.
3. Verify that the DSTDEST for message EQQZ015I matches the SYSDEST in the Data Store message log.

After the server has been started, ensure that the following messages appear in the message log:

```
02/07 20.16.10 EQQZ015I INIT STATEMENT: SYSDEST(TWSFDEST)
02/07 20.16.16 EQQFSK1I PRIMARY KEY FILE INITIALIZATION COMPLETED
02/07 20.16.16 EQQFCM2I Data Store COMMAND TASK IS BEING STARTED
02/07 20.16.16 EQQFCC1I Data Store COMMUNICATION TASK INITIALIZATION COMPLETED
02/07 20.16.16 EQQFV01I FN APPLICATION STARTED
02/07 20.16.16 EQQFV24I ACB SUCCESSFULLY OPENED
02/07 20.16.16 EQQFSF1I DATA FILE EQQSDF01 INITIALIZATION COMPLETED
02/07 20.16.16 EQQFV36I SESSION I9PC33Z3-I9PC33A3 ESTABLISHED
02/07 20.16.16 EQQFSF1I DATA FILE EQQSDF02 INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSF1I DATA FILE EQQSDF03 INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSF1I DATA FILE EQQUDF01 INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSF1I DATA FILE EQQUDF02 INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSF1I DATA FILE EQQUDF03 INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSI1I SECONDARY KEY FILE INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSD5I SYSOUT DATABASE INITIALIZATION COMPLETE
02/07 20.16.16 EQQFSD1I SYSOUT DATABASE ERROR HANDLER TASK STARTED
02/07 20.16.16 EQQFCU1I CLEAN UP TASK STARTED
02/07 20.16.16 EQQFSW1I Data Store WRITER TASK INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSW1I Data Store WRITER TASK INITIALIZATION COMPLETED
02/07 20.16.16 EQQFSW1I Data Store WRITER TASK INITIALIZATION COMPLETED
02/07 20.16.16 EQQFJK3I Data Store JESQUEUE TASK INITIALIZATION COMPLETED
02/07 20.16.21 EQQFSR1I Data Store READER TASK INITIALIZATION COMPLETED
```

Notes:

1. There should be an EQQFSF1I message for each EQQSDFxx file specified in the startup procedure.
2. Verify that the SYSDEST for message EQQZ015I matches the DSTDEST in the controller message log.
3. There should be an EQQFSW1I message for every writer task.
4. There should be an EQQFCC1I message to indicate that the communication completed successfully.

Verifying configuration

When you have installed your Tivoli Workload Scheduler for z/OS controlling system, or when you install a controlled system, review this section to complete the verification of Tivoli Workload Scheduler for z/OS. These topics are described:

- Creating entries in the databases
- Running Tivoli Workload Scheduler for z/OS batch jobs
- Checking the message logs (EQQMLOG)
- Verifying workload submission
- Verifying takeover by a standby controller

Creating entries in the databases

You cannot fully verify Tivoli Workload Scheduler for z/OS until you have created a current plan. Before you can do this, you must create entries in the databases and then produce a long-term plan. If you are not familiar with Tivoli Workload Scheduler for z/OS, see *Tivoli Workload Scheduler for z/OS: Managing the Workload*, SC32-1263 for information about updating the databases and producing a long-term plan and a current plan.

Sample Tivoli Workload Scheduler for z/OS databases come with Tivoli Workload Scheduler for z/OS. They are loaded into the SMP/E target library with ddname SEQQDATA when the tracker software tape is processed. You can load the sample databases by submitting the EQQSAMPI JCL, which is generated by EQQJOBS.

Running batch jobs

When you have created database entries, invoke the LTP dialog and create a long-term plan. Check that the batch job completed successfully, and browse the long-term plan to check that the entries are correct. Next, use the Daily Planning dialog to create a current plan. When this job has ended, browse the current plan to check that the expected application occurrences are present.

You can further test Tivoli Workload Scheduler for z/OS batch jobs by, for example, printing information about the entries you have created in the databases. Table 19 on page 67 lists the Tivoli Workload Scheduler for z/OS batch jobs.

Checking the message logs (EQQMLOG)

When you have created a current plan and have started all Tivoli Workload Scheduler for z/OS address spaces in your configuration, check the message log of the controller and of all trackers.

Controller message log

Look for these messages in the message log of the controller:

Active normal-mode manager messages

```
EQQZ005I OPC SUBTASK NORMAL MODE MGR  IS BEING STARTED
EQQN013I OPC JOB TRACKING IS NOW ACTIVE AND CURRENT PLAN DD-NAME
        IS EQQCP1DS
```

Note: Active job-tracking log archiver messages. In the preceding message, the active current plan ddname is either EQQCP1DS or EQQCP2DS.

```
EQQZ005I OPC SUBTASK JT LOG ARCHIVER IS BEING STARTED
EQQN080I THE LOG ARCHIVER TASK HAS STARTED
```

Active job-tracking log archiver messages

Note: The preceding messages, EQQZ085I and EQQG001I, are repeated for each general service executor that is started. The number of executors started depends on the value you specified on the GSTASK keyword of the OPCOPTS initialization statement. The default is to start all five executors.

Active data-router-task messages

```
EQQZ005I OPC SUBTASK DATA ROUTER TASK IS BEING STARTED
EQQF001I DATA ROUTER TASK INITIALIZATION IS COMPLETE
```

If you have specified that APPC support should be started, check that these messages appear:

Active APPC-task messages

```
EQQZ005I OPC SUBTASK APPC TASK IS BEING STARTED
EQQ0001I APPC TASK INITIALIZATION IS COMPLETE
```

This message must be issued for the first controller or server start after APPC starts; it is issued by APPC to the system log:

APPC scheduler active - system log messages

```
ATB050I LOGICAL UNIT IS4MEOP4 FOR TRANSACTION SCHEDULER EOP4 HAS BEEN
ADDED TO THE APPC CONFIGURATION.
```

If you have specified an event-reader function, check that these messages appear:

Active event-reader messages

```
EQQZ005I OPC SUBTASK EVENT READER IS BEING STARTED
EQQR025I ERDR 01 STARTED
```

The numeric value on message EQQR025I indicates which event reader is started. The same value cannot be specified on more than one ERSEQNO keyword at the same node. No more than 16 event-reader tasks can be specified at the same node.

If the controller uses XCF connections, the XCF group is activated when the controller is started. Several messages can appear in the message log, indicating that a tracker or standby controller has started and that it has joined the group. If the controller communicates with a tracker using XCF, check for this message to verify the connection:

Active tracker-connection message

```
EQQF007I XCF MEMBER TRACK2 HAS JOINED THE GROUP. THE DESTINATION WILL BE
EQQF007I REPORTED ACTIVE
```

If a standby controller is started, check for this message:

Active standby-controller-connection message

```
EQQF008I XCF MEMBER CTRSTBY1 HAS JOINED THE GROUP AS STANDBY FOR THE
EQQF008I OPC CONTROLLER
```

If the controller uses an NCF connection, check that these messages appear (where NCFCON01 is the VTAM application ID of the controller, and NCFTRK01 is the VTAM application ID of the tracker):

Active NCF-connection messages

Checking EQQMLOG

```
EQQZ005I OPC SUBTASK VTAM I/O TASK IS BEING STARTED
EQQV001I NCF APPLICATION STARTED
EQQV024I ACB SUCCESSFULLY OPENED
EQQV036I SESSION NCFCON01-NCFTRK01 ESTABLISHED
```

If the controller uses the end-to-end with fault tolerance capabilities feature to schedule on distributed environments, check that these messages appear in the controller EQQMLOG:

Messages for active end-to-end scheduling with fault tolerance capabilities

```
EQQZ005I OPC SUBTASK END TO END ENABLER IS BEING STARTED
EQQZ085I OPC SUBTASK END TO END SENDER IS BEING STARTED
EQQZ085I OPC SUBTASK END TO END RECEIVER IS BEING STARTED
EQQG001I SUBTASK END TO END ENABLER HAS STARTED
EQQG001I SUBTASK END TO END RECEIVER HAS STARTED
EQQG001I SUBTASK END TO END SENDER HAS STARTED
```

When the end-to-end server is started, with the properties file customized to enable all EQQPT messages to be issued to the Server MLOG by default, check that these messages appear in the server EQQMLOG:

Messages in the server for active end-to-end scheduling with fault tolerance capabilities

```
EQQPH33I THE END-TO-END PROCESSES HAVE BEEN STARTED
EQQPT01I Program "/usr/lpp/TWS/TWS820/bin/translator" has been started,
pid is pid number
EQQPT15I The USS bindir "/usr/lpp/TWS/TWS820" maintenance level
is maintenance level
EQQPT01I Program "/usr/lpp/TWS/TWS820/bin/netman" has been started, pid is pid num
```

If a Symphony file has been created and is active, these messages will follow:

```
EQQPT20I Input Translator waiting for Batchman and Mailman are started
EQQPT21I Input Translator finished waiting for Batchman and Mailman
```

Otherwise, if the Symphony file is not present or a new one must be produced, this message will follow:

```
EQQPT22I Input Translator thread stopped until new Symphony will be available
```

The first time that the controller is being started with the fault-tolerant end-to-end scheduling in use or after the event data sets (EQQTWSIN and EQQTWSOU) have been reallocated, the event data sets need to be formatted. The following messages appear in the controller EQQMLOG before the messages about sender and receiver have started:

```
EQQW030I A DISK DATA SET WILL BE FORMATTED, DDNAME = EQQTWSIN
EQQW030I A DISK DATA SET WILL BE FORMATTED, DDNAME = EQQTWSOU
EQQW038I A DISK DATA SET HAS BEEN FORMATTED, DDNAME = EQQTWSIN
EQQW038I A DISK DATA SET HAS BEEN FORMATTED, DDNAME = EQQTWSOU
```

Also, the following messages might appear in the server EQQMLOG:

```
EQQPT56W The /DD:EQQTWSIN queue has not been formatted yet
EQQPT56W The /DD:EQQTWSOU queue has not been formatted yet
```

If the controller uses the Restart and Clean Up functionality check that the following messages appear in the Controller MLOG:

```

EQQZ005I OPC SUBTASK FL TASK IS BEING STARTED
EQQZ005I OPC SUBTASK PRE-SUBMIT IS BEING STARTED
EQQFSD1I SYSOUT DATABASE ERROR HANDLER TASK STARTED
EQQFSK1I PRIMARY KEY FILE INITIALIZATION COMPLETED
EQQFSF1I DATA FILE EQQSDf01 INITIALIZATION COMPLETED
EQQFSF1I DATA FILE EQQSDf02 INITIALIZATION COMPLETED
...
EQQFSF1I DATA FILE EQQSDfnn INITIALIZATION COMPLETED
EQQFSI1I SECONDARY KEY FILE INITIALIZATION COMPLETED
EQQFSD5I SYSOUT DATABASE INITIALIZATION COMPLETE
EQQPS01I PRE SUBMITTER TASK INITIALIZATION COMPLETE
EQQFL01I JOBL0G FETCH TASK INITIALIZATION COMPLETE

```

If the XCF is used to connect with Data Store Following messages should occur:

```

EQQFCCA1 XCF JOIN STARTED
EQQFCC9I XCF XCFCLC02 HAS JOINED XCF GROUP OPCGRPQ

```

If the SNA is used to connect with Data Store following messages should occur:

```

EQQFV01I FN APPLICATION STARTED
EQQFV24I ACB SUCCESSFULLY OPENED
EQQFV36I SESSION I9PC45RA-I9PC45AA ESTABLISHED

```

Figure 24 shows an example of the MLOG for a controller. The controller is connected to three trackers through shared DASD, XCF, and NCF. A standby controller is also started in this configuration.

```

04/08 12.13.19 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER CONTROLR
04/08 12.13.19 EQQZ015I INIT STATEMENT: OPCOPTS  OPCHOST(YES)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          APPCTASK(YES)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          RECOVERY(YES)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          RODMTASK(YES) RODMPARM(RODM)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          ERDRTASK(0)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          EWTRTASK(NO)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          GTABLE(JOBCARD)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          NCFTASK(YES)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          NCFAPPL(NCFFN002)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          SERVERS(OSR1,OSR2)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          BUILDSSX(REBUILD)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          VARSUB(SCAN)
04/08 12.13.19 EQQZ015I INIT STATEMENT:          SSCMNAME(EQQSSCMJ,TEMPORARY)
04/08 12.13.19 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.19 EQQZ015I INIT STATEMENT: ALERTS   MLOG(DURATION,ERROROPER,OPCERROR,QLIMEXCEED,RESCONT)
04/08 12.13.19 EQQZ015I INIT STATEMENT:

```

Figure 24. Sample Message Log for a controller (Part 1 of 5)

Checking EQQMLOG

```
04/08 12.13.19 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.19 EQQZ015I INIT STATEMENT: AUTHDEF CLASS(IBMOPC) SUBRESOURCES(CP.CPGDDEF LT.LTGDDEF)
04/08 12.13.19 EQQZ015I INIT STATEMENT:
04/08 12.13.19 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.19 EQQZ015I INIT STATEMENT: EXITS CALL00(NO)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL01(NO)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL02(YES)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL03(NO)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL04(NO)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL05(NO)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL06(YES)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL07(NO)
04/08 12.13.19 EQQZ015I INIT STATEMENT: CALL09(YES)
04/08 12.13.19 EQQZ015I INIT STATEMENT:
04/08 12.13.19 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.19 EQQZ015I INIT STATEMENT: ROUTOPTS DASD(SUBCPU1)
04/08 12.13.19 EQQZ015I INIT STATEMENT: SNA(NCFFN003)
04/08 12.13.19 EQQZ015I INIT STATEMENT: USER(OS2LAN1,OS2LAN2)
04/08 12.13.19 EQQZ015I INIT STATEMENT: XCF(SMOPC)
04/08 12.13.19 EQQZ015I INIT STATEMENT: APPC(ROMEAS1:ITIBM200.S44D1288)
04/08 12.13.19 EQQZ015I INIT STATEMENT: TCP(ROM2:9.52.52.11)
04/08 12.13.19 EQQZ015I INIT STATEMENT: TCPIPID(TCPIPROC)
04/08 12.13.19 EQQZ015I INIT STATEMENT:
04/08 12.13.19 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.19 EQQZ015I INIT STATEMENT: XCFOPTS MEMBER(OPC)
04/08 12.13.19 EQQZ015I INIT STATEMENT: GROUP(PLEXM101)
04/08 12.13.19 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.19 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER CONTROLR IS: 0000
04/08 12.13.20 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER RODM
04/08 12.13.20 EQQZ015I INIT STATEMENT: RODMOPTS RODMSYSTEM(EKGXR0DM)
04/08 12.13.20 EQQZ015I INIT STATEMENT: OPCRESOURCE(TAPE)
04/08 12.13.20 EQQZ015I INIT STATEMENT: OPCFIELD(QUANTITY)
04/08 12.13.20 EQQZ015I INIT STATEMENT: RODMCLASS(TAPE)
04/08 12.13.20 EQQZ015I INIT STATEMENT: RODMFIELD(NUMBEROF)
04/08 12.13.20 EQQZ015I INIT STATEMENT: RODMLOST(LAST)
04/08 12.13.20 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.20 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER RODM
04/08 12.13.21 EQQZ172I SSX BLOCK OF VERSION 09HWSZ200 SUCCESSFULLY BUILT
04/08 12.13.21 EQQZ073I OPC HAS RECOGNIZED THAT THIS IS A JES2 SYSTEM WITH
04/08 12.13.21 EQQZ073I COMMAND CHARACTER $ AND THAT THE NJE NODE NAME IS ROMEMVS
04/08 12.13.22 EQQZ005I OPC SUBTASK VTAM I/O TASK IS BEING STARTED
04/08 12.13.22 EQQZ005I OPC SUBTASK NORMAL MODE MGR IS BEING STARTED
04/08 12.13.22 EQQZ005I OPC SUBTASK TCP/IP TASK IS BEING STARTED
04/08 12.13.22 EQQZ005I OPC SUBTASK APPC TRACKER IS BEING STARTED
04/08 12.13.22 EQQZ005I OPC SUBTASK JOB SUBMIT TASK IS BEING STARTED
04/08 12.13.22 EQQZ005I OPC SUBTASK DATA ROUTER TASK IS BEING STARTED
04/08 12.13.22 EQQZ005I OPC SUBTASK RODM TASK IS BEING STARTED
04/08 12.13.22 EQQZ005I OPC SUBTASK APPC TASK IS BEING STARTED
04/08 12.13.23 EQQV001I NCF APPLICATION STARTED
04/08 12.13.23 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER CONTROLR
04/08 12.13.23 EQQZ015I INIT STATEMENT: AUDIT ACCESS(UPDATE) AMOUNT(KEY) FILE(ALL)
04/08 12.13.23 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.23 EQQZ015I INIT STATEMENT: AUDIT ACCESS(UPDATE) AMOUNT(DATA) FILE(JS)
04/08 12.13.23 EQQZ015I INIT STATEMENT:
```

Figure 24. Sample Message Log for a controller (Part 2 of 5)

```

04/08 12.13.23 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.23 EQQZ015I INIT STATEMENT: JTOPTS    BACKUP(NO)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          STATMSG(CPLOCK EVENTS GENSERV)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          ETT(YES)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          HIGHRC(0)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          JOBCHECK(SAME)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          JTLOGS(5)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          OPINFOSCOPE(ALL)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          JOBSUBMIT(YES)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          MAXJSFILE(NO)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          NEWOILIMIT(30)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          OFFDELAY(3)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          PLANSTART(7)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          PRTCOMPLETE(YES)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          DUAL(NO)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          SHUTDOWNPOLICY(75)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          SUBFAILACTION(E)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          SUPPRESSION(C)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          SUPPRESSPOLICY(75)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          TRACK(JOBOPT)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          WSFAILURE(ERROR,REROUTE,IMMED)
04/08 12.13.23 EQQZ015I INIT STATEMENT:          WSOFFLINE(ERROR,REROUTE,IMMED)
04/08 12.13.23 EQQZ015I INIT STATEMENT:
04/08 12.13.23 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.23 EQQZ015I INIT STATEMENT: NOERROR  LIST(MYJOB.*.STEP4.0002
04/08 12.13.23 EQQZ015I INIT STATEMENT:          HISJOB.*.STEP6.0004
04/08 12.13.23 EQQZ015I INIT STATEMENT:          HERJOB.*.STEP9.0016)
04/08 12.13.23 EQQZ015I INIT STATEMENT:
04/08 12.13.23 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.23 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER CONTROLR IS: 0000
04/08 12.13.23 EQQSU12I MAX NUMBER OF WORKSTATIONS CHECKPOINTED BY THIS SUBMIT TASK: 0046
04/08 12.13.23 EQQ0001I APPC TASK INITIALIZATION IS COMPLETE
04/08 12.13.23 EQQF007I XCF MEMBER SMOPC HAS JOINED THE GROUP. THE DESTINATION WILL BE
04/08 12.13.23 EQQF007I REPORTED ACTIVE
04/08 12.13.23 EQQF008I XCF MEMBER SBOPC HAS JOINED THE GROUP AS STANDBY FOR THE
04/08 12.13.23 EQQF008I OPC CONTROLLER
04/08 12.13.23 EQQF001I DATA ROUTER TASK INITIALIZATION IS COMPLETE
04/08 12.13.23 EQQTA01I THE TCP/IP COMMUNICATION TASK HAS STARTED
04/08 12.13.24 EQQSU01I THE SUBMIT TASK HAS STARTED
04/08 12.13.24 EQQAT01I THE APPC TRACKER TASK HAS STARTED
04/08 12.13.26 EQQQ502I SPECIAL RESOURCE DATASPACE HAS BEEN CREATED
04/08 12.13.26 EQQQ502I 0000020 PAGES ARE USED FOR 00000100 SPECIAL RESOURCE RECORDS
04/08 12.13.45 EQQN018I VSAM LSR BUFFERS HAVE BEEN SUCCESSFULLY ALLOCATED FOR VSAM FILE EQQCP1DS
04/08 12.13.45 EQQN018I NUMBER OF INDEX BUFFERS ARE 000006 WITH SIZE 016384
04/08 12.13.45 EQQN018I NUMBER OF DATA  BUFFERS ARE 000010 WITH SIZE 032768
04/08 12.13.45 EQQN012I OPC JOB TRACKING EVENTS ARE NOW BEING LOGGED ON FILE EQQJT01
04/08 12.13.45 EQQN013I OPC JOB TRACKING IS NOW ACTIVE AND CURRENT PLAN DD-NAME IS EQQCP1DS
04/08 12.13.45 EQQZ005I OPC SUBTASK EVENT MANAGER IS BEING STARTED
04/08 12.13.45 EQQZ005I OPC SUBTASK GENERAL SERVICE IS BEING STARTED
04/08 12.13.45 EQQZ005I OPC SUBTASK AUTO RECOVERY IS BEING STARTED
04/08 12.13.45 EQQZ005I OPC SUBTASK JT LOG ARCHIVER IS BEING STARTED
04/08 12.13.45 EQQZ005I OPC SUBTASK EXTERNAL ROUTER IS BEING STARTED
04/08 12.13.45 EQQZ005I OPC SUBTASK WS ANALYZER IS BEING STARTED
04/08 12.13.45 EQQN080I THE LOG ARCHIVER TASK HAS STARTED
04/08 12.13.46 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER STDAR
04/08 12.13.46 EQQZ015I INIT STATEMENT: AROPTS PREDWS(CPU*)
04/08 12.13.46 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.13.46 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER STDAR    IS: 0000
04/08 12.13.46 EQQW505I THE WORK STATION ANALYZER TASK HAS STARTED
04/08 12.13.46 EQQZ085I OPC SUBTASK GS EXECUTOR 01 IS BEING STARTED
04/08 12.13.46 EQQC001I THE AUTOMATIC RECOVERY SUBTASK HAS STARTED

```

Figure 24. Sample Message Log for a controller (Part 3 of 5)

Checking EQQMLOG

```

04/08 12.13.46 EQQZ085I OPC SUBTASK GS EXECUTOR 02 IS BEING STARTED
04/08 12.13.46 EQQEX01I THE EXTERNAL ROUTER TASK HAS STARTED
04/08 12.13.46 EQQZ085I OPC SUBTASK GS EXECUTOR 03 IS BEING STARTED
04/08 12.13.46 EQQZ085I OPC SUBTASK GS EXECUTOR 04 IS BEING STARTED
04/08 12.13.46 EQQZ085I OPC SUBTASK GS EXECUTOR 05 IS BEING STARTED
04/08 12.13.46 EQQG001I SUBTASK GENERAL SERVICE HAS STARTED
04/08 12.13.46 EQQE025I THE EVENT MANAGER HAS STARTED
04/08 12.13.46 EQQE017I THE ETT FUNCTION IS ACTIVATED
04/08 12.13.46 EQQG001I SUBTASK GS EXECUTOR 01 HAS STARTED
04/08 12.13.46 EQQG001I SUBTASK GS EXECUTOR 02 HAS STARTED
04/08 12.13.46 EQQG001I SUBTASK GS EXECUTOR 03 HAS STARTED
04/08 12.13.46 EQQG001I SUBTASK GS EXECUTOR 04 HAS STARTED
04/08 12.13.46 EQQG001I SUBTASK GS EXECUTOR 05 HAS STARTED
04/08 12.17.29 EQQV024I ACB SUCCESSFULLY OPENED
04/08 12.17.59 EQQV036I SESSION NCCFN002-NCCFN003 ESTABLISHED
04/08 12.20.35 EQQN051I A CURRENT PLAN BACKUP PROCESS HAS STARTED. TRIGGER WAS: BACKUP CMD
04/08 12.20.35 EQQN012I OPC JOB TRACKING EVENTS ARE NOW BEING LOGGED ON FILE EQQJT02
04/08 12.20.35 EQQQ507I A SPECIAL RESOURCE DATASPACE BACKUP PROCESS HAS STARTED
04/08 12.20.35 EQQQ508I A SPECIAL RESOURCE DATASPACE BACKUP PROCESS HAS ENDED
04/08 12.20.35 EQQQ508I 00000003 RECORDS WERE WRITTEN TO CX
04/08 12.20.38 EQQN056I A CURRENT PLAN COPY PROCESS HAS STARTED
04/08 12.20.40 EQQN057I A CURRENT PLAN DATA SET WAS SUCCESSFULLY COPIED: FROMDD=EQQCP1DS, TODD=EQQCP2DS
04/08 12.20.40 EQQN023I VSAM LSR BUFFERS HAVE BEEN SUCCESSFULLY DELETED FOR VSAM FILE EQQCP2DS
04/08 12.20.41 EQQN018I VSAM LSR BUFFERS HAVE BEEN SUCCESSFULLY ALLOCATED FOR VSAM FILE EQQCP2DS
04/08 12.20.41 EQQN018I NUMBER OF INDEX BUFFERS ARE 000006 WITH SIZE 016384
04/08 12.20.41 EQQN018I NUMBER OF DATA BUFFERS ARE 000010 WITH SIZE 032768
04/08 12.20.41 EQQN090I THE JOB TRACKING LOG DATA SET DEFINED BY DDNAME EQQJT01 HAS BEEN
04/08 12.20.41 EQQN090I COPIED TO THE JOB TRACKING LOG ARCHIVE DATA SET
04/08 12.20.57 EQQZ000I A STOP OPC COMMAND HAS BEEN RECEIVED
04/08 12.20.57 EQQN000I THE NORMAL MODE MANAGER TASK HAS BEEN REQUESTED TO TERMINATE
04/08 12.20.57 EQQS002I THE SUBMIT TASK HAS ENDED
04/08 12.20.57 EQQEX02I THE EXTERNAL ROUTER TASK HAS ENDED
04/08 12.20.57 EQQZ034I OPC SUBTASK DATA ROUTER TASK HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE 453 SECONDS AND USED 0.1 CPU SECONDS
04/08 12.20.57 EQQZ034I OPC SUBTASK APPC TASK HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE 453 SECONDS AND USED 0.0 CPU SECONDS
04/08 12.20.57 EQQZ034I OPC SUBTASK JOB SUBMIT TASK HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE 453 SECONDS AND USED 0.1 CPU SECONDS
04/08 12.20.57 EQQE000I TOTAL NUMBER OF EVENTS PROCESSED BY THE EVENT MANAGER TASK IS: 34
04/08 12.20.57 EQQE000I NUMBER OF EVENTS SINCE THE PREVIOUS MESSAGE IS: 34
04/08 12.20.57 EQQE000I EVENT MANAGER QUEUE LENGTH STATISTICS FOLLOW:
04/08 12.20.57 EQQE000I TOTAL Q1 Q2 Q5 Q10 Q20 Q50 Q100 >100
04/08 12.20.57 EQQE000I 31 28 2 1 0 0 0 0 0
04/08 12.20.57 EQQE006I EVENT MANAGER EVENT TYPE STATISTICS FOLLOW:
04/08 12.20.57 EQQE006I TYPE NTOT NNEW TTOT TNEW TAVG NAVG NSUS
04/08 12.20.57 EQQE007I ALL 34 34 0.2 0.2 0.00 0.00 0
04/08 12.20.57 EQQE007I 1 3 3 0.0 0.0 0.03 0.03 0
04/08 12.20.57 EQQE007I 2 3 3 0.0 0.0 0.02 0.02 0
04/08 12.20.57 EQQE007I 3S 0 0 0.0 0.0 0.00 0.00 0
04/08 12.20.57 EQQE007I 3J 4 4 0.0 0.0 0.00 0.00 0
04/08 12.20.57 EQQE007I 3P 4 4 0.1 0.1 0.02 0.02 0
04/08 12.20.57 EQQE007I 4 0 0 0.0 0.0 0.00 0.00 0
04/08 12.20.57 EQQE007I 5 12 12 0.0 0.0 0.00 0.00 0
04/08 12.20.57 EQQE007I USER 0 0 0.0 0.0 0.00 0.00 0
04/08 12.20.57 EQQE007I CATM 0 0 0.0 0.0 0.00 0.00 0
04/08 12.20.57 EQQE007I OTHR 8 8 0.0 0.0 0.00 0.00 0
04/08 12.20.57 EQQE004I CP ENQ LOCK STATISTICS SINCE PREVIOUS MESSAGE FOLLOW:
04/08 12.20.57 EQQE004I NAME NEXCL NSHRD THELD TWAIT AHELD AWAIT
04/08 12.20.57 EQQE005I NORMAL MODE MGR 18 0 7.0 0.0 0.39 0.00
04/08 12.20.57 EQQE005I WS ANALYZER 7 0 0.7 0.1 0.10 0.02
04/08 12.20.57 EQQE005I EVENT MANAGER 31 0 0.2 5.6 0.00 0.18

```

Figure 24. Sample Message Log for a controller (Part 4 of 5)


```

04/08 12.20.57 EQQV037I SESSION NCFN002-NCFFN003 ENDED
04/08 12.20.57 EQQC003I THE AUTOMATIC RECOVERY SUBTASK HAS ENDED NORMALLY
04/08 12.20.57 EQQTA02I THE TCP/IP COMMUNICATION TASK HAS ENDED
04/08 12.20.57 EQQAT01I THE APPC TRACKER TASK HAS ENDED
04/08 12.20.57 EQQG010I GENERAL SERVICE REQUEST STATISTICS FOLLOW:
04/08 12.20.57 EQQG010I TYPE          TOTAL  NEWRQS  TOTTIME  NEWTIME  TOTAVG  NEWAVG
04/08 12.20.57 EQQG011I ALL           0         0        0.0      0.0      0.00    0.00
04/08 12.20.57 EQQV020I ACB SUCCESSFULLY CLOSED
04/08 12.20.57 EQQZ034I OPC SUBTASK EXTERNAL ROUTER HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.20.57 EQQE023I THE EVENT MANAGER ENDED NORMALLY
04/08 12.20.57 EQQW503I THE WORK STATION ANALYZER ENDED NORMALLY
04/08 12.20.57 EQQG003I SUBTASK GS EXECUTOR 01 HAS ENDED
04/08 12.20.57 EQQG003I SUBTASK GS EXECUTOR 02 HAS ENDED
04/08 12.20.57 EQQV006I NCF APPLICATION ENDED
04/08 12.20.57 EQQZ034I OPC SUBTASK AUTO RECOVERY HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.20.57 EQQZ034I OPC SUBTASK TCP/IP TASK HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.20.57 EQQZ034I OPC SUBTASK APPC TRACKER HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.20.57 EQQG003I SUBTASK GS EXECUTOR 03 HAS ENDED
04/08 12.20.57 EQQG003I SUBTASK GS EXECUTOR 04 HAS ENDED
04/08 12.20.57 EQQG003I SUBTASK GS EXECUTOR 05 HAS ENDED
04/08 12.20.57 EQQE018I THE ETT FUNCTION IS DEACTIVATED
04/08 12.20.57 EQQZ034I OPC SUBTASK VTAM I/O TASK HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE          454 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.20.57 EQQZ034I OPC SUBTASK WS ANALYZER HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.4 CPU SECONDS
04/08 12.20.57 EQQZ034I OPC SUBTASK EVENT MANAGER HAS ENDED.
04/08 12.20.57 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.03 EQQZ034I OPC SUBTASK GS EXECUTOR 01 HAS ENDED.
04/08 12.21.03 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.03 EQQZ034I OPC SUBTASK GS EXECUTOR 02 HAS ENDED.
04/08 12.21.03 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.03 EQQZ034I OPC SUBTASK GS EXECUTOR 03 HAS ENDED.
04/08 12.21.03 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.03 EQQZ034I OPC SUBTASK GS EXECUTOR 04 HAS ENDED.
04/08 12.21.03 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.03 EQQZ034I OPC SUBTASK GS EXECUTOR 05 HAS ENDED.
04/08 12.21.03 EQQZ034I SUBTASK WAS ACTIVE          431 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.03 EQQG003I SUBTASK GENERAL SERVICE HAS ENDED
04/08 12.21.03 EQQZ034I OPC SUBTASK GENERAL SERVICE HAS ENDED.
04/08 12.21.03 EQQZ034I SUBTASK WAS ACTIVE          437 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.05 EQQN081I THE LOG ARCHIVER TASK HAS ENDED
04/08 12.21.05 EQQZ034I OPC SUBTASK JT LOG ARCHIVER HAS ENDED.
04/08 12.21.05 EQQZ034I SUBTASK WAS ACTIVE          439 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.21.05 EQQN107I THE NORMAL MODE MANAGER TASK HAS ENDED
04/08 12.21.05 EQQZ034I OPC SUBTASK NORMAL MODE MGR HAS ENDED.
04/08 12.21.05 EQQZ034I SUBTASK WAS ACTIVE          461 SECONDS AND USED      7.0 CPU SECONDS
04/08 12.21.06 EQQZ173I SSX BLOCK OF VERSION 09HWSZ200 SUCCESSFULLY RESTORED
04/08 12.21.08 EQQZ006I NO ACTIVE OPC SUBTASKS. OPC IS ENDING

```

Figure 24. Sample Message Log for a controller (Part 5 of 5)

Tracker message log

Look for these messages in the message log of each tracker:

Active data-router-task messages

```

EQQZ005I OPC SUBTASK DATA ROUTER TASK IS BEING STARTED
EQQF001I DATA ROUTER TASK INITIALIZATION IS COMPLETE

```

Active submit-task messages

Checking EQQMLOG

```
EQQZ005I OPC SUBTASK JOB SUBMIT TASK IS BEING STARTED
EQQSU01I THE SUBMIT TASK HAS STARTED
```

Also, verify that the tracker has started an event writer. You should see these messages:

Active event-writer messages

```
EQQZ005I OPC SUBTASK EVENT WRITER IS BEING STARTED
EQQW065I EVENT WRITER STARTED
```

If the tracker forwards events to the controller, ensure that an event-reader function is specified. This can be a separate event-reader task or an event writer that has been started with the EWSEQNO keyword. In some configurations, both functions can be started in a tracker.

- Although no messages are issued if you use EWSEQNO to start the reader function, check that EWSEQNO appears on the EWTROPTS statement and that the return code for this statement is 0.
- If you have specified a separate event-reader function, check that these messages appear:

Active event-reader messages

```
EQQZ005I OPC SUBTASK EVENT READER IS BEING STARTED
EQQR025I ERDR 01 STARTED
```

The numeric value on message EQQR025I indicates which event reader is being started. The same value cannot be specified on EWSEQNO and ERSEQNO keywords at the same node, and no more than 16 reader tasks can be specified at this node.

Note: If the tracker uses an XCF connection, no messages appear in the tracker message log unless an error has occurred. To verify XCF connection messages, check the message log of the controller.

If the tracker uses an NCF connection, check that these messages appear (where NCFTRK01 and NCFCON01 represent the VTAM application IDs of the tracker and controller):

Active NCF-connection messages

```
EQQZ005I OPC SUBTASK VTAM I/O TASK IS BEING STARTED
EQQV001I NCF APPLICATION STARTED
EQQV024I ACB SUCCESSFULLY OPENED
EQQV036I SESSION NCFTRK01-NCFCON01 ESTABLISHED
EQQV040I CURRENTLY RUNNING WITH 'NCFCON01' AS CONTROLLER
```

Figure 25 shows an example of the MLOG for a tracker. The tracker is connected to the controller via a VTAM link. The VTAM application IDs are NCFTRK01 for the tracker and NCFCON01 for the controller. The controller is active.

```

04/08 12.14.08 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER TRKV
04/08 12.14.08 EQQZ015I INIT STATEMENT: OPCOPTS  OPCHOST(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          NCFTASK(YES)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          NCFAPPL(NCFFN003)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          JCCTASK(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          EWTRTASK(YES)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          EWTRPARM(WTRPARMS)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          ERDRTASK(0)

04/08 12.14.08 EQQZ015I INIT STATEMENT:          SSCMNAME(EQQSSCMJ,TEMPORARY)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          BUILDSSX(REBUILD)
04/08 12.14.08 EQQZ015I INIT STATEMENT:
04/08 12.14.08 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.14.08 EQQZ015I INIT STATEMENT: JOBOPTS  CHSMWAIT(1)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          JOBLOGRETRIEVAL(Delayed)
04/08 12.14.08 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.14.08 EQQZ015I INIT STATEMENT: AUTHDEF  CLASS(IBMOPC) SUBRESOURCES(RL.WSNAME SR.SRNAME)
04/08 12.14.08 EQQZ015I INIT STATEMENT:
04/08 12.14.08 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.14.08 EQQZ015I INIT STATEMENT: EXITS    CALL00(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL01(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL02(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL03(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL04(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL05(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL06(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL07(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          CALL09(NO)
04/08 12.14.08 EQQZ015I INIT STATEMENT:
04/08 12.14.08 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.14.08 EQQZ015I INIT STATEMENT: TRROPTS  HOSTCON(SNA)
04/08 12.14.08 EQQZ015I INIT STATEMENT:          SNAHOST(NCFFN002)
04/08 12.14.08 EQQZ015I INIT STATEMENT:
04/08 12.14.08 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.14.08 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER TRKV      IS: 0000
04/08 12.14.08 EQQZ172I SSX BLOCK OF VERSION 09HWSZ200 SUCCESSFULLY BUILT
04/08 12.14.08 EQQZ073I OPC HAS RECOGNIZED THAT THIS IS A JES2 SYSTEM WITH
04/08 12.14.08 EQQZ073I COMMAND CHARACTER $ AND THAT THE NJE NODE NAME IS ROMEMVS
04/08 12.14.08 EQQZ005I OPC SUBTASK EVENT WRITER IS BEING STARTED
04/08 12.14.08 EQQZ005I OPC SUBTASK VTAM I/O TASK IS BEING STARTED
04/08 12.14.08 EQQZ005I OPC SUBTASK JOB SUBMIT TASK IS BEING STARTED
04/08 12.14.08 EQQZ005I OPC SUBTASK DATA ROUTER TASK IS BEING STARTED
04/08 12.14.09 EQQZ013I NOW PROCESSING PARAMETER LIBRARY MEMBER WTRPARMS
04/08 12.14.09 EQQZ015I INIT STATEMENT: EWTROPTS  HOLDJOB(USER)
04/08 12.14.09 EQQZ015I INIT STATEMENT:          EWWAIT(7)
04/08 12.14.09 EQQZ015I INIT STATEMENT:          SUREL(NO)
04/08 12.14.09 EQQZ015I INIT STATEMENT:          EWSEQNO(01)
04/08 12.14.09 EQQZ015I INIT STATEMENT:          STEPEVENTS(NZERO)
04/08 12.14.09 EQQZ015I INIT STATEMENT:          RETCODE(HIGHEST)

```

Figure 25. Sample message log for a tracker (Part 1 of 2)

Verifying workload submission

```
04/08 12.14.09 EQQZ016I RETURN CODE FOR THIS STATEMENT IS: 0000
04/08 12.14.09 EQQZ014I MAXIMUM RETURN CODE FOR PARAMETER MEMBER WTRPARMS IS: 0000
04/08 12.14.09 EQQZ064I OPC WILL USE THE NJE NODE NAME ROMEMVS FOR JOBS HELD ON THIS NODE
04/08 12.14.09 EQQV001I NCF APPLICATION STARTED
04/08 12.14.09 EQQZ065I OPC WILL RELEASE HELD JOBS USING THE JES2 COMMAND CHARACTER $
04/08 12.14.09 EQQW026I THE JOB COMPLETION CHECKER STARTED
04/08 12.14.09 EQQF001I DATA ROUTER TASK INITIALIZATION IS COMPLETE
04/08 12.14.09 EQQSU12I MAX NUMBER OF WORKSTATIONS CHECKPOINTED BY THIS SUBMIT TASK: 0046
04/08 12.14.09 EQQSU01I THE SUBMIT TASK HAS STARTED
04/08 12.14.14 EQQW065I EVENT WRITER STARTED
04/08 12.15.59 EQQV033E ACB OPEN FAILED FOR THE LAST 2 MINUTES - NCF APPLICATION NOT ACTIVE
04/08 12.17.49 EQQV024I ACB SUCCESSFULLY OPENED
04/08 12.17.59 EQQV036I SESSION NCFFN003-NCFFN002 ESTABLISHED
04/08 12.18.09 EQQV040I CURRENTLY RUNNING WITH 'NCFFN002' AS CONTROLLER
04/08 12.22.23 EQQZ000I A STOP OPC COMMAND HAS BEEN RECEIVED
04/08 12.22.23 EQQW011I THE EVENT WRITER ENDED NORMALLY
04/08 12.22.23 EQQV037I SESSION NCFFN003-NCFFN002 ENDED
04/08 12.22.23 EQQSU02I THE SUBMIT TASK HAS ENDED
04/08 12.22.23 EQQZ034I OPC SUBTASK DATA ROUTER TASK HAS ENDED.
04/08 12.22.23 EQQZ034I SUBTASK WAS ACTIVE      494 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.22.23 EQQZ034I SUBTASK WAS ACTIVE      494 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.22.23 EQQZ034I OPC SUBTASK EVENT WRITER HAS ENDED.
04/08 12.22.23 EQQZ034I SUBTASK WAS ACTIVE      494 SECONDS AND USED      0.1 CPU SECONDS
04/08 12.22.23 EQQZ034I OPC SUBTASK JOB SUBMIT TASK HAS ENDED.
04/08 12.22.23 EQQZ034I SUBTASK WAS ACTIVE      494 SECONDS AND USED      0.2 CPU SECONDS
04/08 12.22.27 EQQV020I ACB SUCCESSFULLY CLOSED
04/08 12.22.27 EQQV006I NCF APPLICATION ENDED
04/08 12.22.27 EQQZ034I OPC SUBTASK VTAM I/O TASK HAS ENDED.
04/08 12.22.27 EQQZ034I SUBTASK WAS ACTIVE      498 SECONDS AND USED      0.0 CPU SECONDS
04/08 12.22.28 EQQZ173I SSX BLOCK OF VERSION 09HWSZ200 SUCCESSFULLY RESTORED
04/08 12.22.30 EQQZ006I NO ACTIVE OPC SUBTASKS. OPC IS ENDING
```

Figure 25. Sample message log for a tracker (Part 2 of 2)

Verifying workload submission

Now verify that Tivoli Workload Scheduler for z/OS can submit work and that the work is sent to the correct destination.

Controlling system

You can use this procedure for the controlling system:

1. Create a workstation description, leave the destination field blank. This means that operations will be submitted to the system where the controller is started.
2. Create an application description. Define at least one operation on the workstation you have created. Submit a daily planning extend or replan batch job to include the new workstation in the current plan.
Add an occurrence of this application to the current plan.
3. Verify that the operations run successfully on this system and that they are reported as complete in the current plan.
4. Check that submit events are created in the event data set for the controlling system (see “Verifying job submission” on page 169).

Controlled systems

If you have controlled Tivoli Workload Scheduler for z/OS systems in your configuration, you can use this procedure to check that work is sent to these destinations, that it is submitted, and that events are returned to the controller:

1. Create a workstation description for each destination. In the destination field, remember to specify the submit/release data set name, the XCF member name of the tracker, the tracker VTAM application ID, or the tracker TCP/IP destination name, depending on the connection. Ensure that the trackers are active.

2. Add an application occurrence to the current plan with operations defined on each of the workstations.
3. Verify that the operations run successfully on the correct system and that they are reported as complete in the current plan.
4. Check that submit events are created in the event data set for each controlled system (see “Verifying job submission”).

If your configuration includes a MAS complex, you can specify only one workstation description to represent all systems in the complex. If this is the case, ensure that a job is run on each system in the complex. Verify that events are received at the controller by checking that the operations are reported as complete in the current plan.

Notes:

1. If you create a workstation description after the current plan has been created, you must either replan or extend the current plan to use this workstation.
2. You must also specify workstation destinations in the ROUTOPTS initialization statement. You must restart the controller if you update ROUTOPTS.

Verifying job submission

When Tivoli Workload Scheduler for z/OS submits work, a submit event is written to the event data set. A submit event is prefixed with an I, and can be one of these:

- IJ1** Job JCL. A job has been submitted.
- IJ2** Started-task JCL. A started task has been started.
- IWTO** An operation has been initiated for a general workstation with the WTO option. The submit task causes message EQQW775I to be issued.

Check each system on which Tivoli Workload Scheduler for z/OS is installed to ensure that the destination can be reached by the controller and that the relevant submit events are being created in the event data set. That is, if Tivoli Workload Scheduler for z/OS will submit jobs, start started tasks, and initiate WTO operations, verify that all the event types are created in the event data set. You need not test all these functions if Tivoli Workload Scheduler for z/OS will not be used for a particular operation.

To perform this test for all type I events, start an operation on each of three workstations, all of which specify the destination of the system you are testing. The workstations must be a computer automatic workstation for IJ1 events, a computer automatic workstation with the started-task option for IJ2 events, and a general WTO workstation for IWTO events. Follow this procedure to verify workload submission:

1. Ensure that you have the correct workstations specified in the workstation description database.
2. Create a test application with an operation for each workstation you want to test, and add it to the current plan.
3. When the application has completed, browse the event data set, and verify that the required type I events have been created.
4. If the operations are not started, check that the workstation status is ACTIVE and that the workstation is not WAITING FOR CONNECTION, which means that the controller is waiting for the corresponding tracker to communicate.

Verifying takeover by a standby controller

To verify that a standby controller can take over the functions of the controller, perform these actions:

1. Stop the controller.
2. Unless automatic takeover was not specified in the Tivoli Workload Scheduler for z/OS runtime options, order the standby controller to take over the functions of the controller, using this command:

```
MODIFY ssname,TAKEOVER
```

See the XCF0PTS statement in the Tivoli Workload Scheduler for z/OS *Customization and Tuning* guide to learn how to set up automatic takeover as a runtime option.

3. Check that this message appears in the message log:

Standby controller message

```
EQQZ129I TAKEOVER IN PROGRESS
```

When the standby controller has taken over the functions of the controller, more messages will appear in the message log. These are the same messages that appear when a controller is started. Verify that the takeover was successful by following the procedures used in the verification of the controller.

Chapter 6. Migrating

This chapter provides information to help you plan your migration from Tivoli Workload Scheduler for z/OS Version 8.2, 8.3, 8.5 or 8.5.1 to version 8.6.

Planning for migration

Before attempting to perform a migration, develop a migration plan to ensure a smooth and orderly transition. A well thought-out and documented migration plan can help minimize any interruption of service. Your migration plan should address topics such as:

- Identifying which required and optional products are needed
- Evaluating new, changed, and deleted functions
- Defining which Tivoli Workload Scheduler for z/OS functions you want to add, delete, or modify
- Defining necessary changes to:
 - The configuration
 - The initialization statements
 - Installation modifications
 - Operational procedures
 - Other related products
- Determining any restrictions during the conversion period
- Estimating the amount of time the conversion will take
- Defining education requirements for operators and end users
- Preparing your staff and users for migration.

The content and extent of a migration plan can vary significantly from installation to installation. For example, installations that have many installation-specific modifications might require extensive planning due to the added complexity.

You should also consider the following areas when defining your migration plan:

- Installation
- Initialization
- Customization
- Operation

Migration considerations

IBM attempts to make the installation of new releases as easy as possible. Initially, you should install Tivoli Workload Scheduler for z/OS without taking any customization actions in order to achieve a stable environment. Refer to the *Program Directory* for specific instructions about using System Modification Program/Extended (SMP/E) to install Tivoli Workload Scheduler for z/OS.

You can migrate from or fall back to previous releases **without** IPLing z/OS.

The following are some migration considerations and, in some cases, necessary prerequisite steps to perform, before migrating to the current release, to ensure a proper fallback migration can be performed, if necessary at any time.

Migration considerations

- If you are performing a fallback because of problems experienced on Tivoli Workload Scheduler for z/OS, be sure to keep the Tivoli Workload Scheduler for z/OS data sets for diagnostic purposes.
- If you migrate to and fallback from Tivoli Workload Scheduler for z/OS to test the environment before your *official* migration, ensure that you reallocate all Tivoli Workload Scheduler for z/OS data sets before the next migration exercise.
- Before you migrate to Tivoli Workload Scheduler for z/OS Version 8.6 from Version 8.2, ensure you have applied at least the fix for APARs PK24633 and PK24711 on version 8.2. This is required to successfully perform a fallback migration from Tivoli Workload Scheduler for z/OS Version 8.6 to version 8.2, if necessary at any time.
- If you are migrating from Tivoli Workload Scheduler for z/OS Version 8.2 to Version 8.6, note that the default value of the **GDGNONST** parameter has been changed from NO to YES. Refer to the **OPCOPTS** initialization statement in *Customization and Tuning* for more information about the **GDGNONST** parameter.
- If you are migrating from version 8.2 to version 8.5.1 or from version 8.2 to 8.6, and you have not applied the fix for APAR PK36095 on your version 8.2 installation, then ensure you set **TASKUSER(NO)** to maintain the current behavior of your environment. With version 8.5.1 and later, the default value for **TASKUSER** is **YES**. Refer to the **OPCOPTS** initialization statement in *Customization and Tuning* for more information about the **TASKUSER** parameter.
- If you are migrating from version 8.2 to version 8.5.1 or from version 8.2 to 8.6, and you have not applied the fix for APAR PQ85880 on your version 8.2 installation, be aware that the default setting of the **CODEPAGE** keyword in the **SERVOPTS** initialization statement continues to be IBM-137, however, for version 8.5.1 and later, if you specify a codepage value different from the default value, a check has been implemented to use the default codepage if the first four characters of the codepage you specify are different from "IBM-". Refer to the **SERVOPTS** initialization statement in *Customization and Tuning* for more information about the **CODEPAGE** parameter.
- Before you migrate to Tivoli Workload Scheduler for z/OS Version 8.6 from Version 8.5.1, ensure you have applied at least the fix for APAR PM14386 on version 8.5.1. This is required to successfully perform a fallback migration from Tivoli Workload Scheduler for z/OS Version 8.6 to version 8.5.1, if necessary at any time.
- The handling of operations in X status in a plan has changed beginning with version 8.5.1 and later. A migration operation is interrupted if an operation in X status is found in the current plan. Message EQQIC51E is issued. Before retrying the migration, perform the following steps, depending on the scenario you are running:

A migration from Version 8.5 to either Version 8.5.1 or 8.6

If conditions have been defined and used in the plan, then check if you need the X status propagation to normal successors or, the new logic that makes the normal successor ready is preferred. If you require X status propagation, complete these steps:

1. Change in AD the normal dependencies in conditions ST=C. Refer to the information about conditional logic in *Managing the Workload*.
2. Wait until the occurrences involved in these definitions complete in the Plan.
3. Perform a replan to remove these instances.

A fallback from either Version 8.6 or 8.5.1 to version 8.5

If conditions have been defined and used in the plan, check if you used the new logic that makes the normal successor ready. Consider that after the fallback, they will become X. If you used the new logic for X status, then perform the following steps:

1. Change in AD the definitions so that the new logic is not used.
2. Wait until the occurrences involved in these definitions complete in the Plan.
3. Perform a replan to remove these instances.

Customization considerations

Tivoli Workload Scheduler for z/OS is designed as a general-purpose workload automation subsystem. As such, it might not meet all the requirements for your specific installation. IBM allows installations to implement installation exits and provides many callable services that can be used to supplement Tivoli Workload Scheduler for z/OS processing.

Carefully examine any customization that your site has installed. Determine whether the function is now provided in the product or if you need to modify the logic based on changes made to Tivoli Workload Scheduler for z/OS.

When new functions are added to Tivoli Workload Scheduler for z/OS, installation exits and macros used within installation exits can change. New installation exits or macros might also be introduced in a Tivoli Workload Scheduler for z/OS release. If a release provides a new installation exit, determine if your installation needs to implement the exit. A release can change an existing exit by modifying:

- What the installation exit expects on entry
- Return codes Tivoli Workload Scheduler for z/OS expects when the exit returns control to Tivoli Workload Scheduler for z/OS
- The function that the installation exit performs
- The processing that is performed before or after the exit.

Migration strategies

You need to consider these points when deciding the appropriate migration strategy for your enterprise:

- JES and SMF exits
- Migrating to existing subsystem definitions
- Migrating to new subsystem definitions
- Installation and verification
- Parallel testing

JES and SMF exits

JES and SMF exits supplied with Tivoli Workload Scheduler for z/OS can also track work for previous releases. The exits are **always downward compatible**.

Consider installing JES and SMF exits in your current production environment at least a week before you plan to migrate any of the address spaces to Tivoli Workload Scheduler for z/OS.

Migrating to existing subsystem definitions

You can migrate from or fall back to your current subsystems **without having to IPL the z/OS system**.

Migrating to existing subsystem definitions

By continuing to use your current subsystem names, you do not need to consider the effect of migration on these users of Tivoli Workload Scheduler for z/OS services:

- Host dialog users
- PIF programs
- API programs
- Callable services
- Console automation software

Keeping the same subsystem names reduces the installation effort of a new level of Tivoli Workload Scheduler for z/OS.

Migrating to new subsystem definitions

If you want to parallel-test the new level of Tivoli Workload Scheduler for z/OS with your current level, you must create new subsystems for the Tivoli Workload Scheduler for z/OS address space.

Getting the right software parts

The Tivoli Workload Scheduler for z/OS load modules, panels, messages, and other software parts have the same name as they had in previous Tivoli Workload Scheduler for z/OS releases. You must ensure that the users of Tivoli Workload Scheduler for z/OS services run the same level of software as the subsystem address space.

Load modules: You can decide if you want to use the same data set name for the Tivoli Workload Scheduler for z/OS load modules as your previous environment. However, consider the additional effort required on your part to coordinate the JCL changes required for callers of Tivoli Workload Scheduler for z/OS services such as:

EQQEVPGM
EQQUSIN x
EQQYCOM
EQQYLTOP

If the Tivoli Workload Scheduler for z/OS load library is not referenced in the STEPLIB DD statement, ensure that the Tivoli Workload Scheduler for z/OS library is listed first in the LNKLIST concatenation **and** that the library remains empty until you are ready to cut over to Tivoli Workload Scheduler for z/OS on the production system. Then you should copy the load modules into the library and perform an LLA refresh.

Two Tivoli Workload Scheduler for z/OS load modules **must always** be in a LNKLIST library:

EQQINIT n The subsystem initialization module
EQQSSCM n The subsystem communication module.

However, this does not mean that you must reinitialize the subsystem to cut over Tivoli Workload Scheduler for z/OS to production. The module names defined for EQQINIT and EQQSSCM in the SYS1.PARMLIB subsystem name table (IEFSSN nn) can be overridden when a Tivoli Workload Scheduler for z/OS address space is created.

The EQQMINO x load module requires special attention. EQQMINO x is the scheduler's dialog interface module, is invoked by TSO SERVICES, and passes dialog requests and data to the controller. EQQMINO x must run APF authorized, therefore it must reside in an authorized library. By this token, keep in mind that

any unauthorized library in a STEPLIB or LIBDEF concatenation makes the entire concatenation unauthorized. So remember to identify the library where EQQMINOx resides.

The BUILDSSX keyword of the OPCOPTS initialization statement can be used to rebuild the environment created during subsystem initialization at the new software level. When the address space is terminated, the previous environment is reinstated, thereby ensuring fallback to a previous release of Tivoli Workload Scheduler for z/OS.

SSCMNAME keyword of the OPCOPTS initialization statement can be used to permanently, or temporarily, replace the EQQSSCM n module that was loaded into common storage at IPL. When the TEMPORARY value is defined, the named module is loaded into private storage of the Tivoli Workload Scheduler for z/OS address space, therefore events created while the address space is not active will use the EQQSSCM n from the previous IPL. When PERMANENT is specified, the old EQQSSCM n in common storage is replaced.

Permanent replacement of the EQQSSCM n module effects the SSX and prevents reinstatement of the subsystem to a previous release or version.

Notes:

1. Do not specify PERMANENT in the SSCMNAME keyword and BUILDSSX(REBUILD) until you are certain that you will not need to fall back to a previous version or release.
2. Create backup copies of the old load module library before you replace the objects.

The ISPF environment: Tivoli Workload Scheduler for z/OS ISPF dialog users must run software parts that are at the same level as the controller address space. Again, using the same data set names for software parts libraries, such as messages and panels, negates the requirement to change TSO logon procedures.

If you use the same data set names, instruct dialog users to return to the TSO READY prompt after you have replaced the software parts and before they try to communicate with a Tivoli Workload Scheduler for z/OS controller.

The Tivoli Workload Scheduler for z/OS ISPF profile is automatically reinitialized when the EQQOPCAP panel (the Tivoli Workload Scheduler for z/OS main menu) is first displayed for a new release. Dialog users must enter the Tivoli Workload Scheduler for z/OS options dialog and redefine required values, such as the subsystem name.

Be sure to create backup copies of the old libraries before you replace the objects.

When migrating from one release of Tivoli Workload Scheduler for z/OS to the next, the LOADLIB, PANELLIB, MSGLIB, CLIB, and SKELLIB for the right Tivoli Workload Scheduler for z/OS release must be invoked from the TSO ISPF dialogs. Remember to identify the library where EQQMINOx resides.

Migration overview

This section summarizes the steps you must perform before installing a new release of Tivoli Workload Scheduler for z/OS. You should plan for the migration by installing and stabilizing the new Tivoli Workload Scheduler for z/OS release without incorporating the new functions provided. Installing a new Tivoli

Migration overview

Workload Scheduler for z/OS release without initially exploiting new functions allows you to create a stable environment.

Migration steps overview

To install and activate Tivoli Workload Scheduler for z/OS, you must perform the following steps:

1. Ensure you have the required environment for Tivoli Workload Scheduler for z/OS
2. Make the necessary modifications
3. Stop the scheduler version you are currently running
4. Convert the data sets
5. Start Tivoli Workload Scheduler for z/OS

Only summary information appears in this section. Detailed instructions about how to make specific changes are in the chapters that follow or in referenced publications.

Establishing the required environment

You use SMP/E to install the Tivoli Workload Scheduler for z/OS software. Refer to the *Program Directory* for specific instructions about using SMP/E to install Tivoli Workload Scheduler for z/OS.

IBM Tivoli Workload Scheduler for z/OS Version 8.6 must run in the z/OS environments from Release 1.10 or later.

Program requirements

Before installing Tivoli Workload Scheduler for z/OS, check the preventive service bucket for a current list of the required products, their maintenance levels, and recommendations from the service organizations.

The PSP for this release can be found in the TWSZOS860 upgrade. Read this document carefully before you start to install Tivoli Workload Scheduler for z/OS.

Your installation must have at least the earliest release supported for migration, which is Tivoli Workload Scheduler for z/OS Version 8.2.

Installation and verification

If you are migrating to existing subsystem definitions, you must perform these installation tasks:

1. Load the tracker software ("Step 1. Loading tracker software" on page 53).
2. Load the controller software ("Step 2. Loading controller software" on page 53).
3. Load the NLS software ("Step 3. Loading national language support software" on page 54).
4. Run the EQQJOBS CLIST ("Step 4. Using the EQQJOBS installation aid" on page 54).
5. Install JES and SMF exits at Tivoli Workload Scheduler for z/OS level ("Step 5. Adding SMF and JES exits for event tracking" on page 72).
6. Update PARMLIB ("Step 6. Updating SYS1.PARMLIB" on page 75).
7. Import the new default security certificates for HTTP connections ("Step 8. Securing communications" on page 91).
8. Allocate VSAM and non-VSAM data sets ("Step 9. Allocating data sets" on page 94).

9. Update the JCL procedure for the Tivoli Workload Scheduler for z/OS address space ("Step 10. Creating JCL procedures for address spaces" on page 116).
10. Update the initialization statements ("Step 11. Defining the initialization statements" on page 122).
11. Update the ISPF environment ("Step 13. Setting up the ISPF environment" on page 123).

Ensure that you follow the subsystem verification procedures outlined in Chapter 5, "Verifying your installation," on page 145.

You can use the conversion program for both migration and fallback. You should consider testing your installation by migrating in the production environment and then falling back.

Note: Verify that all the Tivoli Workload Scheduler for z/OS parameters defined in the previous release are still valid in the current release.

Parallel testing

If you want to perform the migration and then continue immediately into parallel testing in a job-tracking environment, you can use the following procedure as a guide. However, you should carefully consider the applicability of this procedure in your own Tivoli Workload Scheduler for z/OS configuration.

1. Stop your production system.
2. Perform data set conversion and copying.
3. Start your production system.
4. Start Tivoli Workload Scheduler for z/OS, Version 8.6.

You should also consider:

- If you start the JCC in both the production system and Tivoli Workload Scheduler for z/OS, the two JCCs cannot delete or requeue SYSOUT from the same SYSOUT class.
- Do not specify HOLDJOB(YES) or HOLDJOB(USER) for more than one of the two systems. If you do, one system might incorrectly release jobs that are held by the other system.
- When you convert the VSAM data sets, it is recommended that you run the conversion of the JS file to verify that conversion has been done correctly. Then, before running the parallel test, reallocate empty JS files. (Otherwise, the test system might find valid production JCL on the active JS file and submit it to the JES subsystem.)
- You should start with an empty JCL library data set (EQQJBLIB). Otherwise, the test system might submit production JCL incorrectly. To test that Tivoli Workload Scheduler for z/OS submits jobs correctly, you should create test applications with job names that are not known to the production system. JCL for those jobs could then safely be inserted into EQQJBLIB.
- On the test system you should specify TRACK(ALL) and SUBFAILACTION(R) on the JTOPTS initialization statement.
- TSO commands or subroutines that have a specific name for the subsystem parameter will not report events to the test system. You should update any procedures, which are dependent on TSO commands or subroutines, if events should also be reported to the test system.
- If you are migrating from a previous release of Tivoli Workload Scheduler for z/OS and you use NetView or a similar product to intercept messages, make sure that WTO (write-to-operator) messages are not issued by the test system.

Otherwise, the test system might trigger some processing that impacts the production system. You should not use alert WTOs, deadline WTOs, or WTO operations on the test system.

- If you want to use Restart and Cleanup when in the old subsystem is running a JCC task, you must specify the keyword DSTCLASS in the RCLOPTS statement of the new controller. The class specified in DSTCLASS must not be one of the classes specified in the JCC parameter CHKCLASS. This will prevent the JCC task from deleting the duplicate SYSOUT copy created for the Data Store before it has been successfully stored. Refer to *Customization and Tuning* for further details.

Using the preceding notes as a guide, you will be able to run one production system and one Tivoli Workload Scheduler for z/OS test system in parallel. The work with the database dialogs and the long-term-planning functions can be fully tested this way. The job-tracking functions of the test system will be incomplete because only the specially created test jobs will be submitted by the test system. However, the tracking of work, including the tracking of applications and jobs in the production area, will be done normally.

Migrating an end-to-end network

All the considerations for a Tivoli Workload Scheduler master domain manager apply to the controller. Refer to the *Release Notes* for the migration path and compatibility information for an end-to-end network.

Migrating DB2

If you want to migrate history data that is already defined, you need only run the EQQICNVH sample as explained in “Sample to migrate the history database” on page 122. Otherwise, if you want to create a new DB2 database, tables, and indexes, you need only run the EQQINIDB sample as explained in “Step 12. Creating the DB2 database” on page 122.

Changing a shared DASD tracker-to-controller connection to an NCF, XCF, or TCP/IP connection

To change a shared DASD tracker connection to an NCF (VTAM), XCF (SYSPLEX), or TCP/IP one, perform the following steps :

1. To remove the DASD connection:
 - a. In the controller started task procedure:
 - 1) Remove the EQQEVDDnn DD statement pointing to the event data set of the specific tracker.
 - 2) Remove the DD statement pointing to the Submit/Release data set of the tracker. Not every DASD-connected tracker has a Submit/Release data set, but if one exists, its DDNAME in the controller procedure is the same as the destination listed under the DASD keyword in the ROUTOPTS initialization statement of the controller.
 - b. In the controller initialization parameters:
 - 1) Decrease the value of the OPCOPTS ERDRNUM() keyword by the number of DASD-connected trackers being removed. If there are no DASD-connected trackers, ERDRNUM() is 0.
 - 2) Remove from the ERDRPARAM() keyword the name of the PARMLIB member containing the parameters for the Event Reader task being deleted.
 - 3) Remove from the DASD keyword in the ROUTOPTS initialization statement the DDNAME of the Submit/Release data set of the tracker.

Changing a shared DASD tracker-to-controller connection

- c. In the controller ISPF dialogs:
 - 1) Remove the DDNAME of the Submit/Release data set of the tracker from the workstation destination under dialog option 1.1.2, using the M (modify) row command.
 - 2) Remove the workstation destination from ROUTOPTS and from the workstation definition.
- d. In the tracker started task procedure, remove the EQQSUDS DD statement.
- e. In the tracker initialization parameters:
 - 1) In the EWTROPTS statement, set the SUREL() keyword to NO .
 - 2) In the TRROPTS statement, remove the HOSTCON(DASD) keyword.
2. To add an NCF connection:
 - a. Define the VTAM LUs for the controller and the tracker. If necessary, create cross-domain definitions. Tivoli Workload Scheduler for z/OS requires that the LU name be the same as the ACBNAME in the APPL. For details, refer to “Step 15. Activating the network communication function” on page 131.
 - b. In the controller initialization parameters:
 - 1) In the OPCOPTS statement, set the NCFTASK() keyword to YES and write the LU name of the controller in the NCFAPPL() keyword.
 - 2) In the ROUTOPTS statement, write the LU name of the tracker in the SNA() keyword.
 - c. In the controller ISPF dialogs, write the LU name of the tracker in the workstation destination under dialog option 1.1.2, using the M (modify) row command.
 - d. In the tracker initialization parameters:
 - 1) In the OPCOPTS statement, set the NCFTASK() keyword to YES and write the LU name of the tracker in the NCFAPPL() keyword.
 - 2) In the TRROPTS statement, set the HOSTCON() keyword to SNA and write the LU name of the controller in the SNAHOST() keyword.
 - 3) In the EWTROPTS statement, set the EWSEQNO() keyword to 1.
3. To add an XCF connection:
 - a. In the SYS1.PARMLIB(COUPLEnn) member:
 - 1) Define the Tivoli Workload Scheduler for z/OS XCF transport class as described in “Updating XCF initialization options” on page 80.
 - 2) Define the XCF group that is to enable the controller to communicate with the trackers.
 - b. In the controller initialization parameters:
 - 1) In the XCFOPTS statement, code the GROUP(), MEMBER(), and TAKEOVER() keywords.
 - 2) In the ROUTOPTS statement, write the XCF MEMBERNAME of the tracker in the XCF() keyword.
 - c. In the controller ISPF dialogs, write the XCF MEMBERNAME of the tracker in the workstation destination under dialog option 1.1.2, using the M (modify) row command.
 - d. In the tracker initialization parameters:
 - 1) In the XCFOPTS statement, code the GROUP() and MEMBER() keywords.
 - 2) In the TRROPTS statement, set the HOSTCON() keyword to XCF.
 - 3) In the EWTROPTS statement, set the EWSEQNO() keyword to 1.
4. To add a TCP/IP connection:

Changing a shared DASD tracker-to-controller connection

- a. Define the IP addresses for the controller and tracker. For details, refer to “Step 16. Using TCP/IP for communication” on page 134.
- b. In the controller initialization parameters:
 - 1) In the TCP0PTS statement, set the values to define the details of the local controller. This statement is optional; if you do not specify it, the default values are taken.
 - 2) In the ROUTOPTS statement, write the TCP/IP destination name and IP address of the remote tracker in the TCPIP keyword.
- c. In the controller ISPF dialogs, write the TCP/IP destination name of the tracker in the workstation destination under dialog option 1.1.2, using the M (modify) row command.
- d. In the tracker initialization parameters:
 - 1) In the TCP0PTS statement, set the values to define the details of the local tracker or leave the default values.
 - 2) In the TRROPTS statement, set the HOSTCON() keyword to TCPIP and write the IP address of the controller in the TCPHOSTNAME() keyword.
 - 3) In the EWTROPTS statement, set the EWSEQNO() keyword to 1.
5. Stop and restart the controller and tracker for the parmlib changes to take effect, and run the CP extend or the CP replan command to update the current plan with the changed workstation destinations.

Running on upgraded operating systems

To run the scheduler on a new version of the z/OS operating system, you must reassemble the SMF and JES exits mentioned in “Step 5. Adding SMF and JES exits for event tracking” on page 72 with the libraries of the new operating system. Load modules EQQTTTOP and EQQPTTCP are pre-linked to support the SOCKET interface. These modules are release-dependent, therefore you must link EQQTTTOP and EQQPTTCP if you are installing Tivoli Workload Scheduler for z/OS for the first time and re-link them every time you upgrade to a later version of z/OS or Tivoli Workload Scheduler for z/OS.

If you upgrade the SMP/E environment to a later version of z/OS by using the SMP/E function BUILD MCS, the relink occurs automatically (ensure that the DDDEF entries for the new operating system are set up correctly by specifying the latest SEZACMTX and SCEELKED libraries). If you do not use BUILD MCS, relink the load modules by using the SMP/E function LINK LMODSCALLLIBS. With this function, all the Tivoli Workload Scheduler for z/OS modules are relinked to the latest SEZACMTX and SCEELKED libraries.

After you upgrade to a later version of z/OS, installing any Tivoli Workload Scheduler for z/OS PTF that updates EQQTTTOP and EQQPTTCP causes the automatic relink to these load modules. The automatic relink occurs regardless if you use the APPLY or APPLY REDO command.

Migrating actions

This chapter describes the tasks you must perform to complete a migration from Tivoli Workload Scheduler for z/OS Version 8.2, 8.3, 8.5 or 8.5.1 to version 8.6. The following topics are covered:

- Migrating data sets
- Switching into production mode
- Performing Fallback

Migrating data sets

For migration purposes, data sets fall into three categories:

- VSAM data sets that are copied and converted by the EQQICTOP migration program
- Non-VSAM data set that you can copy, or use unchanged, in the new version
- VSAM and non-VSAM data sets that must be empty when you migrate to Tivoli Workload Scheduler for z/OS

Each of these categories is described in the following sections.

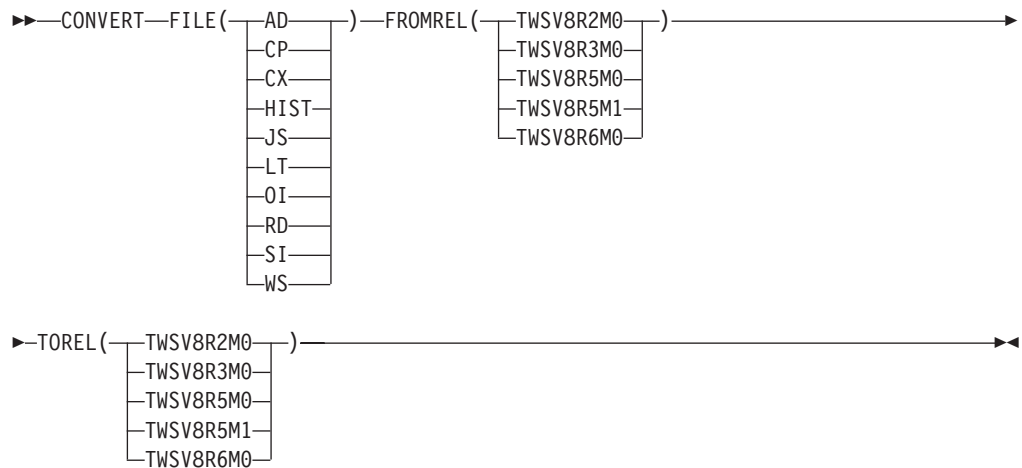
EQQICTOP VSAM data set conversion program

Purpose: With the EQQICTOP conversion program, you can migrate VSAM data sets from earlier releases of Tivoli Workload Scheduler for z/OS. You can also use the program to reverse the procedure in case you need to fall back to your old system.

The EQQJOBS program creates JCL tailored to your installation specifications in the EQQICNVS and EQQICNVH members.

EQQICTOP is controlled by CONVERT statements in the SYSIN file. You can supply any number of these statements to EQQICTOP.

Syntax:



Parameters:

FILE(*file identifier*)

Defines the data set to be converted. You can specify one of the following file identifiers on each CONVERT statement:

- AD** Application descriptions and JCL variable tables
- CP** The current plans, EQQCPnDS and EQQNCPDS
- CX** The current plan extension, EQQCXDS and EQQNCXDS
- HIST** DB2 operation history data from one release to another
- JS** JCL repository and retrieved job logs
- LT** Long-term plan
- OI** Operator instructions

EQQICTOP VSAM data set conversion program

RD Special resource definitions

SI Side information file, ETT criteria and configuration information

WS Workstation descriptions, calendars, and periods

Your conversion JCL should contain DD names EQQxxIN and EQQxxOUT for each data set that you want to convert, where xx is the file identifier.

FROMREL(*product identifier*)

Defines the product and release level of the input data set. You can specify one of the following:

TWSV8R2M0 Tivoli Workload Scheduler for z/OS Version 8 Release 2

TWSV8R3M0 Tivoli Workload Scheduler for z/OS Version 8 Release 3

TWSV8R5M0 Tivoli Workload Scheduler for z/OS Version 8 Release 5

TWSV8R5M1 Tivoli Workload Scheduler for z/OS Version 8 Release 5
Modification Level 1

TWSV8R6M0 Tivoli Workload Scheduler for z/OS Version 8 Release 6

TOREL(*product identifier*)

Defines the product and release level of the output data set. You can specify one of the following:

TWSV8R2M0 Tivoli Workload Scheduler for z/OS for z/OS version 8
Release 2

TWSV8R3M0 Tivoli Workload Scheduler for z/OS Version 8 Release 3

TWSV8R5M0 Tivoli Workload Scheduler for z/OS Version 8 Release 5

TWSV8R5M1 Tivoli Workload Scheduler for z/OS Version 8 Release 5
Modification Level 1

TWSV8R6M0 Tivoli Workload Scheduler for z/OS Version 8 Release 6

Notes:

1. Conversion stops if there is a VSAM I/O error on one of the files. One such error is a duplicate key on the output file. This can occur if the output data set is not empty.
2. Migrate the currently active JCL-repository data set. You can check whether the primary or alternate data set is in use by selecting option 6 in the Query Current Plan dialog. Do this when no work is running and before you stop the controller.
3. You can use one of two methods to convert the current plan:
 - If no error occurred when you stopped your production system, both primary and alternate current plans are the same. Use EQQCP1DS as input to the conversion program.
 - If the last action performed on your production system was to extend or replan the current plan, use the new-current-plan data set that was created on this system as input to the conversion program. This is the preferred method as it ensures you will not lose any job-tracking records, this is relevant if you use the track log (EQQTROUT) as an audit trail.

In both cases, the output file **must** be the new-current-plan data set (EQQNCPDS) on your Tivoli Workload Scheduler for z/OS system. You can convert the current plan extension data set using the same methods.

4. In addition to input and output DD names for each VSAM file, the migration JCL should also contain the EQQMLOG and EQQMLIB DD names. EQQMLOG is an output file for messages. EQQMLIB is an input file that contains the product message library.

Example:

```
//OPCMIG JOB (777777,777),'Migrate to Tivoli Workload Scheduler for z/OS V8R6M0, MSGLEVEL=(1,1),
//      NOTIFY=&SYSUID,MSGCLASS=H,CLASS=A
//*
//CONVERT EXEC PGM=EQQICTOP,REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPC.INST.LOADLIB
//EQQMLIB DD DISP=SHR,DSN=OPC.INST.SEQMSG0
//EQQMLLOG DD SYSOUT=*
//EQQADIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.AD
//EQQADOUT DD DISP=OLD,DSN=CCOPC.OPCC.AD
//EQQWSIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.WS
//EQQWSOUT DD DISP=OLD,DSN=CCOPC.OPCC.WS
//EQQCPIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.NCP
//EQQCPOUT DD DISP=OLD,DSN=CCOPC.OPCC.NCP
//EQQCXIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.NCX
//EQQCXOUT DD DISP=OLD,DSN=CCOPC.OPCC.NCX
//EQQLTIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.LT
//EQQLTOUT DD DISP=OLD,DSN=CCOPC.OPCC.LT
//EQQJSIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.JS1
//EQQJSOUT DD DISP=OLD,DSN=CCOPC.OPCC.JS1
//EQQOIIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.OI
//EQQOIOUT DD DISP=OLD,DSN=CCOPC.OPCC.OI
//EQQSIIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.SI
//EQQSIOUT DD DISP=OLD,DSN=CCOPC.OPCC.SI
//EQQRDIN DD DISP=SHR,DSN=CCOPC.OPCC.OLD.RD
//EQQRDOUT DD DISP=OLD,DSN=CCOPC.OPCC.RD
//SYSIN DD *

/* MIGRATION FROM Tivoli Workload Scheduler for z/OS V8.5.1 to */
/* Tivoli Workload Scheduler for z/OS V8.6.0 IS ASSUMED */
CONVERT FILE(AD) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(CP) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(CX) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(WO) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(LT) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(JS) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(OI) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(RD) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
CONVERT FILE(SI) FROMREL(TWSV8R5M1) TOREL(TWSV8R6M0)
```

In this example, all VSAM files are converted from a previous release to Tivoli Workload Scheduler for z/OS format. The tasks performed immediately before this job was submitted are listed here in order:

1. Verified JS1 as the active JCL repository in option 6.6 on the previous controller. If JS2 is the active JCL repository, use that as input but be sure to use JS1 as output because Tivoli Workload Scheduler for z/OS by default uses JS1 as the active JCL repository when you start a subsystem with an empty checkpoint data set.
2. The previous controller was shut down normally, as verified in the message log. Check that a current plan backup process was completed after the stop command was received by the subsystem.
3. A batch job was submitted to allocate and back up the previous data sets to new DSNs.
4. EQQPCS01 from Tivoli Workload Scheduler for z/OS EQQJOBS was submitted to allocate the VSAM clusters required for Tivoli Workload Scheduler for z/OS.
5. The old NCP is used as input if a daily plan batch process was submitted on the previous system prior to shutdown. Output is the Tivoli Workload Scheduler for z/OS NCP.

Data sets that you need to convert

Allocate new VSAM data sets for Tivoli Workload Scheduler for z/OS. Existing data can then be migrated using EQQICTOP. Keep a copy of the old data sets for backup and fallback purposes. The following data sets must be migrated to Tivoli Workload Scheduler for z/OS format:

Data sets you need to convert

Table 33. Data sets that you need to convert

DD Name	Description
EQQADDS	Application descriptions and JCL variable tables
EQQJSnDS	JCL repository (currently active)
EQQLTDS	Long-term plan
EQQNCPDS, or EQQCPnDS	The current plan, but use the NCP as input if a daily plan process created an NCP after the old system was shut down.
EQQNCXDS, or EQQCXnDS	The current plan extension, but use the NCX as input if a daily plan process created an NCX after the old system was shut down.
EQQOIDS	Operator instructions
EQQRDDS	Special resource definitions
EQQSIDS	Side information, ETT criteria and configuration information
EQQWSDS	Workstation descriptions, calendars, and periods

Data sets that can be used

Tivoli Workload Scheduler for z/OS can use unchanged data from the following data sets:

Table 34. Data sets that Tivoli Workload Scheduler for z/OS can use

DD Name	Description
EQQEVLIB	Configuration file repository for event-triggered resource handling
EQQINCWK	JCC incident work file
EQQJBLIB ¹	JCL library
EQQJCLIB	Job-completion-checker (JCC) message-table library
EQQJTABL	Job table log file
EQQPRLIB	Automatic-recovery-procedure library
EQQSCLIB	Script library for end-to-end scheduling with fault tolerance capabilities
—	JCC incident log
¹ If this library contains jobs for the scheduler planning, the JCL must be modified to reflect the new installation.	

Empty data sets

When you have completed your testing of Tivoli Workload Scheduler for z/OS and have performed data set migration, ensure that the following data sets are empty before you start the product for the first time in production:

EQQCKPT Checkpoint
EQQCXDS Current plan extension
EQQDLnn Dual job-tracking logs
EQQEVDS Event data sets
EQQHHTTP0 Event data set for end-to-end scheduling with z-centric capabilities
EQQJTARC Job-tracking archive
EQQJTnn Job-tracking logs
EQQMLOG Message log
EQQMONDS Monitoring Task Data Set

EQQSCPDS	Secondary Current Plan Data Set
EQQSTC	Started-task submit
EQQSUDS	Submit release
EQQTROUT	Job-tracking log copy created by the daily planning jobs. Input to the EQQAUDIT program, that is not downward compatible
EQQTWSIN	Input event data set for end-to-end scheduling with fault tolerance capabilities
EQQTWSOU	Output event data set for end-to-end scheduling with fault tolerance capabilities
EQQTWSCS	Centralized script work repository

EQQCP n DS , EQQCXDS, and EQQSCPDS do not have to be empty, but they can be. When the product is started for the first time, you **must** specify CURRPLAN(NEW) on the JTOPTS statement. Therefore, any data in the EQQCP n DS and EQQCXDS data sets will immediately be replaced by the contents of EQQNCPDS and EQQNCXDS. Similarly, the inactive EQQJS n DS data set (EQQJS2DS in this example) does not have to be empty, although it can be.

Tracker and Data Store considerations

When you migrate a tracker, it is not necessary for EQQEVDS and EQQSUDS to be empty. The migration enables you to use the subsystem with the new release and modifies the JCL trackers to point to the libraries used, for example EQQMLIB or STEPLIB.

When you migrate a Data Store, consider that:

- In the Data Store, it is not necessary for EQQPKI01, EQQSKI01, EQQSDF nn , EQQUDF nn to be empty.
- In the controller mini Data Store:
 - If the CP files are kept, it is not necessary for EQQPKI01, EQQSKI01, EQQSDF nn to be empty.
 - If the CP files are deleted, EQQPKI01, EQQSKI01, EQQSDF nn must be empty.

The data store and controller tasks might be migrated at different times, provided that the maintenance level of the old and new release of Tivoli Workload Scheduler for z/OS is the same. This means that you should apply any PTF which affects both controller and data store code to both releases of the product. If this level of concurrent PTF maintenance cannot be maintained, it is best to keep the data store and controller on the same release of Tivoli Workload Scheduler for z/OS. If the migration is successfully performed, you should be able to use the Restart and Cleanup function on the new release for any operation which was on the error list on the old release of Tivoli Workload Scheduler for z/OS

If you change a datastore connection type and you want to reflect the naming convention in the FLOPTS destination name, keep the former destination name in the FLOPTS parameter that corresponds to the connection type to be used (SNADEST, XCFDEST, or TCPDEST). For example, suppose that you change the datastore connection from SNA to XCF and the former FLOPTS is SNADEST(OPCTRK1.DST). If you want to use XCFTRK1.DST as new destination name, specify the following FLOPTS parameter: XCFDEST(OPCTRK1.DST, XCFTRK1.DST). Omitting the former destination produces the messages EQQFL18E and EQQM643W in the controller message log, when retrieving any joblog stored with the former destination name.

Switching into production mode

When you have completed the migration steps and tested your system, you should be able to switch Tivoli Workload Scheduler for z/OS into production with a minimum interruption to your normal processing. An example explains the process in the following steps:

1. Closing down your production system
2. Converting VSAM files to Tivoli Workload Scheduler for z/OS format
3. Starting the new system
4. Validating the new system

Consider this scenario:

- An installation is running three z/OS systems: MVS1, MVS2, and MVS3. MVS1 is connected to MVS2 and to MVS3 via a VTAM link. The production control system (OPCA), a tracker (OPCB), and an end-to-end server (OPCS) connected to a network of fault-tolerant workstations are started on MVS1.
- MVS2 and MVS3 are controlled systems. A tracker (OPCC) is started on MVS2. A tracker (OPCD) is started on MVS3.

The objective is to stop the OPCA and OPCB and migrate them to Tivoli Workload Scheduler for z/OS with as little impact on users as possible. One possible method is described here. Modify it as required to suit the specific needs of your installation.

In the example, you should first make sure that:

1. You prepared JCL to:
 - Back up the Tivoli Workload Scheduler for z/OS environment.
 - Allocate the Tivoli Workload Scheduler for z/OS VSAM and non-VSAM files.
2. All trackers (OPCB, OPCC, and OPCD) are active at the beginning of the migration process. The sequence in which the trackers are started and stopped is the key to a successful migration to a new Tivoli Workload Scheduler for z/OS system.

Closing down your production system

If your trackers have a large CSA area defined, you do not have to worry about losing events. It is assumed here that area is quite small, so you should deliberately slow down the event-generating rate as much as possible. To do this, perform the following actions:

1. From the Service Functions dialog on the production system (OPCA), deactivate job submission for jobs running in the host environment and on fault-tolerant workstations, and hold JES job queues, if ETT is used.
2. After all jobs in the current plan that are currently active in the host environment have completed, stop the two controlled systems, OPCC and OPCD. If there are many jobs still on the JES job queues, or if many new jobs are still arriving from outside processes, hold the job queues on MVS2 and MVS3.
3. Stop all fault-tolerant workstations in the network using one of the available Tivoli Workload Scheduler interfaces, or locally on the Fault-Tolerant workstation using the `conman stop` command.
4. Before you proceed with the next steps, wait until in the EQQTWSIN and EQQTWSOU data sets all the events are processed. To verify this, use the sample utility `EQQCHKEV`, provided in the sample library.

The `EQQCHKEV` utility will check the data set structure of `EQQTWSIN` and `EQQTWSOU` which are the input and output end-to-end event data sets from

the version you are migrating. The utility will provide an informational message indicating the number of events that still need to be processed. When the data set will contain 0 unprocessed events you can proceed with the migration. The utility also checks the integrity of the data sets and will issue an appropriate error message in case of corruption or inconsistency.

5. From the Daily Plan dialog on the production system (OPCA), create a replan or plan-extend batch job. Change the job card to contain TYPRUN=HOLD, and submit the job. Save the JCL in a data set in case you have to resubmit it to correct an error.
6. If you specified CHECKSUBSYS(YES) on the BATCHOPT statement used by the batch job, change it to CHECKSUBSYS(NO). In the BATCHOPT statement used by the batch job, comment out the TPLGYPRM keyword if it is used.
7. Using the Query Current Plan dialog on the production system (OPCA), check which JS file is currently in use on this system.
8. Stop the OPCA and OPCB systems. Release the daily plan job from hold, and make sure that it runs successfully.
9. When the daily plan job has finished, verify that it ran successfully. This is indicated by a return code of 0 or 4. If required, correct any problems and rerun the job until a new current plan (NCP) data set has been created. If you have fault-tolerant workstations and you have commented out the TPLGYPRM keyword in the BATCHOPT statement, the warning message EQQ3041W is displayed in the daily plan job output for each fault-tolerant workstation.

Converting VSAM files to the new system format

The next step is to create VSAM files for the new system. You can do this as follows:

1. Create a backup copy of the Tivoli Workload Scheduler for z/OS VSAM files.
2. Allocate VSAM clusters for Tivoli Workload Scheduler for z/OS using the EQQPCS01 job.
3. Review the EQQICNVS sample job. Ensure that input and output data set names are correctly specified. Make sure to select the current JS file. When defining input and output files for the CP file conversion, use the NCP file, as a new current plan has just been created.
4. Run EQQICNVS to convert the VSAM data to Tivoli Workload Scheduler for z/OS format.
5. Verify that the conversion program ran successfully. If there are any problems converting the VSAM files, you should abandon the migration.
6. Backup the Tivoli Workload Scheduler for z/OS non-VSAM data sets.
7. Allocate Tivoli Workload Scheduler for z/OS non-VSAM data sets using the EQQPCS01 and EQQPCS02 jobs.
8. If you have stopped migrating, start OPCA, OPCB, OPCC, and OPCL. Release any held queues and restart any drained initiators.

Starting the new system

In the following procedure, it is assumed that VSAM file conversion was successful. Ensure that the data sets referred to in “Empty data sets” on page 184 are empty. To start the new system, perform the following actions:

1. Modify the JCL procedure for OPCA to include the new DD names and data sets added in Tivoli Workload Scheduler for z/OS. Use ISPF browse to ensure that all job-tracking logs (EQQJTnn), the job-tracking archive (EQQJTARC), and the checkpoint (EQQCKPT) are empty data sets. If you use dual job-tracking logs (EQQDLnn), they should also be empty data sets.

Starting the new system

2. Modify initialization parameters for OPCA. The CKPT data set is not yet initialized the first time you start OPCA after migration, and hence you must specify CURRPLAN(NEW) in JTOPTS. Specify BUILDSSX(REBUILD) and SSCMNAME(EQQSSCMJ,TEMPORARY) on the OPCOPTS initialization statement. Specify the PIFHD keyword on the INTFOPTS initialization statement.

As soon as OPCA starts, change back to CURRPLAN(CURRENT), to prevent OPCA from recovering from the new current plan each time it starts.

Note: You might find it useful to specify JOBSUBMIT(NO) and FTWJSUB(NO) in the JTOPTS initialization statement so that work is not submitted when you start OPCA. When you have checked that OPCA has started without errors, you can activate job submission using the Service Functions dialog.

To initialize the checkpoint data set, you must specify OPCHOST(YES) in OPCOPTS. This so that, when the scheduler starts, the NMM task initializes the checkpoint data set with FMID and LEVEL corresponding to SSX. The OPCHOST value can then be changed. For example, you can change the value to OPCHOST(PLEX) when the subsystem is used as the controlling system in XCF.

3. Run the EQQPCS05 job to create the work directory. Optionally, back up any important data that you have in the old work directory, for example, the LOCALOPTS file, to merge it later in the new work directory.
4. Start OPCA. Verify that no errors occurred during initialization. If required, correct any errors and restart OPCA.
5. Modify initialization parameters for OPCB. Specify BUILDSSX(REBUILD) and SSCMNAME(EQQSSCMJ,TEMPORARY) on the OPCOPTS initialization statement. Specify the PIFHD keyword on the INTFOPTS initialization statement.
6. Start OPCB and OPCS.
7. Restart drained initiators on the MVS1 system.
8. Enter the Service Functions dialog on OPCA, and activate job submission (if it is not already active).
9. Start the OPCC and OPCD systems. Release held queues, and restart drained initiators if required.
10. Change JTOPTS CURRPLAN(NEW) to CURRPLAN(CURRENT).
11. Uncomment the TPLGYPRM keyword in the BATCHOPT statement if you commented it out. Submit a daily plan replan or extend **as soon as possible** after migration. In addition to the current plan, this will also generate a new symphony file. Until a new current plan is created, any references to special resources will cause the resource object to be copied from the EQQRDDS to the current-plan-extension data space. This processing has some performance overheads.

The new-current-plan-extension data set (EQQNCXDS) is built during daily planning to contain all special resources referenced by operations in the new current plan.

Before the next IPL of the system, remove the BUILDSSX and SSCMNAME keywords from OPCA and OPCB initialization statements if the subsystem name table (IEFSSN nm) in SYS1.PARMLIB has been updated to correctly specify EQQINITJ and EQQSSCMJ.

Validating the new system

Now you must validate that your new system works as expected. To do this, perform the following steps:

1. From the Ready List dialog, review the status of active operations.
2. Check that the operations that are becoming ready on the workstations representing the three z/OS systems are successfully submitted to the intended system. Also check that the ending status is correctly reflected in the ready lists.
3. Verify that the current plan and the long-term plan can be extended successfully.
4. Verify that other Tivoli Workload Scheduler for z/OS-related processes (for example, the dialogs, batch programs, and PIF-based programs) work as expected.

Migration steps for a system in a heavy workload environment

If your production environment has such a heavy workload that you cannot suspend job processing and phase out production, you can use the procedure described in the following steps as an alternative to the standard process described in “Switching into production mode” on page 186. The standard process is recommended in all other cases.

The scenario used involves the same systems as in the standard process.

To migrate your production system, you perform the following steps:

1. Close down your production system
2. Convert VSAM files to the new system format
3. Initialize the new system
4. Produce a checkpoint data set containing data from the old production system
5. Start the new system
6. Validate the new system

Close down your production system:

1. From the Service Functions dialog on the production system (OPCA), deactivate job submission for jobs running on fault-tolerant workstations.
2. Stop all fault-tolerant workstations in the network using one of the available Tivoli Workload Scheduler interfaces, or locally on the Fault-Tolerant workstation using the `conman stop` command.
3. Before you proceed with the next steps, wait until all the events are processed in the EQQTWSIN and EQQTWSOU data sets. To verify this, use the sample utility EQQCHKEV, provided in the sample library.

The EQQCHKEV utility checks the data set structure of EQQTWSIN and EQQTWSOU which are the input and output end-to-end event data sets from the version you are migrating. The utility provides an informational message indicating the number of events still to be processed. When the data set contains zero unprocessed events you can proceed with the migration. The utility also checks the integrity of the data sets and issues an appropriate error message in case of corruption or inconsistency.

4. From the Daily Plan dialog on the production system (OPCA), create a replan or plan-extend batch job. Change the job card to contain `TYPRUN=HOLD`, and submit the job. Save the JCL in a data set in case you have to resubmit it to correct an error.
5. If you specified `CHECKSUBSYS(YES)` on the `BATCHOPT` statement used by the batch job, change it to `CHECKSUBSYS(NO)`. In the `BATCHOPT` statement used by the batch job, comment out the `TPLGYPRM` keyword if it is used.

Closing your production system

6. Using the Query Current Plan dialog on the production system (OPCA), check which JS file is currently in use on this system.
7. Stop OPCA and OPCS, release the daily plan from hold and make sure it runs successfully.

Convert VSAM files to the new system format:

1. Create a backup copy of the Tivoli Workload Scheduler for z/OS VSAM files.
2. Allocate VSAM clusters for Tivoli Workload Scheduler for z/OS using the EQQPCS01 job.
3. Review the EQQICNVS sample job. Ensure that input and output data set names are correctly specified. Make sure you select the current JS file. When defining input and output files for the CP file conversion, use the NCP file, because a new current plan has just been created.
4. Run EQQICNVS to convert the VSAM data to Tivoli Workload Scheduler for z/OS format.
5. Verify that the conversion program ran successfully. If there are any problems converting the VSAM files, you should abandon the migration.
6. Back up the Tivoli Workload Scheduler for z/OS non-VSAM data sets.
7. Allocate Tivoli Workload Scheduler for z/OS non-VSAM data sets using the EQQPCS01 and EQQPCS02 jobs.
8. If you have stopped migrating, start OPCA, OPCB, and OPCC. Release any held queues and restart any drained initiators.

Initialize the new system: Before you perform the steps described in this section, ensure that the VSAM file conversion described in the preceding section was successful.

1. Ensure that the data sets referred to in “Empty data sets” on page 184 are empty. Use ISPF browse to ensure that all job-tracking logs (EQQJTnn), the job-tracking archive (EQQJTARC), and the checkpoint (EQQCKPT) data sets are empty. If you use dual job-tracking logs (EQQDLnn), they should also be empty.
2. Modify the JCL procedure for OPCA to include the new DD names and data sets added in IBM Tivoli Workload Scheduler for z/OS.
3. Modify initialization parameters for OPCA. The CKPT data set is not yet initialized the first time you start OPCA after migration, so you must specify CURRPLAN(NEW) in JTOPTS. Specify BUILDSSX(REBUILD) and SSCMNAME(EQQSSCMJ,TEMPORARY) in the OPCOPTS initialization statement. Specify the PIFHD keyword in the INTFOPTS initialization statement. As soon as OPCA has started, change back to CURRPLAN(CURRENT), to prevent OPCA from recovering from the new current plan each time it starts.

Note: You might find it useful to specify JOBSUBMIT(NO) and FTWJSUB(NO) in the JTOPTS initialization statement so that work is not submitted when you start OPCA. When you have checked that OPCA has started without errors, you can activate job submission using the Service Functions dialog.

To initialize the checkpoint data set, specify OPCHOST(YES) in OPCOPTS. This is so that, when the scheduler starts, the NMM task initializes the checkpoint data set with FMID and LEVEL corresponding to SSX. The OPCHOST value can then be changed. For example, you can change the value to OPCHOST(PLEX) when the subsystem is used as the controlling system in XCF.

4. Run the EQQPCS05 job to create the work directory. Optionally, back up any important data that you have in the old work directory, for example, the LOCALOPTS file, to merge it later in the new work directory.
5. Start OPCA. Verify that no errors occurred during initialization. If required, correct any errors and restart OPCA.
6. Stop OPCA.

Produce a checkpoint data set containing data from the old production system:

Produce a checkpoint data set containing data from the old production system:

1. Merge OLD.CKPT, from the version you are migrating, and the newly allocated CKPT, created in the previous section, into CKPT.NEW using a job such as the following, which you customize for your environment:

```
//COPY      EXEC PGM=IDCAMS,REGION=512k
//CKPTOLD   DD DSN=OPCAHLQS.CKPT.OLD,DISP=SHR
//CKPT      DD DSN=OPCAHLQS.CKPT,DISP=SHR
//CKPTNEW   DD dsn=OPCAHLQS.CKPT.NEW,DISP=MOD
//SYSPRINT  DD SYSOUT=*
//SYSIN     DD *
            REPRO IFILE(CKPT) OFILE(CKPTNEW) COUNT(1)
            REPRO IFILE(CKPTOLD) OFILE(CKPTNEW) SKIP(1)
/* "
```

2. Back up the current CKPT and then rename CKPT.NEW to the current CKPT.

Start the new system:

1. Change JTOPTS CURRPLAN(NEW) to CURRPLAN(CURRENT).
2. Start the controller OPCA. The merged checkpoint data set will enable it to continue reading the event records.
3. Start all the trackers without BUILDSSX. Ensure that the load modules invoked are still those for the version from which you are migrating.
4. Stop the trackers after the events in CSA are processed.
5. Modify initialization parameters for OPCB. Specify BUILDSSX(REBUILD) and SSCMNAME(EQQSSCMJ,TEMPORARY) on the OPCOPTS initialization statement. Specify the PIFHD keyword on the INTFOPTS initialization statement.
6. Start the OPCB and OPCS.
7. Restart drained initiators on the MVS1 system.
8. Enter the Service Functions dialog on OPCA, and activate job submission (if it is not already active).
9. Start the OPCC and OPCD systems. Release held queues, and restart drained initiators if required.
10. Submit a daily plan replan or extend **as soon as possible** after migration. Until a new current plan is created, any references to special resources will cause the resource object to be copied from the EQQRDDS to the current-plan-extension data space. This processing has some performance overheads.

The new-current-plan-extension data set (EQQNCXDS) is built during daily planning to contain all special resources referenced by operations in the new current plan.

Validate the new system:

1. From the Ready List dialog, review the status of active operations.

Validating the new system

2. Check that the operations that are becoming ready on the workstations representing the three z/OS systems are successfully submitted to the intended system. Also check that the ending status is correctly reflected in the ready lists.
3. Verify that the current plan and the long-term plan can be extended successfully.
4. Verify that other Tivoli Workload Scheduler for z/OS-related processes (for example, the dialogs, batch programs, and PIF-based programs) work as expected.

Performing fallback

If a problem occurs after Tivoli Workload Scheduler for z/OS has been active as a production system for some time, and the problem is serious enough, you might need to stop the new system and return the workload to the previous system. You can do this using a procedure called *fallback*, if the Tivoli Workload Scheduler for z/OS data sets are usable. The procedure is:

1. Run the EQQPCS01 and EQQPCS02 jobs to allocate new data sets for the old production system. The current data sets, or a copy, used by the Tivoli Workload Scheduler for z/OS systems should be kept for problem determination purposes.
2. Run the EQQPCS05 job to create the work directory.
3. If required, close down the systems in the same way as during migration. This is required if the current plan on the OPCA system is intact and job tracking is working normally.
4. If possible, create up-to-date data sets for the long-term plan and new current plan for the OPCA system. Do not submit a REPLAN job prior to shutdown, unless the PERMANENT option was used for SSCMNAME on the converted system, or if SSCMNAME was not specified. If SSCMNAME(EQQSSCMJ,TEMPORARY) was used, message EQQZ190E will be issued if a REPLAN job is started after the controller is shut down.
5. Build VSAM data sets for the old system by running the EQQICNVS job to convert Tivoli Workload Scheduler for z/OS files to their previous format. Note that the job log to be retrieved by the Data Store will be left as it is.
6. Start the OPCA and OPCB systems again using the converted files and start OPCS.

Note: You might find it useful to specify JOBSUBMIT(NO) and FTWJSUB(NO) on the JTOPTS initialization statement so that work is not submitted when you start OPCA. When you have checked that old system has started without errors, you can activate job submission using the Service Functions dialog.

7. If the new current plan (NCP) data set is not fully up-to-date because you could not run the daily plan program, use the MCP dialog to update the status of operations to make the current plan up-to-date.
8. Start the OPCC and OPCD systems again. Use the SSCMNAME keyword on the JTOPTS initialization to load the current subsystem communication module for the release you are falling back to. Activate MVS systems as required.

Note: If you experience problems with your Tivoli Workload Scheduler for z/OS system and you need to migrate back to an earlier release, you must consider the impact that this will have on all aspects of your configuration. This is especially important for connectivity items. Consider all possible migration actions when planning the migration and fallback procedures for your installation.

In the following example, all VSAM files are converted from the current version of Tivoli Workload Scheduler for z/OS to the format of the previous release of the product. The tasks to perform immediately before this job are listed here in order:

1. Verify JS1 as the active JCL repository in option 6.6 on the controller. If JS2 is the active JCL repository, use that as input, but be sure to use JS1 as output because Tivoli Workload Scheduler for z/OS by default uses JS1 as the active JCL repository when you start a subsystem with an empty checkpoint data set.
2. Shut down the controller normally. Verify this using the message log. Check that a current plan backup process was completed after the stop command was received by the subsystem.
3. Submit a EQQPCS01 job that you generate from EQQJOBS to reallocate the VSAM clusters.
4. Check that a daily plan batch process had not been submitted on the system prior to shutdown. Therefore the Tivoli Workload Scheduler for z/OS CP1 is used as input, and the output is the NCP.

```
//OPCBAK JOB (77777,777),'Fallback to V8R5M1',MSGLEVEL=(1,1),
//      NOTIFY=&SYSUID,MSGCLASS=H,CLASS=A
//*
//CONVERT EXEC PGM=EQQICTOP,REGION=2048K
//STEPLIB DD DISP=SHR,DSN=OPCESA.INST.LOADLIB
//EQQMLIB DD DISP=SHR,DSN=OPCESA.INST.SEQMSG0
//EQQMLLOG DD SYSOUT=*
//EQQADIN DD DISP=SHR,DSN=CCOPC.OPCC.AD
//EQQADOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.AD
//EQQWSIN DD DISP=SHR,DSN=CCOPC.OPCC.WS
//EQQWSOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.WS
//EQQCPIN DD DISP=SHR,DSN=CCOPC.OPCC.CP1
//EQQCPOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.NCP
//EQQLTIN DD DISP=SHR,DSN=CCOPC.OPCC.LT
//EQQLTOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.LT
//EQQJSIN DD DISP=SHR,DSN=CCOPC.OPCC.JS1
//EQQJSOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.JS1
//EQQOIIIN DD DISP=SHR,DSN=CCOPC.OPCC.OI
//EQQOIIOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.OI
//EQQSIIN DD DISP=OLD,DSN=CCOPC.OPCC.SI
//EQQSIOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.SI
//EQQCXIN DD DISP=OLD,DSN=CCOPC.OPCC.CX
//EQQCXOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.NCX
//EQQRDIN DD DISP=OLD,DSN=CCOPC.OPCC.RD
//EQQRDOUT DD DISP=OLD,DSN=CCOPC.OPCC.OLD.RD
//SYSIN DD *
/* FALLBACK FROM Tivoli Workload Scheduler for z/OS V8.6.0 to Tivoli OPC V8.5.1 IS ASSUMED */
CONVERT FILE(AD) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(CP) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(WO) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(LT) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(JS) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(OI) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(CX) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(RD) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
CONVERT FILE(SI) FROMREL(TWSV8R6M0) TOREL(TWSV8R5M1)
```

Part 3. Tivoli Workload Scheduler for z/OS Connector

Chapter 7. Installing, Upgrading, and Uninstalling on the embedded WebSphere Application Server	197	
Preparing.	197	
Authorization roles required for installing, upgrading, or uninstalling	197	
Instances of Tivoli Workload Automation	197	
Installing	198	
Installation and uninstallation log files	199	
Installing using the wizard in interactive mode	200	
Installing using the wizard in silent mode	202	
Installing using response file templates	202	
Installing with an automatically generated response file.	203	
Installing from the launchpad	204	
Upgrading	204	
Upgrading with the wizard in interactive mode	204	
Upgrading from version 8.3	204	
Upgrading from version 8.5 or 8.5.1	205	
Upgrading in silent mode	206	
Upgrading from the launchpad	206	
Uninstalling	207	
Uninstalling using the wizard	207	
Uninstalling in silent mode.	207	
Chapter 8. Installing and uninstalling on WebSphere Application Server for z/OS	209	
Business Scenario	209	
Authorization roles required for installing and uninstalling	209	
Installing on WebSphere Application Server for z/OS	209	
Installing using the Integrated Solutions Console	210	
Installing using the zConnInstall.sh script	211	
Installation and uninstallation log files	213	
Enabling communications with Dynamic Workload Console	213	
Applying maintenance	216	
Uninstalling	217	
Uninstalling using the Integrated Solutions Console	217	
Uninstalling using the zConnUninstall.sh script	217	
Chapter 9. Troubleshooting and maintaining the installation	219	
Troubleshooting the installation	219	
z/OS connector installation step hangs while installing on a TWA instance with an existing embedded WebSphere Application Server	219	
On Windows the z/OS connector installation step fails because the user account does not belong to the Administrators group	219	
Dynamic Workload Console creates wrong connection upon installation	220	
Installation fails because Windows Workstation Service is not started	220	
Failed installation of a dynamic domain manager in the same instance as the z/OS connector.	220	
Maintaining the installation of the z/OS connector	220	
Updating the SOAP properties after changing the WebSphere Application Server user or its password.	221	
Updating the SOAP properties usage	221	
updateWas.sh (.bat)	221	

Chapter 7. Installing, Upgrading, and Uninstalling on the embedded WebSphere Application Server

To use the Dynamic Workload Console you must install the Tivoli Workload Scheduler for z/OS connector. This chapter describes how to install, upgrade, and uninstall this connector on the embedded WebSphere Application Server, and what to do if you encounter problems. It contains the following chapters:

- “Preparing”
- “Installing” on page 198
- “Upgrading” on page 204
- “Uninstalling” on page 207
- Chapter 9, “Troubleshooting and maintaining the installation,” on page 219

Preparing

Before commencing an installation, upgrade, or uninstallation, read the following information:

- “Authorization roles required for installing, upgrading, or uninstalling”
- “Instances of Tivoli Workload Automation”

Authorization roles required for installing, upgrading, or uninstalling

To install, upgrade, or uninstall Tivoli Workload Scheduler for z/OS connector, you must log in as a user with the following authorization roles:

On Windows:

Your login account must be a member of the Windows **Administrators** group or domain administrators with Act as Part of the Operating System.

On UNIX and Linux:

Root access.

Instances of Tivoli Workload Automation

During the installation of Tivoli Workload Scheduler for z/OS, decide if you want to install into an existing instance of Tivoli Workload Automation or to create a new instance.

Each instance of Tivoli Workload Automation can contain the following:

- One instance of the embedded IBM WebSphere Application Server on which can run:
 - One instance of a master domain manager, backup master domain manager, dynamic domain manager, backup dynamic domain manager, domain manager with distributed connector, or fault-tolerant agent with distributed connector
 - One instance of the Dynamic Workload Console
 - One instance of the Tivoli Workload Scheduler for z/OS Connector
- If no other Tivoli Workload Scheduler component (master domain manager, backup master domain manager, dynamic domain manager, backup dynamic domain manager, domain manager with distributed connector, or fault-tolerant

Instances of Tivoli Workload Automation

agent with distributed connector) is installed, one instance of a domain manager or fault-tolerant agent without a distributed connector.

Only one Dynamic Workload Console can be installed on a workstation and can be installed as follows:

- In an existing Tivoli Workload Automation instance
- In a new Tivoli Workload Automation instance
- Outside any Tivoli Workload Automation instance, using an existing external instance of Tivoli Integrated Portal.

Thus, if you are installing into an existing instance of Tivoli Workload Automation, you can install certain products or components, depending on the products or components that currently exist in that instance. Table 35 describes the actions that you can perform in each different scenario.

Table 35. Installing into an existing instance of Tivoli Workload Automation

If the existing Tivoli Workload Automation instance contains:	You can perform the following:
A Tivoli Workload Scheduler version 8.6 master domain manager, domain manager, or backup master or Dynamic Workload Console version 8.6	Install a Tivoli Workload Scheduler for z/OS version 8.6 connector on a common embedded WebSphere Application Server.
A Tivoli Workload Scheduler version 8.5 or 8.5.1 master domain manager, domain manager, or backup master or Dynamic Workload Console version 8.5 or 8.5.1	Take no action.
A Tivoli Workload Scheduler for z/OS version 8.3, 8.5, or 8.5.1 stand-alone connector	Upgrade that component.
A Tivoli Workload Scheduler for z/OS version 8.6 connector	Take no action.
A Tivoli Workload Scheduler for z/OS version 8.5 or 8.5.1 shared connector on a common embedded WebSphere Application Server	Upgrade the connector if the Tivoli Workload Scheduler component is V8.6.

Note: The advantage of installing a product or component into an existing instance of Tivoli Workload Automation is that all of the data that is required to configure the component is already present and displayed in the wizard. In some cases, data from the existing instance is reused automatically. In other cases, data is retrieved as default values that you can choose to use or edit.

Installing

Tivoli Workload Scheduler for z/OS connector requires the IBM WebSphere Application Server. If the installation program does not detect the existence of the WebSphere Application Server on the destination computer, it installs an instance of the embedded WebSphere Application Server.

There are several ways to install the Tivoli Workload Scheduler for z/OS connector. They are:

Installation wizard

Installing Tivoli Workload Scheduler for z/OS connector

Setup files are available to start the installation wizard on all supported operating systems. The wizard guides you through the installation steps. See “Installing using the wizard in interactive mode” on page 200 for details.

This method of installation requires Java Virtual Machine to be installed on the computer where you run it.

Silent mode

You can use this method to run the installation process unattended and in the background. In silent mode, a response file provides the relevant information to the installation process, which is run in the background. First, you customize a *response file* by adding all the configuration settings needed during installation. Then, from the command line, you run the *setup* command. See “Installing using the wizard in silent mode” on page 202 for details.

Launchpad

The *launchpad* is the starting point for installing products that are part of Tivoli Workload Automation, as well as DB2, on Windows, UNIX, and Linux platforms. Using the launchpad, you can:

- Install or upgrade all Tivoli Workload Scheduler components
- Install or upgrade Dynamic Workload Broker
- Install or upgrade Dynamic Workload Console
- Install or upgrade Tivoli Workload Scheduler for z/OS connector
- Install DB2
- Access product information

See “Installing from the launchpad” on page 204 for details.

Installation and uninstallation log files

You can check the following log files for information about the installation. Details of the installation process are kept in log files on the local workstation in the following directories:

Tivoli Workload Scheduler

On Windows operating systems:

c:\Documents and Settings\installing_user\Local
Settings\Temp\TWA\twszconn86

On UNIX and Linux operating systems:

/tmp/TWA/twszconn86

Table 36 lists the log files.

Table 36. Installation log files

Log file name	Content
twstatus.log	Tivoli Workload Scheduler installation status log file. It reports if the installation completed successfully or with errors. In case of errors it indicates if the error is due to an incorrect field value or to a failed step.
twismp.log	Tivoli Workload Scheduler installation trace file.
summary.log	Tivoli Workload Scheduler installation log file. It is generated when the installation completes. It contains summary information on the installation.

Installing Tivoli Workload Scheduler for z/OS connector

For multiple installations on the same workstation, the log header and footer indicate the user ID (*TWS_user*) for which the installation was performed.

Note: If you are running a silent installation and the response file you are using does not have the correct syntax, the installation fails without producing a log file.

Installing using the wizard in interactive mode

To install Tivoli Workload Scheduler for z/OS connector with the installation wizard, follow these steps:

1. Run the setup for the operating system on which you are installing. From the installation media, run the following program to start the wizard:

On Windows:

TWSZOS\SETUP.cmd or TWSZOS\SETUP.exe

On UNIX and Linux

TWSZOS/SETUP.sh or TWSZOS/SETUP.bin

2. Follow the installation wizard panels to complete the installation. The following list describes the fields that you are asked to complete during the process:

- a. Tivoli Workload Scheduler for z/OS connector user details. They are:

User name

The user name (*TWS_user*) for which you want to install the Tivoli Workload Scheduler for z/OS connector.

If no *TWS_user* is defined on the workstation, the installation program creates one with the name and password you provide.

If the installation program finds an existing Tivoli Workload Automation instance, you can choose between installing the z/OS connector there or in a new instance. In the first case you are asked to provide the existing *TWS_user* name and password.

Password

The password of the *TWS_user*.

- b. The port numbers to be used by the WebSphere Application Server embedded in this instance of Tivoli Workload Scheduler for z/OS connector. Leave the default values unless you know that they are already in use. They are:

HTTP transport

The port for the HTTP transport. The default value is **31215**.

HTTPS transport

The port for the secure HTTP transport. The default value is **31216**.

Bootstrap

The port for the bootstrap or RMI. The default value is **31217**.

SOAP connector

The port for the application server protocol SOAP connector. The default value is **31218**.

SAS Server Authentication Listener

The port used by the Secure Association Services (SAS) to listen for inbound authentication requests. The default value is **31219**.

Installing Tivoli Workload Scheduler for z/OS connector

CSIV2 Server Authentication Listener

The port on which the Common Secure Interoperability Version 2 (CSIV2) service listens for inbound server authentication requests. The default value is **31220**.

CSIV2 Client Authentication Listener

The port on which the Common Secure Interoperability Version 2 (CSIV2) service listens for inbound client authentication requests. The default value is **31221**.

ORB Listener

The port used for RMI over IIOP communication. The default value is **31222**.

Administration HTTP transport

The administrative console port. The default value is **31223**.

Administration HTTPS transport

The administrative console secure port. The default value is **31224**.

- c. Connection to a Tivoli Workload Scheduler for z/OS host details. They are:

Tivoli Workload Scheduler for z/OS engine name

The name of the Tivoli Workload Scheduler for z/OS controller which you are connecting.

Tivoli Workload Scheduler for z/OS remote host

The IP address or host name of the remote z/OS system where the scheduler is installed.

Tivoli Workload Scheduler for z/OS remote TCP/IP port

The TCP/IP port number of the remote z/OS system.

You must click **Configure a connection to a Tivoli Workload Scheduler for z/OS host** to enable these fields for entry. If you do not select it at this time, you can specify the connection data later using the **createZosEngine** WebSphere Application Server tool (wastool).

- d. The installation directory on the computer. Unless you specify a different directory, the installation paths used by default are:

Table 37. Default installation paths for Tivoli Workload Scheduler for z/OS connector.

Operating system	Default installation path
Windows	c:\Program Files\IBM\TWA\
Linux	/opt/ibm/TWA/
UNIX	/opt/IBM/TWA/

Note: You cannot use national characters in the installation path.

When the installation process completes, expect the following results:

- If missing, the *TWS_user*, owner of the Tivoli Workload Scheduler for z/OS connector instance, is created.
- If missing, an embedded version of the IBM WebSphere Application Server is installed. Directories *eWAS* and *wastools* are created in the installation path.
- An instance of the Tivoli Workload Scheduler for z/OS connector, version 8.6 is installed in directory *TWSZOS* in the installation path.
- The Tivoli Workload Scheduler for z/OS connector registry file (*TWSZOSConnRegistry.dat*) is created.

Installing Tivoli Workload Scheduler for z/OS connector

- The Tivoli Workload Automation instance property file (`twainstanceinstance_number.TWA.properties`) for Tivoli Workload Scheduler for z/OS connector and WebSphere Application Server (if needed) is created. The file `twainstanceinstance_number.TWA.properties.ext` is also created.

If for some reason, the installation process fails, a diagnostic window is displayed. The diagnostic window provides also the option to continue with the following installation steps.

Installing using the wizard in silent mode

There are two ways to customize a response file to satisfy your installation requirements:

- Edit an existing response file template provided on the installation DVDs. See “Installing using response file templates.”
- Automatically create a customized response file by running the installation wizard. See “Installing with an automatically generated response file” on page 203.

Installing using response file templates

Locate the response files directory on the delivery media of the product, choose the appropriate file, edit it, and save it with a different name. Instructions for customizing the files are included in the files as commented text. See also Appendix F, “z/OS connector response file properties,” on page 341 for a description.

The response file templates provided to install the Tivoli Workload Scheduler for z/OS connector are the following:

TWS86_ZCONN_FRESH_newTWA_WIN.txt

Install on a new instance of Tivoli Workload Automation, installing the Tivoli Workload Automation infrastructure, on Windows operating systems.

TWS86_ZCONN_FRESH_existTWA_WIN.txt

Install on an existing instance of Tivoli Workload Automation, using the Tivoli Workload Automation infrastructure already installed, on Windows operating systems.

TWS86_ZCONN_FRESH_newTWA_UNIX.txt

Install on a new instance of Tivoli Workload Automation, installing the Tivoli Workload Automation infrastructure, on UNIX and Linux operating systems.

TWS86_ZCONN_FRESH_existTWA_UNIX.txt

Install on an existing instance of Tivoli Workload Automation, using the Tivoli Workload Automation infrastructure already installed, on UNIX and Linux operating systems.

Note: Before running a silent installation on UNIX zSeries systems, you must save the response file in UTF 8 format.

To run a silent installation using a response file template, follow these steps:

1. Copy the relevant response file to a local directory and edit it to meet the needs of your environment.

Note: Be sure to review the license agreement information included on the installation media. To accept the terms of the license agreement, set the

licenseAccepted parameter to **true** in the response file you are using.
This value is required to complete the silent installation successfully.

2. Save the file with your changes.
3. Enter the following command:

Windows	SETUP.exe -options <i>local_dir</i> \response_file.txt -silent where <i>response_file.txt</i> is the name of the response file you created. The SETUP.exe file is located in the TWSZOS\WINDOWS directory on 32-bit platforms and in the TWSZOS\WINDOWS_X86_64 directory on 64-bit platforms.
UNIX and Linux	./SETUP.bin -options <i>local_dir</i> /response_file.txt -silent where <i>response_file.txt</i> is the name of the response file you created. The SETUP.bin file is located in the TWSZOS/PLATFORM directory.

4. Review the summary.log file to check that the installation was successful.

Installing with an automatically generated response file

You can first run an interactive installation guided by the wizard and create a response file based on the parameters you enter at this time. You use this response file to run subsequent installations with the same parameters. Creating an automatically generated response file is recommended because all input is automatically validated by the program.

To run a silent installation using an automatically generated response file, perform the following steps:

1. Perform the initial installation using the following command:

Windows	SETUP.exe -options-record <i>local_dir</i> \response_file.txt where <i>response_file.txt</i> is the name of the response file you created. The SETUP.exe file is located in the TWSZOS\WINDOWS directory on 32-bit platforms and in the TWSZOS\WINDOWS_X86_64 directory on 64-bit platforms.
UNIX and Linux	./SETUP.bin -options-record <i>local_dir</i> /response_file.txt where <i>response_file.txt</i> is the name of the response file you created. The SETUP.bin file is located in the TWSZOS/PLATFORM directory.

The installation wizard starts. Follow the prompts and complete the installation. A response file is created in the directory that you specified in the setup command.

Because the response file contains the values that you entered in the installation wizard, you will need to edit the file for the subsequent installations if your configuration requirements change.

Note: The response file that is created contains unencrypted password information.

2. For all subsequent installations, enter the following command:

Windows	SETUP.exe -options <i>local_dir</i> \response_file.txt -silent
UNIX and Linux	./SETUP.bin -options <i>local_dir</i> /response_file.txt -silent

3. After each silent installation, review the summary.log file to check that the installation was successful.

Installing from the launchpad

The launchpad requires some additional installation prerequisites. For more information, see the Tivoli Workload Scheduler System Requirements Document at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27019747>.

Note: When running the launchpad on UNIX and Linux operating systems, make sure that you export the browser location to the BROWSER environment variable.

To install from the launchpad, perform the following steps:

1. Start the launchpad. If you have autorun enabled, the launchpad starts automatically. Otherwise, start it manually from the DVD with the following command:

On Windows

`launchpad.exe.`

On UNIX and Linux

`launchpad.sh.`

The launchpad is displayed.

2. In the left frame of the launchpad, click **Install a specific product** and **Install TWS for z/OS Connector**.

The Install IBM Tivoli Workload Scheduler for z/OS Connector, V8.6 panel is displayed on the right.

3. Click **Install the TWS for z/OS Connector**.

The interactive installation wizard starts. Follow the steps described in “Installing using the wizard in interactive mode” on page 200.

Note: If you try to install on an unsupported platform, the installation menu and screens are not displayed.

Upgrading

You can upgrade from Tivoli Workload Scheduler for z/OS connector version 8.3, version 8.5, and 8.5.1, using one of the following:

- The installation wizard in either interactive or silent (with response files) mode
- The Tivoli Workload Automation launchpad

Upgrading with the wizard in interactive mode

Run the setup for the operating system on which you are upgrading:

On Windows:

`TWSZOS\SETUP.cmd` or `TWSZOS\SETUP.exe`

On UNIX and Linux:

`TWSZOS/SETUP.sh` or `TWSZOS/SETUP.exe`

Note: During the upgrade, you are prompted for the WebSphere Application Server administration user name and password.

Upgrading from version 8.3

Follow the installation wizard panels to complete the installation. The following list describes the fields that you might need to complete during the installation.

Upgrading Tivoli Workload Scheduler for z/OS connector

Backup profile destination directory

This information is needed to perform a backup of your WebSphere Application Server (WAS) profile. Your current settings are transferred to WebSphere Application Server automatically.

User name

Enter the user name (TWSUser) and the password for which you want to install the Tivoli Workload Scheduler for z/OS connector. This user works also as Tivoli Workload Automation user and it is authorized to perform the operations on the Tivoli Workload Scheduler for z/OS connector server. This user is the WebSphere Application Server administrator of the Tivoli Workload Automation instance.

Password

Enter the password of the Tivoli Workload Scheduler user for which you are upgrading the connector instance. If you made any changes to the WebSphere Application Server authentication user name or password from your previous installation, you must supply the new values here.

SAS Server Authentication Listener

The port used by the Secure Association Services (SAS) to listen for inbound authentication requests. The default value is **31219**.

CSIV2 Server Authentication Listener

The port on which the Common Secure Interoperability Version 2 (CSIV2) service listens for inbound server authentication requests. The default value is **31220**.

CSIV2 Client Authentication Listener

The port on which the Common Secure Interoperability Version 2 (CSIV2) service listens for inbound client authentication requests. The default value is **31221**.

ORB Listener

The port used for RMI over IIOP communication. The default value is **31222**.

Administration HTTP transport

The administrative console port. The default value is **31223**.

Administration HTTPS transport

The administrative console secure port. The default value is **31224**.

Upgrading from version 8.5 or 8.5.1

Follow the installation wizard panels to complete the installation. The following list describes the fields that you might need to complete during the installation.

Backup profile destination directory

This information is needed to perform a backup of your WebSphere Application Server (WAS) profile. Your current settings are transferred to WebSphere Application Server automatically.

User name

Enter the user name (TWSUser) and the password for which you want to install the Tivoli Workload Scheduler for z/OS connector version 8.6. This user will work also as Tivoli Workload Automation user and then it will be authorized to perform the operations on the Tivoli Workload Scheduler for z/OS connector server. This user is the WebSphere Application Server administrator of the Tivoli Workload Automation instance.

Password

Enter the password of the Tivoli Workload Scheduler user for which you

Upgrading Tivoli Workload Scheduler for z/OS connector

are upgrading the connector instance. If you made any changes to the WebSphere Application Server authentication user name or password from your previous installation, you must supply the new values here.

Note: When you upgrade from version 8.5 or 8.5.1, you must upgrade the entire instance of Tivoli Workload Automation. To upgrade an instance of Tivoli Workload Automation, you must upgrade all components that are part of the instance. For example, if your instance includes one z/OS connector and also the Dynamic Workload Console, you must upgrade both of these components.

The order in which you upgrade components in the shared instance is very important. You must upgrade the components in the following order:

1. Tivoli Workload Scheduler V8.6
2. Dynamic Workload Console V8.6
3. Tivoli Workload Scheduler for z/OS connector V8.6
4. Any other components

Upgrading in silent mode

To upgrade an existing Tivoli Workload Scheduler for z/OS connector version 8.3, 8.5, or 8.5.1 installation using the silent method, follow the procedure described in “Installing using response file templates” on page 202 with the appropriate response file:

- TWS86_ZCONN_UPGRADE_83Plus_WIN.txt
- TWS86_ZCONN_UPGRADE_83Plus_UNIX.txt

These response files upgrade Tivoli Workload Scheduler for z/OS connector to version 8.6 on an existing instance of Tivoli Workload Automation, using the Tivoli Workload Automation infrastructure already installed.

Upgrading from the launchpad

This section describes how to upgrade using the launchpad.

Note: If you try to upgrade on an unsupported platform, the upgrade menu and screens are not displayed.

To upgrade, perform the following steps:

1. From the DVD, run the Tivoli Workload Automation launchpad as follows:

On Windows

From the root directory of the DVD, run `launchpad.exe`.

On UNIX

From the root directory of the DVD, run `launchpad.sh`.

The launchpad is displayed.

2. In the left frame of the launchpad, click **Upgrade a specific product** and **Upgrade TWS for z/OS Connector**.

The Upgrade IBM Tivoli Workload Scheduler for z/OS Connector, V8.3, V8.5, and V8.5.1 panel is displayed on the right.

3. Click **Upgrade the TWS for z/OS Connector component**.

The interactive installation wizard starts. Follow the steps described in “Upgrading with the wizard in interactive mode” on page 204.

Uninstalling

To uninstall the Tivoli Workload Scheduler for z/OS connector, you can run the `uninstall` program either in interactive or in silent mode.

Uninstalling using the wizard

Perform the following steps:

1. Navigate to the Tivoli Workload Scheduler for z/OS connector installation directory and run the `uninstall` program.
 - On Windows, go to `drive:\Program Files\IBM\TWA\TWSZOS_uninstall`, and run **`uninstall.exe`**.
 - On UNIX and Linux, go to the `opt/ibm/TWA/TWSZOS/_uninstall` path and run **`./uninstall.bin`**.

The uninstallation wizard opens.

2. On UNIX and Linux only, write the WebSphere Application Server authentication user name and password and click **Next** to proceed with the uninstallation process.

The embedded WebSphere Application Server and Tivoli Workload Scheduler for z/OS connector are uninstalled and removed from the Tivoli Workload Automation registry.

Uninstalling in silent mode

The process requires no response file on Windows platforms. On UNIX and Linux you need to customize a response file template with the user credentials of the administrator (*TWSuser*) of the embedded WebSphere Application Server that the Tivoli Workload Scheduler for z/OS connector instance is using.

Perform the following steps:

- On Windows, go to `drive:\Program Files\IBM\TWA\TWSZOS_uninstall`, and run:
`uninstall.exe -silent`
- On UNIX and Linux:
 1. Copy the `TWS86_ZCONN_UNINSTALL_UNIX.txt` response file template located in the `responsefiles` directory of the installation DVD to a local directory.
 2. Edit the response file template with the user name and password of the WebSphere Application Server administrator and save it with a different name.
 3. Go to `opt/ibm/TWA/TWSZOS/_uninstall` and run:
`./uninstall.bin -options local_dir/TWS86_ZCONN_UNINSTALL_UNIX.txt -silent`

If you want to reinstall after running a silent uninstallation, you must first close and reopen the shell to correctly reset the environment variables.

Chapter 8. Installing and uninstalling on WebSphere Application Server for z/OS

This chapter describes how to install, apply maintenance, and uninstall the Tivoli Workload Scheduler for z/OS connector on IBM WebSphere Application Server for z/OS. Install the z/OS connector to maintain your business in a z/OS environment and simultaneously manage your workload using modern applications like EJB and Web Services as described in “Business Scenario”, or to work with the Dynamic Workload Console as described in “Graphical user interfaces” on page 7.

Business Scenario

To save money, skill, and seize new opportunities, a company wants to maintain its business in a z/OS environment and simultaneously manage its workload using modern applications like Web Services. It wants to avoid using scripting languages to schedule jobs on non-IBM software applications, because these scripts can be hard to debug, maintain, and port across different operating systems.

The company can reach this objective by:

- Installing the Tivoli Workload Scheduler for z/OS connector on WebSphere Application Server for z/OS to have all the required functions available in the z/OS environment.
- Using the Java API to define and submit a jobs.

Authorization roles required for installing and uninstalling

To install or uninstall Tivoli Workload Scheduler for z/OS connector, you must have full access (read, write, and execute) to:

- The directory where you installed the WebSphere Application Server for z/OS. The default value is *WebSphere Application Server installation_directory/AppServer/profiles/default/bin* directory. This is the directory where the **wsadmin.sh** script is located.
- The directories where you extract the **JWSZ604** FMID.

Installing on WebSphere Application Server for z/OS

You can install the Tivoli Workload Scheduler for z/OS connector on WebSphere Application Server for z/OS either by using the Integrated Solutions Console or the **zConnInstall.sh** scripts located in the `opt\package` directory.

Integrated Solutions Console

It is a graphical interface to manage your applications and perform system administration tasks for your WebSphere Application Server environment. The administrative console runs in your web browser. You can use it to manage WebSphere Application Server applications. Refer to “Installing using the Integrated Solutions Console” on page 210.

The **zConnInstall.sh** script

It uses the WebSphere Application Server **wsadmin** tool to perform the installation. Refer to “Installing using the **zConnInstall.sh** script” on page 211.

Installing the z/OS connector on WebSphere Application Server for z/OS

The Tivoli Workload Scheduler for z/OS connector is installed using the same user with which you access the WebSphere Application Server for z/OS.

Installing using the Integrated Solutions Console

To install Tivoli Workload Scheduler for z/OS connector with the Integrated Solutions Console, perform the following steps:

1. Ensure that you installed the **JWSZ604** FMID.
2. Install the resource adapter file.
 - a. Select **Resource->Resource Adapters->Resource adapters**.
 - b. On the Resource adapters dialog, select **Install RAR** both to install a RAR file and to configure an associated resource adapter.
 - c. On the Install RAR dialog, in the **Node** field, set the scope to a WebSphere Application Server node on which the resource adapter must be installed.
 - d. Browse to find the appropriate RAR file. If your RAR file is located on your:
 - Local workstation, select **Local file system**, and browse to find the file.
 - Server, select **Remote file system**, and specify the fully qualified path to the file.

Repeat Step 2 for each node that hosts application servers in the cluster.

3. Create the connection factory for a Tivoli Workload Scheduler engine.
 - a. Select **Resource->Resource Adapters->J2C connection factories**.
 - b. On the J2C connection factories page, in the **Scope** pull-down menu, set the scope to the WebSphere Application Server node where you installed the resource adapter.
 - c. Select **New**.
 - d. On the configuration page, specify the following settings:

Table 38. Configuration page settings

Setting	Value
Provider	ZOSConnectorAdapter. It is the name of the resource adapter that you specified when you installed the resource adapter.
Name	The name of the engine that you use to connect to the Dynamic Workload Console.

- e. Click **Apply**.
 - f. On the Custom properties page, in the Additional Properties list, click **Custom properties**.
 - g. From the table displayed, select **HostName**.
 - h. On the General Properties page, in the **Value** field, enter the host name of the Tivoli Workload Scheduler for z/OS server and then click **OK**.
 - i. On the **Custom properties** table, select **PortNumber**.
 - j. On the General Properties page, in the **Value** field enter the port number of the Tivoli Workload Scheduler for z/OS server and then click **OK**.
 - k. In the **JNDI name** field, replace file path `eis/zos_engine_name` with `eis/tws/zconn/zos_engine_name` and then click **OK**.
 - l. In the Messages box, click **Save** to save your changes directly to the master configuration.
4. Install the connector enterprise application.

Installing the z/OS connector on WebSphere Application Server for z/OS

- a. Select **Applications->Applications Types->WebSphere enterprise applications**.
- b. On the Enterprise Applications page, click **Install**.
- c. Specify the path to the ZConnector.ear file.
Browse to find the appropriate EAR file. If your EAR file is located on your:
 - Local workstation, select **Local file system**, and browse to find the file.
 - Server, select **Remote file system**, and specify the fully qualified path to the file.
- d. Deploy the application on the cluster with the name **ZConnector**.
5. Copy the TWSZOSConnConfig.properties file in the `<installation_directory>/AppServer/profiles/default/properties` path.
6. Identify the library files and their classpath by performing the following steps:
 - a. In the console navigation tree, click **Environment->Shared Libraries**. The Shared libraries page is displayed.
 - b. Change the scope of the collection table to see the shared libraries the are in a cell, in a node, or in a server.
 - c. Select the cell, the node or the server where you find the shared library and click **New**.
 - d. On the Configuration page for a shared library, specify the name and the classpath for the library as follows:
Name=applicationJobPlugins
ClassPath=/zConn_instldir/package/apps/applicationJobPlugins
 - e. Click **OK**.
7. Associate the applicationJobPlugins shared library to the ZConnector application by performing the following steps:
 - a. Click **Applications->Applications Types->WebSphere enterprise applications** in the console navigation tree. The General Properties page is displayed.
 - b. Click on the installed application ZConnector.
 - c. Click **Shared libraries references**. The Shared libraries references page is displayed.
 - d. From the Shared libraries references list select ZConnector. The Available list is displayed.
 - e. In the Available list, select the applicationJobPlugins library.
 - f. Click **OK**.

The installation does not show the path where you installed the z/OS connector. The path is specified in the **com.ibm.ws.scripting.traceFile** property in the wsadmin.properties file. The connector enterprise application is installed in the *WebSphere Application Server_installation_directory/AppServer/profiles/default/InstalledApps* directory.

Installing using the zConnInstall.sh script

To install Tivoli Workload Scheduler for z/OS connector with the **zConnInstall.sh** script, perform the following steps:

1. Ensure that you installed the **JWSZ604** FMID.
2. Run the **zConnInstall.sh** script. The script requires the path where you installed the WebSphere Application Server for z/OS and the properties described in Table 39 on page 212 to run.

Installing the z/OS connector on WebSphere Application Server for z/OS

Table 39. *zConnInstall.properties* properties and corresponding values

Property	Value
CellName	The cell name of the profile. This property is required.
nodeName	The node name of the profile. This property is required.
serverName	The server name of the profile. This property is required.
zosEngineName	The engine name for z/OS connection. This property is required.
zosHostName	The host name of the Tivoli Workload Scheduler z/OS server. This property is required.
zosPortNumber	The port number of the Tivoli Workload Scheduler z/OS server. This property is required.
connectionTimeoutCleanup	The connection timeout cleanup for z/OS connection. This property is optional. The default value is 60 .
scaffoldSwitch	The scaffold switch for z/OS connection. This property is optional. The default value is false .
maxConnections	The maximum number of managed connections that can be created for the z/OS connection. This property is optional. The default value is 10 .
connectionTimeout	The connection timeout for z/OS connection. Interval, in seconds after which the z/OS connection request times out and a <code>connectionWaitTimeoutException</code> is thrown. This property is optional. The default value is 1 .
unusedTimeout	Interval in seconds after which an unused connection is discarded by the connection pool maintenance thread. This property is optional. The default value is 60 .
reapTime	The reap time for z/OS connection. This property is optional. The default value is 300 .
startApplication	Start the WebSphere Enterprise Applications. This property is optional. Possible values are true and false. The default value is false .

You can run the script in one of the following ways:

- Specifying the path where you installed the WebSphere Application Server for z/OS and the properties directly in the command line. In this case the path where you installed the WebSphere Application Server for z/OS must be specified as the first property. The following example shows how to run the command specifying all the parameters:

```
./zConnInstall.sh -wasPath /u/wasv7config/bbobase/bbonode/AppServer
-cellName bbobase -nodeName bbonode -serverName server1 -zosEngineName CWSV64
-zosHostName 127.0.0.1 -zosPortNumber 505 -connectionTimeoutCleanup 60
-scaffoldSwitch false -maxConnections 10 -connectionTimeout 1
-unusedTimeout 60 -reapTime 300 -startApplication false
```
 - Specifying in the `zConnInstall.properties` file all the values of the properties and running the command specifying only the path where you installed the WebSphere Application Server for z/OS as follows:

```
./zConnInstall.sh -wasPath app_server_root
```
3. Verify that the installation completed successfully by reading the messages displayed in the screen. Detailed information is logged in the file specified in the **com.ibm.ws.scripting.traceFile** property in the `wsadmin.properties` file.

The installation does not show the path where you installed the z/OS connector. The path is specified in the **com.ibm.ws.scripting.traceFile** property in the

Installing the z/OS connector on WebSphere Application Server for z/OS

wsadmin.properties file. By default, the connector enterprise application is installed in the *WebSphere Application Server_installation_directory/AppServer/profiles/default/InstalledApps* directory.

Installation and uninstallation log files

You can check the following log files for information about the installation. Details of the installation process are displayed on the screen and kept in log files in the following directories:

If you installed using the Integrated Solutions Console

The file path is specified by the **com.ibm.ws.scripting.traceFile** property in the wsadmin.properties file. The default value is **app_server_root/profiles/profiles_name/logs/server_name/was.traceout**.

where:

profiles_name

Is the name of the profile that you used when you installed the WebSphere Application Server. The default name is **default**.

server_name

Is the name of the server that you used when you installed the WebSphere Application Server. The default name is **default**.

If you installed using the zConnInstall.sh script

The file path is specified by the **com.ibm.ws.scripting.traceFile** property in the wsadmin.properties file. The default value is **app_server_root/profiles/profiles_name/logs/was.traceout**, where *profiles_name* is the name of the profile that you used when you installed the WebSphere Application Server. The default name is **default**.

Enabling communications with Dynamic Workload Console

After you installed the Dynamic Workload Console and the z/OS connector you must enable the communication between them. The Dynamic Workload Console and the z/OS connector use RMI/IIOP over SSL to communicate. The SSL security paradigm implemented in the WebSphere Application Server requires two stores to be present on the clients and the server:

A keystore

It contains the private key.

A trust store

It contains the certificates of the trusted counterparts.

Figure 26 on page 214 shows the keys that must be extracted and distributed to enable SSL between the z/OS connector and the Dynamic Workload Console. Each arrow in the diagram includes the following activities performed using an appropriate key management tool on each keystore:

- Create a self-signed certificate or import a third party certificate.
- Extract a new key.
- Open the appropriate trust store.
- Use the new key to add a signed certificate to the trust store.

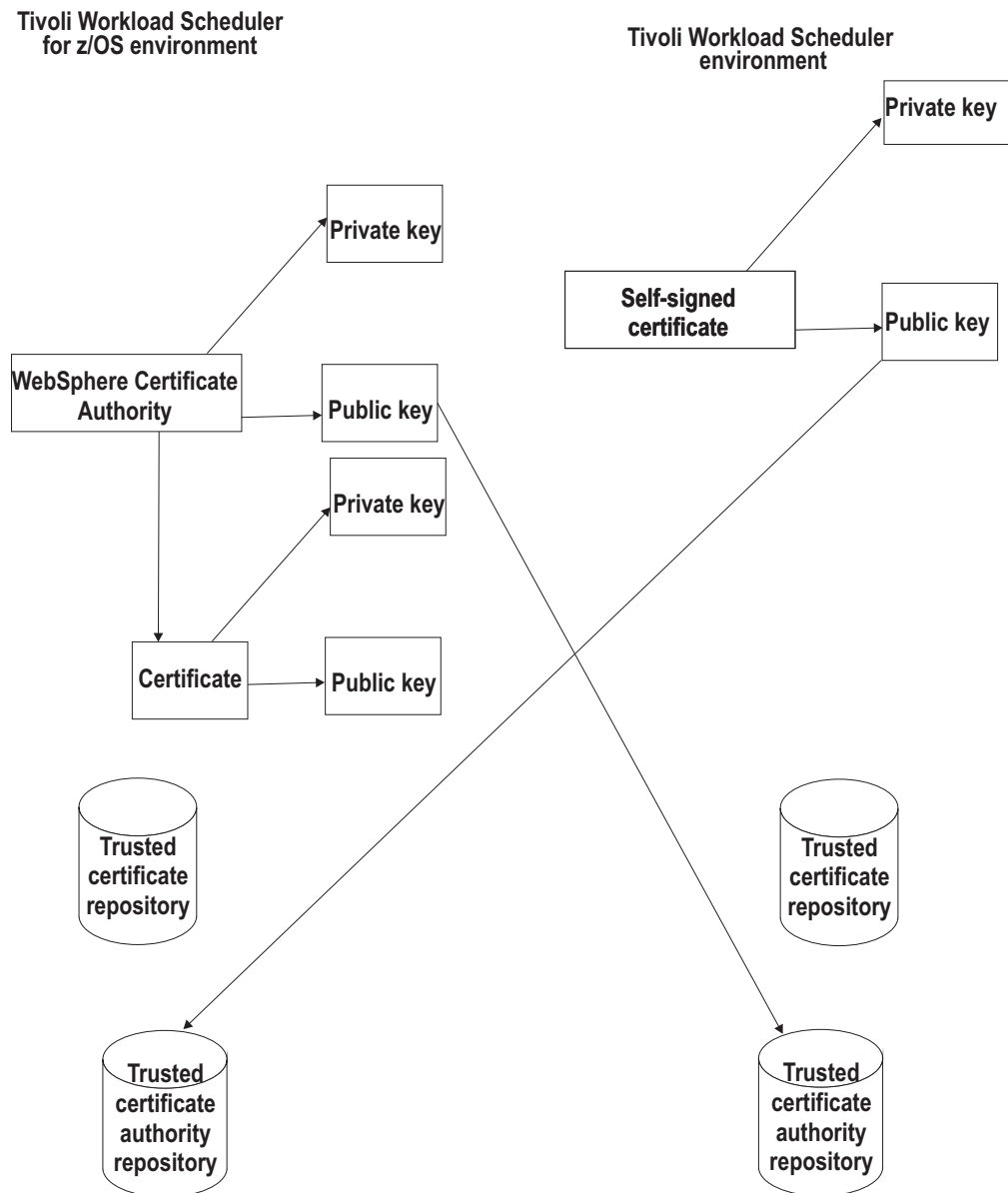


Figure 26. Shows the keys to enable SSL between the z/OS connector and the Dynamic Workload Console

To define SSL basic authentication security, you must first request a signed certificate for your server and a certificate authority (CA) certificate from the certificate authority that signed your server certificate. After you have received both these certificates, you must:

- From the z/OS environment, extract the public key CA certificate and store it in the trusted Certificate Authority repository of the Dynamic Workload Console.
- From the Dynamic Workload Console, extract the public key of the self-signed certificate and store it in the trusted certificate repository of WebSphere Application Server for z/OS.

To perform these operations, complete the following steps:

1. Export the WebSphere Application Server for z/OS certificate to a data set, as follows:

Enabling communications with Dynamic Workload Console

- a. Connect to RACF and select option DIGITAL CERTIFICATES, KEY RINGS, AND TOKENS.
 - b. Select option Digital certificates functions.
 - c. Select option Write a certificate to a data set.
 - d. Export the WebSphere Application Server certificate authority certificate to a data set and transfer the file to the Dynamic Workload Console using the FTP protocol in binary or ASCII mode.
2. Import the file into the trusted certificate authority repository of the Dynamic Workload Console using the iKeyman utility. The iKeyman utility is located in *installation_directory*/TDWC/_jvm/jre/bin.
3. From the Dynamic Workload Console, export the self-signed certificate to a file using the iKeyman utility. For more information, see the section about interface communication in Administration Guide.
4. Transfer the file to the z/OS environment and add it to the RACF database as follows:
 - a. In RACF, select option DIGITAL CERTIFICATES, KEY RINGS, AND TOKENS.
 - b. Select option Digital certificates functions.
 - c. Select option Add, alter, delete or list certificates.
 - d. Select option Add a digital certificate to the RACF database. Set the status to **Trust (T)**.
5. Associate the certificate to the trusted certificate authority repository of WebSphere Application Server for z/OS, as follows:
 - a. In RACF, select option Key Ring functions.
 - b. Select option Connect a digital certificate to a key ring. In field Ring Name, type the name of the WebSphere Application Server controller key ring.
6. Define an EJBROLE profile and then permit a System Authorization Facility (SAF) user to the profile as follows:
 - a. On the WebSphere Application Server, the deployment descriptor of the zConnector defined under the Enterprise Applications, displays the default role, TWSAdmin, that needs to be defined in the RACF class EJBROLE as follows:

```
rdefine EJBROLE <SAF_prefix>.TWSAdmin owner(SYS1)
audit(failures(READ)) uacc(NONE)
```
 - b. Grant READ access to a specific user by issuing the following RACF command:

```
permit <SAF_prefix>.TWSAdmin class(EJBROLE) id(userid) access(READ)
```
7. Restart WebSphere Application Server to make changes effective.

Secure communications is now enabled between the Dynamic Workload Console and the z/OS connector.

Applying maintenance

This section describes how to apply the program temporary fix (PTF) level of the Tivoli Workload Scheduler for z/OS connector. You can apply PTFs using either the Integrated Solutions Console, or the **zConnUpdate.sh** scripts located in the *install_dir/zConnUpdate.sh* directory. This section describes how to apply the PTFs using the **zConnUpdate.sh** scripts.

1. Apply the PTFs as described in the *Program Directory for Tivoli Workload Scheduler for z/OS*.
2. Ensure that the WebSphere Application Server for z/OS Integrated Solutions Console is not running.
3. Ensure that you installed the **JWSZ604** FMID.
4. Move to the directory where the script is located. The default value is *installation directory/opt/package*.
5. Ensure that the **com.ibm.ws.scripting.connectionType** property in the *wsadmin.properties* file is set to the value **SOAP** or **RMI**. The default value is **SOAP**. The script requires the WebSphere Application Server user and password. If you do not want to specify them at run time, set them in the *soap.client.props* file for SOAP connection type and in the *sas.client.props* file for the RMI connection type. The *soap.client.props* and *sas.client.props* files are located in the *properties* directory of your WebSphere Application Server profile. The *WebSphere Application Server_installation_directory/AppServer/profiles/WebSphere Application Serverprofile_name/properties* is the default directory.
6. Run the **zConnUpdate.sh** script. The script requires the path where you installed the WebSphere Application Server for z/OS and the properties described in Table 39 on page 212 to run.

Table 40. *zConnUpdate.properties* properties and corresponding values

Property	Value
CellName	The cell name of the profile. This property is required.
nodeName	The node name of the profile. This property is required.
serverName	The server name of the profile. This property is required.

You can run the script in one of the following ways:

- Specifying the path where you installed the WebSphere Application Server for z/OS and the properties directly in the command line. In this case the path where you installed the WebSphere Application Server for z/OS must be specified as the first property. The following example shows how to run the command specifying all the parameters:

```
./zConnUpdate.sh -wasPath /u/wasv7config/bbobase/bbonode/AppServer
-cellName bbobase -nodeName bbonode -serverName server1 -zosEngineName CWSV64
```
 - Specifying in the *zConnUpdate.properties* file all the values of the properties and running the command specifying only the path where you installed the WebSphere Application Server for z/OS as follows:

```
./zConnUpdate.sh -wasPath app_server_root
```
7. Verify that the PTF was installed successfully by reading the messages displayed on the screen. Detailed information is logged in the file specified in the **com.ibm.ws.scripting.traceFile** property in the *wsadmin.properties* file.

Uninstalling

To uninstall the Tivoli Workload Scheduler for z/OS connector, you can use either the Integrated Solutions Console or the `zConnUninstall.sh` script.

Uninstalling using the Integrated Solutions Console

To uninstall the Tivoli Workload Scheduler for z/OS connector with the Integrated Solutions Console, perform the following steps:

1. Uninstall the resource adapter file.
 - a. Select **Resource->Resource Adapters->Resource adapters**.
 - b. On the Resource adapters dialog, select the resource adapter that you want to delete and click **Delete**. A message is displayed. Click **Save**.

Repeat this step for each node that hosts application servers in the cluster.
2. If still present, uninstall the connection factory for a Tivoli Workload Scheduler engine.
 - a. Select **Resource->Resource Adapters->J2C connection factories**.
 - b. On the J2C connection factories page, in the **Scope** pull-down menu, set the scope to the WebSphere Application Server node where you installed the resource adapter.
 - c. Select the J2C connection factory that you want to delete and click **Delete**. A message is displayed. Click **Save**.
3. Delete the Connector enterprise application.
 - a. Select **Applications->Applications Types->WebSphere enterprise applications**.
 - b. On the Enterprise Applications page, select the ZConnector resource that you want to uninstall and click **Uninstall**. This step deletes the application from the product configuration repository and the application binaries from the file system of all nodes where the application modules are installed. A message is displayed. Click **Save**.

Uninstalling using the `zConnUninstall.sh` script

To uninstall Tivoli Workload Scheduler for z/OS connector with the `zConnUninstall.sh` script, perform the following steps:

1. Ensure that you installed the **JWSZ604** FMID.
2. Ensure that the `com.ibm.ws.scripting.connectionType` property in the `wsadmin.properties` file is set to the **SOAP** or **RMI** value. The default value is **SOAP**. In this case at run time, the script requires the WebSphere Application Server user and password. If you do not want to specify them at run time, set them in the `soap.client.props` file for SOAP connection type and in the `sas.client.props` file for the RMI connection type. The `soap.client.props` and `sas.client.props` files are located in the properties directory of your WebSphere Application Server profile. The *WebSphere Application Server_installation_directory/AppServer/profiles/WebSphere Application Serverprofile_name/properties* is the default directory.
3. Run the `zConnUninstall.sh` script. The script requires the path where you installed the WebSphere Application Server for z/OS and the properties described in Table 39 on page 212 to run.

Uninstalling Tivoli Workload Scheduler for z/OS connector

Table 41. *zConnUninstall.properties* properties and corresponding values

Property	Value
CellName	The cell name of the profile. This property is required.
nodeName	The profile node name. This property is required.
serverName	The profile server name. This property is required.
zosEngineName	The engine name for z/OS connection. This property is required.

You can run the script in one of the following ways:

- Specifying the path where you installed the WebSphere Application Server for z/OS and the properties directly in the command line.

In this case the path where you installed the WebSphere Application Server for z/OS must be specified as the first property. The following example shows how to run the command specifying all the parameters:

```
./zConnUninstall.sh -wasPath /u/wasv7config/bbobase/bbonode/AppServer  
-cellName bbobase -nodeName bbonode -serverName server1 -zosEngineName CWSV64
```

- Specifying in the `zConnUninstall.properties` file all the values of the properties and running the command specifying only the path where you installed the WebSphere Application Server for z/OS as follows:

```
./zConnUninstall.sh -wasPath app_server_root
```

4. Verify that the uninstallation completed successfully by reading the messages displayed on the screen. Detailed information is logged in the file specified in the **com.ibm.ws.scripting.traceFile** property in the `wsadmin.properties` file.

Chapter 9. Troubleshooting and maintaining the installation

This chapter describes how to troubleshoot and maintain the installation of the Tivoli Workload Scheduler for z/OS connector.

Troubleshooting the installation

This section describes how to troubleshoot the installation of the Tivoli Workload Scheduler for z/OS connector.

z/OS connector installation step hangs while installing on a TWA instance with an existing embedded WebSphere Application Server

You might run into the following problem when you install Tivoli Workload Scheduler for z/OS connector under the following conditions:

- You are adding the z/OS connector to an existing Tivoli Workload Automation instance over the embedded WebSphere Application Server
- You are installing on a particularly slow workstation

The step that installs Tivoli Workload Scheduler for z/OS connector hangs indefinitely because the WebSphere Application Server start timeout is exceedingly short. When this happens, do the following:

1. Kill the WebSphere Application Server process if it is still hanging.
2. Remove, if present, the installed ZOSResourceAdapter.rar client:
 - On Windows:
`eWAS_installation_directory\bin\clientRAR.bat delete ZOSResourceAdapter.rar`
 - On UNIX and Linux:
`eWAS_installation_directory/bin/clientRAR.sh delete ZOSResourceAdapter.rar`If the command returns an error related to the inexistence of ZOSResourceAdapter.rar, ignore the error and continue.
3. Rerun the failed installation step.

On Windows the z/OS connector installation step fails because the user account does not belong to the Administrators group

On Windows operating systems, the installation process automatically creates the user (habitually referred to as *TWS_user*) with the appropriate rights, if the user does not already exist. However, if the user is already defined and has all the expected grants correctly set except for being in the **Administrators** group, this leads to the failure of the z/OS connector installation step.

Upon such failure, do check that the *TWS_user* you used is a member of the Windows Administrators group. If it is not, then:

1. Add the user to the Administrators group.
2. Rerun the installation process.

Dynamic Workload Console creates wrong connection upon installation

You might have the following problem when you install Tivoli Workload Scheduler for z/OS connector and Dynamic Workload Console version 8.5.0 (General Availability version) in the same Tivoli Workload Automation instance.

The first time you login to Dynamic Workload Console after installation, it automatically searches the TWA instance for the installed connector. Because of a flaw in the discovery process, however, the installed z/OS connector is erroneously identified as a distributed connector. This makes the connection definition in Dynamic Workload Console to the z/OS host invalid.

As a remedy, take one of the following actions:

- In Dynamic Workload Console, after you login and find that the error has taken place, delete the flawed connection definition and create a new one with the right parameters.
- Avoid the problem by installing Fix pack 1 on Dynamic Workload Console version 8.5.0 before you log in.

Installation fails because Windows Workstation Service is not started

Make sure that Windows Workstation Service is started before installing the z/OS connector and the Dynamic Workload Console on Microsoft Windows Server 2003 and 2008, both on 64-bit and non-64-bit platforms.

Failed installation of a dynamic domain manager in the same instance as the z/OS connector

If you try to install a dynamic domain manager in the same instance where you installed the Tivoli Workload Scheduler for z/OS connector, the installation wizard fails with the error messages similar to the examples below:

```
DISSE0197E Execution of user program userprogram failed.  
DISSE0198I User program exit code:16  
DISSE0123E Unable to execute or complete execution of program userprogram  
DISSE0005E Operation unsuccessful.
```

Cause and solution

It is not possible to install a dynamic domain manager in the same instance of Tivoli Workload Automation as the Tivoli Workload Scheduler for z/OS connector. The installation wizard stops and gives you the opportunity to diagnose the failure. In the output of the step list, double-click the step **Install** to see the log details and, in **Properties**, change the port number of the dynamic domain manager. Then, set the status of the step list to **Ready**, click **Run All**, and your installation continues.

Maintaining the installation of the z/OS connector

This section explains some maintenance procedures you should perform in specific situations.

Updating the SOAP properties after changing the WebSphere Application Server user or its password

If you change the user ID or the password of the WebSphere Application Server administrator, you must also update the SOAP client properties.

To update the properties, run the following command:

- On UNIX and Linux: **updateWas.sh** command from the *TWA_home/wastools* directory.
- On Windows: **updateWas.bat** command from the *TWA_home\wastools* directory.

Updating the SOAP properties usage

To update the SOAP properties, use the syntax in the following section.

updateWas.sh (.bat)

Format:

updateWas.sh (.bat) -user *new_WAS_admin_user* **-password** *pwd*

Parameters:

-user *new_WAS_admin_user* **-password** *pwd*

Supply the user and password of the new WebSphere Application Server administration user that you want to be configured as the credentials in the SOAP client properties.

After using this command you must restart the application server.

Part 4. Dynamic Workload Console

Chapter 10. Preparing	225
Overview of the Dynamic Workload Console.	225
Installation overview	225
Installation considerations	226
Selecting your installation method	226
Instances of Tivoli Workload Automation	227
Installation media	228
Installation log files	228
Interactive wizard installation and uninstallation log files	228
Installation log files for the embedded WebSphere Application Server.	229
Chapter 11. Installing	231
Installing the Dynamic Workload Console.	231
Using the launchpad	231
Using the installation wizard	231
Installing a new instance of the Tivoli Integrated Portal	232
Default installation	232
Advanced installation	232
Installing on an existing instance of the embedded WebSphere Application Server	234
Installing on your existing instance of Tivoli Integrated Portal	235
Performing a silent installation	235
Installing the Tivoli Integrated Portal on an external WebSphere Application Server from the images	237
Post-installation steps to connect to Tivoli Workload Scheduler Version 8.3 Fix Pack 3	237
Post-installation steps to configure the use of Lightweight Third-Party Authentication (LDAP).	238
Accessing the Dynamic Workload Console	238
Quick steps to define a Tivoli Workload Scheduler engine connection	240
Quick steps to define a Dynamic Workload Broker connection	241
Starting and stopping the Dynamic Workload Console	242
Chapter 12. Configuring	245
Chapter 13. Getting started	247
Tivoli Workload Scheduler portfolio	247
Dynamic workload broker portfolio	249
First actions	250
Chapter 14. Upgrading.	251
Updating authentication.	251
Upgrading the console installed on an embedded WebSphere Application Server.	252
Directory structure	252
Program directory.	252
Directory for SSL files	253
Performing the upgrade.	253
Chapter 15. Uninstalling	255
Uninstalling using the wizard	255
Uninstalling in silent mode.	255
Chapter 16. Troubleshooting the installation, upgrade, and uninstallation	257
Installation and uninstallation log and trace files	257
Recovering a failed InstallShield wizard installation	257
Recovering a failed upgrade	257
Uninstalling the Dynamic Workload Console and the Tivoli Integrated Portal manually	258
Troubleshooting scenarios	259
Problems with the launchpad	259
Warning messages displayed when using the launchpad on Linux	259
Undefined error when using launchpad on Windows operating system.	260
Problems with the interactive wizard	260
The Dynamic Workload Console installation hangs	260
Installation hangs during stopWas command	260
Tivoli Integrated Portal installation fails even if into the logs you find successfully installed	261
Installation from a remote shared folder fails on Windows operating system.	262
Installation fails on a Linux 390 system with a hostname which is not a Fully Qualified Domain Name	262
Java Virtual Machine (JVM) failure when installing the Dynamic Workload Console on a Red Hat Enterprise Linux (RHEL) Version 5 or a Suse Linux system Version 11	263
The Dynamic Workload Console graphical installation and uninstallation fail to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64.	263
On Windows, the Dynamic Workload Console installation fails if you try to reinstall on a different profile of an external WebSphere Application Server.	264
Problems with the silent installation.	264
The silent uninstallation does not work and an error code is returned	264
Problems with the upgrade.	264
Upgrade fails with message AWSUI0085E	264
Problems with the uninstallation	265
Uninstall fails on Windows if the installation directory contains the @ character	265
The Dynamic Workload Console interactive uninstallation wizard fails to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64.	266

Installation fails when reinstalling the
Dynamic Workload Console after having
uninstalled it 266

This part describes how to plan, install, configure, and uninstall the IBM Dynamic Workload Console. It contains the following chapters:

- Chapter 10, “Preparing,” on page 225
- Chapter 11, “Installing,” on page 231
- Chapter 12, “Configuring,” on page 245
- Chapter 13, “Getting started,” on page 247
- Chapter 14, “Upgrading,” on page 251
- Chapter 15, “Uninstalling,” on page 255
- Chapter 16, “Troubleshooting the installation, upgrade, and uninstallation,” on page 257

Chapter 10. Preparing

This chapter gives you an overview of what you need to know to prepare for installation of the Dynamic Workload Console. It consists of the following sections:

- “Overview of the Dynamic Workload Console”
- “Installation overview”
- “Installation considerations” on page 226

Overview of the Dynamic Workload Console

The Dynamic Workload Console is a web-based user interface that is used with the following set of products:

- Tivoli Workload Scheduler
- Tivoli Workload Scheduler for z/OS
- Tivoli Workload Scheduler for Applications
- Dynamic workload broker

You can access Tivoli Workload Scheduler and Dynamic Workload Broker environments from any location in your network using one of the supported browsers connected to the Dynamic Workload Console. The Dynamic Workload Console must be installed on a system that can reach either the Tivoli Workload Scheduler or the Dynamic Workload Broker nodes using network connections.

Installation overview

Perform the following steps to prepare, install, and configure the Dynamic Workload Console:

1. Check the installation prerequisites at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27020800> to verify that your system is compliant.
2. Collect the information necessary to fill in the required fields during the installation. See Chapter 11, “Installing,” on page 231.
3. Choose the installation method that best suits your needs as described in “Installing” on page 198.
4. Install the Dynamic Workload Console by following the instructions provided in “Installing the Dynamic Workload Console” on page 231.
5. If you plan to communicate with the Tivoli Workload Scheduler or Tivoli Workload Scheduler for z/OS Connector Version 8.3 Fix Pack 3, perform the post-installation steps as described in “Post-installation steps to connect to Tivoli Workload Scheduler Version 8.3 Fix Pack 3” on page 237.
6. Log in to the Dynamic Workload Console as described in “Accessing the Dynamic Workload Console” on page 238.
7. In the navigation tree on the left, click one of the following:

Tivoli Workload Scheduler

To access the Tivoli Workload Scheduler available functions

Dynamic workload broker

To access the Dynamic Workload Broker available functions

8. To effectively manage the functions available in the Dynamic Workload Console, create *engine connections* to the Tivoli Workload Scheduler and

Dynamic Workload Broker environments that you want to manage. Without defining engine connections, you can use only a limited set of Dynamic Workload Console functions. For more information, see “Quick steps to define a Tivoli Workload Scheduler engine connection” on page 240 and “Quick steps to define a Dynamic Workload Broker connection” on page 241.

Installation considerations

Before you begin an installation or upgrade, consider the following items that might apply to your specific environment.

- Only one Dynamic Workload Console can be installed on a computer and can be installed as follows:
 - In a new Tivoli Workload Automation instance
 - In an existing Tivoli Workload Automation instance where the embedded WebSphere Application Server is already installed, but the Dynamic Workload Console is not installed
 - Outside any Tivoli Workload Automation instance, using an existing external instance of Tivoli Integrated Portal.

For more information about Tivoli Workload Automation instances, see “Instances of Tivoli Workload Automation” on page 197

- You cannot install more than one instance of the current version of the Dynamic Workload Console on the same workstation. If you attempt to install another instance of the Dynamic Workload Console onto a workstation that already has an upgradeable version on it, you will only be able to upgrade it.
- When you upgrade the Dynamic Workload Console, it is automatically upgraded into a new instance of Tivoli Workload Automation.
- If you plan to install the Dynamic Workload Console on already-installed Tivoli Integrated Portal, ensure that the server associated to the profile where you plan to install is active before starting the installation. Only profiles that are created as described and without customization are supported.
- You must restart the Dynamic Workload Console immediately after the installation if you plan to connect to Internet Protocol version 6 (IPv6) enabled engines.
- Before installing Dynamic Workload Console on Windows and Windows 64, you must start the workstation service of Windows. This applies to Windows 2003 and Windows 2008.

Selecting your installation method

You can install the Dynamic Workload Console using one of the following methods:

Launchpad

Use the launchpad to guide you through the installation of the Dynamic Workload Console, and the Tivoli Workload Scheduler components, from a single interface. For more information about how to install using the launchpad, see “Installing from the launchpad” on page 204.

Installation wizard

Access the installation wizard by running the appropriate *setup* command and entering the configuration settings to install and configure your installation. Using this method, you can synchronously monitor the installation processing and results. For more information, see “Using the installation wizard” on page 231.

This method of installation uses a Java Virtual Machine, and therefore has specific system requirements. See the Dynamic Workload Console System Requirements Document at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27020800> for details on installation requirements.

Silent mode

Customize a *response file* by adding all configuration settings to be used during installation, and then invoke from the command line the *setup* command using the **-silent** keyword. Using this method, you can run the installation unattended and in the background. For more information, see “Performing a silent installation” on page 235.

Instances of Tivoli Workload Automation

During the installation of the Dynamic Workload Console you must decide whether to install into an existing instance of Tivoli Workload Automation or whether to create a new instance. For information, see “Instances of Tivoli Workload Automation” on page 197.

If you are installing into an existing instance of Tivoli Workload Automation, you can install certain components, depending on the components or products that currently exist in that instance. Table 42 describes the actions that you can perform in each different scenario.

Table 42. Installing into an existing instance of Tivoli Workload Automation

If the existing Tivoli Workload Automation instance contains:	You can perform the following:
A Dynamic Workload Console version 8.4, 8.5, or 8.5.1	Upgrade
A Dynamic Workload Console version 8.4, 8.5, or 8.5.1 installed on external WebSphere Application Server	Uninstall and reinstall the Dynamic Workload Console. It is not possible to upgrade the Dynamic Workload Console in this case.
A Dynamic Workload Console version 8.6	Take no action. It is not possible to install the Dynamic Workload Console in this case.
Tivoli Workload Scheduler version 8.5 or 8.5.1 master domain manager or backup domain manager	Take no action. It is not possible to install the Dynamic Workload Console in this case.
A Tivoli Workload Scheduler version 8.6 master domain manager or backup domain manager	Install the Dynamic Workload Console on the common embedded WebSphere Application Server.
A Tivoli Workload Scheduler version 8.5 or 8.5.1 agent with connector	Take no action. It is not possible to install a second instance of the Dynamic Workload Console on the same computer.
A Tivoli Workload Scheduler version 8.6 agent with connector	Install the Dynamic Workload Console on the common embedded WebSphere Application Server.
A Tivoli Workload Scheduler for z/OS connector version 8.5 or 8.5.1	Take no action. It is not possible to install a second instance of the Dynamic Workload Console on the same computer.
A Tivoli Workload Scheduler for z/OS connector version 8.6	Install the Dynamic Workload Console on the common embedded WebSphere Application Server.

Table 42. Installing into an existing instance of Tivoli Workload Automation (continued)

If the existing Tivoli Workload Automation instance contains:	You can perform the following:
A Tivoli Workload Scheduler version 8.6 dynamic domain manager or backup dynamic domain manager.	Install the Dynamic Workload Console on the common embedded WebSphere Application Server.

Any components that are not mentioned in this table, such as the agent without connector, the domain manager or the command-line client, are not installed in Tivoli Workload Automation instances, and so do not impact the Dynamic Workload Console installation.

Installation media

The Dynamic Workload Console is packaged into multiple DVDs, one for each of the supported operating systems. Each DVD contains:

- The installable image
- The *setup* file
- The sample response files
- The launchpad

For a complete list of DVDs and supported operating systems, see the Dynamic Workload Console downloadable documentation at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg24029125>.

Notes:

1. If you copy or mount the DVD to a system directory, make sure that the path name to that directory does not contain the following unsupported characters: { } [] < > \$ | ? ! # * + " / % ' or non US-ASCII characters.
2. If you plan to install on a Windows system from a mapped remote drive, make sure you map the remote folder locally on the system where you want to install, and then run the installation using the local path.
3. If you plan to install on Linux, make sure that the files contained in the mounted image have executable permission, and that the `SETUP.bin` file is not located in a path with blanks.

Installation log files

The type of log files you find on your system depends on the type of installation you performed. This section describes the logs associated with the different installations.

For more information about log files, see the *Administration Guide*.

Interactive wizard installation and uninstallation log files

You can check the following log files for information about the installation. Details of the installation process are recorded in log files on the local computer in the following directories:

Note: The following values are valid only if you have not changed the default value of the TEMP system variable.

Windows operating system

%Temp%\TWA\tdwc86

UNIX and Linux operating systems

/tmp/TWA/tdwc86

Table 43 lists the InstallShield wizard log files.

Table 43. Installation log files

Log file name	Content
tdwcstatus.log	Dynamic Workload Console installation status log file. It reports if the installation completed successfully or with errors. In case of errors it indicates if the error is due to an incorrect field value or to a failed step.
tdwcinstall.log	Dynamic Workload Console installation log file
tdwcuninstall.log	Dynamic Workload Console uninstallation log file
securityConfignnnn.log	Dynamic Workload Console log file containing details about the Tivoli Integrated Portal security configuration performed during installation. The numeric value nnnn is automatically assigned. Access the tdwcinstall.log file to see the filename of the securityConfignnnn.log file.
wsadmin.log	Dynamic Workload Console log file containing details about the interaction of the installation with WebSphere Application Server.
TIPInstaller-00.log	Tivoli Integrated Portal installation log file.

For multiple installations on the same workstation, the log header and footer indicate the user ID (*TWS_user*) for which the installation was performed.

Note: If you are running a silent installation and the response file you are using does not have the correct syntax, the installation fails without producing a log.

Installation log files for the embedded WebSphere Application Server

The application server installation has no log. However, if you update the application server, for example during the application of a Tivoli Workload Scheduler fix pack, a log is created which gives information about the update. The log can be found in the directory *TWS_home/ewAS/logs/update*, where you will find a directory that identifies the fix pack that has been installed, for example: *7.0.0-WS-WASEmbedded-AixPPC64-FP0000027.install*, which contains a log file called */update.log.txt*.

The log for the startup of the application server can be found at:

TWS_home/ewAS/profiles/TIPProfile/logs/server1/startServer.log

Installation log files location

Chapter 11. Installing

This chapter describes how to install the Dynamic Workload Console. It is divided into the following sections:

- “Installing the Dynamic Workload Console”
- “Post-installation steps to connect to Tivoli Workload Scheduler Version 8.3 Fix Pack 3” on page 237
- “Post-installation steps to configure the use of Lightweight Third-Party Authentication (LDAP)” on page 238
- “Accessing the Dynamic Workload Console” on page 238
- “Starting and stopping the Dynamic Workload Console” on page 242

Installing the Dynamic Workload Console

This section explains how to install the Dynamic Workload Console using the available installation methods. It is divided into the following topics:

- “Using the launchpad”
- “Using the installation wizard”
- “Performing a silent installation” on page 235
- “Installing the Tivoli Integrated Portal on an external WebSphere Application Server from the images” on page 237

Using the launchpad

You can install the Dynamic Workload Console using the launchpad. Use the instructions for launching and running the launchpad at “Installing from the launchpad” on page 204, and choose the Dynamic Workload Console installation option in the launchpad. Follow the on-screen instructions. The launchpad runs the installation wizard with some of the options pre-filled. Follow the instructions for “Using the installation wizard,” to complete the process.

Using the installation wizard

Follow these steps to install the Dynamic Workload Console using the installation wizard:

1. Browse to the setup directory and start the installation by running the setup file. The installation wizard first checks if there is enough free space available in the Java temporary directory. If not, the installation exits, and you must increase the size of the Java temporary directory, as described in the Tivoli Workload Scheduler System Requirements Document at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27019747>, before rerunning the installation wizard.
2. Select the language to use while installing the Dynamic Workload Console, and click **OK**.
3. In the welcome panel, click **Next** to continue with the installation.
4. Read and accept the license agreement. Click **Next**.
5. Select the Tivoli Integrated Portal instance. Choose among the following:
 - If you choose to install a new instance of the Tivoli Integrated Portal or if you choose to install on an existing instance of Tivoli Workload Automation that does not contain the embedded WebSphere Application Server, perform

Installing using the graphical wizard

the steps in “Installing a new instance of the Tivoli Integrated Portal.” Perform this installation if you do not have the Tivoli Integrated Portal already installed or if you have installed a Tivoli Workload Scheduler component that does not install the embedded WebSphere Application Server for example a fault-tolerant agent.

- If you choose to install on an existing Tivoli Workload Automation instance that contains the embedded WebSphere Application Server, perform the steps in “Installing on an existing instance of the embedded WebSphere Application Server” on page 234. Perform this installation if you have already installed a Tivoli Workload Automation component that installs the embedded WebSphere Application Server also, for example the master domain manager.
- If you choose to install on top of your existing instance of Tivoli Integrated Portal, perform the steps in “Installing on your existing instance of Tivoli Integrated Portal” on page 235. You perform this installation if you have already installed a Tivoli Integrated Portal with another Tivoli product. For a list of supported Tivoli Integrated Portals, see the Tivoli Workload Scheduler System Requirements Document at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27019747>.
- Starting from V8.6 we do not support anymore the Dynamic Workload Console installed on external WebSphere Application Server. If you do not have the Tivoli Integrated Portal installed, you can install it using the installation DVD or the appropriate eImages as described in the “Installing the Tivoli Integrated Portal on an external WebSphere Application Server from the images” on page 237.

6. Select an installation location. Click **Next**.

7. Specify the user name and password of the Tivoli Integrated Portal user that you want to use as the Dynamic Workload Console administrator.

Note: The user name and password must be operating systems credentials. If the user name and password you specify do not exist, a new operating system user will be created

The User Name must be unique, 3 to 60 characters in length, and contain only the characters a-z, A-Z, 0-9, period (.), hyphen (-), underscore (_), and double-byte character set (DBCS) characters.

The password must be 5 to 16 characters in length and contain only the characters a-z, A-Z, 0-9, period (.), hyphen (-), and underscore (_).

Confirm the password and click **Next**.

8. Choose a new path to install into or choose the path of the existing Tivoli Workload Automation instance. Choose the path where you want to install, from now on referred to as *twa_install_dir*, or accept the default path, and click **Next**.

Make sure that the installation path is 32 characters or less in length and that it does not contain special characters.

Installing a new instance of the Tivoli Integrated Portal

The following applies if you are installing a new Tivoli Workload Automation instance or if you are installing over an existing Tivoli Workload Automation instance where the embedded WebSphere Application Server has not yet been installed. You are in this case if you do not have the Tivoli Integrated Portal already installed or if you have installed a Tivoli Workload Scheduler component that does not install the embedded WebSphere Application Server for example a fault-tolerant agent. In this case Tivoli Workload Scheduler installs the embedded WebSphere Application Server and the Tivoli Integrated Portal.

Installing a new instance of Tivoli Integrated Portal

Follow these steps if you selected to install the Tivoli Integrated Portal and the Dynamic Workload Console:

In the installation choice window, select one of the following installation types.

Default Installation

If you want to use the default Tivoli Integrated Portal settings, proceed with the installation as described in “Default installation.”

Advanced Installation

If you want to customize the Tivoli Integrated Portal settings, proceed with the installation as described in “Advanced installation.”

Default installation: Follow these steps to proceed with a default installation:

1. To start the installation, check that the values displayed in the installation summary window are correct and click **Install**.
2. When the installation completes successfully, a window opens showing links to the user interface on the Tivoli Integrated Portal. For more information, see “Accessing the Dynamic Workload Console” on page 238. If the installation fails, the window contains the list of the items that were not installed and the location of the log file. Click **Finish**.

Advanced installation: Perform the following steps to proceed with an advanced installation:

1. Specify the following port numbers for the Tivoli Integrated Portal or accept the default values. These are embedded WebSphere Application Server ports used by Tivoli Integrated Portal.

HTTP transport

The number of the port that the portal uses for HTTP transport. The default value is **29080**.

HTTPS transport

The number of the port that the portal uses for secure HTTP transport (HTTPS). The default value is **29443**.

Bootstrap

The port number for the bootstrap function. The default value is **22809**.

SOAP connector

The port number for the Simple Object Access Protocol (SOAP) connector on the portal. The default value is **28880**.

SAS server authentication listener

The SAS SSL server authentication listener port number on the portal. The default value is **29401**.

CSiv2 server authentication listener

The CSiv2 SSL ServerAuth Listener port number on the portal. The default value is **29403**.

CSiv2 Client Authentication Listener

The CSiv2 SSL ClientAuth Listener port number on the portal. The default value is **29402**.

ORB listener

The ORB listener port number on the portal. The default value is **29100**.

Administrative console

The HTTP administrative console port on the portal. The default value is **29060**.

Advanced installation

Administrative console secure

The HTTP administrative console secure port on the portal. The default value is **29043**.

IPC connector

The IPC connector on the portal. The default value is **29314**.

REST notification

The REST notification port on the portal. The default value is **29324**.

DCS Unicast port

The DCS Unicast port on the portal. The default value is **29353**.

Click **Next**.

2. Complete the installation by following the steps described in “Default installation” on page 233.

Installing on an existing instance of the embedded WebSphere Application Server

You perform this installation if you have already installed a Tivoli Workload Automation component that installs the embedded WebSphere Application Server also, for example the master domain manager. Follow these steps to install the Dynamic Workload Console on an existing instance of the embedded WebSphere Application Server:

1. Select the existing Tivoli Workload Automation directory.
2. Supply the username and password of the existing instance of the embedded WebSphere Application Server.

Note: If you have already installed WebSphere Application Server into your existing Tivoli Workload Automation instance but do not know the username, click **Retrieve**. The username is retrieved but you still must provide the password. This operation may take a few minutes. If you are performing a silent installation, to find these credentials, run **showSecurityProperties** before running the installation.

3. Select if you want the administrator to access the Tivoli Workload Scheduler console, the Dynamic Workload Broker console, or both. Click **Next**.

Note: If you select one of the two available user interfaces, after installing you can authorize the user to access the other user interface by assigning him one of the predefined roles created by the installation process. For more information, see the information about configuring the Dynamic Workload Console in the *Tivoli Workload Scheduler: Administration Guide*.

4. To start the installation, check that the values displayed in the installation summary window are correct and click **Install**.

Specify the following port numbers for the Tivoli Integrated Portal or accept the default values. These are embedded WebSphere Application Server ports used by Tivoli Integrated Portal

IPC connector

The IPC connector on the portal. The default value is **29314**.

REST notification

The REST notification port on the portal. The default value is **29324**.

DCS Unicast port

The DCS Unicast port on the portal. The default value is **29353**.

- When the installation completes successfully, a window opens showing links to the user interface on the Tivoli Integrated Portal. For more information, see “Accessing the Dynamic Workload Console” on page 238. If the installation fails, the window contains a list of the items that were not installed and the location of the log file. Click **Finish**.

Installing on your existing instance of Tivoli Integrated Portal

You perform this installation if you have already installed a Tivoli Integrated Portal with another Tivoli product. Follow these steps to install the Dynamic Workload Console on top of an existing Tivoli Integrated Portal instance:

- Select the existing Tivoli Integrated Portal instance over which you want to install the Dynamic Workload Console by specifying the installation path.
- Specify the user ID and password of an existing Tivoli Integrated Portal user that you want to set as the Dynamic Workload Console administrator.

Note: If you select one of the two available user interfaces, after installing you can authorize the user to access the other user interface by assigning him one of the predefined roles created by the installation process. For more information, see the information about configuring the Dynamic Workload Console in the *Tivoli Workload Scheduler: Administration Guide*.

- To start the installation, check that the values displayed in the installation summary window are correct and click **Install**.
- When the installation completes successfully, a window opens showing links to the user interface on the Tivoli Integrated Portal. For more information, see “Accessing the Dynamic Workload Console” on page 238. If the installation fails, the window contains a list of the items that were not installed and the location of the log file. Click **Finish**.

Performing a silent installation

You can run the installation in unattended mode from the command line by adding the **-silent** parameter when running the setup installation file. Perform the following steps:

- Run the installation as root on UNIX operating systems, or as Administrator on Windows operating systems.
- Specify all the settings that are prompted when installing using the installation wizard.

The installation settings are provided using a *response file*.

Edit the response file templates provided on the installation DVDs in the `\tdwc\responsefiles\` directory. Instructions for customizing the files are included in the files as commented text. For details about response file properties, see Appendix G, “The Dynamic Workload Console response file properties,” on page 345.

Table 44 lists the response files and the types of installation each performs by operating system:

Table 44. Dynamic Workload Console response files

Type of installation	Response file to use on Unix	Response file to use on Windows
Fresh Dynamic Workload Console on existing TWA instance	TDWC86_FRESH_existTWA_UNIX.txt	TDWC86_FRESH_existTWA_WIN.txt

Installing in silent mode

Table 44. Dynamic Workload Console response files (continued)

Type of installation	Response file to use on Unix	Response file to use on Windows
Fresh Dynamic Workload Console on an external Tivoli Integrated Portal	TDWC86_FRESH_extTIP_UNIX.txt	TDWC86_FRESH_extTIP_WIN.txt
Fresh Dynamic Workload Console on a new TWA instance	TDWC86_FRESH_newTWA_UNIX.txt	TDWC86_FRESH_newTWA_WIN.txt
Uninstall the Dynamic Workload Console	TDWC86_UNINSTALL.txt	TDWC86_UNINSTALL.txt
Upgrade the Dynamic Workload Console on existing Tivoli Workload Automation instance (embedded WebSphere Application Server)	TDWC86_UPGRADE_embeddedWAS_UNIX.txt	TDWC86_UPGRADE_embeddedWAS_WIN.txt

Note: In the upgrade scenarios, choose the embedded version of IBM Websphere Application Server that you originally chose when you installed the Dynamic Workload Console version 8.4 or higher.

To install in silent mode, perform these steps on the computer on which you want to install the Dynamic Workload Console:

1. Copy the sample response file for that operating system to a local temporary directory.
2. Customize the options contained in the response file to suit your requirements and environment. For information about the available options, see Appendix G, “The Dynamic Workload Console response file properties,” on page 345.
3. Run the following command:

Windows operating system:

```
SETUP.exe -options response_file.txt -silent
```

UNIX and Linux operating systems:

```
./SETUP.bin -options response_file.txt -silent
```

where *response_file* is the full path name.

4. Check the result of the silent installation as follows:

Windows operating system:

The installation command is asynchronous, meaning that when it is issued it starts an installation procedure and then ends without returning any value or message. To know whether or not the silent installation ran successfully, see the installation result reported in the `tdwcinstall.log` installation log file stored in the temporary directory.

UNIX and Linux operating systems:

The installation command is synchronous and it returns **0** if the installation ran successfully, or a nonzero value if the installation failed.

Note: For information about the installation result, see the `tdwcinstall.log` installation log file stored in the temporary directory.

Installing the Tivoli Integrated Portal on an external WebSphere Application Server from the images

The following procedure applies if you do not have the Tivoli Integrated Portal installed but you have the WebSphere Application Server installed. To install the Tivoli Integrated Portal, perform the following steps:

1. From the installation DVD or from the downloaded eImages, go to the TDWC_<operating_system>\TDWC\WEBUI\<operating_system>\TIP\ directory where the `sample_response.txt` file is located.
2. Follow the instructions provided in the `sample_response.txt` file to customize the properties necessary to perform the installation.
3. Run the following command:

```
install.sh/bat <java_jre_16_home> sample_response.txt
```

where `java_jre_16_home` is the path where the Java V16 is installed.

The Tivoli Integrated Portal installation creates the **TIPProfile** profile into your existing instance of the WebSphere Application Server. After you installed the Tivoli Integrated Portal, you can install the Dynamic Workload Console on this new Tivoli Integrated Portal instance by following the instructions provided in “Installing the Dynamic Workload Console” on page 231.

Post-installation steps to connect to Tivoli Workload Scheduler Version 8.3 Fix Pack 3

To access a Tivoli Workload Scheduler Version 8.3 Fix Pack 3 environment, you must enable Tivoli Workload Scheduler to work with the Dynamic Workload Console.

Notes:

1. These steps are not necessary to connect to a Tivoli Workload Scheduler environment for any version higher than V8.3 Fix Pack 3. Any upgrades performed after version 8.3 Fix Pack 3 will maintain any changes made during this procedure.
2. If you plan to communicate from the Dynamic Workload Console version 8.4 or higher to Tivoli Workload Scheduler, Version 8.3 Fix Pack 3, make sure that the APAR PK47309 is installed on top of the Tivoli Workload Scheduler engine. For more information, contact IBM Software Support.
3. Before proceeding, it is recommended that you run the `backupConfig.sh` or `backupConfig.cmd` script to backup the Tivoli Workload Scheduler configuration. For information about how to run these scripts, see the *Tivoli Workload Scheduler: Administration Guide*.

This task must be run on the system where the Tivoli Workload Scheduler engine that you want to connect to is installed:

Tivoli Workload Scheduler distributed environment

- On the master domain manager.
- On a full status fault-tolerant agents (FTA) workstation where the Tivoli Workload Scheduler connector is installed.

Tivoli Workload Scheduler z/OS environment

On the distributed system where the Tivoli Workload Scheduler z/OS Connector is installed.

Perform the following steps:

Post-installation steps to connect to Tivoli Workload Scheduler 8.3 Fix Pack 3

1. Make sure that the embedded or external version of WebSphere Application Server, as appropriate, is started on the Tivoli Workload Scheduler workstation and then run the following script:

On Windows operating system:

As Administrator, from the directory *TWS_home\wastools*:

```
webui -operation enable -user TWS_user -password TWS_user_pw  
-port TWS_port [-server TWS_server]
```

On UNIX and Linux systems

As root, from the directory *TWS_home/wastools*:

```
./webui.sh -operation enable -user TWS_user -password TWS_user_pw  
-port TWS_port [-server TWS_server]
```

where:

TWS_user The Tivoli Workload Scheduler administrator user ID.

TWS_user_pw The Tivoli Workload Scheduler administrator password.

TWS_port The SOAP port of the WebSphere Application Server where the Tivoli Workload Scheduler is installed. This is a mandatory setting when using the **enable** flag. Its default values are **31118** for distributed environments, and **31128** for z/OS environments.

TWS_server The name of the server specified in the WebSphere Application Server profile used by Tivoli Workload Scheduler. By default the value assigned to this field is **server1**.

2. Stop and start the external or embedded WebSphere Application Server on the Tivoli Workload Scheduler system where you run the script.

When you have completed these steps, you are ready to create engine connections for the Tivoli Workload Scheduler workstation and to manage your Tivoli Workload Scheduler production environment. For information about how to accomplish these tasks, access the Dynamic Workload Console online general help.

Post-installation steps to configure the use of Lightweight Third-Party Authentication (LDAP)

If the Dynamic Workload Console and the Tivoli Workload Scheduler engine or the Tivoli Workload Scheduler z/OS Connector have been configured with the same LDAP user registry, or are installed on the same computer, you might receive a connection failure. If this happens, use the same Lightweight Third-Party Authentication (LTPA) keys on all servers: the Dynamic Workload Console, the Tivoli Workload Scheduler engine server, and the Tivoli Workload Scheduler z/OS Connector server.

To align the LTPA keys, see the section on configuring the use of Lightweight Third-Party Authentication in the *Administration Guide*.

Accessing the Dynamic Workload Console

When the installation of the Dynamic Workload Console completes successfully, a message with links to the Integrated Solutions Console portal is displayed. If you used the silent installation, this information is stored in the *tdwcinstall.log* installation log file. For more details about where to find the installation logs, see “Interactive wizard installation and uninstallation log files” on page 228.

Accessing the Dynamic Workload Console

From a supported browser, access one of the following links provided by the installation program:

`http://dynamic_workload_console_system:http_port/ibm/console`

`https://dynamic_workload_console_system:https_port/ibm/console`

where:

dynamic_workload_console_system

The hostname or IP address of the system where you installed the Dynamic Workload Console.

http_port

The port number used to access the Dynamic Workload Console using an unsecure connection over HTTP. The default value for this port number is 29080 if you installed the Dynamic Workload Console in a new Tivoli Workload Automation instance. If you installed the Dynamic Workload Console into an existing Tivoli Workload Automation instance, the value for this port is inherited. If the existing Tivoli Workload Automation instance contains the current version of Tivoli Workload Scheduler using default ports, the value is **31123**.

https_port

The port number used to access the Dynamic Workload Console using a secure connection over HTTPS. The default value for this port number is 29443 if you installed the Dynamic Workload Console as a new Tivoli Workload Automation instance. If you installed the Dynamic Workload Console into an existing Tivoli Workload Automation instance, the value for this port is inherited. If the existing Tivoli Workload Automation instance contains the current version of Tivoli Workload Scheduler using default ports, the value is **31124**.

When connecting to the Tivoli Integrated Portal using an HTTPS connection, if you receive a security alert, proceed with the Dynamic Workload Console working session. If you receive security information windows while navigating through the Tivoli Integrated Portal, choose to display nonsecure items to proceed. If you are using Internet Explorer, you can prevent these windows from opening by setting **Display mixed content** to **Enable** in the Security settings.

In the Tivoli Integrated Portal login portlet, enter the user ID and password you specified during the installation, and click **Log in**.

On the navigation bar on the left, expand the **Tivoli Workload Scheduler** entry to access the Dynamic Workload Console and then the Tivoli Workload Scheduler components. Expand the **Dynamic Workload Broker** entry to access Dynamic Workload Broker environments.

To effectively use the functions of these two products, you must define connections to the Tivoli Workload Scheduler engines and the Dynamic Workload Broker servers.

Without defining engine connections, you can perform only this limited set of operations:

On Tivoli Workload Scheduler

- Create browse tasks

Accessing the Dynamic Workload Console

- Create report tasks
- Create event management tasks
- Define user preferences

On Dynamic Workload Broker

Define user preferences

If the user ID you used to connect to the Dynamic Workload Console has been assigned a role different from **TWSWEBUIAdministrator** and **TDWBAdministrator**, you will see a subset of the available panels. This subset depends on the authorizations assigned to the role associated to your user ID. For more information about roles, see the information about configuring the Dynamic Workload Console in the *Tivoli Workload Scheduler: Administration Guide*.

If the user ID you used to connect to the Dynamic Workload Console has no role assigned, you do not see the entries for Tivoli Workload Scheduler and Dynamic Workload Broker in the Tivoli Integrated Portal portal navigation tree.

Quick steps to define a Tivoli Workload Scheduler engine connection

After logging in to the Dynamic Workload Console using the administrator userid or another userid with assigned **TWSWEBUIAdministrator** or **TWSWEBUIConfigurator** roles, use the following steps to create an engine connection to one of your supported Tivoli Workload Scheduler engines.

Note: If you installed the Dynamic Workload Console into a Tivoli Workload Automation instance that had the embedded WebSphere Application Server already installed, the connection to the Tivoli Workload Scheduler component (for example, master domain manager, backup master domain manager, or connector) is automatically defined with blank credentials. The connection is shared with all the Dynamic Workload Console users and no further credentials are needed because Single Sign On is automatically implemented for the component. The same situation applies if you install a Tivoli Workload Scheduler component into a Tivoli Workload Automation instance where the Dynamic Workload Console and the embedded WebSphere Application Server are already installed.

1. To expand the tree, click the Dynamic Workload Console and **Tivoli Workload Scheduler**.
2. Select **Quick start**
3. Click **New Engine**.
4. In the Engine Connection Properties window, assign a name to the engine connection and specify:

Engine Type Either z/OS or Distributed. This is the type of the Tivoli Workload Scheduler engine to connect to.

Hostname The hostname or IP address of system where the distributed engine or the z/OS connector, for z/OS engine types, runs.

PortNumber The bootstrap port number for the Tivoli Workload Scheduler engine. Default values are **31117** for distributed engine, and **31217** for z/OS connector.

Remote Server Name

This setting is valid and mandatory only for z/OS engines. It is the value specified when the engine was created in the z/OS connector. It must exactly match the z/OS connector engine

name and is case sensitive. If the engine was defined using the WASTOOLS "createZosEngine" COMMAND, this is the value specified in the **-name** parameter. This is the name of the remote server of the engine as it is specified in the z/OS connector.

Userid and Password

The user ID and password that are used to connect to the engine. This setting allows access to Tivoli Workload Scheduler from the Dynamic Workload Console. The authorization assigned to the user in the Tivoli Workload Scheduler security file determines the operations allowed.

If you want to test the connection to the Tivoli Workload Scheduler database (mandatory for managing reporting and event management functions), you must select **Enable reporting** and specify the user credentials.

5. Click **Test Connection** to check that the configuration was successful and that the Dynamic Workload Console is communicating with the selected engine. If the test connection fails, see *Tivoli Workload Scheduler: Troubleshooting Guide*, SC32-1275.

Note: Make sure you run "Post-installation steps to connect to Tivoli Workload Scheduler Version 8.3 Fix Pack 3" on page 237 before testing the engine connection if you are connecting to a Tivoli Workload Scheduler version 8.3 Fix Pack 3 engine or z/OS Connectors.

Quick steps to define a Dynamic Workload Broker connection

The Dynamic Workload Console supports a single connection to one Dynamic Workload Broker engine at any given time for each authorized user. A different connection is supported for each authorized user.

After having logged in to the Dynamic Workload Console using the administrator user ID, or another user ID with assigned **TDWBAdministrator** or **TDWBConfigurator** roles, follow these steps to create an engine connection to a supported Dynamic Workload Broker engine:

1. In the Dynamic Workload Console, click **Dynamic Workload Broker** to expand the tree.
2. Select **Configuration**.
3. Click **Server connection**.
4. In the Server Connection specify:

Hostname The host name of the Dynamic Workload Broker you want to connect to.

Non secure port The non-secure port to be used for connection.

Secure port The secure port to be used for connection.

Use Secure Connection
Specify whether a secure connection must be used. For more information about security, see the *Tivoli Workload Scheduler: Administration Guide*, SC23-9113.

Username Optionally specify a different user for the server connection.

Accessing the Dynamic Workload Console

The connection to the new server is enabled using the credentials of the user you specified. Each user has access to only one server connection.

Password Specify the password for the authenticated user the connection applies to.

5. Click **OK** to save your changes. The server connection you specified is enabled and is immediately effective.

Starting and stopping the Dynamic Workload Console

To start and stop the Dynamic Workload Console or an engine, you must start and stop the application server instance it is installed on.

Embedded WebSphere Application Server on a Tivoli Integrated Portal in a Tivoli Workload Automation instance

If you installed the Dynamic Workload Console on the embedded WebSphere Application Server, you can start and stop the server as follows:

Windows operating system:

To stop: *install_dir\wastools\stopWas.bat*

To start: *install_dir\wastools\startWas.bat*

UNIX and Linux operating systems:

To stop: *install_dir/wastools/stopWas.sh*

To start: *install_dir/wastools/startWas.sh*

WebSphere Application Server on a Tivoli Integrated Portal outside a Tivoli Workload Automation instance

If you are using an external instance of WebSphere Application Server, use the following WebSphere Application Server scripts to start and stop an application server instance.

Note: These scripts can also be used to start and stop embedded WebSphere Application Server, although it is suggested that you use the method described above.

Windows operating system:

*ewas_install_dir\bin\stopServer.bat app_server
-user user_id -password user_id_pw*

ewas_install_dir\bin\startServer.bat app_server

UNIX and Linux operating systems:

*ewas_install_dir/bin/stopServer.sh app_server
-user user_id -password user_id_pw*

/ewas_install_dir/bin/startServer.sh app_server

where:

ewas_install_dir

Is the directory where the WebSphere Application Server is installed.

| **app_server**

| Is the server name specified in the Tivoli Integrated Portal profile
| related to the Dynamic Workload Console or to the engine. The
| default is **server1**.

user_id

Is the administrator user ID specified when installing the Dynamic Workload Console or the engine.

user_id_pw

Is the administrator user ID password specified when installing the Dynamic Workload Console or the engine.

Chapter 12. Configuring

This chapter describes how to configure the Dynamic Workload Console. You can perform the following optional configuration steps at any time after the installation.

- Configuring new users to access the Dynamic Workload Console
- Configuring the Dynamic Workload Console to use a user registry
 - Configuring the Dynamic Workload Console with LDAP - RACF (for more information, see WebSphere documentation at: Configuring to secure Lightweight Directory Access Protocol user registry using Resource Access Control Facility based on z/OS)
- Configuring roles to access the Dynamic Workload Console
- Configuring the Dynamic Workload Console to use Single Sign-On
- Securing your communication using Secure Socket Layer protocol
- Configure the Dynamic Workload Console to launch in context

Note: If, after installing, you have more than one instance of WebSphere Application Server managing any Tivoli Workload Automation products, you must ensure that they have the same LTPA token_keys.

For information about configuration, see "Configuring the Dynamic Workload Console" in the *Tivoli Workload Scheduler: Administration Guide*, SC23-9113.

Chapter 13. Getting started

This chapter explains how to get started with using the Dynamic Workload Console.

When you connect to the Dynamic Workload Console, you see a portfolio on the left with an entry for each Tivoli product installed inside the Tivoli Integrated Portal, such as, Tivoli Workload Scheduler.

You can access the Dynamic Workload Console from any computer in your environment using a web browser through both the secure HTTPS or HTTP protocol.

For an interactive overview of the product and its features, you can view several demo scenarios, available (in English only) on the product information center at the following link: <https://www.ibm.com/developerworks/wikis/display/tivolimediagallery/Tivoli+Workload+Scheduler>

The Dynamic Workload Console interface consists of the following sections:

Portfolio

Located on the left, it has a tree structure and contains all the entries to launch the Dynamic Workload Console functions. Use the portfolio to navigate to the panels.

Note: The upgrade does not report the customization you have performed on the portfolio. The defaults settings are installed.

Portlet area

Your working area. It displays the panels corresponding to your selection in the portfolio. From each panel you can access the online help by clicking on the "?" symbol at the top right corner of the portlet.

Task bar

Contains a tab to open each active function you called from the portfolio. Each time you click an entry of the portfolio, the corresponding panel is opened in the portlet area. When you open a new panel, the preceding ones are minimized to tabs on the task bar and you can switch between the panels by clicking on these tabs. The browser task bar contains up to five open tabs. If you open more than five tabs, a new browser window opens and you can move from one page to another by opening the Select Action menu.

The portfolio has separate sections for Tivoli Workload Scheduler and Dynamic Workload Broker.

Tivoli Workload Scheduler portfolio

The Tivoli Workload Scheduler portfolio contains the following entries:

Quick Start

Open this entry to run some basic operations. Click here to create and manage queries of objects on the plan and to create and modify connections to the Dynamic Workload Console engines.

All Configured Tasks

Open this entry to view a list of all your saved tasks to monitor objects in the plan. A set of predefined tasks is provided to help you start using the application for the first time. These tasks cover the most common queries you might want to launch to find information about scheduling objects running on distributed, z/OS, or both platforms.

All Configured Reports

Open this entry to view a list of all your saved reports. From this view you can create new reports and customize existing ones.

Dashboard

Open this entry to have a graphical view that shows the progress of the current plan on the engines for which you configured a connection and specified its inclusion in the dashboard.

Workload

Manage your workload to design objects in the database, to handle plans, to submit jobs or job streams to monitor objects in the plan, or to generate reports.

Design

Open this entry to create, list, and edit object and object definitions in the database. Click here, for example, to create and modify jobs, job streams, and event rules.

Forecast

Open this entry to work with plans, creating and viewing trial and forecast plans and listing archived plans.

Submit

Open this entry to submit jobs and job streams on request

Monitor

Open this entry to create, list, and edit tasks to monitor objects in the plan. Click here, for example, to create and modify queries for jobs or job streams in the plan. Also, click here to handle queries about workload dependencies and events.

Scheduling Environment

Design and control the topology of your scheduling environment: the workstations and domains.

Design

Open this entry to create, list, and edit workstations and domains in your environment.

Monitor

Open this entry to create, list, and edit tasks to monitor workstations and domains in the plan.

Reporting

Define and run reports.

Generate Historical Reports

Open this entry to create reports that gather historical data.

Generate Plan Reports

Open this entry to create reports with details about your plans.

Generate Custom SQL Reports

Open this entry to generate and run customized SQL reports.

Settings

Configure and modify general settings about Tivoli Workload Scheduler

Manage Engines

Open this entry to create, list, and edit your connections to the Tivoli Workload Scheduler engine.

Manage User Preferences

Open this entry to configure and modify settings about table layout, time zone, and dashboard layout for Tivoli Workload Scheduler.

Manage Settings

Open this entry to import and export user preferences, configured tasks, and engine connections and to configure your settings repository.

Dynamic workload broker portfolio

The Dynamic Workload Broker portfolio contains the following entries:

Scheduling Environment

Define and control logical resources and resource groups in your dynamic scheduling environment

Define New Logical Resource

Open this entry to define a new logical resource required to run jobs dynamically.

Define New Resource Group

Open this entry to create a new group definition to aggregate different logical resources in a group.

Logical Resources

Open this entry to list and edit defined logical resources.

Resource Groups

Open this entry to list and edit defined resource groups

Configuration

Define a connection to the dynamic workload broker component.

Server Connections

Open this entry to create or edit a connection to the dynamic workload broker component.

Definitions

Manage dynamic workload to create list and submit jobs.

Define a New Job

Open this entry to create new dynamic job definitions.

Jobs Open this entry to list, edit and submit dynamic workload objects.

Tracking

Monitor your dynamic workflow and the environment status.

Job Instances

Open this entry to monitor submitted dynamic job instances, see job output, and cancel jobs.

Computers

Open this entry to monitor and edit status and details of dynamic workstations.

Preferences

Customize display settings for Dynamic Workload Broker.

User Preferences

Open this entry to customize number of rows in each table page and set the displayed time zone.

First actions

The following sections describe the first and main actions you perform when you connect to the Dynamic Workload Console.

Creating a connection to a Tivoli Workload Scheduler engine

You type the details (such as IP address, user name, and password) to access a Tivoli Workload Scheduler engine, and, optionally, a database to operate with objects defined in plans or stored in the database. From the Dynamic Workload Console you can access the current plan, a trial plan, a forecast plan, or an archived plan for the distributed environment or the current plan for the z/OS® environment. You might want to access the database to perform actions against objects stored in it or generate reports showing historical or statistical data. In addition, working both on the database and on plans, you can create and run event rules to define and trigger actions that you want to run in response to events occurring on Tivoli Workload Scheduler nodes.

Defining a scheduling environment

You define your Tivoli Workload Scheduler network. You create workstation definitions on the database representing the physical machines or computer systems on which your workload is scheduled to run. Tivoli Workload Scheduler network is made up of the workstations where job and job stream processing occurs. When you design your network, you assign roles to these workstations to suit your specific business requirements. You can design your network with multiple domains, to divide control of a large network into smaller manageable groups. A typical Tivoli Workload Scheduler network consists of a workstation acting as master domain manager and at least one domain.

Defining scheduling objects in the database

You define your workload, which consists of jobs that are concatenated in job streams. Then, you specify the calendars and run cycles according to which job streams must run. Moreover, you define possible dependencies to condition the workload processing. All these definitions can be done within the Workload Designer.

Creating tasks to manage Tivoli Workload Scheduler objects in the plan

You specify some filtering criteria to query a list of scheduling objects whose attributes satisfy the criteria you specified. Starting from this list, you can navigate and modify the content of the plan, switching between objects, opening more lists, and accessing other plans or other Tivoli Workload Scheduler environments.

Creating a connection to a Tivoli dynamic workload broker scheduling environment

You type the details (such as IP address, user name, password, and port) to access a dynamic workload broker workstation. Specify if you want to work in secure HTTPS or HTTP protocol. After creating the connection, opening the tracking computer you can view status and details of broker workstations, and define resources and dynamic jobs. For more details about dynamic scheduling, see: *Scheduling Workload Dynamically*.

Chapter 14. Upgrading

This chapter describes how to upgrade the Dynamic Workload Console to the current version.

Note: The upgrade on a external WebSphere Application Server is not supported. If you have already installed the WebSphere Application Server with another product you must uninstall the Dynamic Workload Console and install it again.

To upgrade an instance of Tivoli Workload Automation:

- You must upgrade all components that are part of the instance. For example, if your instance includes one master domain manager and also the Dynamic Workload Console, you must upgrade both of these components.
- You must upgrade all the components part of that instance before you can upgrade the Dynamic Workload Console.

If you installed the Dynamic Workload Console sharing the WebSphere Application Server with other Tivoli Workload Automation components, when you upgrade those components, the existing Dynamic Workload Console will not work until you upgrade it to the new version.

When you upgrade the Dynamic Workload Console on the embedded WebSphere Application Server, changes are made to the directory structure of the console. Thus, the section contains the following topics:

- “Updating authentication”
- “Upgrading the console installed on an embedded WebSphere Application Server” on page 252
- “Performing the upgrade” on page 253

Updating authentication

This section describes how your configured authentication mechanism is upgraded.

In versions of Tivoli Workload Scheduler before V8.6, authentication was configured to use stand-alone user registries, managed by the embedded WebSphere Application Server. The available options were:

- Local operating system
- Custom (through PAM - Pluggable Authentication Module)
- LDAP
- File Registry

If you enabled LDAP, you could use one of the following servers:

- IBM Tivoli Directory Server
- Sun ONE
- Microsoft Windows Active Directory
- RACF configured on IBM Tivoli Directory Server

Updating authentication mechanism

Versions of Tivoli Workload Scheduler from V8.6 onwards are configured for authentication (through the embedded WebSphere Application Server) in VMM (Virtual Member Manager) mode. This creates a *Federated User Registry*, which supports the simultaneous use of more than one user registry. The user registry choices and LDAP server options are similar to those in versions before V8.6.

During the upgrade, your existing configuration are migrated, so that when the upgrade is complete the product is configured to use the same authentication mechanism as before, but within a Federated User Registry.

For detailed information, see *Tivoli Workload Scheduler: Administration Guide*.

Upgrading the console installed on an embedded WebSphere Application Server

This section provides information about the upgrade of the Dynamic Workload Console on an embedded WebSphere Application Server.

Directory structure

This section describes the program directory structure and the directory structure for SSL files that was implemented in version 8.5.1. This section applies if you are upgrading from version 8.3 or 8.4. If you are upgrading from version 8.5 or 8.5.1, this directory structure already exists.

Program directory

When you upgrade the Dynamic Workload Console to the current version, a new program directory structure is created. During the upgrade process, components of the Dynamic Workload Console are moved from the old directory structure and then updated into the new directory structure. The Dynamic Workload Console program files remain in the original installation directory.

If you have any custom configurations (for example, custom scripts or backup processes) existing in your Dynamic Workload Console structure, you must update them so that they work in the new directory structure.

For example, if you originally installed the Dynamic Workload Console into the default directory `c:\Program Files\IBM\webui\`, you have a directory structure as follows:

```
c:\Program Files\IBM\TWA\webui\appserver
c:\Program Files\IBM\TWA\webui\wastools
c:\Program Files\IBM\TWA\webui\_webuiutils
c:\Program Files\IBM\TWA\webui\_webuiuninst
c:\Program Files\IBM\TWA\webui\_jvm
```

When you upgrade the Dynamic Workload Console, the new directory structure is:

```
c:\Program Files\IBM\TWA\eWAS
c:\Program Files\IBM\TWA\wastools
c:\Program Files\IBM\TWA\TDWC
c:\Program Files\IBM\TWA\TDWC\_tdwcscripts
c:\Program Files\IBM\TWA\TDWC\_tdwcuninst
c:\Program Files\IBM\TWA\TDWC\_tdwcutils
c:\Program Files\IBM\TWA\TDWC\_jvm
```

The new directory structure includes new embedded WebSphere Application Server tools that are common to Tivoli Workload Scheduler.

Directory for SSL files

When you upgrade to the current version, a new directory for SSL files is created. The following describes the old and new directory structures.

Note: Before upgrading you must backup any customized SSL keys and copy them to the default installation path.

After upgrading, the old SSL files stored in PKCS12 format are imported into new SSL files in JKS format.

The old PKCS12 files are also copied to the new directory as a backup. The key.p12 file becomes **TWSServerKeyFile.jks**. The trust.p12 file becomes **TWSServerTrustFile.jks**.

Previous directory structure

- *TWSInstallationPath*\appServer\profiles\webuiprofile\config\cells*CellName*\nodes\NodeName\key.p12
- *TWSInstallationPath*\appServer\profiles\webuiprofile\config\cells*CellName*\nodes\NodeName\trust.p12

New directory structure

- *TWSInstallationPath*\eWAS\profiles\TIPProfile\config\cells\TIPNode\nodes\TIPCell\key.p12
- *TWSInstallationPath*\eWAS\profiles\TIPProfile\config\cells\TIPNode\nodes\TIPCell\trust.p12
- *TWSInstallationPath*\eWAS\profiles\TIPProfile\etc\TWSServerKeyFile.jks
- *TWSInstallationPath*\eWAS\profiles\TIPProfile\etc\TWSServerTrustFile.jks

Note: The files key.p12 and trust.p12 are not used by the Dynamic Workload Console, but are backed up.

Performing the upgrade

You can upgrade the Dynamic Workload Console using the following methods:

Interactive wizard

To upgrade using the interactive wizard, run the setup for the operating system on which you are installing:

On Windows operating system:

WINDOWS\SETUP.exe

Before beginning, stop the **appserverman** process by running the following commands:

```
Shutdown.cmd -appsrv
StartWas.bat -direct
```

After running these commands, verify that all Tivoli Workload Scheduler processes are stopped with the exception of the embedded WebSphere Application Server. The embedded WebSphere Application Server must remain running.

On UNIX and Linux operating systems:

SETUP.sh or *operating_system*/SETUP.bin.

Updating authentication mechanism

Note: SETUP.sh copies the entire image to a temporary directory. Ensure there is enough space available.

Launchpad

Start the launchpad and select the Dynamic Workload Console upgrade. The installation wizard is launched with some options pre-selected to upgrade the console.

Silent You can run the upgrade silently and in background by creating a response file from the template provided and running the installation wizard with the **-silent** option. See “Performing a silent installation” on page 235 for more details on how to run a silent installation or upgrade. See the following section for the upgrade information that you must supply in the response file.

Follow the installation panels to complete the upgrade. The following list describes the fields you must provide during the upgrade.

Use an existing instance of the Dynamic Workload Console

When you are prompted that a previous version of the Dynamic Workload Console has been found, select **Use an existing instance**. From the drop-down list, choose the instance that you are upgrading.

Administrative credentials of application server

Enter the external or embedded WebSphere Application Server user name and password.

Backup directory

Choose a backup directory. This directory contains only configuration information and other program-related objects and not the external or embedded WebSphere Application Server files. Note that this directory remains on your computer even after the upgrade is complete.

IPC connector

The IPC connector on the portal. The default value is **29314**.

REST notification

The REST notification port on the portal. The default value is **29324**.

DCS Unicast port

The DCS Unicast port on the portal. The default value is **29353**.

Notes:

1. For information about Tivoli Workload Automation instances, see “Instances of Tivoli Workload Automation” on page 227.
2. During an upgrade on Windows, the embedded WebSphere Application Server Windows Service account name in the local OS user registry is changed to the administrator user name of the Tivoli Integrated Portal. If you use a custom registry or LDAP registry, the service is upgraded to the installation user.
3. It is not necessary to manually stop the embedded WebSphere Application Server prior to upgrading, as it is stopped automatically during the upgrade procedure.

Chapter 15. Uninstalling

This chapter describes how to uninstall the Dynamic Workload Console. It is divided into the following sections:

- “Uninstalling using the wizard”
- “Uninstalling in silent mode”

Uninstalling using the wizard

To uninstall the Dynamic Workload Console using the wizard, perform the following steps:

1. Start the Tivoli Integrated Portal.
2. Start the uninstall as follows:

On Windows operating system:

Perform one of the following:

- From the *TWA_home\TDWC* directory, run the command:
`uninstall.bat`
- From the Control Panel, click **Add/Remove Programs**. Scroll down the list of software, and select the Dynamic Workload Console. Click **Change/Remove**.

On UNIX operating systems:

From the *TWA_home/TDWC* directory, run the command:
`uninstall.sh`

3. Select the language.
4. Click **Next** in the Dynamic Workload Console uninstall welcome window.
5. Provide the external or embedded WebSphere Application Server administrator user name and password, and click **Next**.
6. In the uninstall summary window, check that the directory from where the product is to be removed and the features to be removed are correct, and then click **Uninstall**. If you installed the Dynamic Workload Console and the Tivoli Integrated Portal, they are both uninstalled. If you installed the Dynamic Workload Console on a existing Tivoli Integrated Portal only the Dynamic Workload Console is uninstalled.
7. When the uninstall completes, a window showing a message about the success of the operation is displayed. Click **Finish** to exit the InstallShield Wizard.

Uninstalling in silent mode

You can perform a silent uninstall of the Dynamic Workload Console.

Before starting to uninstall ensure that the Tivoli Integrated Portal is active, and move to a directory different from the *tdwc_install_dir*.

Run the uninstall command as follows:

On Windows operating system:

```
tdwc_home\tdwc\uninstall.bat -options  
response_file.txt -silent
```


Uninstalling in silent mode

On UNIX or Linux operating systems:

```
twa_home/tdwc/uninstall.bin -options  
response_file.txt -silent
```

where *response_file* is the full path name.

Chapter 16. Troubleshooting the installation, upgrade, and uninstallation

This chapter describes how to troubleshoot the installation, upgrade, and uninstallation of the Dynamic Workload Console. It is divided into the following sections:

- “Installation and uninstallation log and trace files”
- “Recovering a failed InstallShield wizard installation”
- “Recovering a failed upgrade”
- “Uninstalling the Dynamic Workload Console and the Tivoli Integrated Portal manually” on page 258
- “Troubleshooting scenarios” on page 259

Note: See the section “Uninstalling the Dynamic Workload Console and the Tivoli Integrated Portal manually” on page 258 to manually uninstall or recover from a failed installation.

Installation and uninstallation log and trace files

For information about installation log files, see “Installation log files” on page 228.

Recovering a failed InstallShield wizard installation

The recovery of a failed installation is fully described in the *Tivoli Workload Scheduler: Planning and Installation Guide*.

Follow the instructions in the *Tivoli Workload Scheduler: Planning and Installation Guide* up to the point where you want to modify the values of a step, and then follow these instructions:

1. The values used in each step for the Dynamic Workload Console are all stored in one place - *Step 0*. So if you discover, for example, that the step that configures the embedded Tivoli Integrated Portal has failed because a port is in use, you must go to *Step 0* and modify the value for the port in that step.
2. Set the status of *Step 0*, plus the status of the step that failed, to *Ready*.
3. In *all* cases, run *Step 0* in the Step List, using the **Run next** option. Step 0 uses the original data, as modified by you, to regenerate all of the scripts that run the steps.
4. Resume the wizard from the failed step, either running **Run all** to complete the installation without stopping at each step, or **Run next**, to complete the installation step by step.

Note: You cannot rerun any step that has completed successfully, other than *Step 0*.

Recovering a failed upgrade

In the case of a failed upgrade, contact IBM Software Support.

Uninstalling the Dynamic Workload Console and the Tivoli Integrated Portal manually

Perform the following steps to manually remove an instance of Tivoli Workload Automation that contains the Dynamic Workload Console and uses the embedded Tivoli Integrated Portal. In the case of a failed installation, you may find some of the steps are unnecessary, depending on when the installation failed.

To remove an instance of Tivoli Workload Automation that contains an integrated installation of Tivoli Workload Scheduler and the Dynamic Workload Console, do the following actions:

1. Uninstall Tivoli Workload Scheduler and as described in the *Tivoli Workload Scheduler: Planning and Installation Guide*.
2. Remove the Dynamic Workload Console by performing the steps described in the procedure below.

If you want to remove the Dynamic Workload Console from an instance of Tivoli Workload Automation without removing the Tivoli Workload Automation instance, contact IBM Software Support.

Note: Only perform these manual steps on systems where the Dynamic Workload Console is installed, otherwise you delete the Composite Offering Installer registry.

On Windows operating system:

1. If you have already removed the Dynamic Workload Console, using the installation DVD or the downloaded eImages, run the following command from the `\tdwc\webui\<operating_system>\scripts\` directory:

```
# ./clean.bat - installRoot <eWAS_installation_directory> -force true
```

If, instead, the Dynamic Workload Console is still installed but not working, run the following command from the `TWA\tdwc_tdwcutils\scripts\` directory:

```
# ./cleanDE.bat - installRoot <eWAS_installation_directory> -force true
```

2. Stop the service:
`install_dir\bin\WASService -stop TIPProfile_Port_defaulthost_port`
3. Remove the service:
`install_dir\bin\WASService -remove`
4. Navigate to the `install_dir` and note the name of the ID file `twainstancexxx.id`. You will need this information later in the procedure.
5. Remove the directory:
`install_dir`
6. Remove the directory:
`C:\Program Files\Common Files\InstallShield\Universal\TDWC`
7. Delete the following registry key:
`HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall\ e625666383dedb70850864e2a6feaa2e1371705039`
8. Remove the file:
`%windir%\TWA\twainstancexxx.properties`
9. Restart the system.

On UNIX and Linux operating systems:

Manually uninstall the Dynamic Workload Console

1. If you have already removed the Dynamic Workload Console, using the installation DVD or the downloaded eImages, run the following command from the `/tdwc/webui/<operating_system>/scripts/` directory:

```
# ./clean.sh - installRoot <eWAS_installation_directory> -force true
```

If, instead, the Dynamic Workload Console is still installed but not working, run the following command from the `TWA/tdwc/_tdwcutils/scripts/` directory:

```
# ./cleanDE.sh - installRoot <eWAS_installation_directory> -force true
```

2. Stop the server by running the command:
`install_dir/wastools/stopWas.sh`
3. Navigate to the `install_dir` and note the name of the ID file `twainstancexxx.id`. You will need this information later in the procedure.
4. Remove the directory:
`install_dir`
5. Remove the directory:

On AIX

```
/usr/lib/objrepos/InstallShield/Universal/TDWC
```

On all UNIX systems, except AIX

```
ROOT_USER_HOME/InstallShield/Universal/TDWC
```

6. Remove the file:
`etc/TWA/twainstancexxx.properties`

Troubleshooting scenarios

The troubleshooting scenarios are listed in the following categories:

- “Problems with the launchpad”
- “Problems with the interactive wizard” on page 260
- “Problems with the silent installation” on page 264
- “Problems with the upgrade” on page 264
- “Problems with the uninstallation” on page 265

Problems with the launchpad

The following problems might be encountered while using the launchpad to install the Dynamic Workload Console:

- “Warning messages displayed when using the launchpad on Linux”
- “Undefined error when using launchpad on Windows operating system” on page 260

Warning messages displayed when using the launchpad on Linux

Problem description:

Warning messages might be displayed on the standard output when using the launchpad on Linux.

Cause and solution

You can ignore these messages because they do not indicate a malfunction of the launchpad.

Undefined error when using launchpad on Windows operating system

Problem description:

You try to install the Dynamic Workload Console on a Windows operating system using the launchpad and you get an "Undefined" error message. The launchpad does not start.

Cause and solution

Make sure that the path from where you launched the installation does not contain folder names longer than eight characters. If it does, then map the path to the launchpad.exe, and run the launchpad from that new path.

Problems with the interactive wizard

The following problems might be encountered while running the Dynamic Workload Console interactive installation:

- "The Dynamic Workload Console installation hangs"
- "Installation hangs during stopWas command"
- "Tivoli Integrated Portal installation fails even if into the logs you find successfully installed" on page 261
- "Installation from a remote shared folder fails on Windows operating system" on page 262
- "Installation fails on a Linux 390 system with a hostname which is not a Fully Qualified Domain Name" on page 262
- "Java Virtual Machine (JVM) failure when installing the Dynamic Workload Console on a Red Hat Enterprise Linux (RHEL) Version 5 or a Suse Linux system Version 11" on page 263
- "The Dynamic Workload Console graphical installation and uninstallation fail to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64" on page 263
- "On Windows, the Dynamic Workload Console installation fails if you try to reinstall on a different profile of an external WebSphere Application Server" on page 264

The Dynamic Workload Console installation hangs

Problem description:

The installation of the Dynamic Workload Console does not proceed. This occurs regardless of the method you used to install.

Cause and solution

Make sure an active personal firewall is not preventing the installation process from connecting to the network. If it is, allow the connection and then continue with the installation.

Installation hangs during stopWas command

Problem description:

The installation of the Dynamic Workload Console does not proceed.

Cause and solution

Manually uninstall the Dynamic Workload Console

The installation of the Dynamic Workload Console does not proceed because the stopWas command is hanging.

To continue the installation, open the Task Manager and locate the Java process of the embedded WebSphere Application Server. This process is the java.exe process with the associated installation username. Stop this process.

Then, find the associated WASService process. This process is the WASService.exe process with the associated the installation username. Stop this process.

Continue with the installation.

Tivoli Integrated Portal installation fails even if into the logs you find successfully installed

Problem description:

You are installing the Tivoli Integrated Portal, either using the wizard or the silent installation, and the installation fails with the following error:

Start installing TIP - Tivoli Integrated Portal 2.2

For log details, please see

C:\Documents and Settings\Administrator\TIPInstaller-00.log

and {install location}\logs.zip

Preparing SILENT Mode Installation...

```
=====
GenericInstaller      (created with InstallAnywhere by Macrovision)
-----
=====
Installing...
=====|=====|=====|=====]
```

Installation Complete.

SUCCESS: The overall installation is successful.

Current OS is Windows XP

Executing

'C:\IBM\TWS\TDWC\eWAS\profiles\TIPProfile\bin\tipcli.bat' with arguments:

'Version'

The ' characters around the executable and arguments are not part of the command.

Execute:Java13CommandLauncher:

Executing

'C:\IBM\TWS\TDWC\eWAS\profiles\TIPProfile\bin\tipcli.bat' with arguments:

'Version'

The ' characters around the executable and arguments are not part of the command.

The following error occurred while executing this line:

C:\temp\TDWC\WEBUI\WINDOWS\xml\commonTargets.xml:1735:

Execute failed: java.io.IOException:

Cannot run program "C:\IBM\TWS\TDWC\eWAS\profiles\TIPProfile\bin\tipcli.bat"

in directory "C:\IBM\TWS\TDWC\eWAS\profiles\TIPProfile\bin"):

CreateProcess error=2, The system cannot find the file specified.

Installation Complete.

SUCCESS: The overall installation is successful.

Current OS is Windows XP

Executing

'C:\IBM\TWS\TDWC\eWAS\profiles\TIPProfile\bin\tipcli.bat' with arguments:

'Version'

The ' characters around the executable and arguments are not part of the command.

Manually uninstall the Dynamic Workload Console

```
Execute:Java13CommandLauncher:  
Executing  
'C:\IBM\TWS\TDWC\ewAS\profiles\TIPProfile\bin\tipcli.bat' with arguments:  
'Version'
```

Cause and solution

This error occurs if you remove an instance of the Dynamic Workload Console, and the Tivoli Integrated Portal is also installed in the same path, without running the uninstaller (for example, removing manually only the content of the TWA directory). In this case, the Tivoli Integrated Portal instance remains registered in the Deployment Engine installation registries and when you perform another installation of the Tivoli Integrated Portal the installer reports the error indicated previously.

Run the procedure described in “Uninstalling the Dynamic Workload Console and the Tivoli Integrated Portal manually” on page 258 to clean your environment. Restart the installation from the failed step.

Installation from a remote shared folder fails on Windows operating system

Problem description:

You try to install the Dynamic Workload Console on a Windows operating system from a shared network folder that uses Universal Naming Convention (UNC). The installation fails.

Cause and solution

You must map the remote folder locally on the Windows system where you want to install the Dynamic Workload Console and then run the installation using the local path.

Installation fails on a Linux 390 system with a hostname which is not a Fully Qualified Domain Name

Problem description:

You install the Dynamic Workload Console with the embedded WebSphere Application Server on a server with a hostname is not a Fully Qualified Domain Name. The installation fails and the following error is stored in the twainstall.log file:

```
ADMU3011E: Server launched but failed initialization. startServer.log,  
SystemOut.log(or job log in zOS) and other log files under  
/oracle/ibm/TDWC/ewAS/profiles/TIPProfile/logs/tdwcserver  
should contain failure information.
```

```
WASX7023E: Error creating "SOAP" connection to host "localhost";  
exception information:  
com.ibm.websphere.management.exception.ConnectorNotAvailableException:  
com.ibm.websphere.management.exception.ConnectorNotAvailableException:  
ADMC0016E: The system cannot create a SOAP connector to connect to host  
localhost at port 28880.
```

Cause and solution

Run the following command from the system prompt on the Linux 390 system where you tried to install the Dynamic Workload Console:

```
hostname --fqdn
```


If the command returns:

```
hostname: Unknown host
```

the host name is not resolved. You must specify a hostname with a fully qualified domain name to install the embedded WebSphere Application Server. Update the hostname notation, as explained in the embedded WebSphere Application Server documentation, and then rerun the installation.

Java Virtual Machine (JVM) failure when installing the Dynamic Workload Console on a Red Hat Enterprise Linux (RHEL) Version 5 or a Suse Linux system Version 11

Problem description:

When installing the Dynamic Workload Console on a Red Hat Enterprise Linux Version 5 or a Suse Linux Version 11 system, you might receive the error "Failed to find VM - aborting".

Cause and solution

Linux systems have a security feature named 'Security Enhanced Linux', or SELinux for short. A weaker version of SELinux was included in Red Hat Enterprise Linux Version 4, and was disabled by default. On these versions of Red Hat Enterprise Linux and Suse Linux, this security feature is enabled by default. SELinux helps to keep the host secure from certain types of malicious attacks. However, the default settings have been known in many cases to prevent Java from running properly.

To fix this issue, choose one of the following options:

- Configure SELinux so that it knows that the Dynamic Workload Console Java related processes are acceptable to run.
- Change the mode of SELinux to *Permissive* by entering `setenforce 0` on the command line. SELinux will be fully enabled again the next time the system is rebooted or if `setenforce 1` is entered on the command line. For the Dynamic Workload Console to function, you must set `setenforce 0`. For more information about `setenforce`, see the documentation for your operating system.

The Dynamic Workload Console graphical installation and uninstallation fail to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64

Problem description:

When launching the Dynamic Workload Console installation or uninstallation wizard in interactive mode on Red Hat Enterprise Linux (RHEL) Version 5 x86-64, you might receive the following error:

For the installation:

The installer is unable to run in graphical mode.
Try running the installer with the `-console` or `-silent` flag.

For the uninstallation:

The uninstaller is unable to run in graphical mode.
Try running the uninstaller with the `-console` or `-silent` flag.

Cause and solution

Manually uninstall the Dynamic Workload Console

If you run into this problem, launch the installation or the uninstallation in silent mode. For more information, see “Performing a silent installation” on page 235 and “Uninstalling in silent mode” on page 255.

On Windows, the Dynamic Workload Console installation fails if you try to reinstall on a different profile of an external WebSphere Application Server

Problem description:

This situation applies to a Windows operating system. You install the Dynamic Workload Console on a profile, for example, ProfileA, of an existing WebSphere Application Server installation. You remove the Dynamic Workload Console successfully and then try to install it on a different profile of the same WebSphere Application Server. The installation fails.

Cause and solution

A possible cause is that when you removed the Dynamic Workload Console, some files belonging to ProfileA were not removed. To solve this problem, stop ProfileA and then install the Dynamic Workload Console again on the other profile.

Problems with the silent installation

The following problems might be encountered while running the Dynamic Workload Console silent installation:

- “The silent uninstallation does not work and an error code is returned”
- “Tivoli Integrated Portal installation fails even if into the logs you find successfully installed” on page 261

The silent uninstallation does not work and an error code is returned

Problem description:

If you try to perform a silent uninstall with a response file that does not exist, either because the file name is incorrect or because you specified the wrong directory, an error code is returned and the uninstallation does not run. Nothing is logged in the temporary directory and no messages are issued.

Cause and solution

Ensure that you specify a valid response file name.

Problems with the upgrade

The following problem might be encountered while running the Dynamic Workload Console upgrade:

- “Upgrade fails with message AWSUI0085E”

Upgrade fails with message AWSUI0085E

Problem description:

You are running the upgrade of the Dynamic Workload Console and the following error occurs:

The instance of the Dynamic Workload Console that you want to upgrade uses an LDAP user registry.

The LDAP server type you are using is not supported.

The supported LDAP server types are:

IBM Tivoli Directory Server, Microsoft Active Directory, Sun ONE DS, and RACF configured on IBM Tivoli Directory Server. Use a supported LDAP server type or install a fresh instance of the Dynamic Workload Console

Cause and solution

The upgrade fails because you are using an LDAP server type that is not supported. The supported LDAP server types are:

- IBM Tivoli Directory Server
- Microsoft Active Directory
- Sun ONE DS
- RACF configured on IBM Tivoli Directory Server

Configure the Dynamic Workload Console to use:

- One of the supported LDAP server types.
- The local operating system - the default authentication system at installation on Windows operating systems.
- The custom (through PAM - Pluggable Authentication Module) - the default authentication system at installation on UNIX and Linux operating systems.

or install a fresh instance of the Dynamic Workload Console and then configure it to use one of the supported LDAP server types.

Problems with the uninstallation

The following problems might be encountered while running the Dynamic Workload Console uninstallation:

- "Uninstall fails on Windows if the installation directory contains the @ character"
- "The Dynamic Workload Console interactive uninstallation wizard fails to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64" on page 266
- "Installation fails when reinstalling the Dynamic Workload Console after having uninstalled it" on page 266

Uninstall fails on Windows if the installation directory contains the @ character

Problem description:

When running **uninstaller.exe** to remove the Dynamic Workload Console installed in a directory that contains the @ character, for example C:\Program Files\ibm\TDWC\@-._~a, the uninstall fails and the following error message is displayed:

```
CreateProcess failed ==> The system cannot find the file specified.
```

Cause and solution

The uninstall fails because '@' is a special character for ISMP. ISMP is not able to manage an installation directory containing this character.

You can bypass the problem by running the uninstall as follows:

```
"C:\Program Files\ibm\TDWC\@-._~a\_jvm\jre\bin\java.exe"
-cp "C:\Pr Fi\ibm\TDWC\@-._~a\_tdwcuninst\uninstall.jar" run
```

Run this command outside the installation directory, otherwise the installation directory is not removed.

The Dynamic Workload Console interactive uninstallation wizard fails to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64

Problem description:

When launching the Dynamic Workload Console interactive uninstallation wizard on Red Hat Enterprise Linux (RHEL) Version 5 x86-64, the following error is displayed:

The uninstaller is unable to run in graphical mode.
Try running the uninstaller with the -console or -silent flag.

See “The Dynamic Workload Console graphical installation and uninstallation fail to start on Red Hat Enterprise Linux (RHEL) Version 5 on x86-64” on page 263 for the solution.

Installation fails when reinstalling the Dynamic Workload Console after having uninstalled it

Problem description:

Installation fails when trying to reinstall the Dynamic Workload Console on a Windows system where the Dynamic Workload Console has been uninstalled.

Cause and solution

This problem can be due to the fact that eWAS directory was not correctly deleted during uninstallation. If during the Dynamic Workload Console uninstallation eWAS directory cannot be deleted because it is locked by another process, the uninstallation wizard does not fail but completes successfully without removing the directory. The solution for this problem is to force the uninstallation to fail when eWAS directory cannot be deleted. In this way you can kill all the processes related to the eWAS directory. Alternatively, you can manually delete it and finally rerun the installation step.

Part 5. Tivoli Workload Scheduler for z/OS Agent

Chapter 17. Installing the Tivoli Workload Scheduler for z/OS Agent	269	Coexistence with previous versions	287
User authorization requirements	270	User authorization requirements	287
Authorization roles for running the wizard and a silent installation	270	Upgrading notes	287
Authorization roles for running the twsinst script	270	Upgrading using the installation wizard	288
Authorization roles for Software Distribution	270	Upgrading using a silent installation	288
Starting the launchpad	271	Upgrading using twsinst	289
Installing with the installation wizard	271	Upgrading process	289
Performing a silent installation	273	Examples	291
Silent installation using response file templates	274	Upgrading using Software Distribution	291
Silent installation using an automatically generated response file	275	Creating and installing the software package block	291
Installing using twsinst	276	Upgrading procedure overview	292
twsinst	276	Prerequisite: Install the Common Inventory Technology	292
Installing using Software Distribution	280	Upgrading the agent	293
Software packages and parameters	281	Upgrading the Java runtime to run job types with advanced options	294
Installation procedure	283	Adding Java runtime or enabling dynamic capabilities after upgrade	295
Prerequisite: Installing the Common Inventory Technology (CIT)	283	Chapter 19. Uninstalling the Tivoli Workload Scheduler for z/OS Agent	297
Installing the Tivoli Workload Scheduler for z/OS Agent	284	User authorization requirements	297
Enabling dynamic capabilities	285	Uninstalling using the wizard	297
Adding the Java runtime to run job types with advanced options	285	Performing a silent uninstallation	298
Adding Java runtime environment after installation or upgrade	285	Uninstalling the Tivoli Workload Scheduler for z/OS Agent using the twsinst script	298
Enabling dynamic capabilities after installation or upgrade	286	Uninstalling using the Software Distribution CLI	299
Chapter 18. Upgrading the Tivoli Workload Scheduler for z/OS Agent	287		

Chapter 17. Installing the Tivoli Workload Scheduler for z/OS Agent

This chapter describes how to perform a first-time installation of the current version of Tivoli Workload Scheduler for z/OS Agent (z-centric).

You install this agent to run workload from the mainframe to distributed systems with a low cost of ownership.

Using this agent you can run your workload:

Statically

To run existing job types, for example scripts, on a specific Tivoli Workload Scheduler for z/OS Agent. In this case, you install the Tivoli Workload Scheduler for z/OS Agent on the distributed systems and connect it to the z/OS system through the Tivoli Workload Scheduler for z/OS controller.

Statically including job types with advanced options

In this case, you install the Tivoli Workload Scheduler for z/OS Agent on the distributed systems adding the Java runtime, and connect it to the z/OS system through the Tivoli Workload Scheduler for z/OS controller.

Dynamically

To run existing job types allowing the product to assign them to the workstation that best meets both the hardware and software requirements needed to run them. In this case, you install the Tivoli Workload Scheduler for z/OS Agent on the distributed systems adding the dynamic capabilities, and connect it to the dynamic domain manager. Refer to the *Tivoli Workload Scheduler: Planning and Installation Guide* for a detailed explanation on how to install a dynamic domain manager for a z/OS controller. During the installation of the dynamic domain manager for a z/OS controller, you must provide the **master domain manager** and the **Tivoli Workload Scheduler Netman port** values, even if these values are not used in a z/OS lightweight end-to-end configuration because the fault-tolerant agents is not needed.

Dynamically including job types with advanced options

To run existing job types and job types with advanced options allowing the product to assign them to the workstation that best meets both the hardware and software requirements needed to run them. In this case, you install the Tivoli Workload Scheduler for z/OS Agent on the distributed systems adding the dynamic capabilities and the Java runtime, then connecting it to the dynamic domain manager. Refer to the *Tivoli Workload Scheduler: Planning and Installation Guide* for a detailed explanation on how to install a dynamic domain manager for a z/OS controller. During the installation of the dynamic domain manager for a z/OS controller, you must provide the **master domain manager** and the **Tivoli Workload Scheduler Netman port** values, even if these values are not used in a z/OS lightweight end-to-end configuration because the fault-tolerant agents is not needed.

The chapter contains the following sections:

- “User authorization requirements” on page 270
- “Installing with the installation wizard” on page 271

Installing Tivoli Workload Scheduler for z/OS Agent

- “Performing a silent installation” on page 273
- “Installing using twsinst” on page 276
- “Installing using Software Distribution” on page 280
- “Adding Java runtime environment after installation or upgrade” on page 285
- “Enabling dynamic capabilities after installation or upgrade” on page 286

For information how to use it see *Scheduling End-to-end with z-centric Capabilities*.

User authorization requirements

Check the authorization roles before beginning the installation procedure.

Authorization roles for running the wizard and a silent installation

The following table provides the authorization roles required to use the installation, uninstallation, and upgrade wizards, and to run a silent installation.

Table 45. Required authorization roles for running the installation wizard

Activity	Required role
Installing using the wizard Upgrading from version 8.5.1	UNIX and Linux root access. Windows Your login account must be a member of the Windows Administrators group or domain administrators with Act as Part of the Operating System .

Authorization roles for running the twsinst script

The following table provides the authorization roles required to use the **twsinst** method.

Table 46. Required authorization roles for running twsinst

Activity	Required role
Running the twsinst script	UNIX and Linux root access. Windows Your login account must be a member of the Windows Administrators group or domain administrators with Act as Part of the Operating System .

Authorization roles for Software Distribution

The following table provides the authorization roles required to use the Software Distribution method.

Table 47. Required authorization roles for Software Distribution

Activity	Required role
Using Software Distribution to install a software package block	admin, senior, or super
	UNIX and Linux root access. Windows Your login account must be a member of the Windows Administrators group with Act as Part of the Operating System .

Starting the launchpad

The launchpad requires some additional installation prerequisites. For detailed information, see the Tivoli Workload Scheduler System Requirements Document at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27019747>.

Note: When running the launchpad on UNIX and Linux operating systems, make sure that you export the browser location to the BROWSER environment variable.

The launchpad automatically accesses and runs the related installation setup file in interactive mode. If you have autorun enabled, the launchpad starts automatically. To start the launchpad installation program, perform the following steps:

1. From the DVD that contains the component you want to install, run the launchpad as follows:

UNIX and Linux operating systems:

From the root directory, run `launchpad.sh`.

Windows operating systems:

From the root directory, run `launchpad.exe`.

The launchpad opens.

2. In the launchpad, click to install the configuration that you want. The related installation program starts. To proceed with the installation of the selected Tivoli Workload Scheduler for z/OS component, follow the instructions described in the following sections.

To access information about product installation prerequisites, click the different options in the left frame of the launchpad.

Installing with the installation wizard

This section describes how to install the Tivoli Workload Scheduler for z/OS Agent by using the installation wizard.

For a complete list of the supported operating systems, refer to <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27019747>.

For information to download the agent eImage from the Passport Advantage® Online website, refer to <http://www.ibm.com/support/docview.wss?rs=672&uid=swg24030243>.

Note: IBM i is an exception. The Tivoli Workload Scheduler for z/OS Agent on this platform can be installed only using the **twinsinst** command line.

Installing using the installation wizard

For a graphical installation, from the installation DVD, start the launchpad as described in “Starting the launchpad” on page 271.

Or, run the setup for the operating system on which you are installing. From the TWS directory on the DVD, perform the following:

On UNIX and Linux operating system:

`SETUP.sh` or `op_system/SETUP.bin`

On Windows operating system:

`op_system\SETUP.cmd`

Note: `SETUP.sh` copies the entire image to a temporary directory. Ensure there is enough space available.

To install the Tivoli Workload Scheduler for z/OS Agent, perform the following steps:

1. From the **Welcome** panel, click **Next**.
2. Select the options you want:

Dynamic capabilities

To add dynamic scheduling capabilities to your distributed environment.

Java runtime to run job types with advanced options

To add the runtime environment for Java that is used to run jobs supplied with the product or implemented through the custom plug-ins. The runtime environment also enables the capability to remotely run, from the agent, the Dynamic Workload Broker resource command on the server.

3. Follow the installation wizard to complete the installation. The following list describes the installation options.

Note: At the end of the installation, when a summary screen appears, you are unable to click **Back** to return to the previous wizard screens.

User name and password

Specify the Tivoli Workload Scheduler for z/OS user name and password. Spaces are not allowed.

On UNIX and Linux operating systems:

This user account must be created manually before running the installation. Create a user with a home directory. By default, Tivoli Workload Scheduler for z/OS Agent is installed under the HOME directory of the selected user.

On Windows operating systems:

If the user account does not already exist, it is automatically created by the installation wizard. If you specify a domain user, specify the name as *domain_name\user_name*. If you are installing in a domain controller, the user name must always be *domain_name\user_name*. If you specify a local user with the same name as a domain user, the local user must first be created manually by an administrator and then specified as *system_name\user_name*. Type and confirm the password.

Note: The password must comply with the password policy in your Local Security Settings, otherwise the installation fails.

Enable HTTPS communication for the JobManager port

This option enables the HTTPS communication between the Tivoli Workload Scheduler master domain manager or Tivoli Workload Scheduler for z/OS

controller and the agent. If you accept this default, ensure that you also configure the HTTPS communication on the z/OS master. For secure connections, it is recommended that you use HTTPS. To use the HTTP communication, clear this checkbox. However, if your situation requires improved performance of communication between the Tivoli Workload Scheduler for z/OS controller and the agent, you can choose to use HTTP.

JobManager port number

The port used by the Tivoli Workload Scheduler for z/OS or the Dynamic Workload Broker component to connect to the Tivoli Workload Scheduler agent. It is used by JobManager to run dynamic workload and to run workload coming from a z/OS environment in a distributed environment. JobManager is the network process that controls the dynamic scheduling environment and the z-centric environment. The installation default value is **31114**. The valid range is from 1 to 65535.

Host name or IP address

The fully qualified host name on which the agent will be contacted by the Dynamic Workload Broker.

Agent display name

The name of the agent as shown by the Dynamic Workload Console.

Dynamic workload broker host name

The fully qualified host name of the master or of the backup master used by the Tivoli Workload Scheduler for z/OS Agent to connect to the Dynamic Workload Broker.

Dynamic workload broker HTTPS port number

The HTTPS transport port specified when installing the master or backup master. It is used by the Tivoli Workload Scheduler for z/OS Agent to connect to the Dynamic Workload Broker. The installation default value is **31116** although if you leave the field blank, it defaults to **0**. The valid range is from 1 to 65535.

Install location

Enter the name of the directory where to install the Tivoli Workload Scheduler for z/OS Agent for the specified user. On UNIX and Linux, the default directory is `/opt/IBM/TWA`. On Windows, the default directory is `%ProgramFiles%\IBM\TWA`.

On UNIX and Linux, optionally check **Create symbolic links** to create links in the `/usr/bin` directory. Any existing Tivoli Workload Scheduler for z/OS symbolic links are overwritten. The maximum field length is 46 characters and the name must not contain numbers. Parentheses `()` are not allowed. You cannot use national characters.

On UNIX and Linux, the name must be longer than one character and the first character must be `/`.

On Windows, the name must be longer than three characters, the second character must be `:`, and the third character must be `\`.

Performing a silent installation

A silent installation runs according to parameters set in a response file. The response file includes all the installation information required to run the installation without user intervention.

Installing using the installation wizard

There are two ways to customize a response file to satisfy your installation requirements:

- Edit an existing response file template provided on the installation DVDs. See “Silent installation using response file templates.”
- Automatically create a customized response file by running the installation wizard. See “Silent installation using an automatically generated response file” on page 275.

Note: IBM i is an exception. The Tivoli Workload Scheduler for z/OS Agent on this platform can be installed only using the **twsinst** command line.

Silent installation using response file templates

Edit the response file templates provided in \TWS\RESPONSEFILES\, on the installation DVD. Instructions for customizing the files are included in the files as commented text.

Table 48 lists, by operating system, the response files and the types of installation each performs:

Table 48. Response files

Type of installation	Response file to use
Installing on UNIX and Linux	
Fresh installation of Tivoli Workload Scheduler for z/OS Agent	TWS86_FRESH_ZCENTRIC_Agent_UNIX.txt
Upgrade of Tivoli Workload Scheduler for z/OS Agent	TWS86_UPGRADE_ZCENTRIC_Agent_UNIX.txt
Installing on Windows	
Fresh installation of Tivoli Workload Scheduler for z/OS Agent	TWS86_FRESH_ZCENTRIC_Agent_WIN.txt
Upgrade of Tivoli Workload Scheduler for z/OS Agent	TWS86_UPGRADE_ZCENTRIC_Agent_WIN.txt

Note: When you are performing a silent installation on UNIX zSeries systems, you must first save the response file in UTF 8 format.

To perform a silent installation using a response file template, perform the following steps:

1. Copy the relevant response file to a local directory and edit it to meet the needs of your environment.

Note: Be sure to review the license agreement information included in the installation media. To accept the terms of the license agreement, set the **licenseAccepted** parameter to **true** in the response file you are using. This value is required to complete the silent installation successfully.

2. Save the file with your changes.
3. Enter the following command:

UNIX and Linux operating systems:

```
./SETUP.bin -options local_dir/response_file.txt -silent
```

where *response_file.txt* is the name of the response file to be used for installation. The *SETUP.bin* file is located in the root directory of the relevant installation DVD.

Windows operating systems:

```
SETUP.exe -options local_dir\response_file.txt -silent
```

where *response_file.txt* is the name of the response file to be used for installation. The *SETUP.exe* file is located in *TWS\WINDOWS* on 32-bit operating systems, and in *TWS\WINDOWS_X86_64* on 64-bit operating systems.

4. To check that installation was successful, review the installation messages in the *summary.log* file.

Silent installation using an automatically generated response file

During the initial installation of the current version of Tivoli Workload Scheduler for z/OS, you can create a response file based on the parameters of the initial installation. You use this response file to run subsequent installations with the same parameters. Creating an automatically generated response file is recommended because all input is automatically validated by the program.

To perform a silent installation using an automatically generated response file, perform the following steps:

1. Perform the initial installation using the following command:

UNIX and Linux operating systems:

```
./SETUP.bin -options-record local_dir/response_file.txt
```

where *response_file.txt* is the name of the response file to be created. The *SETUP.bin* file is located in the root directory of the relevant installation DVD.

Windows operating systems:

```
SETUP.exe -options-record local_dir\response_file.txt
```

where *response_file.txt* is the name of the response file to be created. The *SETUP.exe* file is located in *TWS\WINDOWS* on 32-bit operating systems, and in *TWS\WINDOWS_X86_64* on 64-bit operating systems.

The installation wizard is launched. Follow the prompts and complete the installation (for details, see “Installing with the installation wizard” on page 271). A response file is created in the directory that you specified in the setup command.

Although the response file contains the parameters that you entered in the installation wizard, be aware that you might need to edit the file for each subsequent installation. This depends on the configuration of each workstation.

Note: The response file that is created will contain unencrypted password information.

2. For all subsequent installations, enter the following command:

UNIX and Linux	<code>./SETUP.bin -options local_dir/response_file.txt -silent</code>
Windows	<code>SETUP.exe -options local_dir\response_file.txt -silent</code>

Installing using the installation wizard

3. After each silent installation, review the installation messages in the `summary.log` file to check that installation was successful.

Installing using `twsinst`

You can use the **`twsinst`** script to install Tivoli Workload Scheduler for z/OS Agent if you are not running a Java Virtual Machine (JVM). The **`twsinst`** utility provides an alternative to the silent installation wizard. See “Performing a silent installation” on page 273.

Optionally, you can add to the Tivoli Workload Scheduler for z/OS Agent:

- Dynamic capabilities, to provide your distributed environment with dynamic scheduling capabilities.
- Java runtime to run job types with advanced options, both those supplied with the product and the additional types implemented through the custom plug-ins. The runtime environment also enables the capability to remotely run, from the agent, the Dynamic Workload Broker resource command on the server.

Agents installed using **`twsinst`** can be uninstalled only by using **`twsinst`**.

For a complete list of the supported operating systems, see <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27019747>.

`twsinst`

During the installation process, **`twsinst`** creates a file in the following directories for each of the installation steps:

On UNIX and Linux operating systems:

`/user's_home/TWS`

On Windows operating systems:

`%ProgramFiles%\IBM\TWA\TWS`

If you stop and restart the installation, the installation process starts from the installation step where it was stopped.

To install the Tivoli Workload Scheduler for z/OS Agent and all the supported language packs, perform the following steps:

On UNIX and Linux operating systems:

1. Insert the DVD for your operating system or download the agent eImage.
2. Create the Tivoli Workload Scheduler for z/OS user. The software is installed by default in the user's home directory, referred to as `/installation_dir/TWS`

User: `TWS_user`

Home: `/installation_dir/TWS` (for example: `/home/user1/TWS` where `user1` is the name of the Tivoli Workload Scheduler for z/OS user.)

3. Log in as root on the workstation where you want to install the product.
4. From the `DVD_root/TWS/operating_system` directory, run **`twsinst`** using the synopsis described in the following section.

On Windows operating systems:

1. Insert the DVD for your operating system or download the agent eImage.

2. Log in as administrator on the workstation where you want to install the product.
3. From *DVD_root/TWS/operating_system*, run **twsinst** using the synopsis described in the following section.

Note: **twsinst** for Windows is a Visual Basic Script (VBS) that you can run in CScript and WScript mode.

The Tivoli Workload Scheduler for z/OS user is automatically created. The software is installed by default in the Tivoli Workload Scheduler for z/OS installation directory. The default value is %ProgramFiles%\IBM\TWA.

On IBM i operating systems:

1. On the IBM i system, sign on as **QSECOFR** user.
2. Create the IBM i user profile for which the Tivoli Workload Scheduler for z/OS Agent is installed. This user profile is not to be confused with the user performing the installation logged on as **QSECOFR**, but instead is the user that you specify in the **-uname username** parameter when running the **twsinst** script. For descriptions of the syntax parameters, see “twsinst” on page 276. You cannot use an existing IBM i system user profile, an application supplied user profile, or any of the following reserved IBM i user profiles:
QDBSHR, QDFTOWN, QDOC, QLPAUTO, QLPINSTALL, QRJE, QSECOFR, QSPL, QSYS, QTSTRQS
3. On the IBM i system, verify that no library exists with the same name as the user profile supplied for the Tivoli Workload Scheduler for z/OS Agent.
4. Use the DVD or download the eImage from the Passport Advantage Online website.
5. Open a QSH command entry running the **qsh** command.
6. Run the **twsinst** script as described in “twsinst” on page 276. During the installation the product creates an IBM i library and a job description with the same name as the user profile created in the Step 2. The installation procedure adds this library to the user profile library list of the Tivoli Workload Scheduler for z/OS Agent user profile and sets this job description as the job description of the Tivoli Workload Scheduler for z/OS Agent user profile. By default, the software is installed in the user's home directory.

A successful installation using **twsinst** issues the return code RC = 0. A failed installation issues the return code RC = 1. In case of a failed installation, refer to the installation messages documented in *Tivoli Workload Automation: Messages and Codes*.

Synopsis

On UNIX and Linux operating systems:

Show command usage and version

```
twsinst -u | -v
```

Install a new instance

```
twsinst -new -uname username
[-addruntime true|false]
[-displayname agent_name]
[-hostname host_name]
[-inst_dir install_dir]
[-jimport port_number]
[-jimportssl true|false]
[-lang lang_id]
```

Installing by using twsinst

```
[-reset_perm]  
[-skip_usercheck]  
[-tdwbport tdwbport_number]  
[-tdwbhostname host_name]
```

On Windows operating systems:

Show command usage and version

```
twsinst -u | -v
```

Install a new instance

```
twsinst -new -uname username  
-password user_password  
[-addjruntime true|false]  
[-displayname agent_name]  
[-domain user_domain]  
[-hostname host_name]  
[-inst_dir install_dir]  
[-jimport port_number]  
[-jimportssl true|false]  
[-lang lang_id]  
[-skip_usercheck]  
[-tdwbport tdwbport_number]  
[-tdwbhostname host_name]
```

Parameters

-addjruntime *true|false*

Adds the Java runtime to run job types with advanced options, both those supplied with the product and the additional types implemented through the custom plug-ins. Valid values are **true** and **false**. The default is **true**.

-domain *user_domain*

Windows only. The domain name of the Tivoli Workload Scheduler user. The default is the name of the workstation on which you are installing the Tivoli Workload Scheduler for z/OS Agent.

-displayname *agent_name*

The name to be assigned to the Tivoli Workload Scheduler for z/OS Agent. The default is the host name.

-hostname *host_name*

The fully qualified host name or IP address on which the agent will be contacted by the Dynamic Workload Broker.

-inst_dir *installation_dir*

The directory where to install the Tivoli Workload Scheduler for z/OS Agent. On UNIX and Linux, this path cannot contain blanks. On Windows, if you specify a path that contains blanks, enclose it in double quotes. If you do not manually specify a path, the path is set to the default home directory. On UNIX and Linux, the path is set to the *user_name* home directory, and on Windows it is set to %ProgramFiles%\IBM\TWA.

-jimport *port_number*

The port used by the Tivoli Workload Scheduler for z/OS controller or the Dynamic Workload Broker to connect to the Tivoli Workload Scheduler for z/OS Agent. The default value is **31114**. The valid range is from 1 to 65535.

-jimportssl *true|false*

The port used by the Tivoli Workload Scheduler for z/OS controller, or by the dynamic workload broker to connect to the Tivoli Workload Scheduler for z/OS Agent. This number is registered in the *ita.ini* file, located in *ITA\cpa\ita* on Windows and *ITA/cpa/ita* on UNIX, Linux, and IBM i.

For communication using SSL or HTTPS

Set **jimportssl = true**. To communicate with the Dynamic Workload Broker, it is recommended that you set the value to **true**. In this case, the port specified in **jimport** communicates in HTTPS. If you specify **true**, ensure that you also configure the HTTPS communication on the z/OS master.

For communication without using SSL, or through HTTP

Set **jimportssl = false**. In this case the port specified in **jimport** communicates in HTTP.

The default value is **true**.

To increase the performance of the Tivoli Workload Scheduler for z/OS server, it is recommended that you set this value to **false**.

-lang *lang_id*

The language in which the **twsinst** messages are displayed. If not specified, the system LANG is used. If the related catalog is missing, the default C language catalog is used.

Note: This is the language in which the installation log is recorded, and not the language of the installed engine instance. **twsinst** installs all languages as default.

-new A fresh installation of the agent. Installs an agent and all supported language packs.

-password *user_password*

Windows only. The password of the user for which you are installing Tivoli Workload Scheduler for z/OS Agent.

-reset_perm

UNIX and Linux only. Reset the permissions of the **libatrc** library.

-skip_usercheck

Enable this option if the authentication process within your organization is not standard, thereby disabling the default authentication option. On UNIX and Linux, skip the check of the user in the `/etc/passwd` file or using the **su** command. On Windows, does not create the user you specified in the `-uname username` parameter. If you specify this parameter you must create the user manually before running the script.

-tdwbhostname *host_name*

The fully qualified host name of the dynamic domain manager of backup dynamic domain manager used to connect to the Tivoli Workload Scheduler for z/OS Agent. It is used together with the `-tdwbport tdwbport_number` parameter. It adds the capability to run dynamic workload to the Tivoli Workload Scheduler for z/OS Agent. If not specified, the default value is **localhost**. This value is registered in the `ResourceAdvisorUrl` property in the `JobManager.ini` file.

-tdwbport *tdwbport_number*

The dynamic domain manager of backup dynamic domain manager HTTP or HTTPS port number used to connect to the Tivoli Workload Scheduler for z/OS Agent. It is used together with the `-tdwbhostname host_name` parameter to add the capability to run dynamic workload to the Tivoli Workload Scheduler for z/OS Agent. This number is registered in the `ResourceAdvisorUrl` property in the `JobManager.ini` file. The default value

Installing by using twsinst

is **0**, meaning that the capability to run dynamic workload to the agent is not added. The valid range is from 0 to 65535.

-u Displays command usage information and exits.

-uname *username*

The name of the user for which the Tivoli Workload Scheduler for z/OS Agent is installed. This user name is not to be confused with the user performing the installation logged on as **root** on UNIX and Linux, as **administrator** on Windows, and as **QSECOFR** on IBM i.

On UNIX, Linux and IBM i, this user account must be created manually before running the installation. Create a user with a home directory. By default, Tivoli Workload Scheduler for z/OS Agent is installed in the home directory of the specified user.

-v Displays the command version and exits.

Examples

This example describes how to install the Tivoli Workload Scheduler for z/OS Agent and accept the default value to add the runtime environment for Java. The runtime environment is used to run jobs supplied with the product or implemented through the custom plug-ins, it also enables the capability to remotely run, from the agent, the Dynamic Workload Broker resource command on the server.

On UNIX and Linux operating systems:

```
./twsinst -new
-uname TWS_user
-jmportssl false
-jmport 31114
-inst_dir /home/user1/TWA
```

On Windows operating systems:

```
twsinst -new
-uname TWS_user
-password qaz12qaz
-jmportssl false
-jmport 31114
-inst_dir "c:\Program Files\IBM\TWA"
```

Installing using Software Distribution

This section describes how to install the Tivoli Workload Scheduler for z/OS Agent by using Software Distribution software package blocks. During the installation, you can add the following:

- Dynamic capabilities
- The runtime environment for Java that is used to run both jobs supplied with the product and additional types implemented through the custom plug-ins.

The agent installed using the Software Distribution software package blocks has the following characteristics:

- It is installed in its own path, independent of any other Tivoli Workload Automation products or components installed on the same system.
- It cannot share components of the Tivoli Workload Automation network.
- It cannot have a connector added to it and therefore cannot be directly connected to the Dynamic Workload Console.

Use Software Distribution software package blocks to install the Tivoli Workload Scheduler for z/OS Agent only if you do not run a JVM on the workstation. If this is not your situation, you might choose to perform a silent installation instead. See “Performing a silent installation” on page 273.

Agents installed using Software Distribution software package blocks can be uninstalled *only* using Software Distribution.

Software packages and parameters

The Tivoli Workload Scheduler for z/OS Agent can be installed by distributing a software package block (SPB), using the Software Distribution component of Tivoli Configuration Manager, Versions 4.1, 4.2, 4.2.1, 4.2.2, or 4.2.3. You can distribute the SPB, by using either the command line interface or from the Tivoli desktop.

Note: Do not modify the SPB supplied with the product.

An SPB exists for each supported operating system located on the installation disks under the directory of the operating system. The SPBs are named according to the operating system: `Tivoli_LWA_operating_system.SPB`. For the packages to be distributed, they must be imported in software package profiles. The software package profiles must be named according to the operating system and user: `TWS_LWA_operating_system_TWS_user.8.6.0.00`. Possible values for operating system are:

- AIX
- HP
- SOLARIS
- WINDOWS
- LINUX_I386
- LINUX_PPC
- LINUX_S390
- SOLARIS_I386
- HPIA64
- LINUX_X86_64
- WINDOWS_X86_64

The Tivoli Workload Scheduler for z/OS Agent installation parameters are defined as default variables in the software package. Following is the list of installation parameters.

backup Optional. Indicates a backup. For a fresh install, specify **false**. The default value is **false**.

display_name

The name of the agent, as shown by the Dynamic Workload Console.

domain Optional unless the user is a domain user. Windows operating systems only. The domain name of the user. The default value is *computer_name*.

fresh_install

Required. Indicates if this is a first time install. To perform a fresh installation, specify **true**. To perform an upgrade, specify **false**. The default value is **true**.

group The group, at operating system level, to which the user belongs.

host_name

The fully qualified host name or IP address on which the agent will be contacted by the Dynamic Workload Broker.

Installing using Software Distribution

installer

Windows only. The user ID of the installer of the Tivoli Workload Scheduler for z/OS Agent. The default value is Administrator.

install_dir

Required. The fully qualified path to the location of the Tivoli Workload Scheduler for z/OS Agent installation. This path must be a fully qualified path and cannot contain blanks. On Windows workstations, the path is created if it does not already exist. On UNIX, Linux and IBM i, the path is the same as the user's home directory. The default values are:

- UNIX, Linux and IBM i, operating systems: *user_home*
- Windows operating systems: $\$(system_drive)\win32app1TWS\TWS_user$

jm_port

Specify the value of the HTTP port that is used for communication between the Tivoli Workload Scheduler server and the Tivoli Workload Scheduler agent with dynamic capabilities enabled or the communication between the Tivoli Workload Scheduler for z/OS server and the Tivoli Workload Scheduler agent with z-centric capabilities.

To communicate in HTTP you must also set the following parameters: -D *jm_port=nnnnn* where *nnnnn* is the port number and -D *jm_sec_port=0*. The default number of *jm_port* is **31114**.

jm_sec_port

Specify the value of the HTTPS port that is used for communication between the Tivoli Workload Scheduler server and the Tivoli Workload Scheduler agent with dynamic capabilities enabled or the communication between the Tivoli Workload Scheduler for z/OS server and the Tivoli Workload Scheduler agent with z-centric capabilities.

To communicate in HTTPS, you must also set the following parameters: -D *jm_port=0* -D *jm_sec_port=nnnnn* where *nnnnn* is the port number. The default number of *jm_sec_port* is 31114.

password (for Windows only)

Required for Windows operating systems when performing a first time install. The password associated with *TWS_user*.

tdwb_hostname

Optional. The Dynamic Workload Broker fully qualified host name. Used together with the **-tdwbport** *tdwbport_number* parameter. Adds to the Tivoli Workload Scheduler agent the capability to run dynamic workload. If not specified, the default value is **localhost**. This value is registered in the ResourceAdvisorUrl property in the JobManager.ini file.

tdwb_port

Optional. The Dynamic Workload Broker HTTP or HTTPS port number. Used together with the **-tdwb_hostname** *host_name* parameter. Adds to the Tivoli Workload Scheduler agent the capability to run dynamic workload. This number is registered in the ResourceAdvisorUrl property in the JobManager.ini file. The default value is **0**, meaning that the capability to run dynamic workload to the agent is not added. The valid range is from 0 to 65535.

tws_user

Required. The user name for which Tivoli Workload Scheduler instance is being installed. On Windows systems, if this user account does not already exist, it is automatically created. If you specify a domain user or domain controller, you must specify the domain in the *domain* variable. If you

specify a local user with the same name as a domain user, the local user must first be created manually by an administrator and then identified as `system_name\user_name`.

On UNIX and Linux, this user account must be created manually before running the installation.

upgrade

Required. Indicates if the install is an upgrade. To perform an upgrade, specify **true**. To perform a fresh installation, specify **false**. The default value is **false**.

Notes:

1. *fresh_install* and *upgrade* are mutually exclusive.
2. The variables that are not documented here are for debugging purposes only. See *Administration Guide*.

Installation procedure

To perform the installation, complete the following steps. The installation procedure checks that there is sufficient space for the Tivoli Workload Scheduler for z/OS Agent to be installed.

1. Create a software package profile:
`TWS_LWA_operating_system_TWS_user.8.6.0.00` where *operating_system* is the operating system where you are installing and *TWS_user* is the user of the installation.
2. Import the software package blocks using the **wimpspo** command. When you import the software package blocks, you must pass the name of the profile to **wimpspo** so that the Configuration Manager endpoint catalogs the name correctly.
3. Install the software package blocks using the **wdinstsp** command.

Note: The supplied software packages must be installed as COMMITTED. The packages cannot be installed as UNDOABLE because the UNDO action does not rollback the product registry entries.

Note: For complete instructions on performing these tasks, refer to **wimpspo** and **wdinstsp** in the *IBM Tivoli Configuration Manager, Reference Manual for Software Distribution*, and the *IBM Tivoli Configuration Manager, User's Guide for Software Distribution*.

Prerequisite: Installing the Common Inventory Technology (CIT)

You must install CIT before installing the Tivoli Workload Scheduler for z/OS Agent and optionally adding the following capabilities:

- Dynamic capabilities
- Runtime environment for Java, used to run both jobs supplied with the product or additional types implemented through the custom plug-ins

The following are examples of the commands you run to install CIT on UNIX, Linux, and Windows workstations. For a description of the parameters, see "Software packages and parameters" on page 281.

UNIX and Linux operating systems

1. **wdinstsp -D CIT_ExploiterID=TWA /TWS_86/UNIX/CIT_Preinstall.spb**
2. **wdinstsp /TWS_86/UNIX/CIT.spb**

Windows operating systems

1. **wdinstsp -D CIT_ExploiterID=TWA D:\TWS_86\WINDOWS\CIT_Preinstall.spb**

2. **wdinstsp** D:\TWS_86\WINDOWS\CIT.spb

Installing the Tivoli Workload Scheduler for z/OS Agent

This section describes how to install the Tivoli Workload Scheduler for z/OS Agent and add dynamic capabilities to it by using the **wdinstsp** command. If you add dynamic capabilities to the agent, you can also add the runtime environment for Java. The runtime environment is used to run job types with advanced options. See “Enabling dynamic capabilities” on page 285 and “Adding the Java runtime to run job types with advanced options” on page 285.

To add the Java runtime, perform the following steps:

1. Verify the authorizations required to run the procedure in the “User authorization requirements” on page 270 section.
2. Locate the .spb as described in the “Software packages and parameters” on page 281 section.
3. Set the Software Distribution environment launching the following command located in the TWA/TWS/_uninstall/CLI directory:

Windows operating systems:

swd_env.bat

UNIX and Linux operating systems:

swd_env.sh

4. Run the **wdinstsp** command located in the TWA/TWS/_uninstall/CLI as shown in the following examples:

See “Software packages and parameters” on page 281 for a description of the parameters.

UNIX and Linux operating systems:

The following example describes an installation with the user **twuser** and the endpoint **Tivoli_TWS_LINUX_I386**.

wdinstsp

```
-f
-uy
-D install_dir="/home/twsuser/TWS"
-D tws_user="twuser"
-D domain="null"
-D group="group_name"
-D installer="root"
-D jm_port="0"
-D jm_sec_port="31114"
-D host_name="IT041924-T61.rot.ibm.com"
-D display_name="IT041924-T61_1"
-D fresh_install="true"
-D upgrade="false"
-n "TWS_LWA_twsuser.8.6.0.00"
  /mnt/gsa/home/SPB_INSTALL/LINUX_I386/Tivoli_LWA_LINUX_I386.SPB
```

Windows operating systems:

The following Windows example describes an installation with the user **twuser** and the endpoint **Tivoli_TWS_WINDOWS**. In this example, you are installing on a domain controller or in a Windows node agent because the **-D domain="domain_name"** is specified.

wdinstsp

```
-f
-uy
-D install_dir="C:\ibm\TWS\twsuser\TWS"
-D tws_user="twuser"
-D password="twspasswd"
-D domain="domain_name"
```

```

-D group="group_name"
-D installer="Administrator"
-D jm_port="0"
-D jm_sec_port="31114"
-D host_name="IT041924-T61.rot.ibm.com"
-D display_name="IT041924-T61"
-D fresh_install="true"
-D upgrade="false"
-n "TWS_LWA_twsuser.8.6.0.00"
"C:\Output\TWS_VLAST\WINDOWS\Tivoli_LWA_WINDOWS.SPB"

```

Enabling dynamic capabilities

To enable dynamic capabilities to the agent, specify the **-D tdwb_port="31116"** - **D "tdwb_hostname=slutri2.romelab.it.ibm.com"** parameter to the **wdinstsp** command.

Adding the Java runtime to run job types with advanced options

The runtime environment is used to run job types with advanced options, both those supplied with the product and the additional types implemented through the custom plug-ins. The runtime environment also enables the capability to remotely run, from the agent, the Dynamic Workload Broker resource command on the server. To add the runtime environment for Java jobs to the agent, see “Adding Java runtime environment after installation or upgrade.”

Adding Java runtime environment after installation or upgrade

The Java runtime environment allows you to:

- Run job types with advanced options, both those types supplied with the product and the additional types implemented through the custom plug-ins.
- Enable the capability to remotely run, from the Tivoli Workload Scheduler for z/OS Agent, the dynamic workload broker resource command on the server.

To add the Java runtime, perform the following steps:

1. Verify the authorizations required to run the procedure in the “User authorization requirements” on page 270 section.
2. Locate the .spb as described in the “Software packages and parameters” on page 281 section.
3. Set the Software Distribution environment launching the following command located in the TWA/TWS/_uninstall/CLI directory:

Windows operating systems:

```
swd_env.bat
```

UNIX and Linux operating systems:

```
swd_env.sh
```

4. Run the **wdinstsp** command located in the TWA/TWS/_uninstall/CLI as shown in the following examples:

See “Software packages and parameters” on page 281 for a description of the parameters.

Windows operating systems:

The following Windows example describes a command example:

```

wdinstsp
-f
-uy
-D install_dir="C:\ibm\TWS\twsuser\TWS"
-D tws_user="twuser"
-D group="group_name"

```

Installing using Software Distribution

```
-D installer="Administrator"
-n "TWS_Eclipse_twsuser.8.6.0.00"
"C:\Output\TWS_VLAST\WINDOWS\Tivoli_Eclipse_WINDOWS.SPB"
```

UNIX and Linux operating systems:

The following UNIX example describes an installation with the user <TWS_user>:

```
wdinstsp
-f
-uy
-D install_dir="/home/twsuser/TWS"
-D tws_user="twsuser"
-D group="group_name"
-D installer="root"
-n "TWS_Eclipse_twsuser.8.6.0.00"
/mnt/gsa/home/s/l/user1/web/public/SPB_INSTALL/
LINUX_I386/Tivoli_Eclipse_LINUX_I386.SPB
```

Enabling dynamic capabilities after installation or upgrade

This section describes the procedure that you must perform to enable dynamic scheduling capabilities after you installed or upgraded the Tivoli Workload Scheduler for z/OS Agent, without enabling them:

1. Update the JobManager.ini configuration file located in:

UNIX and Linux operating systems:

tws_home/TWS/ITA/cpa/config/JobManager.ini

Windows operating systems:

tws_home\TWS\ITA\cpa\config\JobManager.ini

by assigning to the *tdwb_hostname* and *mdm_httpsport* variables contained in the ResourceAdvisorUrl property, the following values:

tdwb_hostname

The fully qualified host name of the workload broker server.

mdm_httpsport

The value that the **httpsPort** has on the master domain manager, as shown by the **showHostProperties** wastool. The default is 31116, which is the dynamic workload broker port number. The port is currently set to zero because at installation time you specified that you would not use the dynamic workload broker.

The **ResourceAdvisorUrl** property has the following syntax:

```
ResourceAdvisorUrl = https://tdwb_hostname:mdm_httpsport
/JobManagerRESTWeb/JobScheduler/resource
```

2. Start the Tivoli Workload Scheduler for z/OS Agent by running the following command from *TWS_home*:

UNIX and Linux operating systems:

StartUpLwa.cmd

Windows operating systems:

StartUpLwa

Chapter 18. Upgrading the Tivoli Workload Scheduler for z/OS Agent

This chapter describes how to upgrade Tivoli Workload Scheduler for z/OS Agent (z-centric) from version 8.5.1 to the current version. It contains the following sections:

- “Coexistence with previous versions”
- “User authorization requirements”
- “Upgrading notes”
- “Upgrading using the installation wizard” on page 288
- “Upgrading using a silent installation” on page 288
- “Upgrading using twsinst” on page 289
- “Upgrading using Software Distribution” on page 291
- “Adding Java runtime or enabling dynamic capabilities after upgrade” on page 295

Coexistence with previous versions

The current version of the Tivoli Workload Scheduler for z/OS Agent (z-centric) can be installed on any workstation containing a prior version, provided that the *TWS_user*, **JobManager** port, and installation path are different from those of the previous versions.

User authorization requirements

Check the authorization roles before beginning the upgrade procedure. For detailed information, see “User authorization requirements” on page 270.

Upgrading notes

Before upgrading the Tivoli Workload Scheduler for z/OS Agent, ensure that there are no jobs running on the agent.

If you are upgrading Tivoli Workload Scheduler for z/OS Agent from an installation where you did not install the dynamic capabilities or the Java runtime to run job types with advanced options, you cannot add them during the upgrade process. To add them, perform the procedure described in the following sections:

- “Adding Java runtime environment after installation or upgrade” on page 285
- “Enabling dynamic capabilities after installation or upgrade” on page 286

When the upgrade procedure is successful, it is not possible to roll back to the previous version. Rollback is possible only for upgrades that fail. Refer to the following sections for detailed instructions about how to upgrade the agent using the various installation methods:

- “Upgrading using the installation wizard” on page 288
- “Upgrading using a silent installation” on page 288
- “Upgrading using twsinst” on page 289
- “Upgrading using Software Distribution” on page 291

Upgrading using the installation wizard

Use the installation wizard to upgrade the Tivoli Workload Scheduler for z/OS Agent by satisfying the following objectives:

Use a graphical interface that guides the user through the upgrade

In interactive mode, the wizard guides you through the upgrade steps.

Manage both UNIX and Windows operating system workstations

It runs both on UNIX and Windows agents.

To upgrade the agent using the installation wizard, run the setup for the operating system on which you are upgrading:

UNIX and Linux operating systems:

From the *operating_system* directory, run:

SETUP.bin

Windows operating systems:

From the WINDOWS directory, run:

SETUP.exe

Alternatively, start the launchpad as follows and select the Tivoli Workload Scheduler for z/OS Agent installation:

UNIX and Linux operating systems:

From the root directory of the DVD, run `launchpad.sh`.

Windows operating systems:

From the root directory of the DVD, run `launchpad.exe`.

When the installation wizard is launched, follow the prompts to complete the upgrade process.

Upgrading using a silent installation

Use a silent installation to upgrade the Tivoli Workload Scheduler for z/OS Agent by satisfying the following objectives:

Use a method that runs unattended and in background

It uses a response file that you customize by adding all the configuration settings to be used during installation. Then, from a command line, running the setup command. Using this method the you can run the installation unattended and in the background.

Manage both UNIX and Windows operating system workstations

It runs both on UNIX and Windows agents.

To upgrade the agent using a silent installation, follow the procedure described in “Performing a silent installation” on page 273 with the appropriate response files:

UNIX and Linux operating systems:

TWS86_UPGRADE_ZCENTRIC_Agent_UNIX.txt

Windows operating systems:

TWS86_UPGRADE_ZCENTRIC_Agent_WIN.txt

Upgrading using twsinst

Use **twsinst** to upgrade Tivoli Workload Scheduler for z/OS Agent by satisfying the following objectives:

Save time, disk space, and RAM when upgrading the product

It performs the agent upgrade in 30% less time than the ISMP upgrade. It saves disk space and RAM because it is not Java-based.

Use a very simple command

It consists of a single line command.

Manage both UNIX and Windows operating system workstations

It runs both on UNIX and Windows agents.

For a list of the supported operating systems and requirements, refer to <http://www.ibm.com/support/docview.wss?rs=672&uid=swg27019747>.

Upgrading process

During the upgrade process, **twsinst** creates a file in the following directories for each of the installation steps:

UNIX and Linux operating systems:

`/user's_home/TWS`

Windows operating systems:

`C:\%Program Files%\IBM\TWA`

If you stop and restart the installation, the installation process starts from the installation step where it was stopped.

According to your operating system, to upgrade the Tivoli Workload Scheduler for z/OS Agent with **twsinst** perform the following steps:

UNIX and Linux operating systems

1. Insert the installation DVD related to your operating system.
2. From `DVD_root/TWS/operating_system`, run the **twsinst** script using the synopsis described in this section.

Windows operating systems

1. Insert the DVD related to your operating system.
2. Log in as administrator on the workstation where you want to upgrade the agent.
3. From the `DVD_root/TWS/operating_system` directory of the DVD, run the **twsinst** script using the synopsis described in this section.

Note: **twsinst** for Windows is a Visual Basic Script (VBS) that you can run in CScript and WScript mode.

A successful upgrade using the **twsinst** issues the return code RC = 0. A failed upgrade issues the return code RC = 1. In the case of a failed installation, refer to the installation messages documented in *Tivoli Workload Scheduler: Planning and Installation Guide*.

Synopsis:

UNIX and Linux operating systems:

Upgrading agents with twsinst

Show command usage and version

```
twsinst -u | -v
```

Upgrade an instance

```
twsinst -update -uname user_name  
[-backup_dir backup_dir]  
[-inst_dir install_dir]  
[-lang lang-id]  
[-nobackup]  
[-reset_perm]  
[-skip_usercheck]
```

On Windows operating systems:

Show command usage and version

```
twsinst -u | -v
```

Upgrade an instance

```
twsinst -update -uname user_name  
-password user_password  
[-backup_dir backup_dir]  
[-domain user_domain]  
[-inst_dir install_dir]  
[-lang lang_id]  
[-nobackup]  
[-skip_usercheck]
```

-backup_dir *backup_dir*

An alternative directory (which must be created manually) as the destination for the backup copy of a previous version.

If you do not specify this option when running an upgrade, the following default value is used:

```
$BACKUP_DIR = $INST_DIR_backup_$TWS_USER
```

where:

- *\$INST_DIR* is the installation path (the user home directory on UNIX and Linux).
- *\$TWS_USER* is the user name.

For example:

```
$INST_DIR=/opt/TWS/TWS83  
$TWS_USER=user83  
$BACKUP_DIR=/opt/TWS/TWS83_backup_user82  
$BACKUP_SUBDIR=/opt/TWS/TWS83_backup_user83/TWS83
```

In the backup directory you must also create a subdirectory to include as the latest directory of the installation path.

-domain *user_domain*

Windows only. The domain name of the Tivoli Workload Scheduler for z/OS Agent user. The default is the name of the workstation on which you are upgrading the agent.

-inst_dir *install_dir*

The directory of the Tivoli Workload Scheduler for z/OS Agent installation. On UNIX this path cannot contain blanks. On window if you specify a path that contains blanks, enclose it in double quotes. If not specified, On UNIX the path is set to the *user_name* home directory, on Windows the path is set to %ProgramFiles%\IBM\TWA.

-lang

The language in which the twsinst messages are displayed. If not specified, the system LANG is used. If the related catalog is missing, the default C language catalog is used.

Note: The **-lang** option does not relate to the supported language packs. By default, all supported language packs are installed when you install using the **twsinst** script.

-nobackup

The upgrade process does not back up the instance you are upgrading.

-uname *username*

The name of the user for which Tivoli Workload Scheduler for z/OS Agent is being upgraded. The software is updated in this user's home directory. This user name is not to be confused with the user performing the upgrade. This user name is not to be confused with the user performing the installation logged on as **root** on UNIX and Linux, as **administrator** on Windows, and as **QSECOFR** on IBM i.

-update

Upgrades an existing agent that was installed using **twsinst**.

-password *user_password*

Windows only. The password of the user for which you are upgrading Tivoli Workload Scheduler for z/OS Agent.

-reset_perm

UNIX and Linux only. Reset the permissions of the **libatrc** library.

-skip_usercheck

Enable this option if the authentication process within your organization is not standard, thereby disabling the default authentication option. On UNIX and Linux, skip the check of the user in the `/etc/passwd` file or using the **su** command. On Windows, does not create the user you specified in the **-uname *username*** parameter. If you specify this parameter you must create the user manually before running the script.

Examples

To upgrade the agent installed in the user home directory that does not have the dynamic scheduling capabilities and the Java runtime to run job types with advanced options :

```
./twsinst -update -uname twsuser
```

Upgrading using Software Distribution

This section describes how to upgrade Tivoli Workload Scheduler for z/OS Agent using Software Distribution software package blocks.

Creating and installing the software package block

To create, import, and install the software package block (SPB), complete the following steps:

1. Create a software package profile that has the following name:

```
TWS_LWA_operating_system_TWS_user.8.6.0.00
```

where *operating_system* is the operating system where you are installing and *TWS_user* is the user of the installation.

Upgrading Tivoli Workload Scheduler for z/OS Agent using Software Distribution

When you import the software package block, you must pass the name of the profile to **wimpspo** so that the Configuration Manager endpoint catalogs the name correctly.

2. Import the software package block using the **wimpspo** command.
3. Install the software package block using the **wdinstsp** command.

Note: When upgrading using the **wdinstsp** command, make sure that you specify the *install_dir* variable. If you installed the previous version in a directory other than the default and you do not specify *install_dir*, Tivoli Workload Scheduler is installed as a fresh installation.

For complete instructions about performing these tasks, refer to the *IBM Tivoli Configuration Manager, Reference Manual for Software Distribution* and the *IBM Tivoli Configuration Manager, User's Guide for Software Distribution*.

Upgrading procedure overview

To upgrade the Tivoli Workload Scheduler for z/OS Agent, after installing the software package block, follow this procedure:

1. Install the Common Inventory Technology (CIT). See “Prerequisite: Install the Common Inventory Technology.”
2. Upgrade the Tivoli Workload Scheduler for z/OS Agent to version 8.6.

Some Tivoli Workload Scheduler parameters are used by the software package block to perform the upgrade. You can assign values to each variable to reflect the installation that is being upgraded, otherwise the default value is assigned.

When you upgrade agents using Software Distribution, the following variables are required:

- *install_dir*
- *tws_user*
- *pwd* (This parameter is not required on UNIX and Linux.)
- *fresh_install*
- *upgrade*
- *from_release*

For a list of Software Distribution parameters, see “Software packages and parameters” on page 281.

Prerequisite: Install the Common Inventory Technology

Before upgrading the agent, you must install the current version of Common Inventory Technology (CIT).

The following are examples of the commands that you run to install CIT on Windows and UNIX workstations. See “Software packages and parameters” on page 281 for a description of the parameters.

UNIX and Linux operating systems:

1. **wdinstsp -D CIT_ExploiterID=TWA /TWS_86/UNIX/CIT_Preinstall.spb**
2. **wdinstsp /TWS_851/UNIX/CIT.spb**

Windows operating systems:

1. **wdinstsp -D CIT_ExploiterID=TWA D:\TWS_86\WINDOWS\CIT_Preinstall.spb**
2. **wdinstsp D:\TWS_851\WINDOWS\CIT.spb**

Upgrading the agent

The following is an example of the settings required to upgrade the Tivoli Workload Scheduler for z/OS Agent version 8.5.1 on Windows and UNIX workstations. See “Software packages and parameters” on page 281 for a description of the parameters.

1. Verify the authorizations required to run the procedure in the “User authorization requirements” on page 270 section.
2. Locate the .spb as described in the “Software packages and parameters” on page 281 section.
3. Set the Software Distribution environment launching the following command located in the TWA/TWS/_uninstall/CLI directory:

Windows operating systems:

```
swd_env.bat
```

UNIX and Linux operating systems:

```
swd_env.sh
```

4. Run the **wdinstsp** command located in the TWA/TWS/_uninstall/CLI as shown in the following examples:

See “Software packages and parameters” on page 281 for a description of the parameters.

UNIX and Linux operating systems:

The following UNIX example describes an upgrade with the user **twuser** and the endpoint **Tivoli_TWS_LINUX_I386**.

```
wdinstsp
-f
-uy
-D install_dir="/home/twsuser/TWS"
-D tws_user="twuser"
-D domain="null"
-D group="group_name"
-D installer="root"
-D jm_port="0"
-D jm_sec_port="31114"
-D host_name="IT041924-T61.rot.ibm.com"
-D display_name="IT041924-T61_1"
-D fresh_install="false"
-D upgrade="true"
-n "TWS_LWA twsuser.8.6.0.00"
/mnt/gsa/home/SPB_INSTALL/LINUX_I386/Tivoli_LWA_LINUX_I386.SPB
```

Windows operating systems:

The following Windows example describes an upgrade with the user **twuser** and the endpoint **Tivoli_TWS_WINDOWS**. In this example, you are upgrading on a domain controller or in a Windows node agent because the **-D domain="domain_name"** is specified.

```
wdinstsp
-f
-uy
-D install_dir="C:\ibm\TWS\twsuser\TWS"
-D tws_user="twuser"
-D password="twspasswd"
-D domain="domain_name"
-D group="group_name"
-D installer="Administrator"
-D jm_port="0"
-D jm_sec_port="31114"
-D host_name="IT041924-T61.rot.ibm.com"
-D display_name="IT041924-T61"
-D fresh_install="false"
```

```
-D upgrade="true"
-n "TWS_LWA_twsuser.8.6.0.00"
"C:\Output\TWS_VLAST\WINDOWS\Tivoli_LWA_WINDOWS.SPB"
```

Upgrading the Java runtime to run job types with advanced options

The following are examples of the settings required to add the Java runtime to run job types with advanced options to the agent. The runtime environment is used to:

- Run on the agent, job types with advanced options, both those supplied with the product and the additional types implemented through the custom plug-ins.
 - Enable the capability to remotely run, from the agent, the Dynamic Workload Broker resource command on the server.
1. Verify the authorizations required to run the procedure in the “User authorization requirements” on page 270 section.
 2. Locate the .spb as described in the “Software packages and parameters” on page 281 section.
 3. Set the Software Distribution environment launching the following command located in the TWA/TWS/_uninstall/CLI directory:

Windows operating systems:

```
swd_env.bat
```

UNIX and Linux operating systems:

```
swd_env.sh
```

4. Run the **wdinstsp** command located in the TWA/TWS/_uninstall/CLI as shown in the following examples:

See “Software packages and parameters” on page 281 for a description of the parameters.

Windows operating systems:

The following example describes an installation with the user **twsuser**.

wdinstsp

```
-f
-uy
-D install_dir="C:\ibm\TWS\twsuser\TWS"
-D tws_user="twsuser"
-D group="group_name"
-D installer="Administrator"
-n "TWS_Eclipse_twsuser.8.6.0.00"
"C:\Output\TWS_VLAST\WINDOWS\Tivoli_Eclipse_WINDOWS.SPB"
```

UNIX and Linux operating systems:

The following example describes an installation with the user **twsuser**.

wdinstsp

```
-f
-uy
-D install_dir="/home/twsuser/TWS"
-D tws_user="twsuser"
-D group="group_name"
-D installer="root"
-n "TWS_Eclipse_twsuser.8.6.0.00"
"/mnt/gsa/home/public/SPB_INSTALL/LINUX_I386/
Tivoli_Eclipse_LINUX_I386.SPB"
```

Adding Java runtime or enabling dynamic capabilities after upgrade

To add the Java runtime environment or enable dynamic scheduling after you upgraded the Tivoli Workload Scheduler for z/OS Agent without adding or enabling it, refer to the following sections:

- “Adding Java runtime environment after installation or upgrade” on page 285
- “Enabling dynamic capabilities after installation or upgrade” on page 286

Chapter 19. Uninstalling the Tivoli Workload Scheduler for z/OS Agent

This chapter describes how you uninstall the Tivoli Workload Scheduler for z/OS Agent. It is divided into the following sections:

- “User authorization requirements”
- “Uninstalling using the wizard”
- “Performing a silent uninstallation” on page 298
- “Uninstalling the Tivoli Workload Scheduler for z/OS Agent using the `twsinst` script” on page 298
- “Uninstalling using the Software Distribution CLI” on page 299

The uninstaller program is created during the installation procedure. Wherever possible, use the same method you chose to install the agent when uninstalling it. For example, if you installed the agent using the installation wizard, use the **uninstaller** program to subsequently remove it.

The Tivoli Workload Scheduler for z/OS Agent on IBM i can be uninstalled using only the **twsinst** script.

Uninstalling the agent does not remove files created after the agent was installed, nor files that are open at the time of uninstallation. If you do not need these files, you must remove them manually. If you intend to reinstall and therefore need the files, make a backup before starting the installation process.

User authorization requirements

Check the authorization roles before beginning the uninstallation procedure. See “User authorization requirements” on page 270.

Uninstalling using the wizard

The **uninstaller** program removes product files, registry keys, and services. It also removes the binaries related to the Tivoli Workload Scheduler agent installed, the distributed connector, and the language packs.

To uninstall Tivoli Workload Scheduler for z/OS Agent, perform the following steps:

1. Ensure that all processes and services are stopped, and that there are no active or pending jobs.
2. Navigate to the *twshome* path.
3. Run the uninstall script:
 - On Windows operating systems:
`uninstaller.exe`
 - On UNIX and Linux operating systems:
`./uninstall.bin`
4. Select the instance you want to uninstall:

Performing a silent uninstallation

For a silent uninstallation, perform the following steps:

1. Ensure that all processes and services are stopped, and that there are no active or pending jobs.
2. Navigate to the `/TWA/TWS/_uninstall` path.
3. Enter the following command:
 - On Windows operating systems:
`uninstaller.exe -silent`
 - On UNIX and Linux operating systems:
`./uninstall.bin -silent`

Note: If you want to reinstall after performing a silent uninstallation, you must first close and reopen the shell to correctly reset the environment variables.

Uninstalling the Tivoli Workload Scheduler for z/OS Agent using the `twsinst` script

Follow these steps to uninstall the Tivoli Workload Scheduler for z/OS Agent using the **twsinst** script. Only agents installed using **twsinst** can be uninstalled using **twsinst**. Depending on the operating system, proceed as follows:

- On UNIX, Linux, and IBM i operating systems:
 1. Ensure that all processes and services are stopped, and that there are no active or pending jobs. For information about stopping the processes and services, see *Administration Guide*.
 2. Log on as root and change your directory to `/installation_dir/TWS` (for example: `/home/user1/TWS` where `user1` is the name of Tivoli Workload Scheduler user.)
 3. From the TWS directory, run the **twsinst** script as follows:

```
twsinst -uninst -uname username [-wait minutes]  
[-lang lang_id]
```

The uninstall is performed in the language of the locale and not the language set during the installation phase. If you want to uninstall agents in a language other than the locale of the computer, run the **twsinst** script from the `/installation_dir/TWS` (for example, `/home/user1/TWS`) as follows:

```
./twsinst -uninst -uname user_name -lang language
```

where *language* is the language set during the uninstallation.

- On Windows operating systems:
 1. Ensure that all Tivoli Workload Scheduler processes and services are stopped, and that there are no active or pending jobs.
 2. Log on as administrator on the workstation where you want to uninstall the product.
 3. From the `installation_dir\TWS` (for example, `c:\Program Files\IBM\TWA`), run the **twsinst** script as follows:

```
twsinst -uninst -uname username [-wait minutes]  
[-domain domain_name] [-lang lang_id]
```

Note: **twsinst** for Windows is a Visual Basic Script (VBS) that you can run in CScript and WScript mode.

The uninstallation is performed in the language of the locale and not the language set during the installation phase. If you want to uninstall agents in a language other than the locale of the computer, run the **twsinst** script from the */installation_dir/TWS* (for example, */home/user1/TWS*) as follows:

```
twsinst -uninst -uname user_name -lang language
```

where *language* is the language set during the uninstallation.

-uninst

Uninstalls the agent

-uname username

The name of the user for which the agent is uninstalled. This user name is not to be confused with the user performing the installation logged on as **root** on UNIX and Linux, as **administrator** on Windows, and as **QSECOFR** on IBM i.

-wait minutes

The number of minutes that the product waits for jobs that are running to complete before starting the uninstallation. If the jobs do not complete during this interval the uninstallation stops and an error message is displayed. Valid values are integers or **-1** for the product to wait indefinitely. The default is **60**.

-domain domain_name

Windows only. The domain name of the Tivoli Workload Scheduler user. The default is the name of the workstation on which you are uninstalling the product.

-lang lang_id

The language in which the **twsinst** messages are displayed. If not specified, the system LANG is used. If the related catalog is missing, the default C language catalog is used.

Note: The **-lang** option is not to be confused with the Tivoli Workload Scheduler supported language packs.

The following is an example of a **twsinst** script that uninstalls the Tivoli Workload Scheduler agent, originally installed for user named **twsuser**:

On UNIX, Linux, and IBM i operating systems:

```
./twsinst -uninst -uname TWS_user
```

On Windows operating systems:

```
twsinst -uninst -uname TWS_user
```

Uninstalling using the Software Distribution CLI

You can uninstall Tivoli Workload Scheduler for z/OS Agent using a Software Distribution/Configuration Manager command. To uninstall a software package from a disconnected target, use the command **wdrmvsp**. Tivoli Workload Scheduler uses the disconnected catalog.

Ensure that all Tivoli Workload Scheduler processes and services are stopped, and that there are no active or pending jobs. For information about stopping the processes and services see *Administration Guide*.

For example, to uninstall the Tivoli Workload Scheduler for z/OS Agent on UNIX, run the following command:

```
conman "stop;wait"
conman "shut;wait"
```

Uninstalling using the Software Distribution CLI

```
|           Ensure all processes are stopped.
|
|           As root:
|           cd <twshome>/_uninstall/CLI. ./swd_env
|
|           To display package names and versions: wdlssp
|
|           wdrmvsp -f packagename.version
|
|           Using the same procedure, you can also remove the software package block that
|           installs language packs, the Java runtime to run job types with advanced options,
|           and the dynamic agent.
```

Appendix A. Sample library (SEQQSAMP)

The SEQQSAMP library contains samples to help you install, migrate, and customize Tivoli Workload Scheduler for z/OS. In most cases, you need only add installation-specific JCL to adapt a member in SEQQSAMP to your requirements. Table 49 lists all members in the SEQQSAMP library and provides a brief description of each member. The following pages describe the samples relating to installing Tivoli Workload Scheduler for z/OS in more detail. Descriptions of other sample-library members are included in the book that describes the function demonstrated by the sample. For example, program-interface samples are described in *Tivoli Workload Automation: Developer's Guide: Driving Tivoli Workload Scheduler for z/OS*, SC32-1266

Some of the samples provided address a specific function and you might be able to use the sample unchanged in your environment. If you need to change a sample member, it is advisable to copy the source to a separate library. The original sample member is then available for reference. It is also recommended that you create an SMP/E *usermod* for each sample member you run in the production environment. Changes to the sample source code will then be flagged for your attention, and subsequent updates can be reflected in the production code as soon as possible.

Table 49. SEQQSAMP library members

Member	Brief description
EQQ9RF01	Sample RACF router table entry to enable security environment
EQQ9RFDE	Sample RACF class descriptor entry to enable security environment
EQQ9SM01	JCL to install RACF router table update
EQQ9SMDE	JCL to install RACF class descriptor update
EQQACPTx	Sample SMP/E ACCEPT JCL for the controller software, where the value of <i>x</i> depends on the language
EQQACTR1	Sample SMF exit IEFACTRT, written in assembler, to enable job-tracking
EQQAIXST	Parameters used by the EQQX9AIX and EQQAIXTR samples
EQQAIXTR	Sample tracker running on AIX, used with EQQX9AIX
EQQBENCR	Sample EQQE2EPW JCL to run the utility that encrypts the Windows passwords set in the USRPSW parameter of the USRREC statements.
EQQDDDEF	Sample job to allocate DDDEFs in SMP/E
EQQALLOC	JCL to allocate the Tivoli Workload Scheduler for z/OS distribution and target libraries.
EQQALSMP	Sample JCL to allocate and initialize the SMP/E environment needed to install Tivoli Workload Scheduler for z/OS
EQQAPISM	ASCII file containing a sample API application
EQQAPPLx	Sample SMP/E APPLY JCL for the controller software, where the value of <i>x</i> depends on the language
EQQAUDIB	Sample to invoke EQQAUDIT in batch mode outside of the dialog Note: EQQAUDIB can be used successfully only if the EQQTROUT dsname and the EQQAUDIT output dsn fields in the EQQJOBSA panel are filled out.
EQQBSCAN	Batch loader sample to validate an application description

Sample library (SEQQSAMP)

Table 49. SEQQSAMP library members (continued)

Member	Brief description
EQQBSubS	Batch loader sample to create four application descriptions and two operator instructions. Output is directed to a subsystem.
EQQBVSAM	Batch loader sample to create an application description and two operator instructions. Output is directed to a VSAM data set that is allocated by the sample.
EQQCHKEV	A sample JCL to display EQQTWSIN and EQQTWSOU event data set content information
EQQCLEAN	Sample procedure invoking EQQCLEAN program
EQQCONOP	Sample parameters used by EQQCONO
EQQCONO	Sample started task procedure for controller only
EQQCONP	Sample initial parameters for a controller and tracker in the same address space
EQQCON	Sample started task procedure for a controller and tracker in the same address space
EQQCVM2	Sample to enable submission and tracking on VM systems using EQQUX009
EQQCVM	Sample to enable job-tracking facilities on VM systems
EQQDBENC	Contains the JCL to encrypt the password in the DBOPT statement
EQQDBOPT	Sample DBOPT statement
EQQDELDI	JCL and usage notes for the data set deletion function
EQQDLFX	Assembler installation sample of DLF connect/disconnect exit
EQQDPCOP	JCL and usage notes for copy VSAM function
EQQDPX01	DP batch sample user exit to update the scheduling environment
EQQDSCL	Batch Clean Up sample
EQQDSCLP	Batch Clean up sample parameters
EQQDSECT	Assembler version of PIF data areas
EQQDSEX	Batch Export sample
EQQDSEXP	Batch Export sample parameters
EQQDSIM	Batch Import sample
EQQDSIMP	Batch Import sample parameters
EQQDSRG	Batch sample reorg
EQQDSRI	Batch Recovery index
EQQDSRIP	Batch Recovery index parameters
EQQDST	Sample procedure to start Data Store
EQQDSTP	Parameters for sample procedure to start Data Store
EQQE2EP	Sample initial parameters for server and batch to define if the end-to-end scheduling with fault tolerance capabilities is active
EQQICNVH	Sample job to migrate history DB2 tables
EQQICNVS	Sample job to migrate VSAM files
EQQINIDB	Sample to create the history data base
EQQISMKD	Sample job to run EQQMkdir exec for directories
EQQJCCTB	JCL to assemble a JCC message table macro definition
EQQJCLIN	Sample JCL to start program EQQPDFL
EQQJER2U	Sample to restore the EXIT7 as a JES2 usermod
EQQJER2V	Sample to restore the EXIT5 as a JES2 usermod
EQQJER3U	Sample to restore the EQQUX191 and EQQUX291 as JES3 usermods

Table 49. SEQQSAMP library members (continued)

Member	Brief description
EQQJES2	JCL to assemble and link-edit a JES2 exit
EQQJES21	JCL to assemble and link-edit the JES2 EXIT51
EQQJES2U	JCL to install the JES2 EXIT7 as an SMP/E usermod
EQQJES2V	JCL to install the JES2 EXIT51 as an SMP/E usermod
EQQJES3	JCL to assemble and link-edit a JES3 exit
EQQJES3U	JCL to install a JES3 exit as an SMP/E usermod
EQQJVXIT	Sample assembler JCL-variable-substitution exit. Also used for variable substitution in System Automation commands
EQQLSJCL	Sample JCL to invoke the EQQLSENT macro
EQQMKDIR	Sample exec to create directories
EQQNCFCF	Sample parameters for an SNA connection between controller and tracker
EQQNETW1	REXX EXEC that receives Tivoli Workload Scheduler for z/OS WTO messages and issues MVS commands
EQQNETW2	PL/I NetView command processor that uses EQQUSINT to change the status of operations
EQQNETW3	REXX EXEC that uses EQQEVPGM to change the status of operations
EQQOCWTO	Sample job to assemble and linkedit the IPOWTO routine used by the PIF REX sample
EQQORST	Resets the USS environment for the end-to-end scheduling with fault tolerance capabilities
EQQOS2ST	Parameters used by the EQQX9OS2 and EQQOS2TR samples
EQQOS2TR	Sample tracker running on OS/2, used with EQQX9OS2
EQQPCS01	Allocates data sets that need to be unique within the SYSPLEX
EQQPCS02	Allocates data sets that need to be unique to each MVS image in the SYSPLEX
EQQPCS03	Generates a job that allocates VSAM copy data sets
EQQPCS04	Defines Data Store VSAM files and initializes them
EQQPCS05	Allocates files used by a controller to enable fault-tolerant workstations
EQQPCS06	Allocates VSAM data sets for integration with the end-to-end scheduling with fault tolerance capabilities
EQQPCS07	Allocates VSAM data sets for Restart and Cleanup
EQQPIFAD	Program-interface PL/I sample that creates a two-operation application in the AD database
EQQPIFAP	Program-interface PL/I sample that resolves JCL variables
EQQPIFCB	Program-interface assembler samples for various current plan or long-term plan actions
EQQPIFCL	Program-interface assembler sample that uses the DAYSTAT command to return work or free status for a particular date
EQQPIFDJ	Program-interface assembler sample, deletes JCL for completed occurrences from JS data set
EQQPIFJC	Program-interface COBOL sample to manipulate JCL variable tables
EQQPIFJD	Program-interface PL/I sample that can either list or delete records in the JCL repository data set (JS)
EQQPIFJV	Program-interface PL/I sample to manipulate JCL variable tables
EQQPIFJX	Sample to maintain the JCL repository
EQQPIFOP	Program-interface REXX sample to modify an operation in the current plan
EQQPIFPR	Program-interface REXX sample to list all cyclic periods
EQQPIFWI	Program-interface PL/I sample to modify capacity values in an open interval of a current plan workstation

Sample library (SEQQSAMP)

Table 49. SEQQSAMP library members (continued)

Member	Brief description
EQQPROC	Sample procedure, started by Tivoli Workload Scheduler for z/OS, to initiate purge of DLF objects
EQQRECV x	Sample SMP/E RECEIVE JCL for the controller software, where the value of x depends on the language
EQQRETWT	Sample program to simulate abends, return codes and waits
EQQRMD5	Usage notes for the job-log-retrieval exit object code to interface to RMD5
EQQRXSTG	An assembler routine to get and free storage for the REXX program-interface samples
EQQSAMPI	JCL to load sample data for application descriptions, operator instructions, and workstation descriptions to the databases
EQQSERP	Sample initial parameters for a Server
EQQSER	Sample started task procedure for a Server
EQQSLCHK	JCL to perform a syntactic check on SCRIPT library members Note: EQQSLCHK sample JCL is generated with the following DD card: EQQMLOG DD SYSOUT=*" <p>If the EQQMLOG ddname is associated with a physical data set that has not sufficient size , a D37 abend followed by a user abend U4036 might occur. In this case, you must reallocate the EQQMLOG data set with more space.</p> <p>To recreate a new EQQSLCHK sample JCL, run again option 1 (Create sample job JCL) of EQQJOBS</p>
EQQSMF	JCL to assemble and install the SMF exits
EQQTCPC	Sample definitions for TCP/IP communication between tracker and controller.
EQQTRAP	Sample initial parameters for a Tracker
EQQTRA	Sample started task procedure for a Tracker
EQQTROPT	Sample TRGOPT statement
EQQXML01	Sample XML file for data set triggering event rule definitions
EQQU831	Sample SMF exit IEFU83 to enable job tracking and optionally include data set triggering support
EQQUJI1	Sample SMF exit IEFUJI to enable job-tracking
EQQUSIN1	EQQUSIN subroutine sample to change the status of an operation
EQQUSIN2	EQQUSIN subroutine sample to change the availability of a special resource
EQQUSIN3	EQQUSIN subroutine sample to change the status of a workstation
EQQUSIN4	EQQUSIN subroutine sample to backup a Tivoli Workload Scheduler for z/OS resource data set
EQQUSIN5	EQQUSIN subroutine sample to update the USERDATA field of an operation.
EQQUX001	Sample job-submit exit
EQQUX002	Sample job-library-read exit
EQQUX004	Sample event-filtering exit
EQQUX011	Sample job-tracking log write exit
EQQUX013	Sample job-tailoring prevention exit
EQQUX0N	Sample PL/I start/stop exit, EQQUX000
EQQUX191	Sample JES3 exit IATUX19 to enable job tracking
EQQUX291	Sample JES3 exit IATUX29 to enable job tracking

Table 49. SEQQSAMP library members (continued)

Member	Brief description
EQQUX9N	Sample PL/I operation-initiation exit, communicating with VM (EQQUX009)
EQQUXCAT	Sample restart and clean up exit for the EQQCLEAN program
EQQUXPIF	Sample user exit to validate application descriptions
EQQUXSAZ	Sample assembler system command exit, communicating with System Automation invoked in place of EQQUX007 for automation workstations
EQQVTAMN	Sample VTAM definition for SNA connection between tracker and controller
EQQVTAMS	Sample VTAM definition for server SNA connection
EQQX5ASM	Sample SYSOUT archiving exit
EQQX6ASM	Sample incident-record-create exit
EQQX6JOB	Sample batch-job skeleton JCL used by EQQX6ASM
EQQX7ASM	Sample change-of-status exit
EQQX7JOB	Sample batch-job skeleton JCL used by EQQX7ASM
EQQX9AIX	Sample assembler operation-initiation exit, communicating with AIX
EQQX9OS2	Sample assembler operation-initiation exit, communicating with OS/2
EQQXCFCF	Sample definitions for XCF connection between tracker and controller
EQQXIT51	Sample JES2 EXIT51 to enable job tracking for JES2 with z/OS version 1 release 7, and later
EQQXIT74	Sample JES2 EXIT7 to enable job tracking for JES2 level version 4 release 1 and later
EQQYCBAG	Sample to unload a group application (and all the applications belonging to it) into a sequential file in Batch Loader Control statement format
EQQYCBAT	Run the Batch Command Interface tool
EQQYRJCL	Sample JCL to run the Control Language tool
EQQYRMSG	Messages used by the Control Language tool
EQQYRPRC	Sample procedure to run the Control Language tool
EQQYRPRM	Sample initialization parameter file for the Control Language tool

Using the Visual Age compiler

With the z/OS operating system, the Visual Age PL/I compiler replaces all the previous PL/I compilers. Therefore, if you use this compiler, you need to customize the samples in PL/I as follows:

1. Replace the PL/I compiler invocation statement:

```
EXEC PGM=IEL0AA
```

with:

```
EXEC PGM=IBMZPLI
```

2. Link into a PDS/E data set for SYSLMOD or include a pre-link edit step in the JCL.

As an example, here is the JCL for the EQQPIFJV sample using the Visual Age PL/I compiler:

```
//EQQPIFJV JOB MSGCLASS=N, .....
//PLI1     EXEC PGM=IBMZPLI,REGION=1024K,
//          PARM='OBJECT,OPTIONS'
//STEPLIB DD DSN=IBMZ.V2R2M1.SIBMZCMP,DISP=SHR
//SYSPRINT DD SYSOUT=*
```

Using Visual Age compiler

```
//SYSLIN DD UNIT=SYSDA,SPACE=(CYL,(2,1)),DISP=(,PASS),
// DSN=&&OBJ1
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(3,3))
//SYSIN DD *
/*
/*
//PLI2 EXEC PGM=IBMZPLI,REGION=1024K,
// COND=(4,LT,PLI1),PARM='OBJECT,OPTIONS'

//SYSPRINT DD SYSOUT=*
//STEPLIB DD DSN=IBMZ.V2R2M1.SIBMZCMP,DISP=SHR
//SYSLIN DD UNIT=SYSDA,SPACE=(CYL,(2,1)),
// DISP=(,PASS),DSN=&&OBJ2
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(3,3))
//SYSIN DD *
.....
/*
/*
//*****
/* PRE-LINK-EDIT STEP *
//*****
//PLKED EXEC PGM=EDCPRLK,COND=(8,LT,PLI1),
// REGION=2048K
//SYSDEFSD DD DSN=&&DEF1,LRECL=80,BLKSIZE=3200,
// DISP=(,PASS)
//STEPLIB DD DSN=CEE.SCEERUN,DISP=SHR
//SYMSGS DD DSN=CEE.SCEEMSGP(EDCPMSG),DISP=SHR
//SYSLIB DD DUMMY
//SYSMOD DD DSN=&&PLNK,DISP=(,PASS),
// UNIT=SYSALLDA,SPACE=(CYL,(1,1)),
// DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSIN DD DSN=&&OBJ1,DISP=(OLD,DELETE)
// DD DSN=&&OBJ2,DISP=(OLD,DELETE)
//SYSPRINT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//*****
/* SCEELKED ADDED TO SYSLIB ON LINK STEP *
//*****
//LKED EXEC PGM=IEWL,PARM='XREF',
// COND=(4,LT,PLI2),REGION=4M
//SYSPRINT DD SYSOUT=*
//SYSLIB DD DISP=SHR,DSN=CEE.SCEELKED
// DD DISP=SHR,DSN=USER.OPC23.LINKLI
//SYSLMOD DD DISP=SHR,DSN=SVIOLA.SEQQLMD0
//OPCLIB DD DISP=SHR,DSN=USER.OPC23.LINKLI
//SYSUT1 DD UNIT=SYSDA,SPACE=(CYL,(3,3))
//SEQOBJ1 DD DISP=(OLD,DELETE),DSN=&&PLNK
//SYSLIN DD *
INCLUDE SEQOBJ1
INCLUDE OPCLIB(EQQYCOM)
SETCODE AC(1)
ENTRY CEESTART
NAME EQQPIFT(R)
/*
/*
//EQQPIFT EXEC PGM=EQQPIFT,PARM='NOSTAE,NOSPIE',
// COND=(4,LT,LKED), REGION=4096K
//STEPLIB DD DISP=SHR,DSN=SVIOLA.SEQQLMD0
// DD DISP=SHR,DSN=USER.OPC23.LINKLIB
//EQQMLIB DD DSN=EQQ.V2R3M0.SEQQMSG0,DISP=SHR
//EQQYPARM DD DISP=SHR,DSN=XXXX.YYYY.ZZZZ(YPARM)
//EQQMLOG DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//EQQDUMP DD SYSOUT=*
//EQQMSG DD SYSOUT=*
```

```
//CARDIN DD *
.....
/*
//*
```

SMP/E samples

The following SEQQSAMP members relate to SMP/E processes.

Environment setup

You can use the sample library members EQQALSMP, EQQDDDEF, and EQQALLOC to create and initialize the SMP/E environment and the Tivoli Workload Scheduler for z/OS product libraries that are needed to support the installation and continuing maintenance of Tivoli Workload Scheduler for z/OS.

The EQQALSMP job performs the following functions:

- Initializes an SMP/E CSI, adding a global zone, and the Tivoli Workload Scheduler for z/OS FMID.

The EQQDDDEF job sets up DDDEFs for all Tivoli Workload Scheduler for z/OS data sets to provide for basic JCL requirements for RECEIVE, APPLY, and ACCEPT processing.

The EQQALLOC job allocates all Tivoli target and distribution libraries. The JCL also contains a number of steps, which are currently commented out. You can use those steps to delete the Tivoli Workload Scheduler for z/OS libraries if you need to reinstall the product.

RECEIVE processing

The sample library members EQQRECVE, EQQRECVS, EQQRECVJ, EQQRECVD, and EQQRECVK contain JCL that you can use to run SMP/E RECEIVE processing for Tivoli Workload Scheduler for z/OS data sets. These library members enable you to receive the following Tivoli Workload Scheduler for z/OS features:

- Tracker
- Controller
- z/OS connector enable
- End-to-end and Java enabler

Each of these jobs performs RECEIVE processing for a particular NLS feature:

EQQRECVE Receives all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the English feature for the controller (FMID JWSZ6A4).

EQQRECVS Receives all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604) plus the Spanish feature for the controller (FMID JWSZ6A1).

EQQRECVJ Receives all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Japanese feature for the controller (FMID JWSZ6A2).

EQQRECVD Receives all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the German feature for the controller (FMID JWSZ6A3).

EQQRECVK Receives all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Korean feature for the controller (FMID JWSZ6A5).

You might need to change the distribution library and zone name to reflect those defined in the Tivoli Workload Scheduler for z/OS CSI.

For further details, see the Tivoli Workload Scheduler for z/OS program directory.

Note: The program directory refers to trackers as *agents*, and to the controller as the *engine*.

APPLY processing

The sample library members EQQAPPLE, EQQAPPLS, EQQAPPLJ, EQQAPPLD, and EQQAPPLK contain JCL that you can use to run SMP/E APPLY processing for Tivoli Workload Scheduler for z/OS. These members enable you to apply the following features:

- Tracker
- Controller
- z/OS connector enable
- End-to-end and Java enabler

Each of these jobs performs APPLY processing for a particular NLS feature:

EQQAPPLE Applies all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the English feature for the controller (FMID JWSZ6A4).

EQQAPPLS Applies all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Spanish feature for the controller (FMID JWSZ6A1).

EQQAPPLJ Applies all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Japanese feature for the controller (FMID JWSZ6A2).

EQQAPPLD Applies all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the German feature for the controller (FMID JWSZ6A3).

EQQAPPLK Applies all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Korean feature for the controller (FMID JWSZ6A5).

You might need to change the distribution library and zone name to reflect those defined in the Tivoli Workload Scheduler for z/OS CSI.

For further details, see the Tivoli Workload Scheduler for z/OS program directory.

Note: The program directory refers to trackers as *agents*, and to the controller as the *engine*.

ACCEPT processing

The sample library members EQQACPTE, EQQACPTS, EQQACPTJ, EQQACPTD, and EQQACPTK contain JCL that you can use to run SMP/E ACCEPT processing for Tivoli Workload Scheduler for z/OS. These members enable you to accept the following features:

- Tracker
- Controller
- z/OS connector enable
- End-to-end and Java enabler

Each of these jobs performs ACCEPT processing for a particular NLS feature:

EQQACPTE	Accepts all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the English feature for the controller (FMID JWSZ6A4).
EQQACPTS	Accepts all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Spanish feature for the controller (FMIDJWSZ6A1).
EQQACPTJ	Accepts all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Japanese feature for the controller (FMIDJWSZ6A2).
EQQACPTD	Accepts all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the German feature for the controller (FMID JWSZ6A3).
EQQACPTK	Accepts all the scheduler base and tracker components (FMIDs HWSZ600, JWSZ602-JWSZ604), plus the Korean feature for the controller (FMIDJWSZ6A5).

You might need to change the distribution library and zone name to reflect those defined in the Tivoli Workload Scheduler for z/OS CSI.

For further details, see the Tivoli Workload Scheduler for z/OS program directory.

Note: The program directory refers to trackers as *agents*, and to the controller as the *engine*.

SMF exits

The following text provides details of the SEQQSAMP members relating to SMF exits.

Note: If version ASMA90 of the compiler reports errors, and the RMODE=ANY statement is defined, remove the RMODE=ANY statement from the sample exit.

Exit installation

The sample library member EQQSMF contains the JCL needed to assemble the SMF exits required for Tivoli Workload Scheduler for z/OS. The job also defines an SMP/E usermod to connect the SMF exits to your target zone.

A single usermod is used to define the three SMF exits. You can, if you prefer, define usermods for each exit.

To restore the JES exits as SMP/E usermods, use the samples EQQJER2U, EQQJER2V, and EQQJER3U.

Job step termination exit

The sample library member EQQACTR1 contains the assembler source code of an SMF job/step termination exit, IEFACTRT. The sample contains two subroutines:

- OPCASUB provides the necessary Tivoli Workload Scheduler for z/OS code to track job- and step-end events.
- LOCALSUB generates WTO messages for step- and job-end.

If you use the Tivoli Workload Scheduler for z/OS Restart and Cleanup functionality or other functions, such as the one related to the NOERROR table (described in the *Customization and Tuning* manual), you are required to install this exit. Use the sample provided with the product to install this exit.

From the introduction of the usability enhancement on, the IEFACTRT exit creates two different tables in the joblog, the Steptable and the Not_Executed_Step_Table.

If you use the Tivoli Workload Scheduler for z/OS Restart and Cleanup functionality or other functions, such as the one related to the NOERROR table, and you want to replace this subroutine with your own, you need to comply with the following restrictions:

- The fields JOBNAME, STEPNAME, PROCSTEP and STEPNO must continue to be filled on the basis of the following logic:
 - JOBNAME must contain the name of the job
 - STEPNAME is the label of the EXEC PROC=... Card and must be filled only if a PROC is used
 - PROCSTEP is the label of the EXEC PGM=... Card and must be filled also if a PROC is not used
 - STEPNO must contain the sequence number of the steps inside the job
- The JOBNAME, STEPNAME, PROCSTEP identifiers in the tables header must match the values specified in the *HDRJOBNAME*, *HDRSTEPNAME*, and *HDRPROCNAME* parameters of the DSTOPTS DATASTORE statement.
- The layout of STEPTABLE and NOT_EXECUTED_STEP_TABLE must be in compliance with the following rules:
 - JOBNAME must be preceded by a hyphen sign (-), some characters can be inserted between a hyphen sign and the JOBNAME
 - JOBNAME must be followed by a blank
 - STEPNAME must be preceded and followed by a blank
 - PROCSTEP must be preceded and followed by a blank
 - STEPNO must be preceded by a blank
 - STEPNO must follow the PROCSTEP in the NOT_EXECUTED_STEP_TABLE
- NOT_EXECUTED_STEP_TABLE must be aligned to STEPTABLE as far as it concerns JOBNAME, STEPNAME and PROCSTEP information.
- The string "JOBXXXXX ENDED. NAME-" must be aligned so that the JOBNAME JOBXXXXX is under the JOBNAME header.
- JOBNAME, STEPNAME, PROCSTEP position in the STEPTABLE and in the NOT_EXECUTED_STEP_TABLE must match the values specified in the *HDRJOBLENGTH*, *HDRSTEPLength*, *HDRPROCLength* parameters of the DSTOPTS DATASTORE statement. STEPNO position in the STEPTABLE must match the value specified in *HDRSTEPNOLENGTH*.
- User-customized records issued in the STEPTABLE and in the NOT_EXECUTED_STEP_TABLE must be avoided.

Initialization exit

The sample library member EQQUJI1 contains the assembler source code of an SMF initialization exit, IEFUJI. Tivoli Workload Scheduler for z/OS uses events generated from the exit to track job start information.

If your installation is already using an IEFUJI, incorporate the code into your existing exit and reassemble.

Record write exits

The sample library member EQQU831 contains the assembler source code of a record write exit, IEFU83. Tivoli Workload Scheduler for z/OS uses events generated from the exit to track print group and purge information.

If your installation is already using an IEFU83, incorporate the code into your existing exit and reassemble.

You can optionally include support for both the data set triggering and job-tracking functions using the EQQU831 sample. This provides you with a method to automatically generate a special resource availability depending on specific actions affecting data sets. The event can be used by Tivoli Workload Scheduler for z/OS to change the status of a special resource to make it available for operations and/or to trigger an application to be added to the current plan. You specify the data sets you want special resource availability events for using a specific macro, as described in Appendix D, “Invoking the EQQLSENT macro,” on page 335. See “Implementing support for data set triggering” on page 83 for more information about data set triggering. Use the EQQSMF sample to install EQQU831.

If you do not track print operations through Tivoli Workload Scheduler for z/OS, and you do not want to include data set triggering support, you need not change IEFU83.

JES exits

The following text provides details of the SEQQSAMP members relating to JES exits.

Note: If version ASMA90 of the compiler reports errors, and the RMODE=ANY statement is defined, remove the RMODE=ANY statement from the sample exit.

Exit installation

The sample library contains a number of members to assemble and link-edit JES exits. EQQJES2, EQQJES21, and EQQJES3 provide sample JCL to assemble and link-edit JES2 and JES3 exits respectively. However, it is recommended that you use members EQQJES2U, EQQJES2V, and EQQJES3U. These samples provide the JCL to install the JES exits as SMP/E usermods. The usermods are defined so that both the JES and Tivoli Workload Scheduler for z/OS target zones are informed of the dependencies. This ensures that future maintenance, to either the JES component or the Tivoli Workload Scheduler for z/OS component, will be handled correctly.

JES2 QMOD phase change exit

The sample library member EQQXIT51 contains the assembler source code of the JES2 QMOD Phase Change exit, JES EXIT51. This sample must be used with z/OS 1.7 or later. Tivoli Workload Scheduler for z/OS uses JES2 EXIT51 to detect job errors occurring during the JES2 input phase, and to trigger the creation of IJ2 events for started task.

JES2 JCT I/O exit

The sample library member EQQXIT74 contains the assembler source code of a JES2 JCT I/O exit, JESEXIT7. EQQXIT74 is used for JES2. Tivoli Workload Scheduler for z/OS uses JESEXIT7 to detect new jobs on the internal reader and also to detect output group purge.

If you are already using a JESEXIT7, and want to keep the Tivoli Workload Scheduler for z/OS job-tracking support in a separate load module, you can specify that JES use multiple EXIT7 modules in your JES2 parameters.

JES3 OSE modification exit

The sample library member EQQUX191 contains the assembler source code of a JES3 OSE modification exit, IATUX19. Tivoli Workload Scheduler for z/OS uses events generated from the exit to detect output group purge.

If you are already using an IATUX19, you should include the code in your existing exit and reassemble.

Note: If you are using JES3 Exit IATUX72 then this exit must return with R15 = 8 to call IATUX19.

JES3 input service final-user exit

The sample library member EQQUX291 contains the assembler source code of a JES3 input service final-user exit, IATUX29. Tivoli Workload Scheduler for z/OS uses events generated from the exit to detect new jobs on the internal reader.

If you are already using an IATUX29, then you should incorporate the code into your existing exit and reassemble.

RACF samples

The following text provides details of the SEQQSAMP members relating to RACF changes, which are required for Tivoli Workload Scheduler for z/OS security.

Class descriptor table

The sample library member EQQ9RFDE provides the class descriptor entry required to define the Tivoli Workload Scheduler for z/OS security environment to RACF, or a functionally equivalent product.

Use this sample if you are running RACF Release 1.7, 1.8, or 1.9. Each class descriptor contains control information needed by RACF to validate class names and is a CSECT in the load module ICHRRCDE.

You can use member EQQ9SMDE to install ICHRRCDE as an SMP/E usermod on the RACF target zone.

Router table

The sample library member EQQ9RF01 provides the router table entry required to define the Tivoli Workload Scheduler for z/OS security environment to RACF, or a functionally equivalent product.

Use this sample if you are running RACF Release 1.7, 1.8, or 1.9. This is a sample RACF router table that provides action codes to determine if RACF is invoked on behalf of the RACROUTE macro.

You can use member EQQ9SM01 to install ICHRF01 as an SMP/E usermod on the RACF target zone.

Sample library (SEQQSAMP)

The EQQYCBAG member of the EQQSAMP library provides a sample in which the Batch Command Interface Tool (BCIT) is used to unload a group application, and all applications belonging to it, into a sequential file in batchloader control statement format.

The group applications, as well as other applications, can be modified via the batchloader control statements. From then on, you can use the sequential file as input for the batchloader run.

This sample consists of two jobs:

1. The unload job, that uses the batch command interface tool
2. The load job, that uses the batchloader.

Before running the job, you need to customize it with correct values for the job card name, data set names, subsystem name, and so on.

The EQQBENCR member of the EQQSAMP library provides a sample of the EQQE2EPW JCL that you can use to encrypt the passwords written in plaintext in the USRREC statement of the USRINFO configuration member, or to insert additional USRREC statements through the SYSIN data.

Following is an example of the sample EQQE2EPW JCL.

```
//EQQE2EPW EXEC PGM=EQQUPTOP,REGION=64M,TIME=1440
//*****
/* THIS IS A SAMPLE JCL TO ENCRYPT THE PASSWORDS IN THE USRREC      *
/* STATEMENT CONTAINED IN THE EQQPARM LIBRARY MEMBER AS SPECIFIED *
/* BY THE USRMEM KEYWORD IN THE TOPOLOGY STATEMENT, FOR EXAMPLE   *
/* USRMEM(USRINFO).                                              *
/* THE TWS FOR ZOS DEFAULT FOR THIS MEMBER NAME IS USRINFO, AS    *
/* DEFINED IN THE EQQE2EP INSTALLATION SAMPLE.                    *
/* SPECIFY THE LIBRARY THAT CONTAINS THE USRINFO MEMBER,          *
/* INCLUDING THE MEMBER NAME, IN THE EQQSRIN DD OF THIS JCL.      *
/* SPECIFY IN THE SYSIN DD EITHER THE NAME OF A DATA SET (INCLUDING *
/* THE MEMBER NAME, IF PDS) CONTAINING THE USRREC STATEMENTS OR THE *
/* USRREC STATEMENTS DIRECTLY AS INLINE PARAMETERS. THESE ARE THE *
/* USRREC STATEMENTS THAT YOU WOULD LIKE TO ADD TO THE USRINFO    *
/* DATASET MEMBER.                                              *
/* NOTICE THAT ALL THE THREE KEYWORDS OF THE USRREC STATEMENT    *
/* (USRCPU, USRNAM, USRPSW) ARE REQUIRED IN THE SYSIN. INSERT ONE  *
/* USRREC STATEMENT KEYWORD PER ROW.                             *
/* FOR EXAMPLE:                                                  *
/*      SYSIN DD *                                              *
/*          USRCPU(WS01)                                         *
/*          USRNAM('TEST1')                                     *
```

Sample library (SEQQSAMP)

```
//*          USRPSW('ABC123')          *
//*          USRCPU(WS02)              *
//*          USRNAM('TEST2')          *
//*          USRPSW('EFG567')          *
//* AS RESULT THE PASSWORDS SPECIFIED IN THE USRPSW KEYWORDS WILL BE *
//* ENCRYPTED (EITHER IF IN THE SYSIN OR IN THE USRINFO MEMBER) AND *
//* THESE USRREC STATEMENTS ARE STORED IN THE USRINFO DATA SET AS *
//* SPECIFIED IN THE EQQUSRIN DD CARD. *
//* IF THE SYSIN IS NOT SPECIFIED, OR SPECIFIES DUMMY, ONLY THE *
//* PASSWORDS PRESENT IN THE USRINFO DATA SET WILL BE ENCRYPTED. *
//* NOTE: *
//* REVIEW ALL THE JCL CONTENT AND SETTINGS *
//* BEFORE SUBMITTING THIS JCL DOUBLE CHECK THAT THE DD DEFINITIONS *
//* SUIT YOUR INSTALLATION. MAINLY DOUBLE CHECK THAT EQQUSRIN AND *
//* SYSIN DD ARE PROPERLY DEFINED. CHECK THE JOB REGION SIZE. *
//* ADD YOUR JOB CARD. CONSIDER BACKING UP YOUR PARMLIB DATASET. *
//*****
//EQQE2EPW EXEC PGM=EQQUPTOP,REGION=64M,TIME=1440
//STEPLIB DD DISP=SHR,DSN=&STEPDSN
//EQQMLOG DD SYSOUT=&FPCLA
//EQQMLIB DD DISP=SHR,DSN=&MSGLIB
//EQQPARM DD DISP=SHR,DSN=&PARMDSN
//EQQDUMP DD SYSOUT=&FPCLA
//SYSUDUMP DD SYSOUT=&FPCLA
//EQQUSRIN DD DISP=SHR,DSN=&PARMDSN(USRINFO)
//SYSIN DD DISP=SHR,DSN=dsname(member_name)
//*SYSIN DD *
//*USRCPU(TEST)
//*USRNAM('DUMMY')
//*USRPSW('ABC123')
```

Notes:

1. Insert the keywords contained in SYSIN, either inline or in a data set, one per row.
2. The keywords are the same as the ones used for the USRREC statement: USRCPU, USRNAM, and USRPSW. These three keywords are all required in the SYSIN.
3. Rows containing only comments are inserted into the USRINFO data set member (pointed by the EQQUSRIN DD card) as they are, starting at column 13.
4. You can write comments on every row, but, depending on the statement length, they can result truncated or be overwritten by the row content. The suggested range is from column 50 to column 60. Rows containing only comments are allowed.
5. The password length after the encryption is always 31 bytes and the statements start at column 13, therefore you can use maximum 60 characters per row.
6. During the data set scanning process, if duplicated USRREC statements (same values for USRCPU and USRNAM) are found, the last USRREC found is inserted and the first USRREC is removed. The scan is performed from the top of the data set. The statements contained in SYSIN are considered more recent compared with the statements in USRINFO.
7. A light syntax checking is performed on the SYSIN data set content. Only few checks on the USRINFO data set. A complete syntax checking is performed on the content of the final USRINFO data set, when a DP batch or Symphony renew is performed, as usual.
8. New rows added to the USRINFO member are flagged with the /*JADD*/ comment starting at column 73. This will help to locate the modified lines. Following is an example of how the inserted rows look like:

```

USRREC USRCPU(WS01)                                /*JADD*/
    USRNAM('TEST01')                                /*JADD*/
    USRPSW('¿M($H7ggTDê;Dê ä ä7LN};ôä°Nä))|¿')    /*JADD*/

```

Remove the /*JADD*/ flag manually before the next run of job EQQE2EPW, to distinguish the new lines that will be added. To do this, edit the USRINFO member and do one of the following actions:

- Remove manually the comments /*JADD*/ using the edit command CUT of the host emulator.
- Use the TSO edit command CHANGE ALL:
CHANGE '/*JADD*/' ' ' ALL
- Use the TSO edit commands RENUM and UNNUM.

Sample library (SEQQSAMP)

Appendix B. Configuration examples

This appendix gives examples of Tivoli Workload Scheduler for z/OS configuration. The examples are based on z/OS JES2, but they are also valid for z/OS JES3 systems, or a combination of JES2 and JES3 systems. Each example shows:

- The controlling system, with the controller and the tracker started in separate address spaces
- All Tivoli Workload Scheduler for z/OS address spaces as Tivoli Workload Scheduler for z/OS systems
- A summary of actions that the workload restart function could take automatically
- Sample initialization statements that you can use to create the configuration
- The Tivoli Workload Scheduler for z/OS components that are required, the flow of automatic work submission, and event collection in various system combinations.

The controlling system

The controlling system is shown in the examples only with the controller and the tracker connected via either shared DASD or XCF. But you can connect them via NCF or TCP/IP, if you prefer this method.

Tivoli Workload Scheduler for z/OS can support remote systems that are in different time zones from the controlling system. Refer to *Tivoli Workload Scheduler for z/OS: Managing the Workload* for more information on time zone support and daylight saving time changes.

Automatic restart actions

The possible actions vary according to the type of connection between the controller and the tracker.

Initialization statements

Default values are used for statements that do not specifically relate to the configuration. The statements are specified in one or more parameter library members.

Multi-access spool systems connected through shared DASD

Figure 27 on page 318 shows two z/OS JES2 multi-access spool (MAS) complexes that are connected through shared DASD.

Systems A and B form a MAS complex. System A is the Tivoli Workload Scheduler for z/OS controlling system. It shares spool with System B, which is a controlled Tivoli Workload Scheduler for z/OS system. Work is sent directly to this complex by the controller on System A. The work is processed on one of these two systems, depending on installation parameters. You represent this complex to Tivoli Workload Scheduler for z/OS by defining a computer workstation with a blank destination field. That is, all work for this workstation is submitted to the system

Multi-access spool systems

that the controller is started on.

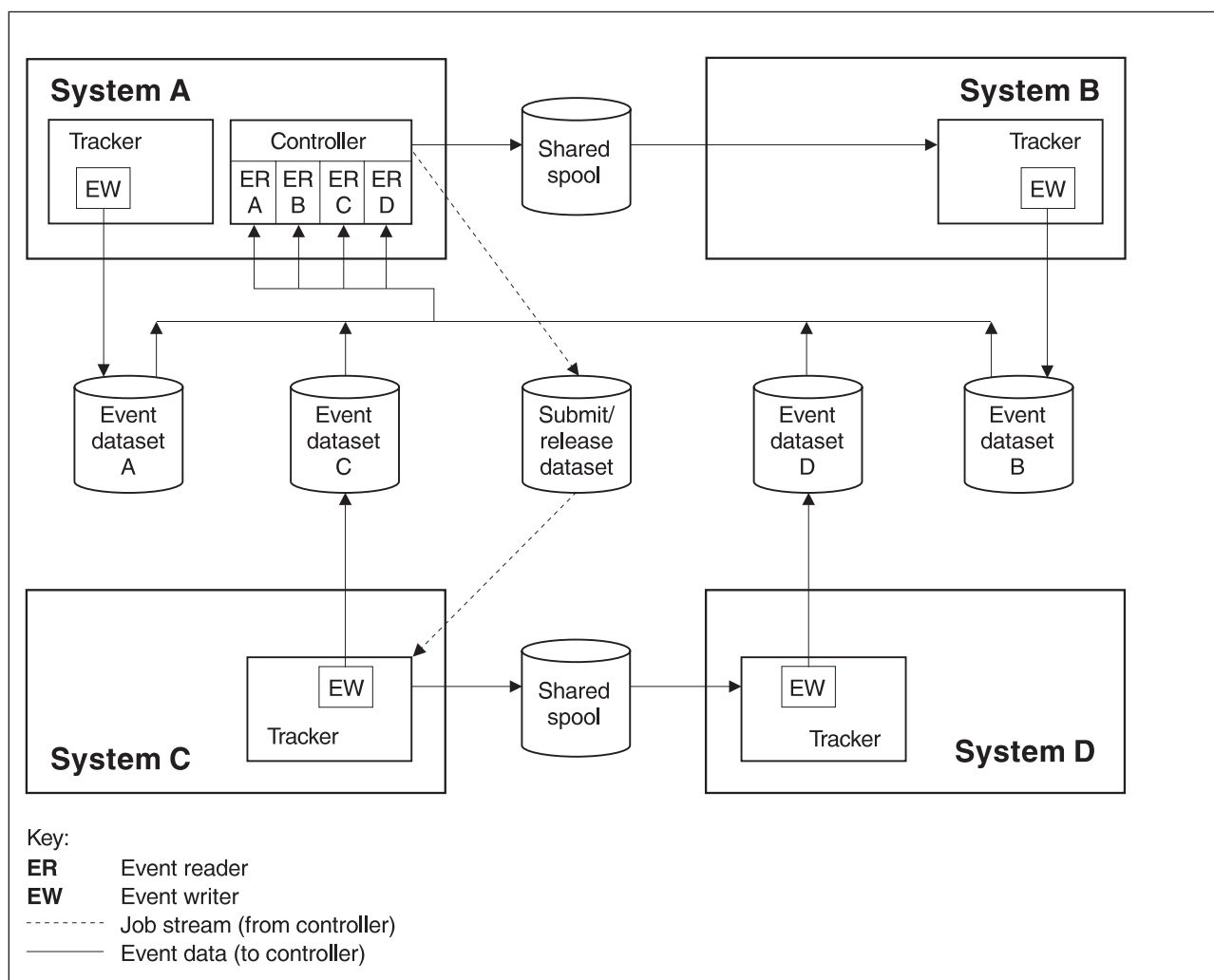


Figure 27. Two z/OS JES2 MAS complexes connected through shared DASD

Systems C and D, which are both Tivoli Workload Scheduler for z/OS controlled systems, form a second MAS complex. Work is sent to this complex via a submit/release data set. The destination field in the workstation description that represents this complex contains the DD name of the submit/release data set. The tracker on System C reads this data set and passes any new work to the complex for processing.

A tracker is installed on each system in the configuration. The event writer subtask of the tracker on each system writes events to an event data set on that system. Four event-reader subtasks, one for each of the event data sets, are started in the controller on System A. The controller reads the event data sets and updates the current plan.

When the controller is started on system A, it attempts to open the submit/release data set. If an I/O error occurs, the status of the workstation that represents the controlled MAS complex is set to offline. Tivoli Workload Scheduler for z/OS can then take automatic-workload-restart actions for operations at this workstation. These actions depend on the values that you specified on the WSOFFLINE keyword of the JTOPTS initialization statement.

Table 50 shows the initialization statements you can use to create the configuration in Figure 27 on page 318. This example assumes that some of the planned Tivoli Workload Scheduler for z/OS work on System C is submitted by a non-Tivoli Workload Scheduler for z/OS process. To control this work, the hold/release function is used. HOLDJOB(USER) is specified on the EWTROPTS statement for the tracker on System C. The RELDDNAME keyword is specified on the ERDROPTS statement of the event reader that reads event data set C. This keyword identifies the DD name of the submit/release data set that the controller should write release commands to.

Table 50. Example EQQPARM members for the previous figure

EQQPARM members for System A			
CONTROLR		TRACKERA	
OPCOPTS	OPCHOST(YES) ERDRTASK4(4) ERDRPARM(ERDRA,ERDRB) ERDRC,ERDRD DASD(SUDSC)	OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKAEW) HOSTCON(DASD)
ROUTOPTS		TRROPTS	
ERDRA		TRKAEW	
ERDROPTS	ERSEQNO(1)	EWTROPTS	
ERDRB			
ERDROPTS	ERSEQNO(2)		
ERDRC			
ERDROPTS	ERSEQNO(3) RELDDNAME(SUDSC)		
ERDRD			
ERDROPTS	ERSEQNO(4)		
EQQPARM members for System B			
TRACKERB		TRKBEW	
OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKBEW) HOSTCON(DASD)	EWTROPTS	
TRROPTS			
EQQPARM members for System C			
TRACKERC		TRKCEW	
OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKCEW) HOSTCON(DASD)	EWTROPTS	SUREL(YES) HOLDJOB(USER)
TRROPTS			
EQQPARM members for System D			
TRACKERD		TRKDEW	
OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKDEW) HOSTCON(DASD)	EWTROPTS	
TRROPTS			
Note: In this example, SUDSC is used for the user-defined DD name of the submit/release data set. This DD name appears in the started-task JCL of the controller, and in the destination field of the workstation that represents the controlled MAS system.			

Individual systems connected via shared DASD

Figure 28 shows three z/OS systems connected via shared DASD.

System A is the Tivoli Workload Scheduler for z/OS controlling system. Systems B and C are controlled systems, each of which shares a submit/release data set with the controlling system. Each of the three systems is represented by a computer workstation. The destination field in the workstation description for System A is left blank, indicating that Tivoli Workload Scheduler for z/OS should submit work to the system that the controller is started on. The destination field in the workstation descriptions for Systems B and C contains the DD name of the submit/release data set connecting them to the controller. Work is sent to the correct submit/release data set and is then passed to the corresponding system for processing by the event writer on that system.

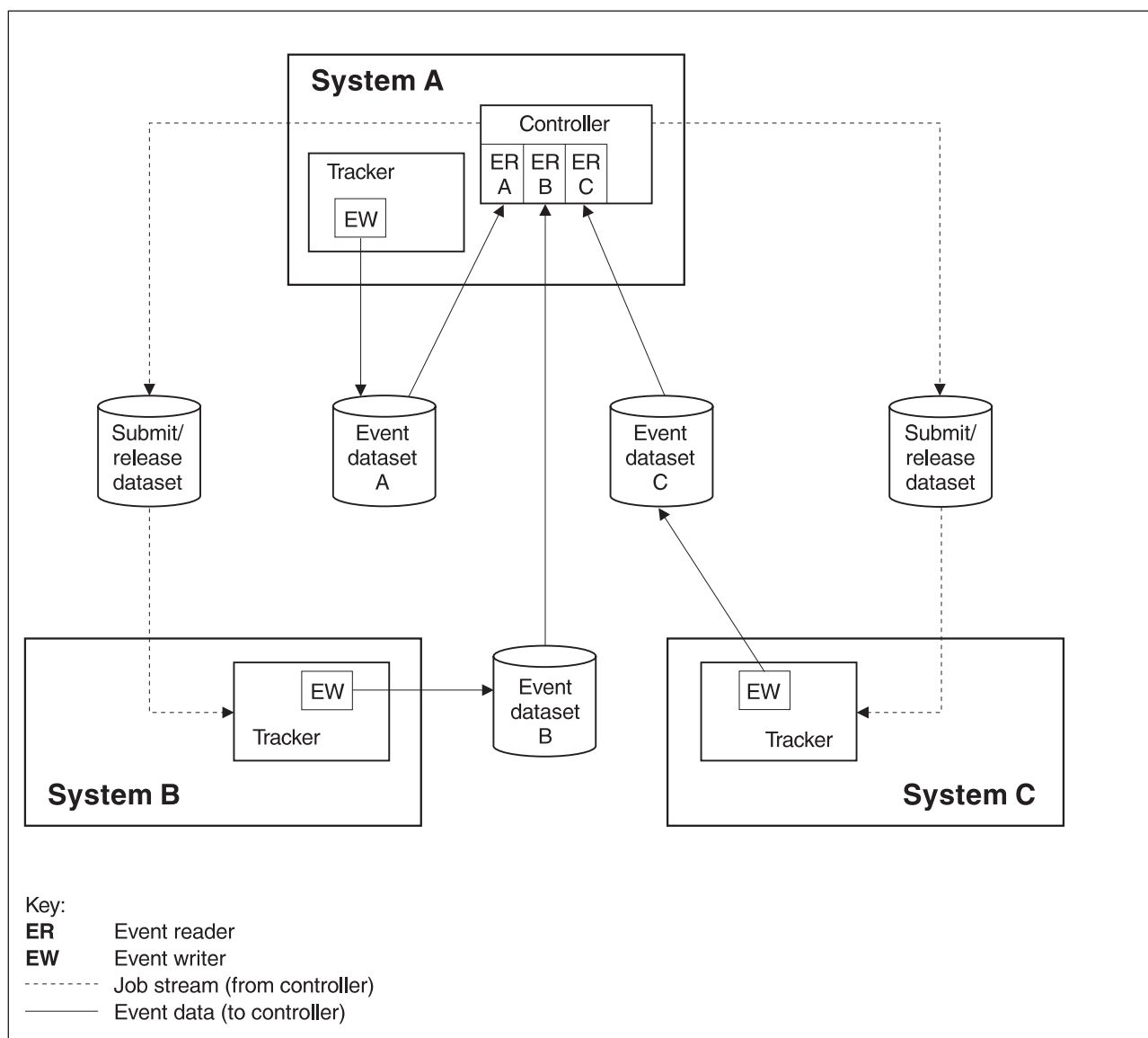


Figure 28. Individual systems connected through shared DASD

The event writer subtask on each system writes event information to its event data set. Three event-reader subtasks, one for each of the event data sets, are started in the controller on System A. The controller reads the event data sets and updates the current plan.

Automatic workload restart can be invoked in this configuration if an I/O error occurs when the controller attempts to open a submit/release data set. The workstation that has this submit/release data set as a destination is given the offline status, and WLR actions are taken according to the options specified on the WSOFFLINE keyword of the JTOPTS initialization statement.

Table 51 shows the initialization statements you can use to create the configuration in Figure 28 on page 320.

Table 51. Example EQQPARM members for the previous figure

EQQPARM members for System A	
CONTROLR	TRACKERA
OPCOPTS OPCHOST(YES)	OPCOPTS OPCHOST(NO)
ERDRTASK(3)	ERDRTASK(0)
ERDRPARM(ERDRA,ERDRB,ERDRC)	EWTRTASK(YES)
ROUTOPTS DASD(SUDSB,SUDSC)	EWTRPARM(TRKAEW)
ERDRA	TRROPTS HOSTCON(DASD)
ERDROPTS ERSEQNO(1)	TRKAEW
ERDRB	EWTROPTS
ERDROPTS ERSEQNO(2)	
ERDRC	
ERDROPTS ERSEQNO(3)	
EQQPARM members for System B	
TRACKERB	TRKBEW
OPCOPTS OPCHOST(NO)	EWTROPTS SUREL(YES)
ERDRTASK(0)	
EWTRTASK(YES)	
EWTRPARM(TRKBEW)	
TRROPTS HOSTCON(DASD)	
EQQPARM members for System C	
TRACKERC	TRKCEW
OPCOPTS OPCHOST(NO)	EWTROPTS SUREL(YES)
ERDRTASK(0)	
EWTRTASK(YES)	
EWTRPARM(TRKCEW)	
TRROPTS HOSTCON(DASD)	

Note: In this example, SUDSB and SUDSC are used for the user-defined DD names of the submit/release data sets. Both of these DD names appear in the JCL procedure of the controller. They also appear in the destination field of the respective workstations.

A z/OS Sysplex

Figure 29 on page 322 shows four systems, each connected by cross-system coupling facility (XCF) communication links.

System A is the controlling Tivoli Workload Scheduler for z/OS system and Systems B, C, and D are controlled systems. You represent each system in the systems complex (Sysplex) by a computer workstation. The destination field

contains the XCF-group-member name of the Tivoli Workload Scheduler for z/OS started task. On the controlling system, you can leave the destination field of the workstation that represents System A blank, or you can specify the XCF-group-member name of the tracker on that system. If you leave the field blank, the controller passes work to the system for processing. If you specify the tracker XCF-group-member name, the controller transmits work to the tracker, which in turn passes the work to this system. The way that you define this workstation depends on the recovery strategy you want to use.

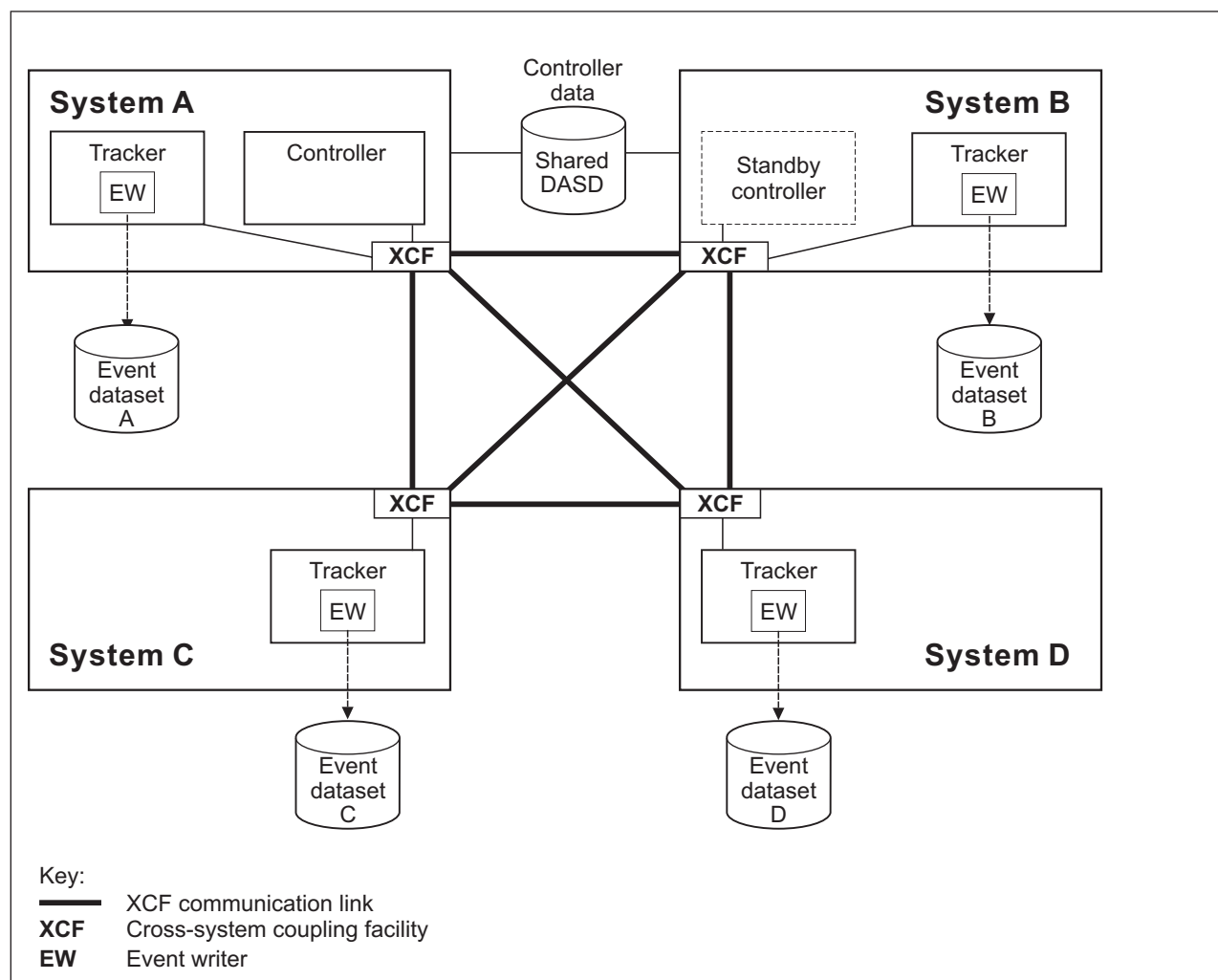


Figure 29. A z/OS Sysplex

A tracker is installed on each system in the sysplex. Each tracker event-writer subtask is started with a reader function, EWSEQNO is defined in the EWTROPTS statement. This means that the event writer passes the events to XCF for transfer to the controller at the same time as they are written to the event data set. This eliminates the need for separate event-reader subtasks.

XCF services let you define standby controllers, which act as a backup to the controller in case a failure occurs on the controlling system. This support is referred to as the hot standby function. In Figure 29, a Tivoli Workload Scheduler for z/OS address space is started on System B in standby mode. It is a copy of the controller but does not perform any functions unless the controller fails or System A fails. The standby controller must have access to all Tivoli Workload Scheduler for z/OS data, because it becomes the controller in the event of a failure.

The full functions of workload restart are available in this configuration. If a z/OS system failure occurs, the workstation that represents that destination is set to failed. Actions are taken according to the WSFAILURE keyword of the JTOPTS initialization statement. If a tracker fails or if the communication link between the controller and the tracker fails, the workstation is set to offline. Tivoli Workload Scheduler for z/OS takes actions according to the WSOFFLINE keyword of JTOPTS.

Table 52 shows the initialization statements you can use to create the configuration in Figure 29 on page 322.

Table 52. Example EQQPARM members for the previous figure

EQQPARM members for System A	
CONTROLR	
OPCOPTS	OPCHOST(YES) ERDRTASK(0)
ROUTOPTS	XCF(SYSATRK,SYSBTRK, SYSCTRK,SYSDTRK)
XCFOPTS	GROUP(OPCGRP) MEMBER(CONTR)
TRACKERA	
OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKAEW)
XCFOPTS	GROUP(OPCGRP) MEMBER(SYSATRK)
TRROPTS	HOSTCON(XCF)
TRKAEW	
EWTRROPTS	EWSEQNO(1)
EQQPARM members for System B	
TRACKERB	
OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKBEW)
XCFOPTS	GROUP(OPCGRP) MEMBER(SYSBTRK)
TRROPTS	HOSTCON(XCF)
TRKBEW	
EWTRROPTS	EWSEQNO(1)
EQQPARM members for System C	
TRACKERC	
OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKCEW)
XCFOPTS	GROUP(OPCGRP) MEMBER(SYSCTRK)
TRROPTS	HOSTCON(XCF)
EQQPARM members for System D	
TRACKERD	
OPCOPTS	OPCHOST(NO) ERDRTASK(0) EWTRTASK(YES) EWTRPARM(TRKDEW)
XCFOPTS	GROUP(OPCGRP) MEMBER(SYSDTRK)
TRROPTS	HOSTCON(XCF)
TRKDEW	
EWTRROPTS	EWSEQNO(1)
Note: In this example, the XCF group is called OPCGRP. This group has members CONTR, SYSATRK, SYSBTRK, SYSCTRK, SYSDTRK, and STBYCTRB.	

A PLEX configuration

Figure 30 shows four systems running in a sysplex environment, connected using cross-system coupling facility (XCF) communication links.

One controller and one tracker are started on each tracker image of the sysplex; one controller becomes the active one, while the others start as standby controllers. One server is started on the z/OS image where the active controller runs, to handle requests from dialogs and PIF applications.

The &SYSCONE system variable is assumed to be set to KA, KB, KB, and KC on systems A, B, C, and D respectively.

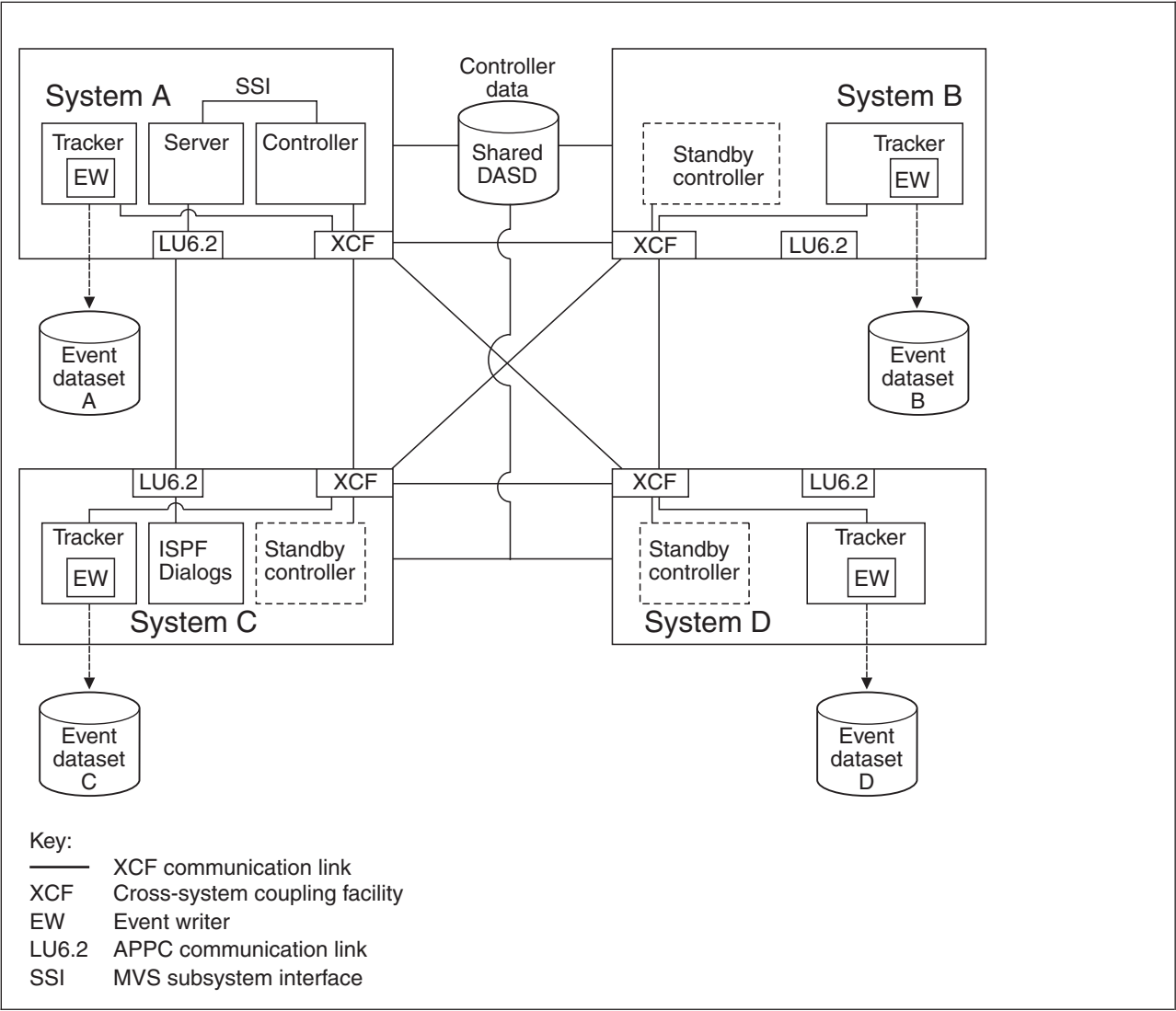


Figure 30. A Tivoli Workload Scheduler for z/OS PLEX environment

Table 53 shows the initialization statements you can use to create the configuration in Figure 30.

Table 53. Example EQQPARM Members for the previous figure

EQQPARM members, shared among z/OS images

Table 53. Example EQQPARM Members for the previous figure (continued)

CONTROLR		SERVER	
OPCOPTS	OPCHOST(PLEX)	SERVOPTS	SUBSYS(OPCC)
	ERDRTASK(0)		SCHEDULER(OSRV)
	SERVERS(OSRV)		
ROUTOPTS	XCF(TRKA,TRKB, TRKC,TRKD)		
XCFOPTS	GROUP(OPCGRP)		
	MEMBER(CONTR)		
TRACKER		TRKEW	
OPCOPTS	OPCHOST(NO)	EWTROPTS	EWSEQNO(1)
	ERDRTASK(0)		
	EWTRTASK(YES)		
	EWTRPARM(TRKBW)		
XCFOPTS	GROUP(OPCGRP)		
	MEMBER(TR&SYSCONE.)		
TRROPTS	HOSTCON(XCF)		

Controlling a z/OS system through a VTAM link

Figure 31 shows a z/OS system connected to the Tivoli Workload Scheduler for z/OS host via a VTAM link.

You represent each system by a computer workstation. The destination field in the workstation description for System A is left blank. Work for this workstation is started on System A. The destination field for the System B workstation contains the VTAM application ID of the tracker at this node. Work is transmitted from the host to the tracker and is then initiated on System B.

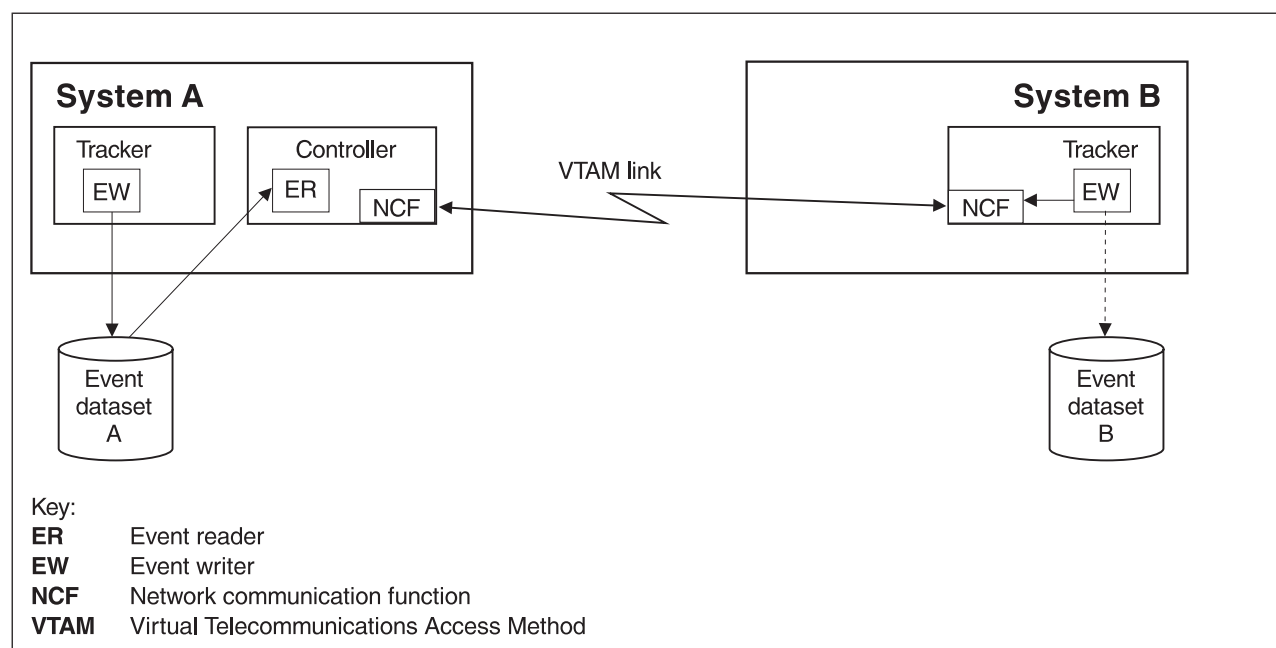


Figure 31. Controlling a z/OS system through a VTAM link

On System A, an event writer writes events to event data set A, which is read by an event reader subtask at the controller. On system B the tracker event-writer subtask is started with a reader function, EWSEQNO is defined in the EWTROPTS statement. This means that the event writer passes the events to NCF for transfer to the controller at the same time as they are written to the event data set.

Controlling z/OS system through VTAM link

Automatic workload restart can be used in this configuration if the controller cannot communicate with the tracker on system B. The status of the workstation for System B is set to offline if z/OS is stopped or fails, if the tracker is stopped or fails, or if the VTAM link is lost. WLR actions are taken according to the WSOFFLINE keyword of the JTOPTS initialization statement.

Table 54 shows the initialization statements you can use to create the configuration in Figure 31 on page 325.

Table 54. Example EQQPARM Members for the previous figure

EQQPARM members for System A	
CONTROLR	
OPCOPTS	OPCHOST(YES)
	ERDRTASK(1)
	ERDRPARM(ERDR1)
	NCFTASK(YES)
	NCFAPPL(NCFAPPL1)
ROUTOPTS	SNA(NCFAPPL2)
ERDR1	
ERDROPTS	ERSEQNO(1)
EQQPARM members for System B	
TRACKERB	
OPCOPTS	OPCHOST(NO)
	ERDRTASK(0)
	EWTRTASK(YES)
	EWTRPARM(TRKBEW)
	NCFTASK(YES)
	NCFAPPL(NCFAPPL2)
TRROPTS	HOSTCON(SNA)
	SNAHOST(NCFAPPL1)
Note: In this example, the controller has VTAM application ID NCFAPPL1, and the tracker on System B has VTAM application ID NCFAPPL2.	

Controlling a z/OS system through a TCP/IP link

Figure 32 on page 327 shows a z/OS system connected to the Tivoli Workload Scheduler for z/OS host via a TCP/IP link.

You represent each system by a computer workstation. The destination field in the workstation description for System A is left blank. Work for this workstation is started on System A. The destination field for the System B workstation contains the destination name associated with the IP address of the tracker on this system. Work is transmitted from the host to the tracker and is then initiated on System B.

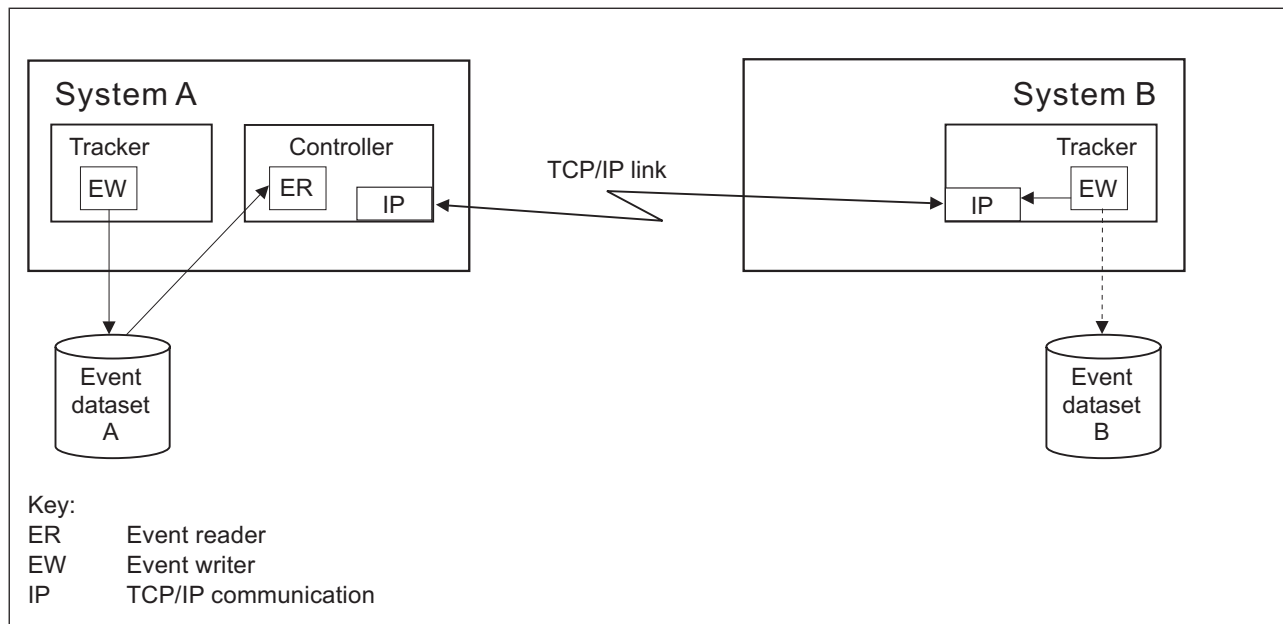


Figure 32. Controlling a z/OS system through a TCP/IP link

On System A, an event writer writes events to event data set A, which is read by an event reader subtask at the controller. On system B the tracker event-writer subtask is started with a reader function, EWSEQNO is defined in the EWTROPTS statement. This means that the event writer passes the events to NCF for transfer to the controller at the same time as they are written to the event data set.

Automatic workload restart can be used in this configuration if the controller cannot communicate with the tracker on system B. The status of the workstation for System B is set to offline if z/OS is stopped or fails, if the tracker is stopped or fails, or if the link is lost. WLR actions are taken according to the WSOFFLINE keyword of the JTOPTS initialization statement.

Table 55 shows the initialization statements you can use to create the configuration in Figure 32.

Table 55. Example EQQPARM Members for the previous figure

EQQPARM members for System A		TRACKERA	
CONTROLR			
OPCOPTS	OPCHOST(YES)	OPCOPTS	OPCHOST(NO)
	ERDRTASK(1)		ERDRTASK(0)
	ERDRPARM(ERDR1)	TRROPTS	HOSTCON(DASD)
TCPOPTS	TCPIPJOBNAME('TCPIP')		
	HOSTNAME('9.12.134.1')		
	TRKPORTNUMBER(8888)		
ROUTOPTS	TCPIP(DEST1:'1.111.111.111'/4444)		
ERDR1		TRKAEW	
ERDROPTS	ERSEQNO(1)	EWTROPTS	
EQQPARM members for System B			

Controlling z/OS system through TCP/IP link

Table 55. Example EQQPARM Members for the previous figure (continued)

TRACKERB	TRKBEW
OPCOPTS	OPCHOST(NO)
	ERDRTASK(0)
	EWTRTASK(YES)
	EWTRPARM(TRKBEW)
TCPOPTS	TCPIPJOBNAME('TCPIP')
	HOSTNAME('1.111.111.111')
	TRKPORTNUMBER(4444)
TRROPTS	HOSTCON(TCP)
	TCPHOSTNAME('9.12.134.1')
	TCPPORTNUMBER(8888)

Note: In this example, the name of the destination is DEST1. The destination is defined also in the destination field of the workstation.

Controlling a JES2 MAS system through a VTAM link

Figure 33 shows a z/OS JES2 MAS system connected to the Tivoli Workload Scheduler for z/OS host via a VTAM link.

System A and the systems in the JES2 MAS complex (System B and System C) are each represented by a computer workstation. The destination field for the workstation on System A is left blank so that work is initiated by the controller on that system. The destination field of the workstation descriptions for the MAS complex contains the VTAM application ID of the tracker on System B. The controller sends work to the tracker on System B via the network communication function. The tracker passes the work to the complex, and the work then processes on either System B or System C, depending on installation parameters.

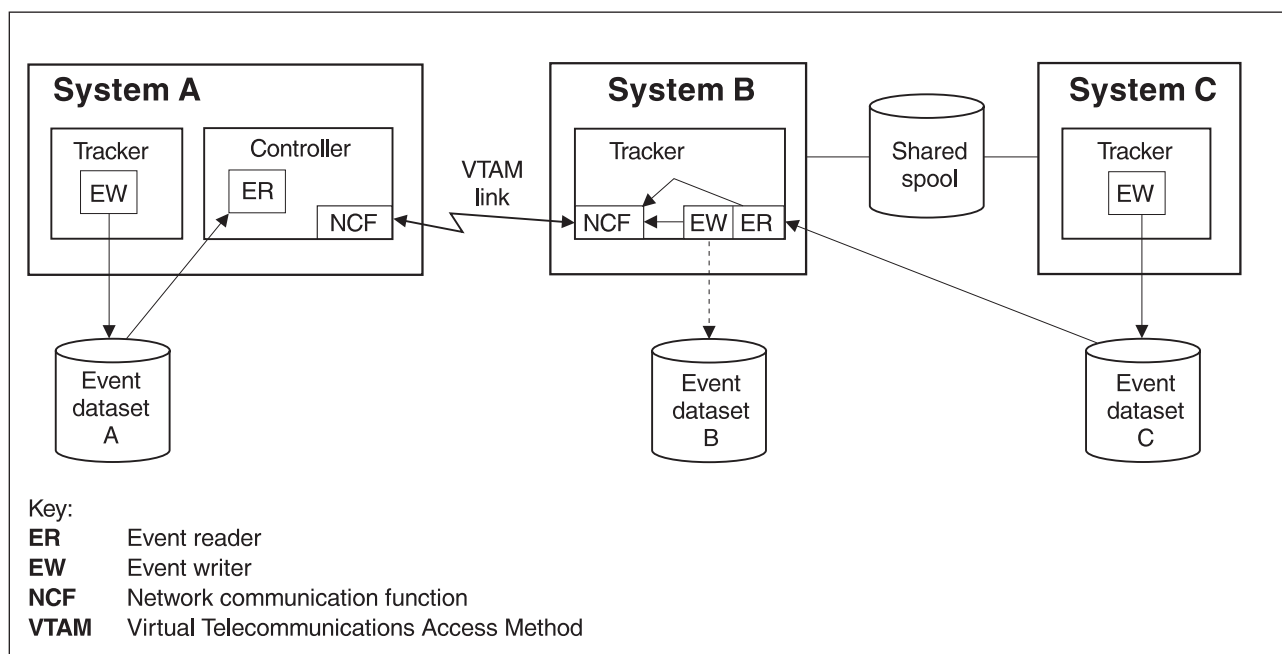


Figure 33. Controlling a JES2 MAS system through a VTAM link

A tracker is started on each system in the configuration. An event-reader subtask in the controller reads events from System A. The event-reader on System B reads the event information from System C and passes the events to NCF for transmission to the controller. This event-reader is required because System C does

not have its own link to the controller. The event-writer subtask on System B is started with a reader function—EWSEQNO is defined in the EWTROPTS statement. This means that the event writer passes the events for System B to NCF for transfer to the controller at the same time as they are written to the event data set.

Note: This figure demonstrates the need for an event reader task where System C does not have a direct link to the controller. But if the required resources are available, try to give each tracker its own link to the controller.

Automatic workload restart can be used in this configuration if the controller cannot communicate with the tracker on System B. The status of the workstation for System B is set to offline if z/OS is stopped or fails, if the tracker is stopped or fails, or if the VTAM link is lost. WLR actions are taken according to the WSOFFLINE keyword of JTOPTS. Workload restart is not affected by failures on System C, because the controller has no direct link with this system.

Table 56 on page 330 shows the initialization statements you can use to create the configuration in Figure 33 on page 328.

Controlling JES2 MAS system through VTAM link

Table 56. Example EQQPARM members for the preceding figure

EQQPARM members for System A	
CONTROLR	
OPCOPTS	OPCHOST(YES)
	ERDRTASK(1)
	ERDRPARM(ERDR1)
	NCFTASK(YES)
	NCFAPPL(NCFAPPL1)
ROUTOPTS	SNA(NCFAPPL2)
ERDR1	
ERDROPTS	ERSEQNO(1)
EQQPARM members for System B	
TRACKERB	
OPCOPTS	OPCHOST(NO)
	ERDRTASK(1)
	ERDRPARM(ERDR2)
	EWTRTASK(YES)
	EWTRPARM(TRKBEW)
	NCFTASK(YES)
	NCFAPPL(NCFAPPL2)
TRROPTS	HOSTCON(SNA)
	SNAHOST(NCFAPPL1)
ERDR2	
ERDROPTS	ERSEQNO(2)
EQQPARM members for System C	
TRACKERC	
OPCOPTS	OPCHOST(NO)
	ERDRTASK(0)
	EWTRTASK(YES)
	EWTRPARM(TRKCEW)
TRROPTS	HOSTCON(DASD)
<p>Note: In this example, the controller has VTAM application ID NCFAPPL1, and the tracker on System B has VTAM application ID NCFAPPL2.</p>	
TRACKERA	
OPCOPTS	OPCHOST(NO)
	ERDRTASK(0)
	EWTRTASK(YES)
	EWTRPARM(TRKAEW)
TRROPTS	HOSTCON(DASD)
TRKAEW	
EWTROPTS	
TRKBEW	
EWTROPTS EWSEQNO(1)	
TRKCEW	
EWTROPTS HOLDJOB(NO)	

Appendix C. Invoking the EQQEXIT macro

The sample event-tracking exits shipped with Tivoli Workload Scheduler for z/OS are written in assembler language. The event-tracking code in these exits is generated by an assembler macro called EQQEXIT. The following sections describe how you invoke the EQQEXIT macro. This appendix contains General-use Programming Interface and Associated Guidance Information.

Invoking EQQEXIT in SMF exits

EQQEXIT establishes its own addressability in SMF exits. It saves and restores all used registers. To do this, it expects Register 13 to point to a standard z/OS save area.

There are two ways to invoke the EQQEXIT macro in an SMF exit:

- Invoke EQQEXIT with all registers unchanged since the exit was called (except Register 15).
- Save all registers on entry to the exit and then invoke EQQEXIT by specifying the address of the initial save area.

In both cases, the EQQEXIT macro must be invoked in Supervisor state, PSW key 0.

Invoking EQQEXIT in JES exits

In JES exits, EQQEXIT must be invoked in Supervisor state, PSW key 1. EQQEXIT expects code addressability to be already established. It also expects registers to be set up as follows:

- EXIT7
 - R0** JCT read/write indicator (JES2 SP Version 3 and earlier); address of a parameter list mapped by the JES2 \$XPL macro (JES2 SP Version 4 and later)
 - R1** Address of the JCT being read or written
 - R13** Address of the current PCE
- EXIT51
 - R1** Address of a parameter list mapped by the JES2 \$XPL macro (JES2 with z/OS 1.7, or later)
- IATUX19
 - R8** Address of the current JDS entry
 - R9** Address of the current RESQUEUE entry
 - R11** Address of the current FCT entry
 - R12** Address of the TVTABLE entry
- IATUX29
 - R11** Address of the current FCT entry
 - R13** Address of the input-service data area for the current function.

Note that these register conventions are already set up when the exit is called. You must invoke EQQEXIT while these registers are unchanged.

Invoking EQQEXIT in JES exits

If a shipped JES exit example (or the EQQEXIT macro) has been user-modified, make sure that it does not prevent or filter the tracking of Tivoli Workload Scheduler for z/OS itself.

See the NOTES section of the EQQEXIT prolog for information about the register contents that are destroyed by EQQEXIT in JES exits.

Macro invocation syntax for EQQEXIT

Purpose

EQQEXIT produces Tivoli Workload Scheduler for z/OS event-tracking exit code by generating assembler code to perform in an SMF or JES exit.

Syntax

EXIT=*exit name*
REG13=*address of save area*
MAPMAC={YES|NO}
SETUID={YES|NO}
SRREAD={YES|NO|NONE}
SNA={YES|NO}

Parameters

EXIT=*exit name*

A required keyword defining the name of the exit in which the macro is used. The following names can be specified: IEFACRT, IEFUJI, IEFU83, EXIT7, IATUX19, and IATUX29. Except for the EXIT7 exit, a warning message is issued if the name of the current CSECT differs from the name specified by the EXIT keyword.

REG13=*address of save area*

An optional keyword defining the address of the current-register save area when the SMF or JES exit was called. The default for this keyword depends on the name specified by the EXIT keyword. If the current exit is EXIT7, the default is PCELPV. If the current exit is IATUX19 or IATUX29, the default is FCTSAVCH. In all other cases, the default is the second fullword in the current save area (if the current save area is properly chained, and the previous save area contains the registers at entry to the exit).

If the default does not apply, the REG13 keyword must be specified. Its value must be a fullword pointing to the save area that was used to store all the registers when the exit was entered.

MAPMAC={YES|NO}

An optional keyword specifying whether the macro should generate the required assembler mapping macros. The default is to generate these mapping macros. The following mapping macros are required by EQQEXIT code: CVT, IEFJESCT, IEFJSSOB, and IEFJSSIB. The IEFACRT exit also requires the IEFJMR macro.

If you specify NO, the IEFU83 exit requires mapping of the SMF records IFASMFR 14 and IFASMFR 64. You must label them SMF14REC and SMF64REC, respectively. For example:

```
SMF14REC DSECT      * SMF RECORD 14 MAPPING
               IFASMFR 14      * DATA SET ACTIVITY RECORD
```

SETUID={YES|NO}

An optional keyword specifying whether the macro should generate code to place the current user ID in the JMRUSEID field when the IEFUJI exit is taken. Specify YES to generate this code. If you specify NO, which is the default, the JMRUSEID field is not updated. You are recommended to specify YES if you use the current user ID to filter data set close events. You need these mapping macros when you specify YES: IHAPSA, IHAASCB, IHAASXB, and IHAACEE.

Macro invocation syntax for EQQEXIT

SRREAD={YES|NO|NONE}

An optional keyword defining whether a resource availability event should be generated when a data set is closed after being opened for read processing.

When YES is specified, an SR event is generated each time a data set is closed after being opened for either read or output processing.

When NO is specified or defaulted, the SR event is generated only when a data set has been opened for output processing. The event is not generated if the data set has been opened for read processing.

When you specify NONE, no data set triggering is performed.

See “Implementing support for data set triggering” on page 83 for more information about the data set triggering function.

SNA={YES|NO}

An optional keyword specifying whether JES3 SNA NJE is supported.

Return codes: The following return codes can be generated at assembly time:

- 4** Input invalid, check for warning messages.
- 12** Unsupported exit specified for the EXIT keyword.

Messages

The following messages can be generated at assembly time:

- WARNING: SNA KEYWORD IS ONLY USED FOR EXIT = IATUX19
- WARNING: SNA VALUE SNA IS NOT RECOGNIZED
- WARNING: EXIT NAME DIFFERS FROM CURRENT CSECT NAME
- WARNING: MAPMAC VALUE MAPMAC IS NOT RECOGNIZED
- WARNING: SRREAD KEYWORD IS ONLY USED FOR EXIT=IEFU83
- WARNING: SRREAD VALUE NOT RECOGNIZED, YES OR NO ARE THE ONLY VALID VALUES
- EXIT NAME EXIT IS NOT SUPPORTED

Appendix D. Invoking the EQQLSENT macro

The following procedure is supported for backward compatibility only. To exploit the current support for data set triggering, see the procedure for running event-driven workload automation described in *Managing the Workload*.

When the data set triggering function is used, you specify the data sets for which you want events generated by building the data set selection table EQQDSLST. The EQQDSLST is created by invoking the EQQLSENT macro. The following sections describe how you invoke the EQQLSENT macro. This appendix contains General-use Programming Interface and Associated Guidance Information.

Note: The current support for data set triggering is based on the EQQEVLSST configuration file. If EQQJCLIB contains both EQQEVLSST and EQQDSLST, the resulting triggering selection table is the union of EQQEVLSST and EQQDSLST. In this case, EQQEVLSST data is processed first. If EQQJCLIB contains only EQQDSLST, the tracker loads it as triggering selection table.

Invoking EQQLSENT to create EQQDSLST

The EQQLSENT macro is used to create entries in the data set triggering selection table. The selection table is loaded into ECSA when the Tivoli Workload Scheduler for z/OS event writer is started.

The sample EQQLSJCL in the SEQQSAMP library can be used to invoke the EQQLSENT macro.

Macro invocation syntax for EQQLSENT

Purpose

EQQLSENT produces an entry in the data set triggering selection table, EQQDSLST. EQQDSLST is used in SMF exit IEFU83 by the data set triggering function to decide which SMF records to process. When an SMF 14, 15, or 64 record matches a condition in EQQDSLST, a special resource availability event is created and broadcast to all Tivoli Workload Scheduler for z/OS subsystems defined on the system where the SMF record was created.

Format

STRING= *string* | **LASTENTRY**
POS= *numeric position*
USERID= *user ID filter criteria*
JOBNAME= *jobname filter criteria*
AINDIC={Y|N}
LIFACT={Y|N|R}
LIFTIM=*interval*

Parameters

STRING=*string* | **LASTENTRY**

Required keyword specifying the character string to be searched for. The string can be 1 to 44 characters long. To identify the fully-qualified last level of a data set name, add a space as the last character and enclose the string in single quotes. Consider this example. You have two data sets:

DSN.NAME.AB
DSN.NAME.ABC

Specify **STRING=DSN.NAME.AB,POS=1** if you want SR availability events created for both data sets. Specify **STRING='DSN.NAME.AB ',POS=1** if you want events created only for the first data set.

When EQQLSENT is invoked with **STRING=LASTENTRY** it generates an end of table indicator. After having invoked EQQLSENT with keyword parameters **STRING** and **POS** a number of times, EQQLSENT must be invoked one last time with **STRING=LASTENTRY** in order to complete the table.

To create an empty EQQDSLST, just invoke EQQLSENT once, with **STRING=LASTENTRY**. When an empty list is used by IEFU83, no SR events are created.

POS=*numeric position*

A required keyword specifying the numeric position where the string begins.

USERID=*string*

Optional keyword specifying a generic character string to be compared with the SMFxxUID field, which contains the user identification associated with the job, started task, or TSO user that requested the activity against the data set that resulted in the data set close. The string can be 1 to 8 characters long.

Note: The SMF user ID field may contain a blank value. See *z/OS System Management Facilities* for more information about the SMFxxUID or SMFxxUIF field.

If you need to control SR availability events based on the user ID and the SMF value is blank in your installation, consider using the IEFUJI exit to insert the user ID. You are recommended to specify SETUID=YES on the EQQEXIT macro when you generate the IEFUJI exit: this sets the JMRUSEID field, which SMF then copies to the SMF user ID field.

If you want to update the JMRUSEID field yourself, the user ID is most easily taken from the ACEEUSRI field in the ACEE, pointed to from the ASXB, pointed to from the ASCB. This can be located as follows: PSAA0LD
 ===> ASCB ACSBASXB ===> ASXB
 ASXBSENV ===> ACEE
 ACEEUSRI ===> userid

The DSECTs needed are mapped by these macros:

Area	Macro	Library
PSA	IHAPSA	SYS1.MACLIB
ASCB	IHAASCB	SYS1.MACLIB
ASXB	IHAASXB	SYS1.MODGEN
ACEE	IHAACEE	SYS1.MACLIB

The JMR, mapped by IEFJMR, is already available in the EQQEXIT expansion in IEFUJI.

JOBNAME=*string*

Optional keyword specifying a generic character string to be compared with the SMF14JBN, SMF15JBN, or SMF64JMN field, which contains the name of the job, started task or TSO user that requested the activity against the data set that resulted in the data set close. The string can be 1 to 8 characters long.

If the data set is to be processed by FTP, JOBNAME corresponds to the **USERID ** under which the dataset is received. That is, the USERID supplied when the remote host opened the FTP session to PUT the dataset, or when a local user (or batch job) opened the FTP session to GET the dataset.

AINDIC={Y|N}

Optional keyword specifying that the special resource is available (Y) or unavailable (N). The default is that the resource available.

LIFACT={Y|N|R}

Optional keyword specifying the value to which the global availability of the special resource is reset, after the interval of time specified by LIFTIM has expired. Allowed values are:

- Y** Sets the global availability to Yes
- N** Sets the global availability to No
- R** Sets the global availability to blank

This keyword is valid only if LIFTIM is specified. The default is R.

LIFTIM=*interval*

Optional keyword specifying the interval of time, in minutes, after which the global availability of the special resource is reset to the value specified by LIFACT. The allowed range is from 1 to 999999.

Macro invocation syntax for EQQLSENT

Notes:

1. The output from assembling the EQQLSENT macro must be placed in the EQQDSLST member in the data set referenced by the ddname EQQJCLIB.
2. Generation Data Group data sets are specified by the group name. For example, when a GDG data set with the name 'DSN.OPCSUBS.GDG.G0001V00' is closed the special resource event contains resource name 'DSN.OPCSUBS.GDG'.
3. For a partitioned data set, the member name is not part of the resource name in the SR event.
4. For VSAM data sets the resource name in the SR event is the cluster name (without the DATA or INDEX suffix).

Examples

```
EQQLSENT STRING=SYS1.MAN,POS=1
EQQLSENT STRING='TEST.DSCLOSE ',POS=1,USERID=SYSOP
EQQLSENT STRING=CP2,POS=12
EQQLSENT STRING=EQQDATA.EXCL,POS=5
EQQLSENT STRING='DSN.OPCSUBS.GDG ',POS=1
EQQLSENT STRING=LASTENTRY
END
```

In this example, SMF records with:

- A data set name beginning with SYS1.MAN, or
- Data set name TEST.DSCLOSE and user ID SYSOP
- Records with CP2 in position 12, such as DSN.OPCSUB.CP2, or
- Records that have EQQDATA.EXCL starting in position 5
- The root of a GDG dataset name

will cause SR availability events to be generated.

Return codes: The following return code can be generated at assembly time:

12 Input invalid, check error messages.

Messages

The following messages can be generated at assembly time:

- KEYWORD STRING IS REQUIRED
- KEYWORD POS IS REQUIRED
- POSITION MUST BE BETWEEN 1 AND 43
- NULL NAME NOT VALID
- NAME (STRING) GREATER THAN 44 CHARACTERS
- POSITION INVALID FOR NAME (STRING)
- USERID STRING NOT VALID
- JOBNAME STRING NOT VALID
- AINDIC MUST BE EITHER Y OR N
- POSITION NOT VALID FOR NAME (STRING)
- LIFACT MUST BE Y, N, OR R
- LIFTIM LENGTH NOT VALID
- LIFTIM VALUE NOT VALID
- LIFTIM VALUE 0 NOT ALLOWED

Appendix E. Using response files

All components of Tivoli Workload Scheduler and the Dynamic Workload Console that can be installed by the InstallShield wizard can also be installed silently, using a response file. A response file is a flat text list of property-value pairs each of which corresponding to a data item that the wizard needs to determine what is to be installed, where, and with what configuration. Silent installations can be used to install, upgrade or uninstall components locally, or remotely.

Tivoli Workload Scheduler and the Dynamic Workload Console components are provided with template response files, containing the appropriate properties to perform one installation, upgrade, or uninstallation action.

To perform a silent installation, provide the following command line arguments when running the wizard:

```
-options "<response_file_name> -silent
```

The provided files are template files, so you are recommended to edit the properties appropriately, and then save a copy of the file with a file name which identifies the component to be installed and the system on which it is to be installed.

The properties have unique names and uses, and are described in the following sections. Many of them will contain default values that you can use. The defaults are not listed here as they may change, depending on which template file they are used in.

Note: Where the same template file is provided for Windows and UNIX platforms, default paths are supplied for both environments, with the keys duplicated and one commented out. Note that if you uncomment one and omit to comment the other, the wizard utilizes the last of the duplicated keys.

Response file properties: general

Appendix F. z/OS connector response file properties

This section describes the properties used in the Tivoli Workload Scheduler for z/OS connector response files.

The provided files are template files, so you are recommended to edit the properties appropriately, and then save a copy of the file with a file name which identifies the component to be installed and the system on which it is to be installed.

The properties described in the following table have unique names and uses. Many of them contain default values that you can use.

The response file properties are shown in alphabetical order.

Notes:

1. All values must be written between double quotation marks ("), for example: `ZOSInstanceConfiguration.ZOSToConfigure="true"`.
2. Properties are written in mixed case for ease of reading, but are not case-sensitive.
3. Keywords (for example, "true") used in values, are not case-sensitive.
4. If you are installing a z/OS connector in an existing Tivoli Workload Automation instance, the WebSphere Application Server port definitions are commented. You must uncomment these properties only if your existing Tivoli Workload Automation instance does not have a WebSphere Application Server installed. Follow the instructions provided in the response file as commented text.

Table 57. Tivoli Workload Scheduler for z/OS connector response file properties

Name	Description	Permitted values
InstallationActions. TWA_INSTANCE_PATH	Tivoli Workload Automation instance path Identifies the path where an instance of Tivoli Workload Automation has already been installed. The z/OS connector will be installed in this instance.	The fully qualified path of the existing Tivoli Workload Automation instance. You cannot use national characters in the installation path.
licenseAccepted	Accept license agreement To install a component using a response file you must explicitly accept the license agreement, a copy of which is in the License directory of the product install media (DVD or downloaded image).	true To accept the license agreement. false To not accept the license agreement. In this event the component is <i>not</i> installed. This is the default value.
twPortsPanel. portAdmin	Administration HTTP transport port	The default value is 31223 .
twPortsPanel. portAdminSec	Administration HTTPS transport port	The default value is 31224 .

Response file properties: Tivoli Workload Scheduler for z/OS connector

Table 57. Tivoli Workload Scheduler for z/OS connector response file properties (continued)

Name	Description	Permitted values
twPortsPanel. portHTTP	HTTP transport port	The default value is 31215 .
twPortsPanel. portHTTPS	HTTPS transport port	The default value is 31216 .
twPortsPanel. portMtlAuth	CSIV2 Client Authentication Listener port	The default value is 31221 .
twPortsPanel. portORB	ORB Listener port	The default value is 31222 .
twPortsPanel. portRMI	Bootstrap port	The default value is 31217 .
twPortsPanel. portSAS	SAS Server Authentication Listener port	The default value is 31219 .
twPortsPanel. portSOAP	SOAP connector port	The default value is 31218 .
twPortsPanel. portSrvAuth	CSIV2 Server Authentication Listener port	The default value is 31220 .
upgradePanel. bckpDirectory	The backup path for the profile used by the embedded version of WebSphere Application Server. The profile is backed up by the installation procedure into the path you supply with this property.	Any valid fully qualified path outside the path of any existing Tivoli Workload Scheduler component.
userUnixCfgPanel. inputUserName	The user ID of the <i>TWS_user</i> (on UNIX).	The ID must already exist on the system where the silent wizard will be run. This user will own the instance of the z/OS connector.
userUnixCfgPanel. twPassword	The password of the <i>TWS_user</i> (on UNIX).	
userWinCfgPanel. inputUserName	The ID of the <i>TWS_user</i> - the user that will "own" the instance of the z/OS connector on the agent workstation (on Windows).	If this user does not already exist, it will be created. In this case, the format of the ID must follow the rules for User IDs on the computer where it is to be created.
userWinCfgPanel. twPassword	The password of the <i>TWS_user</i> (on Windows).	If the user is to be created, the format of the password must follow the rules for passwords on the computer where it is to be created.

Response file properties: Tivoli Workload Scheduler for z/OS connector

Table 57. Tivoli Workload Scheduler for z/OS connector response file properties (continued)

Name	Description	Permitted values
ZOSInstanceConfiguration. ZOSToConfigure	Specify if you want to configure a connection to a Tivoli Workload Scheduler for z/OS controller.	<p>true To configure a connection to a Tivoli Workload Scheduler for z/OS controller.</p> <p>false To skip the configuration of a connection to a Tivoli Workload Scheduler for z/OS controller. After the installation, you can create connections using WebSphere Application Server tools (wastools).</p>
ZOSInstanceConfiguration. engineName	The name of the Tivoli Workload Scheduler for z/OS controller.	It is a label that identifies the z/OS connector instance.
ZOSInstanceConfiguration. remoteHost	The host name or TCP/IP address of the Tivoli Workload Scheduler for z/OS controller.	A valid hostname or TCP/IP address.
ZOSInstanceConfiguration. remotePort	The host name or TCP/IP port number used to communicate with the Tivoli Workload Scheduler for z/OS controller.	This value must correspond to the value specified in the SERVOPTS member on the controller. The default value is 11111 .

Appendix G. The Dynamic Workload Console response file properties

This section describes the properties used in the Dynamic Workload Console response files, in alphabetical order:

Notes:

1. All values must be written between double quotation marks ("), for example: `InstallationActions.INSTALL_METHOD="new"`
2. Property names are written in mixed case for ease of reading, but are not case-sensitive
3. Keywords used in values are not case-sensitive.

Table 58. Dynamic Workload Console response file properties

Name	Description	Permitted values
BOOTSTRAP_ADDRESS	The bootstrap port.	See "Advanced installation" on page 233 for more details.
CREATE_WAS_SERVICE	On Windows, the embedded WebSphere Application Server can be defined to start automatically at system startup. To do this, set this property, which creates a Windows service that starts up the embedded WebSphere Application Server.	true A Windows service is created to automatically start the embedded WebSphere Application Server false The Windows service is <i>not</i> created
CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS	CSIV2 Client Authentication Listener port.	See "Advanced installation" on page 233 for more details.
CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS	CSIV2 Server Authentication Listener port	See "Advanced installation" on page 233 for more details.
DCS_UNICAST_ADDRESS	The DCS Unicast port.	See "Advanced installation" on page 233 for more details.
ENABLE_TDWB	Enable Dynamic Workload Broker The Dynamic Workload Console can be used to access either of the following: <ul style="list-style-type: none">• Tivoli Workload Scheduler (includes Tivoli Workload Scheduler for z/OS)• Dynamic workload broker All users must be given specific access to one or both of these products. It is useful to give these access rights to the WebSphere Application Server administrator from the outset, so that the administrator can immediately perform any tasks that might be required:	true Gives the administrator access to Dynamic Workload Broker false Denies the administrator access to Dynamic Workload Broker

Response file properties: Dynamic Workload Console

Table 58. Dynamic Workload Console response file properties (continued)

Name	Description	Permitted values
ENABLE_TWS	Enable Tivoli Workload Scheduler See the description of "ENABLE_TDWB"	true Gives the administrator access to Tivoli Workload Scheduler false Denies the administrator access to Tivoli Workload Scheduler
INSTALL_METHOD	Installation instance choice The Dynamic Workload Console must be installed in an instance of Tivoli Workload Automation (see "Instances of Tivoli Workload Automation" on page 197 for an explanation). This property lets you choose whether you want to install the component in a new instance (installing also the embedded WebSphere Application Server and other infrastructure support), or an existing instance. In the former case, the path you want to use for the new instance must be defined in the property IS_DESTINATION. In the latter case you must also identify the path of the existing instance, using the property: TWA_INSTANCE_PATH This property also lets you install the Dynamic Workload Console outside the Tivoli Workload Automation structure, on your own external supported version of WebSphere Application Server. In this case, the path must be supplied using the property ISC_APPSERVER_DIR	new Install the Dynamic Workload Console in a new instance of Tivoli Workload Automation (and install the infrastructure support). Use this value also when upgrading an existing instance of the Dynamic Workload Console. ONTWA Install the Dynamic Workload Console in an existing instance of Tivoli Workload Automation onwas Install the Dynamic Workload Console on your own external supported version of WebSphere Application Server
IPC_CONNECTOR_ADDRESS	The IPC connector.	See "Advanced installation" on page 233 for more details.
IS_BACKUP_DIR	Backup directory for upgrade When upgrading the Dynamic Workload Console, the wizard needs to back up the application server configuration while it is upgrading embedded WebSphere Application Server (part of the Dynamic Workload Console upgrade process).	Any valid, fully qualified path outside: any existing instance of Tivoli Workload Automation, and the installation path of the Embedded Version of WebSphere Application Server

Response file properties: Dynamic Workload Console

Table 58. Dynamic Workload Console response file properties (continued)

Name	Description	Permitted values
IS_DESTINATION	<p>Console installation path</p> <p>On a new instance of Tivoli Workload Automation: the path of a new instance of Tivoli Workload Automation where the Dynamic Workload Console is to be installed.</p> <p>On your existing external instance of WebSphere Application Server: when installing the Dynamic Workload Console on your own external version of WebSphere Application Server, supply the console installation path.</p> <p>This does <i>not have to be</i> a path related to the instance of the WebSphere Application Server on which you are going to install it. The path must not be within an instance of Tivoli Workload Automation:</p>	Any valid, fully qualified path outside any existing instance of Tivoli Workload Automation.
IS_UPGRADE	Boolean property that determines whether the wizard is being run to upgrade an existing instance.	<p>true The wizard will use the supplied properties to upgrade an existing instance of the Dynamic Workload Console</p> <p>false The wizard will use the supplied properties to install an instance of the Dynamic Workload Console</p>
ISC_ADMIN_FULL_USER	<p>Your WebSphere Application Server administrator user ID</p> <p>On a new instance of Tivoli Workload Automation: supply the user ID to be used for the Integrated Solutions Console administration user</p> <p>On your existing external instance of WebSphere Application Server: when installing, upgrading, or uninstalling the Dynamic Workload Console on your own external version of WebSphere Application Server, supply the existing user ID of the Integrated Solutions Console administration user.</p>	The user ID must exist.

Response file properties: Dynamic Workload Console

Table 58. Dynamic Workload Console response file properties (continued)

Name	Description	Permitted values
ISC_ADMIN_PASSWORD	<p>Your WebSphere Application Server administrator user password</p> <p>On a new instance of Tivoli Workload Automation: supply the password to be used for the Integrated Solutions Console administration user</p> <p>On your existing external instance of WebSphere Application Server: when installing, upgrading, or uninstalling the Dynamic Workload Console on your own external version of WebSphere Application Server, supply the password of the user ID of the existing Integrated Solutions Console administration user.</p>	
ISC_APPSERVER_DIR	<p>Existing instance installation directory</p> <p>The installation directory of the external Integrated Solutions Console on which the Dynamic Workload Console must be installed or upgraded.</p>	See “Installing on your existing instance of Tivoli Integrated Portal” on page 235 for more details.
licenseAccepted	<p>Accept license agreement</p> <p>To install the Dynamic Workload Console using a response file, you must explicitly accept the license agreement, a copy of which is in the License directory of the product install media (DVD or downloaded image).</p>	<p>true To accept the license agreement.</p> <p>false To not accept the license agreement. In this event, the Dynamic Workload Console is <i>not</i> installed.</p>
ORB_LISTENER_ADDRESS	ORB Listener port	See “Advanced installation” on page 233 for more details.
REST_NOTIFICATION_ADDRESS	The REST notification port.	See “Advanced installation” on page 233 for more details.
SAS_SSL_SERVERAUTH_LISTENER_ADDRESS	SAS SSL Port	See “Advanced installation” on page 233 for more details.
SOAP_CONNECTOR_ADDRESS	SOAP connector port	See “Advanced installation” on page 233 for more details.
TWA_INSTANCE_PATH	<p>Existing Tivoli Workload Automation instance path</p> <p>The path of an existing instance of Tivoli Workload Automation where the Dynamic Workload Console is to be installed.</p>	Any valid, fully qualified Tivoli Workload Automation instance path.
UPDATE_INSTALLER_DIR	<p>The WebSphere Application Server update installer path</p> <p>The directory of the external WebSphere Application Server update installer.</p>	See “Installing on your existing instance of Tivoli Integrated Portal” on page 235 for more details.

Response file properties: Dynamic Workload Console

Table 58. Dynamic Workload Console response file properties (continued)

Name	Description	Permitted values
WAS_CELL_NAME	The WebSphere Application Server cell name The external WebSphere Application Server cell name.	See “Installing on your existing instance of Tivoli Integrated Portal” on page 235 for more details.
WAS_NODE_NAME	The WebSphere Application Server node name The external WebSphere Application Server node name.	See “Installing on your existing instance of Tivoli Integrated Portal” on page 235 for more details.
WAS_PROFILE_NAME	The WebSphere Application Server profile name The external WebSphere Application Server profile name.	See “Installing on your existing instance of Tivoli Integrated Portal” on page 235 for more details.
WAS_SERVER_NAME	The WebSphere Application Server server name The external WebSphere Application Server server name.	See “Installing on your existing instance of Tivoli Integrated Portal” on page 235 for more details.
WC_adminhost	Administrative console	See “Advanced installation” on page 233 for more details.
WC_adminhost_secure	Administrative Console Secure	See “Advanced installation” on page 233 for more details.
WC_defaulthost	HTTP transport	See “Advanced installation” on page 233 for more details.
WC_defaulthost_secure	HTTPS transport	See “Advanced installation” on page 233 for more details.

Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this publication in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

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

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan, Ltd.
1623-14, Shimotsuruma, Yamato-shi
Kanagawa 242-8502 Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement might not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
2Z4A/101
11400 Burnet Road
Austin, TX 78758 U.S.A.

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

The licensed program described in this publication and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Trademarks

IBM, the IBM logo, and `ibm.com`[®] are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol ([®] or [™]), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Intel, Intel Centrino, and Itanium are registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Other product and service names might be trademarks or service marks of others.

Index

Special characters

@ character in install directory name, causing the Dynamic Workload Console uninstallation to fail 265

A

ACCEPT processing 308

accessibility xv

activating

API (application programming interface) 135

Dynamic Workload Console 142

NCF 131

server 137

subtasks 10

add

option to add runtime environment for Java jobs with wdinstsp 285

option to add the dynamic workload broker resource command with wdinstsp 285

the Java runtime to run job types with advanced options using wdinstsp 294

agent

installation

JobManager port 272

installing 271

APAR

PK93917 80

APARs

IZ79105 245

PK00502 133

PK04155 84, 86

PK05336 185

PK06007 73, 78

PK06227 61, 73, 74, 148, 149, 301, 311, 312, 331

PK06712 132

PK06763 180

PK11767 310

PK16903 124, 125

PK18760 39

PK20776 81

PK20879 185

PK22761 177

PK23181 115

PK25245 114

PK25268 99

PK25979 304

PK28707 106

PK30374 3

PK31005 90

PK34310 90, 114

PK39432 179

PK40356 61, 141, 301, 313

PK40969 7, 8, 9, 15, 16, 17, 24, 40, 58, 60, 70, 91, 134, 155, 168

PK41519 66

PK45614 143

PK53014 127

PK53476 118, 120

PK54782 185

PK56520 90, 118, 121

APARs (*continued*)

PK57062 126

PK64650 337

PK65147 312

PK65527 125

PK69493 84

PK77418 44

PK81179 338

PK83161 185

PK86682 55, 63, 67

PK88734 73

PK92042 74, 132, 309, 311

PK94896 4

PK96348 98, 99, 193

PM01090 101, 113

PM02690 113

PM04245 124, 127

PM06648 101, 180

PM07439 62, 302, 309

PM08778 83, 106

PM21607 123

PM32308 115

PM45677 81

PQ65923 3

PQ78043 3

PQ78350 90

PQ81700 95

PQ84095 310

PQ86050 101

PQ87576 84, 88

PQ87710 115

PQ89715 112

PQ91074 117

PQ96400 75

PQ96540 39, 51, 66, 75, 109

PQ97143 90

PQ98852 116

PQ99317 61, 304

PQ99366 192

API (application programming interface)

activating support for 83, 135

APPC/MVS

options, updating in SYS1.PARMLIB 83, 137, 140

APPCPMnn member of SYS1.PARMLIB 83, 137, 140

APPL

resource class 87

statements

API (application programming interface) 135

local LU for the server, defining 139

local LU, defining 135

NCF 131, 133

application description data set (EQQADDS) 95

considerations when allocating 98

application job plug-ins

option to add runtime environment for Java jobs with wdinstsp 285

option to add the Java runtime to run job types with advanced options using wdinstsp 294

application node definitions, NCF 131

application programming interface (API)

activating support for 83, 135

- application server
 - installation log files 229
- APPLY processing 308
- applying maintenance
 - z/OS connector 216
- z/OS connector on WebSphere Application for z/OS 209
- AUTHCMD statement, updating for Tivoli OPC TSO
 - commands 82
- authentication mechanism
 - Tivoli Dynamic Workload Console
 - updating 251
- authority, problem determination procedures 154
- authorization roles
 - installation wizard 270
 - Software Distribution 270
 - twsinst 270
 - z/OS connector 197
 - z/OS connector on WebSphere Application for z/OS 209
- AUTHTSF statement, updating for EQQMINOX 81
- automatic-recovery-procedure library (EQQPRLIB) 101
 - considerations when allocating 110
- automatically generated response file
 - installing the z/OS connector 203
- available functions
 - for Dynamic Workload Broker 240
 - for Tivoli Workload Scheduler 239

B

- BACKUP command 82
- batch jobs
 - security 84
 - user ID of Tivoli OPC submitted jobs 84
- batch-jobs
 - generating skeleton JCL 63
- books
 - See* publications
- BOOTSTRAP_ADDRESS response file property 345
- BULKDISC command 82
- business scenario
 - z/OS connector on WebSphere Application Server for z/OS 209

C

- calendar and workstation data set (EQQWSDS) 96
- CDRSC statements
 - API 136
 - NCF 131
- CDs
 - Dynamic Workload Console
 - installation 228
- checklist for installing 40
- checkpoint data set (EQQCKPT) 100
 - considerations when allocating 103
- class of service table 133
- CLI
 - wdinstsp 283
 - wimspo 283
- CLIST library 100, 124
- COFDLFnn member of SYS1.PARMLIB 82
- command line
 - See* CLI
- commands
 - TSO
 - BACKUP 82

commands (*continued*)

- TSO (*continued*)
 - BULKDISC 82
 - OPINFO 82
 - OPSTAT 82
 - SRSTAT 82
 - WSSTAT 82
- wdinstsp 283
- wdinstsp agent installation 284
- wdinstsp CIT installation 283, 292
- wdinstsp to add dynamic capabilities 285
- wdinstsp to add runtime environment for Java jobs 285
- wdinstsp to add the Java runtime to run job types with
 - advanced options 294
- wimspo 283
- COMMNDnn member of SYS1.PARMLIB 83
- compatibility, software 5
- configurations
 - connecting Tivoli Workload Scheduler for z/OS
 - systems 16
 - cross-system coupling facility (XCF) 17
 - description 9
 - event data set 15
 - examples
 - introduction 20, 317
 - NCF connection 325, 328
 - PLEX 324
 - server 19
 - shared DASD connection 20, 317, 320
 - single address space 27
 - sysplex 321
 - TCP/IP connection 24, 326
 - VTAM connection 22, 325, 328
 - XCF connection 25, 321
 - MAS (Multi-Access Spool) restrictions 18
 - Multi-Access Spool (MAS) restrictions 18
 - NCF (network communication function) 17
 - planning 15
 - shared DASD 16
 - VTAM 17
 - workload restart
 - description 18
 - examples 318, 321, 323, 326, 327, 329
 - workstation destination 17
 - XCF (cross-system coupling facility) 17
- connection
 - to Tivoli Workload Scheduler 240
- console
 - portfolio 247
 - start 247
- controlled systems
 - description 9
 - software requirements 4
- controller
 - checking the message log 152
 - description 6
 - IBMOPC resource class 86
 - loading national language support (NLS) software 54
 - loading software 53
 - RACF 84, 86
 - software requirements 4
 - started task data sets 118, 120
 - verifying installation 151
- controlling system
 - description 9
- conventions used in publications xiv
- COS table 133

- COUPLEnn member of SYS1.PARMLIB 80
- CREATE_WAS_SERVICE response file property 345
- creating sample JCL 55
- critical job table data set (EQQJTABL) 101
- cross-domain resource definitions
 - API 136
 - NCF 131
- cross-system coupling facility (XCF)
 - groups 129
 - including 129
 - initialization statements 130
 - MVS initialization options 80
 - run time options 130
- CSIV2_SSL_MUTUALAUTH_LISTENER_ADDRESS response file property 345
- CSIV2_SSL_SERVERAUTH_LISTENER_ADDRESS response file property 345
- current plan data set (EQQCPnDS) 95
 - considerations when allocating 98
- customer support
 - See* Software Support

D

- data
 - considerations when allocating
 - dual job-tracking log (EQQDLnn) 107
 - job-tracking archive (EQQJTARC) 107
 - job-tracking log (EQQJTnn) 107
- data lookaside facility (DLF) 82
- data set triggering
 - implementing 83
- data set triggering selection table macro (EQQLSENT)
 - invoking 335
 - syntax 336
- data sets
 - allocating 94
 - VSAM 95
 - considerations when allocating
 - (event-driven workload automation configuration file (EQQEVLIB) 106
 - application description (EQQADDs) 98
 - automatic-recovery-procedure library (EQQPRLIB) 110
 - checkpoint (EQQCKPT) 103
 - current plan (EQQCPnDS) 98
 - diagnostic (EQQDUMP) 104
 - diagnostic message and trace (EQQDMSG) 104
 - dump (SYSMDUMP) 104
 - event (EQQEVDnn) 104
 - event (EQQEVDS) 104
 - event (EQQHTTP0) 104
 - extended data (EQQXDnDS) 99
 - general 94
 - JCC incident log 107
 - JCC incident work (EQQINCWK) 107
 - JCC message table (EQQJCLIB) 107
 - JCL repository (EQQJSnDS) 99
 - job library (EQQJBLIB) 106
 - job-tracking archive (EQQJTARC) 107
 - loop analysis (EQQLOOP) 109
 - message log (EQQMLOG) 108
 - parameter library (EQQPARM) 109
 - PIF parameter data set (EQQYPARM) 109
 - started-task submit (EQQSTC) 110
 - submit/release (EQQSUDS) 110
 - in Tivoli OPC procedure JCL
 - optional 120

- data sets (*continued*)
 - in Tivoli OPC procedure JCL (*continued*)
 - required 118
 - ISPF profile 123
 - migrating 181, 183, 184
 - security 86
 - Tivoli OPC dialog 126
- databases, migrating 181, 183, 184
- DB2
 - migrating 178
- DB2 database
 - creating 122
- DCS_UNICAST_ADDRESS response file property 345
- default dialog-controller connection table 124
- diagnostic data set
 - EQQDMSG 100, 104
 - EQQDUMP 100, 104, 134
 - SYSMDUMP 79, 101, 104
- dialog
 - description 12
 - entering 127
 - EQQMINOX, authorizing 81
 - initializing 123
 - ISPF command table 124
 - problem determination procedures 153
 - security 87
- dispatching priority 82
- distribution tape 13
- DLF (data lookaside facility) 82
- domain manager
 - backup
 - See* backup domain manager
 - backup master
 - See* backup master domain manager
 - master
 - See* master domain manager
- dump content definitions 79
- dump data set (SYSMDUMP) 79, 101, 104
- dynamic exits (PROGnn) 77, 79
- Dynamic Workload Broker
 - available functions 240
 - server connection 241
- Dynamic workload broker host name
 - installation
 - Tivoli Workload Scheduler for z/OS Agent 273
- Dynamic workload broker HTTPS port number
 - installation
 - Tivoli Workload Scheduler for z/OS Agent 273
- Dynamic Workload Console
 - accessibility xv
 - activating 142
 - activating support for 142
 - configuration 245
 - for Tivoli Workload Scheduler Version 8.3 Fix Pack 3 237
- connection
 - to Dynamic Workload Broker components 241
- getting started 247
- installation
 - advanced 233
 - CDs 228
 - default 233
 - images 228
 - log files 228
 - methods 226
 - on embedded WebSphere Application Server 232
 - on existing external WebSphere Application Server 235

Dynamic Workload Console *(continued)*

- installation *(continued)*
 - on existing instance of the embedded WebSphere Application Server 234
 - sample scenarios 226
 - setup file 228
 - silent 235
 - types 233
 - using launchpad 231
 - using response file 235
 - using wizard 231
 - installation and uninstallation log files 257
 - installing 223
 - log files 228
 - overview 225
 - remove
 - manually 258
 - server support 142
 - starting and stopping 242
 - troubleshooting 257
 - uninstall 255
 - clean-up 258
 - in silent mode 255
 - manually 258
 - using wizard 255
 - updating
 - authentication mechanism 251
 - upgrade
 - failed, recovering 257
 - upgrading
 - overview 251
 - silently 253
 - upgrading on embedded WebSphere Application Server 252
 - using launchpad 253
 - using wizard 253
 - user interface 238
- Dynamic Workload Console and z/OS connector
- secure communications 213
- Dynamic Workload Console problems with
- uninstallation 265
 - upgrade 264

E

- ECSA (extended common service area) 75
- education
 - See* Tivoli technical training
- EDWA configuration file repository (EQQEVLIB) 101
- ENABLE_TDWB response file property 345
- ENABLE_TWS response file property 346
- end-to-end centralized script data set (EQQTWSCS) 111
- End-to-end data set for centralized script support (EQQTWSCS) 101
- end-to-end event data sets (EQQTWSIN/OUT) 101
- end-to-end input events data set (EQQTWSIN) 111
- end-to-end output events data set (EQQTWSOU) 111
- end-to-end script library (EQQSCLIB) 101
- ended-in-error-list layout table 126
- environment setup 307
- EQQADDS (application description data set) 95
 - considerations when allocating 98
- EQQBRDS (internal reader data set) 103
- EQQCKPT (checkpoint data set) 100
 - considerations when allocating 103
- EQQCPnDS (current plan data set) 95
 - considerations when allocating 98
- EQQDLnn (dual job-tracking-log data set) 100
 - considerations when allocating 107
- EQQDMSG (diagnostic message and trace data set) 100
 - considerations when allocating 104
- EQQDUMP (diagnostic data set) 100, 134
 - considerations when allocating 104
- EQQEVDnn (event data set for an event reader) 101
 - calculating optimum LRECL 105
 - considerations when allocating 104
- EQQEVDs (event data set) 101
 - calculating optimum LRECL 105
 - considerations when allocating 104
- EQQEVLIB (EDWA configuration file repository) 101
- EQQEVLIB (event-driven workload automation configuration file data set)
 - considerations when allocating 106
- EQQEXIT (event-tracking-code generation macro)
 - in JES exits 73, 331
 - in SMF exits 73, 331
 - invoking 331
 - syntax 333
- EQQHTTP0 (event data set) 101
 - considerations when allocating 104
- EQQICNVH (sample to migrate the history database) 122
- EQQINCWK (JCC incident work data set) 101
 - considerations when allocating 107
- EQQINITL 75
- EQQJBLIB (job library data set) 101
 - considerations when allocating 106
- EQQJCLIB (JCC message table data set) 101
 - considerations when allocating 107
- EQQJOBS installation aid
 - creating sample JCL 55
 - description 54
 - generating batch-job skeletons 63
 - setting up 55
- EQQJSnDS (JCL repository data set) 95
 - considerations when allocating 99
- EQQJTABL (critical job table data set) 101
- EQQJTARC (job-tracking-archive data set) 101
 - considerations when allocating 107
- EQQJTnn (job-tracking-log data set) 101
 - considerations when allocating 107
- EQQLDDS (long-term plan work data set) 95
- EQQLOOP (loop analysis data set) 101
- EQQLOOP (loop analysis log data set)
 - considerations when allocating 109
- EQQLSENT (data set triggering selection table macro)
 - invoking 335
 - syntax 336
- EQQLTBKP (long-term-plan backup data set) 95
- EQQLTDS (long-term plan data set) 95
- EQQMINOX, authorizing for TSO 81
- EQQMLOG
 - checking at the controller 152
 - checking at the server 152
- EQQMLOG (message log data set) 101
 - considerations when allocating 108
- EQQMONDS (monitoring data set) 101
- EQQNCPDS (new current plan data set) 95
- EQQNCXDS (new current plan extension data set) 95
- EQQOCPBK (data set) 101
- EQQOIDS (operator instruction data set) 95
- EQQPARM (parameter library) 101
 - considerations when allocating 109
- EQQPRLIB (automatic-recovery-procedure library) 101
 - considerations when allocating 110

- EQQRDDS (resource description data set) 95
- EQQSCLIB (end-to-end script library) 101
- EQQSIDS (side information data set) 95
- EQQSTC (started-task-submit data set) 101
 - considerations when allocating 110
- EQXSUDS (submit/release data set) 17, 101
 - considerations when allocating 110
- EQQTROUT (input to EQQAUDIT) 101
- EQQTROUT (tracklog data set) 108
- EQQTSWCS (end-to-end centralized script dataset) 111
- EQQTSWCS (End-to-end data set for centralized script support) 101
- EQQTWSIN (end-to-end input events data set) 111
- EQQTWSIN/OUT (end-to-end event data sets) 101
- EQQTWSOU (end-to-end output events data set) 111
- EQQWSDS (workstation and calendar data set) 96
- EQQXDnDS (extended data, data set)
 - considerations when allocating 99
- EQQYPARM (parameter data set)
 - considerations when allocating 109
- EQQYPARM (PIF parameter data set) 101
- event data set
 - description 15
 - for an event reader (EQQEVDnn) 101
 - for an event writer (EQQEVDs) 101
 - for submit checkpointing 15, 101
 - verify 148
- event types 147
- event writer
 - using with event reader function 40
- event-driven workload automation configuration file data set (EQQEVLIB)
 - considerations when allocating 106
- event-tracking-code generation macro (EQQEXIT)
 - in JES exits 73, 331
 - in SMF exits 73, 331
 - invoking 331
 - syntax 333
- exits
 - event tracking 72
- extended common service area (ECSA) 75
- extended data (EQQXDnDS)
 - considerations when allocating 99

F

- fallback 192
- fault-tolerant domain manager
 - See domain manager
- fault-tolerant switch manager
 - See domain manager
- file sets
 - See files
- file system
 - See files
- files
 - /etc/password 291
 - swdis.ini 283
 - temporary
 - See temporary files
- firewall stopping installation of the Dynamic Workload Console 260
- functions, Tivoli Workload Scheduler for z/OS
 - subtasks, activating 10

G

- GDDM 126
- generating batch-job skeletons 63
- glossary xiv
- graphical attribute table 126

H

- hardware requirements 3
- Hiperbatch support
 - COFDLFnn, updating 82
- history function
 - installing 122
- host name
 - not FQDN causing installation to fail on Linux 262
- host name or IP address
 - installation
 - Tivoli Workload Scheduler for z/OS Agent 273
- hot standby 39

I

- IBM i Tivoli Workload Scheduler for z/OS Agent 276
- IBMOPC resource class 86
- ICHRIN03 84
- IKJTSONn member of SYS1.PARMLIB
 - AUTHCMD statement, updating for Tivoli OPC TSO commands 82
 - AUTHTSF statement, updating for EQQMINOX 81
- images
 - Dynamic Workload Console
 - installation 228
- incident log (JCC)
 - considerations when allocating 107
- initialization statements
 - defining 122
- input to EQQAUDIT (EQQTROUT) 101
- INSTALL_METHOD response file property 346
- installation
 - agent
 - JobManager port 272
 - checking
 - See installation, verifying
 - Dynamic Workload Console
 - CDs 228
 - images 228
 - in silent mode 227
 - methods 226
 - on embedded WebSphere Application Server 232
 - on existing external WebSphere Application Server 235
 - on existing instance of the embedded WebSphere Application Server 234
 - sample scenarios 226
 - setup file 228
 - types 233
 - using launchpad 226
 - using wizard 226
 - Dynamic Workload Console log files 257
 - from shared folder fails on Windows 262
 - hangs (Dynamic Workload Console) 260
 - log files, embedded WebSphere Application Server 229
 - of the Dynamic Workload Console fails when installing on different external WebSphere Application Server profile 264
 - of the Dynamic Workload Console, fails to start on Linux RHEL 5 (x86-64) 263

- installation (*continued*)
 - silent 273
 - of the Dynamic Workload Console, problems with 264
 - response file template 274
 - software package blocks 281
 - steps
 - See* steps, installation
 - Tivoli Integrated Portal
 - from the DVD or eImages 237
 - Tivoli Integrated Portal installation fails 261
 - Tivoli Workload Scheduler for z/OS Agent
 - Dynamic workload broker host name 273
 - Dynamic workload broker HTTPS port number 273
 - host name or IP address 273
 - JobManager port 273
 - troubleshooting scenarios
 - Dynamic Workload Console 259
- installation and uninstallation log files
 - z/OS connector 199, 213
- installation method
 - twinsinst 276
- installation methods
 - z/OS connector 198
 - z/OS connector on WebSphere Application for z/OS 209
- installation wizard
 - authorization roles 270
- InstallationActions.TWA_ 341
- installing
 - agent 271
 - authorization roles z/OS connector 197
 - authorization roles z/OS connector on WebSphere Application for z/OS 209
 - availability 39
 - checklist 40
 - considerations 39
 - DB2 database 122
 - detailed instructions 51
 - EQQJOBS installation aid
 - creating sample JCL 55
 - description 54
 - generating batch-job skeletons 63
 - setting up 55
 - event tracking exits 72
 - hot standby 39
 - initialization statements, defining 122
 - loading controller software 53
 - loading national language support (NLS) software 54
 - loading tracker software 53
 - MAS (Multi-Access Spool) restrictions 18
 - Multi-Access Spool (MAS) restrictions 18
 - NCF 131
 - overview 13
 - planning 39
 - RACF 84
 - security 84
 - started-task operations, implementing support for 117
 - using the installation wizard 271
 - verifying 145
 - configuration 158
 - controller 151
 - standby controller 154
 - submit events 169
 - tracker 145
 - tracking events 147
 - z/OS connector 197
 - z/OS connector from the launchpad 204
 - z/OS connector in silent mode 202, 211

- installing (*continued*)
 - z/OS connector on WebSphere Application for z/OS 209
 - z/OS connector using the wizard 200
 - z/OS connector with automatically generated response file 203
 - z/OS connector with response file template 202
- installing on WebSphere Application for z/OS
 - z/OS connector using the Integrated Solutions Console 210
 - z/OS connector using zConnInstall.sh 211
- InstallShield wizard
 - installation and uninstallation log files 228
 - problem using with the Dynamic Workload Console 260
- INSTANCE_PATH response file property 341
- Integrated Solutions Console
 - installing the z/OS connector 210
 - uninstalling the z/OS connector 217
- interactive mode
 - installing the z/OS connector 200
 - upgrading the z/OS connector 204
 - upgrading the z/OS connector version 8.3 204
 - upgrading the z/OS connector version 8.5 or 8.5.1 205
- interactive wizard
 - problem using with the Dynamic Workload Console 260
- internal reader data set (EQQBRDS) 103
- IPC_CONNECTOR_ADDRESS response file property 346
- IS_BACKUP_DIR response file property 346
- IS_DESTINATION response file property 347
- IS_UPGRADE response file property 347
- ISC_ADMIN_FULL_USER response file property 347
- ISC_ADMIN_PASSWORD response file property 348
- ISC_APPSERVER_DIR response file property 348
- ISMP
 - See* InstallShield wizard
- ISPF (Interactive System Productivity Facility)
 - command table 124
 - table library 124

J

- J2SE
 - See* Java Runtime Environment
- Java 2 Platform, Standard Edition
 - See* Java Runtime Environment
- Java development kit
 - See* Java Runtime Environment
- Java Development Kit
 - See* Java Runtime Environment
- Java utilities
 - activating support for 144
- Java Virtual Machine
 - See* Java Runtime Environment
- JCL repository data set (EQQJSnDS) 95
 - considerations when allocating 99
- JDK
 - See* Java Runtime Environment
- JES exits, installing 72
- JES2
 - EXIT51 331
 - EXIT7 331
 - MAS (Multi-Access Spool) restrictions 18
 - Multi-Access Spool (MAS) restrictions 18
- JESJOBS RACF resource class 88
- JESSPOOL RACF resource class 89, 91
- job completion checker (JCC)
 - data sets
 - considerations when allocating 106

- job completion checker (JCC) *(continued)*
 - data sets *(continued)*
 - incident log 101, 107
 - incident work 107
 - incident work (EQQINCWK) 101
 - message table (EQQJCLIB) 101, 107
- job library data set (EQQJBLIB) 101
 - considerations when allocating 106
- job-tracking data sets
 - considerations when allocating 107
 - dual job-tracking-log data set (EQQDLnn) 100
 - job-tracking-archive data set (EQQJTARC) 101
 - job-tracking-log data set (EQQJTnn) 101
 - tracklog data set (EQQTROUT) 108
- joblog and Restart Information requests log data sets
 - EQQLOGRC (joblog and Restart Information requests log data sets) 101
- JobManager port
 - installation
 - agent 272
 - Tivoli Workload Scheduler for z/OS Agent 273
- JRE
 - See* Java Runtime Environment
- JVM
 - See also* Java Runtime Environment
 - causing installation to fail on Linux RHEL V5 263
 - causing installation to fail on Suse Linux 263

L

- language packs
 - installing 279, 291
 - removing 297
 - uninstalling 300
- launchpad
 - installation prerequisites 271
 - installing the z/OS connector 204
 - installing Tivoli Workload Scheduler for z/OS Agent 271
 - problems using with the Dynamic Workload Console 259
 - starting 271
- LDAP
 - upgrading
 - Tivoli Dynamic Workload Console 251
- libraries
 - CLIST 124
 - ISPF table 124
 - link 80
 - sample (SEQQSAMP) 301
- licenseAccepted response file property, TDWC 348
- licenseAccepted response file property, TWS 341
- Lightweight Directory Access Protocol
 - See* LDAP
- link library (LNKLSTnn) 80
- Linux
 - erroneous warning messages displayed from
 - launchpad 259
 - installation fails if host name not FQDN 262
 - RHEL 5 (x86-64) install or uninstall of the Dynamic Workload Console fails to start 263
 - RHEL V5 and Suse V11 installation fails (JVM) 263
- LNKLSTnn member of SYS1.PARMLIB 80
- load module library (IEAAPFnn) 77
- log file
 - z/OS connector 199, 213
- log files
 - Dynamic Workload Console 257
 - embedded WebSphere Application Server installation 229

- log successfully
 - but Tivoli Integrated Portal installation fails 261
- logon mode table 132
- long-term plan data set (EQQLTDS) 95
- long-term plan work data set (EQQLDDS) 95
- long-term-plan backup data set (EQQLTBKP) 95
- LookAt message retrieval tool xiv
- loop analysis data set
 - EQQLOOP 101
- loop analysis log data set
 - EQQLOOP
 - considerations when allocating 109

M

- macros
 - EQQEXIT (event-tracking-code generation macro) 73, 331
 - EQQSENT (data set triggering selection table macro) 335
- maintaining
 - z/OS connector 219
- manually
 - Dynamic Workload Console
 - uninstall 258
- manuals
 - See* publications
- MAS (Multi-Access Spool) restrictions 18
- master domain manager
 - backup
 - See* backup master domain manager
- MAXECSA values 76
- message log data set
 - checking at the controller 152
 - checking at the server 152
 - EQQMLOG 101
 - considerations when allocating 108
- message retrieval tool, LookAt xiv
- migrating 186
 - data sets 181, 183, 184
 - databases 181, 183, 184
 - DB2 178
 - EQQICTOP conversion program 181
 - overview 175
- Monitoring data set (EQQMONDS) 101
- Multi-Access Spool (MAS) restrictions 18
- multiple systems
 - configuration examples 317
- MVS
 - relationship with Tivoli Workload Scheduler for z/OS 12
 - router service 86
 - SSI (subsystem interface) 12
 - subsystem interface (SSI) 12

N

- national language support (NLS) 54
- NCF (network communication function)
 - activating 131
 - application node definitions 131
 - cross-domain resource definitions 131
- network communication function (NCF)
 - activating 131
 - application node definitions 131
 - cross-domain resource definitions 131
- new current plan data set (EQQNCPDS) 95
- new current plan extension data set (EQQNCXDS) 95
- NLS (national language support) 54

O

- operator instruction data set (EQQOIDS) 95
- OPERCMDS RACF resource class 89, 90
- OPINFO command 82
- OPSTAT command 82
- ORB_LISTENER_ADDRESS response file property 348
- overview
 - upgrading
 - Dynamic Workload Console 251

P

- parallel sysplex
 - configuration examples 317
- parallel test 177
- parameter library (EQQPARM) 101
 - considerations when allocating 109
- parameter twsinst update
 - backup_dir 290
 - domain 290
 - inst_dir 290
 - lang 291
 - nobackup_dir 291
 - password 291
 - reset_perm 291
 - skip_usercheck 291
 - uname 291
- performance considerations
 - dispatching priority 82
 - event data set (EQQEVDs), calculating optimum LRECL 105
 - program properties table (PPT) 82
 - starting an event writer with an event reader 40
- PIF parameter data set (EQQYPARM) 101
 - considerations when allocating 109
- planned maintenance
 - See* maintenance
- planning
 - installation 39
- PLEX configuration 324
- port
 - JobManager 272, 273
- portfolio
 - console 247
- ports
 - WebSphere Application Server 233
- PPT (program properties table) 82
- problem determination procedures
 - authority 154
 - dialog 153
 - job tracking 147
 - security 154
 - tracker 147
- problems
 - See* troubleshooting
- profile, WebSphere Application Server, different, causing the Dynamic Workload Console installation to fail 264
- PROGnn member of SYS1.PARMLIB 77, 79
- program directory 13
- program properties table (PPT) 82
- program temporary fix (PTF) 13
- PTF (program temporary fix) 13
- publications xiv

Q

- queues, message
 - See* message queues

R

- RACF 12
 - APPL resource class 87
 - batch jobs 84
 - IBMOPC resource class 86
 - ICHRIN03 84
 - modifying 84, 86
 - STARTED resource class 84
 - started task 86
 - user ID of Tivoli OPC submitted jobs 84
 - using functions of RACF 1.9
 - JESJOBS resource class 88
 - JESSPOOL resource class 89, 91
 - OPERCMDS resource class 89, 90
 - SURROGAT resource class 88
 - using functions of RACF 2.1
 - STARTED resource class 84
- RACF security
 - Dynamic Workload Console and z/OS connector 213
- ready-list layout table 126
- RECEIVE processing 307
- Red Hat Enterprise Linux V5, (x86-64), install or uninstall of the Dynamic Workload Console failing on 263
- Red Hat Enterprise Linux V5, installation failing on (JVM) 263
- related software 5
- remove
 - See also* uninstallation
 - Dynamic Workload Console manually 258
- removing the product
 - Tivoli Workload Scheduler for z/OS Agent 297
 - twsinst 298
- removing the Tivoli Workload Scheduler for z/OS Agent
 - silent 298
 - wizard 297
- required maintenance
 - See* maintenance
- requirements 3
 - hardware 3
 - software 4
- resource description data set (EQQRDDS) 95
- response file automatically generated
 - installing the z/OS connector 203
- response file missing, causing silent installation to fail 264
- response file template
 - installing the z/OS connector 202
- response files
 - silent installation 273
 - template 274
- REST_NOTIFICATION_ADDRESS response file property 348
- RHEL
 - See* Red Hat Enterprise Linux
- RHEL 5 (x86-64), install or uninstall of the Dynamic Workload Console failing on 263
- RHEL V5 and Suse V11, installation failing on (JVM) 263
- roles
 - authorization z/OS connector 197
 - authorization z/OS connector on WebSphere Application for z/OS 209

S

- SAF (system authorization facility) 12, 86
- sample JCL, creating 55
- sample library (SEQQSAMP) 13, 301
- sample to migrate the history data base (EQQICNVH) 122
- SAS_SSL_SERVERAUTH_LISTENER_ADDRESS response file property 348
- SCHEDnn member of SYS1.PARMLIB 82
- schedules
 - See* job streams
- script
 - webui 238
- scripts
 - See* commands and scripts
- secure communications
 - Dynamic Workload Console and z/OS connector 213
 - RACF security 213
- security 12
 - APPL resource class 87
 - batch jobs 84
 - IBMOPC resource class 86
 - modifying 84, 86
 - router table 86
 - SAF (system authorization facility) 86
 - STARTED resource class 84
 - started task 86
 - system authorization facility (SAF) 86
 - user ID of the Tivoli OPC address space 84
 - user ID of Tivoli OPC submitted jobs 84
 - using functions of RACF 1.9
 - JESJOBS resource class 88
 - JESSPOOL resource class 89, 91
 - OPERCMDS resource class 89, 90
 - SURROGAT resource class 88
- security, problem determination procedures 154
- SEQQSAMP (sample library) 13, 301
- server
 - activating 137
 - checking the message log 152
 - description 7
 - optional data sets 120
 - required data sets 118
 - sample configuration 324
 - updating APPC/MVS options 140
- setup file
 - Dynamic Workload Console installation 228
- shared Windows folder, installation fails from 262
- side information data set (EQQSIDS) 95
- silent
 - uninstalling 298
- silent installation 273
 - Dynamic Workload Console 235
 - of the Dynamic Workload Console 227
 - response file template 274
- silent installation of the Dynamic Workload Console problems with 264
- silent mode
 - installing the z/OS connector 202, 211
- silent uninstall
 - of the Dynamic Workload Console 255
- SMF exits, installing 72
- SMF parameters (SMFPRMnn) 77, 79
- SMFPRMnn member of SYS1.PARMLIB 77, 79
- SOAP_CONNECTOR_ADDRESS response file property 348
- software compatibility 5
- Software Distribution
 - authorization roles 270
- space, disk
 - See* disk space
- SRSTAT command 82
- standby controller
 - verifying installation 154
- Standby controller
 - description 9
- STARTED resource class 84
- started-task operations, implementing support for 117
- started-task procedure
 - controller 116
 - tracker 116
- started-task-submit data set (EQQSTC) 101
 - considerations when allocating 110
- starting
 - console 247
 - Dynamic Workload Console 242
 - server 242
- stopping
 - Dynamic Workload Console 242
 - server 242
- stopWas command hangs during install of the Dynamic Workload Console 260
- submit checkpointing 15
- submit/release data set (EQQSUDS) 17, 101
 - considerations when allocating 110
- subsystem
 - APPL resource class 87
 - name table (IEFSSNnn) 75
- subtasks, activating 10
- Sun
 - See* Solaris
- supported operating systems for wizard 271
- SURROGAT RACF resource class 88
- Suse Linux V11, installation failing on (JVM) 263
- switch manager, fault-tolerant
 - See* backup domain manager
- syntax
 - wdinstsp to add runtime environment for Java jobs 285
 - wdinstsp to add the Java runtime to run job types with advanced options 294
- syntax agent installation
 - wdinstsp 284
- syntax CIT installation
 - wdinstsp 283, 292
- syntax to add dynamic capabilities
 - wdinstsp 285
- SYS1.PARMLIB
 - APPC/MVS options (APPCPMnn) 83, 137, 140
 - defining subsystems (IEFSSNnn) 75
 - dispatching priority 82
 - dynamic exits (PROGnn) 77, 79
 - EQQMINOx, authorizing for TSO (IKJTSONn) 81
 - Hiperbatch support (COFDLFnn) 82
 - link library (LNKLSTnn) 80
 - load module library (IEAAPFnn) 77, 80
 - performance (SCHEDnn) 82
 - program properties table (PPT) 82
 - SMF parameters (SMFPRMnn) 77
 - starting Tivoli OPC (COMMNDnn) 83
 - SYSMDUMP 79
 - updating dump content definitions 79
 - XCF initialization options (COUPLEnn) 80
- SYS1.PROCLIB
 - controller 116

- SYS1.PROCLIB (*continued*)
 - tracker 116
- SYSMDUMP (dump data set) 79, 101
 - considerations when allocating 104
- sysplex
 - configuration examples 317
- sysplex (system complex) 321
- system authorization facility (SAF) 12, 86
- system complex (sysplex) 321

T

- technical training
 - See* Tivoli technical training
- template response file
 - installing the z/OS connector 202
- text files, used for backup and restore
 - See* files
- Tivoli Dynamic Workload Console
 - configuration 245
 - getting started 247
 - overview 225
 - starting and stopping 242
 - troubleshooting 257
 - uninstall 255
 - updating
 - authentication mechanism 251
 - upgrading
 - overview 251
 - user interface 238
- Tivoli Integrated Portal
 - installation
 - from the DVD or eImages 237
- Tivoli OPC server
 - activating support for 137
 - server support 137
- Tivoli technical training xv
- Tivoli Workload Scheduler 225
 - available functions 239
 - engine connection 240
- Tivoli Workload Scheduler agents uninstalling
 - wizard 297
- Tivoli Workload Scheduler for Applications 225
- Tivoli Workload Scheduler for z/OS 225
- Tivoli Workload Scheduler for z/OS Agent 276
 - installation
 - Dynamic workload broker host name 273
 - Dynamic workload broker HTTPS port number 273
 - host name or IP address 273
 - JobManager port 273
 - user name and password 272
 - on IBM i 276
 - runtime for application job plug-ins 284
 - uninstalling 297
 - upgrading 287
 - upgrading with installation wizard 288
- Tivoli Workload Scheduler for z/OS Agent uninstalling
 - silent 298
 - twinsinst 298
 - wizard 297
- Tivoli Workload Scheduler Version 8.3 Fix Pack 3
 - configuration
 - for Dynamic Workload Console 237
- Tokensrv
 - See* Tivoli Token Service
- tracker
 - description 6
- tracker (*continued*)
 - loading software 53
 - optional data sets 120
 - RACF 84
 - started task data sets 118
 - verifying installation 145
- tracklog data set (EQQTROUT) 108
- training
 - See also* Tivoli technical training
 - technical xv
- troubleshooting
 - installation scenarios
 - Dynamic Workload Console 259
 - z/OS connector 219
- TSO
 - IKJTSONn member of SYS1.PARMLIB
 - AUTHCMD statement, updating for Tivoli OPC TSO
 - commands 82
 - AUTHTSF statement, updating for EQQMINOx 81
 - RACF user 87
- TSO commands
 - BACKUP 82
 - BULKDISC 82
 - OPINFO 82
 - OPSTAT 82
 - SRSTAT 82
 - WSSTAT 82
- TWA_INSTANCE_PATH response file property 348
- twinsinst 276
 - authorization roles 270
 - installation method 276
 - uninstalling 298
 - usage 276
- twPortsPanel.portAdmin response file property 341
- twPortsPanel.portAdminSec response file property 341
- twPortsPanel.portHTTP response file property 342
- twPortsPanel.portHTTPS response file property 342
- twPortsPanel.portMtlAuth response file property 342
- twPortsPanel.portORB response file property 342
- twPortsPanel.portRMI response file property 342
- twPortsPanel.portSAS response file property 342
- twPortsPanel.portSOAP response file property 342
- twPortsPanel.portSrvAuth response file property 342
- twUpgradePanel.bckpDirectory response file property 342

U

- uninstall
 - Dynamic Workload Console 255
 - manually 258
 - of the Dynamic Workload Console
 - in silent mode 255
 - using response file 255
- uninstallation
 - Dynamic Workload Console log files 257
 - of the Dynamic Workload Console, fails to start on Linux
 - RHEL 5 (x86-64) 263
 - of the Dynamic Workload Console, problems with 265
- uninstalling
 - authorization roles z/OS connector 197
 - authorization roles z/OS connector on WebSphere
 - Application for z/OS 209
 - Tivoli Workload Scheduler for z/OS Agent 297
 - z/OS connector 197
 - z/OS connector on WebSphere Application for z/OS 209

- uninstalling on WebSphere Application Server for z/OS
 - z/OS connector using the Integrated Solutions Console 217
 - z/OS connector using zConnUninstall.sh 217
- uninstalling Tivoli Workload Scheduler for z/OS Agent
 - silent 298
 - twinsinst 298
 - wizard 297
- UPDATE_INSTALLER_DIR response file property 348
- updateWas, using to update the SOAP properties after changing application server user or password 221
- updating
 - Tivoli Dynamic Workload Console authentication mechanism 251
- upgrade
 - of the Dynamic Workload Console, problems with 264
- upgrading
 - authentication
 - Tivoli Dynamic Workload Console 251
 - authorization roles z/OS connector 197
 - Dynamic Workload Console
 - on embedded WebSphere Application Server 252
 - overview 251
 - Tivoli Workload Scheduler for z/OS Agent 287
 - Tivoli Workload Scheduler for z/OS Agent with installation wizard 288
 - z/OS connector 197, 204
 - z/OS connector from version 8.3 using the wizard 204
 - z/OS connector from version 8.5 or 8.5.1 using the wizard 205
 - z/OS connector using the wizard 204
- usage
 - twinsinst 276
- user name
 - creating 282
- userUnixCfgPanel.inputUserName response file property 342
- userUnixCfgPanel.twsPassword response file property 342
- userWinCfgPanel.inputUserName response file property 342
- userWinCfgPanel.twsPassword response file property 342

V

- variables
 - Software Package Block
 - backup 281
 - display_name 281
 - domain 281
 - fresh_install 281
 - group 281
 - host_name 281
 - install_dir 282
 - installer 282
 - jm_port 282
 - jm_sec_port 282
 - pwd 282
 - tdwb_hostname 282
 - tdwb_port 282
 - twinsinst 282
 - upgrade 283
- VARY ACT command 133
- verifying installation 145
 - configuration 158
 - controller 151
 - standby controller 154
 - submit events 169
 - tracker 145
 - tracking events 147

- VTAM
 - activate network resources 133
 - class of service table 133
 - defining NCF 131
 - logon mode table 132
 - VARY ACT command 133

W

- WAS_CELL_NAME response file property 349
- WAS_NODE_NAME response file property 349
- WAS_PROFILE_NAME response file property 349
- WAS_SERVER_NAME response file property 349
- WC_adminhost response file property 349
- WC_adminhost_secure response file property 349
- WC_defaulthost response file property 349
- WC_defaulthost_secure response file property 349
- wdinstsp
 - syntax agent installation 284
 - syntax CIT installation 283, 292
 - syntax to add dynamic capabilities 285
 - syntax to add runtime environment for Java jobs 285
 - syntax to add the Java runtime to run job types with advanced options 294
- Web User Interface
 - See Dynamic Workload Console
- WebSphere Application for z/OS
 - installing the z/OS connector using zConnInstall.sh 211
- WebSphere Application for z/OS installing
 - z/OS connector 209
- WebSphere Application for z/OS uninstalling
 - z/OS connector on 209
- WebSphere Application for z/OS upgrading
 - z/OS connector on 209
- WebSphere Application Server
 - See also application server
 - choosing instance 231
 - ports 233
- WebSphere Application Server for z/OS
 - uninstalling the z/OS connector using zConnUninstall.sh 217
- WebSphere Application Server, installation of the Dynamic Workload Console fails when installing on different profile 264
- webui
 - script 238
- Windows
 - installation of the Dynamic Workload Console fails on different external WebSphere Application Server profile 264
 - shared folder, installation fails from 262
 - undefined error message displayed from launchpad 260
- wizard
 - installing the z/OS connector 200
 - uninstalling 297
 - upgrading the z/OS connector 204
 - upgrading the z/OS connector version 8.3 204
 - upgrading the z/OS connector version 8.5 or 8.5.1 205
- wizard, supported operating systems 271
- workload restart (WLR)
 - description 18
 - examples 318, 321, 323, 326, 327, 329
- workstation
 - destination 17
- workstation and calendar data set (EQQWSDS) 96
- WSSTAT command 82

X

XCF (cross-system coupling facility)
 groups 129
 including 129
 initialization statements 130
 MVS initialization options 80
 run time options 130

Z

z-centric agent
 runtime for application job plug-ins 285
z/OS connector
 applying maintenance 216
 authorization roles installing 197
 authorization roles installing on WebSphere Application for
 z/OS 209
 authorization roles uninstalling 197
 authorization roles uninstalling on WebSphere Application
 for z/OS 209
 authorization roles upgrading 197
 authorization roles upgrading on WebSphere Application
 for z/OS 209
 installation and uninstallation log files 199, 213
 installation methods 198
 installing 197
 installing from the launchpad 204
 installing in silent mode 202
 installing on WebSphere Application for z/OS 209
 installing with automatically generated response file 203
 installing with response file template 202
 installing with the wizard 200
 maintaining 219
 on WebSphere Application Server for z/OS business
 scenario 209
 troubleshooting 219
 uninstalling 197
 uninstalling on WebSphere Application for z/OS 209
 upgrading 197, 204
 upgrading on WebSphere Application for z/OS 209
 upgrading with the wizard 204
 upgrading with the wizard from version 8.3 204
 upgrading with the wizard from version 8.5 or 8.5.1 205
z/OS connector and Dynamic Workload Console
 secure communications 213
z/OS connector on WebSphere Application for z/OS
 installation methods 209
 installing with the Integrated Solutions Console 210
 installing with zConnInstall.sh 211
z/OS connector on WebSphere Application Server for z/OS
 secure communications 213
 uninstalling with the Integrated Solutions Console 217
 uninstalling with zConnUninstall.sh 217
z/OS connector on zWAS
 secure communications 213
zConnInstall.sh
 installing the z/OS connector 211
zConnUninstall.sh
 uninstalling the z/OS connector 217
ZOSInstanceConfiguration.engineName response file
 property 343
ZOSInstanceConfiguration.remoteHost response file
 property 343
ZOSInstanceConfiguration.remotePort response file
 property 343

ZOSInstanceConfiguration.ZOSToConfigure response file
 property 343



Product Number: 5698-A17

Printed in USA

SC32-1264-06



Spine information:

Workload Scheduler for z/OS

Version 8.6

IBM Tivoli Workload Scheduler for z/OS: Planning and Installation

