

Tivoli Workload Scheduler
Version 8.5.1 (Revised October 2010)

Administration Guide

IBM

Tivoli Workload Scheduler
Version 8.5.1 (Revised October 2010)

Administration Guide

IBM

Note

Before using this information and the product it supports, read the information in "Notices" on page 297.

This edition applies to version 8, release 5, modification level 1, Fix Pack 1 of IBM Tivoli Workload Scheduler (program number 5698-WSH) and to all subsequent releases and modifications until otherwise indicated in new editions.

This edition replaces SC23-9113-01

© **Copyright IBM Corporation 2001, 2010.**

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

Contents

List of figures	vii		Configuring the Tivoli Workload Scheduler agent	59
List of tables	ix		1 Configuring log and trace properties	60
About this publication	xi		Configuring common launchers properties	62
What is new in this release	xi		Configuring properties of the native job launcher	62
What is new in this release for administration	xi		Configuring properties of the Java job launcher	63
What is new in this publication for version 8.5.1	xi		Configuring properties of the Resource advisor agent	63
1 What is new in this publication for version 8.5.1.1	xi		Configuring properties of the System scanner	65
Who should read this publication	xii		Chapter 4. Configuring the Tivoli Dynamic Workload Console	67
Publications	xii		Launching in context with the Tivoli Dynamic Workload Console	67
Accessibility	xiii		Scenarios	67
Tivoli technical training	xiii		Advanced optional parameters	69
Support information	xiii		Configuring access to the Tivoli Dynamic Workload Console	73
Chapter 1. Getting started with administration	1		Configuring a User Registry	73
Where products and components are installed	1		Configuring roles to access to the Tivoli Dynamic Workload Console	77
Finding out what has been installed in which Tivoli Workload Automation instances	2		Configuring Tivoli Dynamic Workload Console to use Single Sign-On	79
Chapter 2. Customizing and configuring Tivoli Workload Scheduler	5		LTPA token-keys	80
Setting global options	5		Configuring the use of Lightweight Third-Party Authentication	80
Global options - summary	6		Configuring to use the same LTPA token_keys	81
Global options - detailed description	9		Disabling the automatic generation of LTPA token_keys	83
Setting local options	20		Configuring Tivoli Dynamic Workload Console to use SSL	84
Localopts summary	21		1 Customizing Tivoli Dynamic Workload Console (Advanced configuration)	84
Localopts details	23		1 Customizing your global settings	85
Local options file example	32		Configuring Tivoli Dynamic Workload Console to view reports	89
Setting user options	34		Configuring for a DB2 database	89
Sample useropts file	35		Configuring for an Oracle database	90
Multiple product instances	35		Configuring for an Informix Dynamic Server database	91
Configuring command-line client access authentication	36		Preventing a connection to specific Tivoli Workload Scheduler Version 8.3 engines	92
Connection parameters	36		Chapter 5. Configuring user authorization (Security file)	93
Entering passwords	37		Security management overview	93
Tivoli Workload Scheduler console messages and prompts	37		Getting started	94
Setting sysloglocal on UNIX	37		Updating the security file	94
console command	38		dumpsec	96
Enabling the time zone feature	38		makesec	97
Configuring to use the report commands	39		Centralized security management	98
Modifying jobmon service rights for Windows	39		Centralized security usage notes	98
Chapter 3. Configuring for dynamic scheduling	41		Configuring the security file	99
Configuring the dynamic workload broker instance on the master domain manager	41		Security file syntax	99
ResourceAdvisorConfig.properties file	42		Specifying user attributes	102
JobDispatcherConfig.properties file	44		Specifying object types	107
Archiving job data	45			
Configuring to schedule J2EE jobs	48			
Configuring security roles for users and groups	56			

Specifying object attributes	108
Specifying access	112
The <i>TWS_user</i> - special security file considerations	131
Sample security file	131
Configuring for LDAP	135
Example Configuration of an LDAP server for Microsoft Active Directory	136
1 Configuring RACF LDAP servers.	139
Using the Pluggable Authentication Module	141

Chapter 6. Network administration 143

Network overview	143
Network definitions	144
Network communications	145
Network links	145
Network operation	146
Network processes	147
Optimizing the network	151
Data volumes	151
Connectivity.	152
Planning space for queues	153
Tuning mailman servers	160
Netman configuration file	160
Determining internal Symphony table size.	162
Extended agents	162
UNIX extended agents	163
IP address validation	165
Support for Internet Protocol version 6.	165
Operating system configuration (UNIX only)	166
IP address validation messages	166
Impact of network changes	167

Chapter 7. Setting connection security 169

Connection security overview	169
Using an SSL connection for communication across the network by netman and conman	169
Setting up private keys and certificates	170
Creating Your Own Certification Authority	171
Creating private keys and certificates	172
Configuring SSL attributes	173
Configuring the SSL connection protocol for the network	174
Configuring full SSL security	175
Interface communication	178
Overview.	179
Customizing the connector configuration files	180
Changing a server key	182
Customizing the SSL connection for the Job Scheduling Console	183
Customizing the SSL connection for a command-line client	184
Command-line SSL connection settings	186
Working across firewalls.	186
LDAP connection authentication	187
Configuring the Job Scheduling Console to work in a Network Address Translation (NAT) environment	187
1 Federal Information Processing Standards	188
1 FIPS overview	188
1 Using FIPS certificates	190

Configuring SSL to be FIPS-compliant	193
Configuring DB2 for FIPS	196
Using Tivoli Dynamic Workload Console and FIPS	198
Configuring dynamic workload broker for FIPS	199
Configuring LDAP for FIPS	199
Finding the GSKit version on agents running on UNIX and Linux platforms	199

Chapter 8. Data maintenance. 201

Maintaining the database	201
Backing up and restoring	201
Reorganizing the database	203
Maintaining the file system.	204
Avoiding full file systems	204
Log files and archived files.	207
Temporary files.	209
Managing event message queue file sizes	209
Administrative tasks - DB2.	209
Changing DB2 passwords	209
Locating the DB2 tools	209
User permissions for running the DB2 tools	210
Administering the DB2 maintenance feature	210
Reorganizing the DB2 database	212
Monitoring the lock list memory	213
Administrative tasks - Oracle	215
Changing the Oracle access password	215
Locating the Oracle tools	215
Maintaining the Oracle database	216
Obtaining information about the Tivoli Workload Scheduler databases installed on an Oracle instance.	216
User permissions for running the Oracle tools	216
Migrating data from DB2 to Oracle and <i>vice versa</i>	217
Parallel data migration from DB2 to Oracle	217
Parallel data migration from Oracle to DB2	219
Reconfiguration from DB2 to Oracle.	220
Reconfiguration from Oracle to DB2.	225
Upgrading your database	231

Chapter 9. Administrative tasks 233

Changing a domain manager	234
Choosing a backup domain manager	235
Setting up a backup domain manager	235
Network security	235
Switching a domain manager	235
Changing a master domain manager	236
Choosing a workstation for backup master domain manager	236
Setting up a backup master domain manager	237
Copying files to use on the backup master domain manager	237
Switching a master domain manager	238
Extended loss or permanent change of master domain manager	238
1 Implications on dynamic workload broker.	239
Changing key Tivoli Workload Scheduler passwords	241
Determining the role of the user whose password has changed	242

Determining the actions to take	244	Backup usage	272
Action 1 - change the WebSphere Application Server user ID password	244	Restore usage	272
Action 2 - change password used by command-line clients to access the master domain manager	245	Application server - changing the host name or TCP/IP ports	273
Action 3 - change password used by fault-tolerant agent systems to access the master domain manager (for conman)	246	Changing host properties	274
Action 4 - update the engine connection parameters in the GUIs	246	Application server - changing the trace properties	276
Action 5 - change the j2c user ID password	246	Application server - using the utilities that change the properties - reference	277
Action 6 - UNIX - update SOAP properties	247	Understanding the templates	278
Action 7 - Windows - update Windows services	247	Application server - background information about the utilities	278
Action 8 - change the Tivoli Workload Scheduler Windows user definition	248	Chapter 10. Performance 281	
Using the changePassword script.	248	Network traffic	281
Unlinking and stopping Tivoli Workload Scheduler	250	Tracing	281
Changing the database host name, port, or database name	251	Logging	282
Change the DB2 host name, port, or database name	251	Maintaining the database	282
Changing the Oracle host name, port, or database name	257	Symphony file sizing	282
Changing the security settings.	258	Tuning a UNIX domain manager to handle large numbers of fault-tolerant agents	282
Changing security settings - general.	258	Tuning job processing on a workstation	282
Modify or display security settings - custom user registry.	263	Tuning the database	283
Managing the event processor.	263	Tuning the embedded WebSphere Application Server	284
Starting, stopping and displaying dynamic workload broker status	264	Inadequate Java heap size	284
Application server - starting and stopping.	264	Too many manual job submissions	284
Application server - automatic restart after failure	266	Too many file dependency checks	284
Appservman - how it works	266	Workload spreading	284
Controlling appservman	267	Improving job-processing performance	285
Starting and stopping the application server and appservman	268	Mailbox caching - advantages and disadvantages	285
1 Configuring user and password for running conman stopappserver	268	Setting the synch level parameter.	286
1 Monitoring the application server status	269	The fault-tolerant switch manager - impact on performance.	287
Obtaining information about application server failures	269	Network Traffic	287
Events created by appservman	269	Disk Space	287
Application server - encrypting the profile properties files	269	Scalability	288
Encrypting profile properties usage	270	Impact on JnextPlan	288
Application server - updating the Windows services after modifications.	270	Impact on reporting	288
Updating the embedded WebSphere Application Server services usage.	270	Impact on event rule deployment	289
Application server - updating the SOAP properties after changing the WebSphere Application Server user or its password	271	Increasing application server heap size	289
Updating the SOAP properties usage	271	Increasing maximum DB2 log capacity	289
Application server - configuration files backup and restore.	272	Multiple Tivoli Dynamic Workload Console production plan reports	292
		Tivoli Dynamic Workload Console - adjusting session timeout settings	293
		Chapter 11. Availability 295	
		Resolving Windows user ID account	295
		Using a temporary directory on UNIX	296
		Notices 297	
		Trademarks	298
		Index 301	

List of figures

1.	List of tasks	72	5.	Typical Tivoli Workload Scheduler network	
2.	Tivoli Workload Scheduler network domain		flows.	153	
	structure	143	6.	SSL server and client keys	180
3.	Symphony file synchronization	147			
4.	Process creation on domain manager and				
	fault-tolerant agent.	148			

List of tables

1.	Workload service assurance feature	6	1	29.	Syntax for special characters	87
2.	Event-driven workload automation feature - general	7	1	30.	Variables used in the URL definition	87
3.	Event-driven workload automation feature - event mailing	7		31.	Object attribute types for each object type	109
4.	SSL	7		32.	Access keywords for composer actions	114
5.	Job management	8		33.	Actions - access keywords	116
6.	Job stream management.	8		34.	Calendar - additional access keywords	117
7.	Stageman	8		35.	Cpus - additional access keywords	118
8.	Planman	8		36.	Events - access keywords	119
9.	Logging	9		37.	Files - access keywords	120
10.	General	9		38.	Jobs - additional access keywords.	121
11.	Shortcuts for encryption ciphers.	30		39.	Parameters - additional access keywords	124
12.	JOA_JOB_ARCHIVES database table	46		40.	Prompts - additional access keywords	125
13.	JRA_JOB_RESOURCE_ARCHIVES database table	47		41.	Files- access keywords	126
14.	MEA_METRIC_ARCHIVES database table	47		42.	Resources - additional access keywords	127
15.	Job statuses in the historical tables	48		43.	Job streams - additional access keywords	128
16.	J2EEJobExecutorConfig.properties file keywords	49		44.	Users - additional access keywords	129
17.	Workload broker workstation configuration properties	55	1	45.	Variable tables - access keywords	130
18.	Actors and their roles	56	1	46.	Critical flow errors.	154
19.	Trace and log configurable properties of the Tivoli Workload Scheduler agent	60		47.	Queue sizing conditions.	155
20.	Common launchers configurable properties of the Tivoli Workload Scheduler agent	62		48.	Example for the ge operator	156
21.	Native job launcher configurable properties of the Tivoli Workload Scheduler agent	63		49.	Example for the le operator	157
22.	Java job launcher configurable properties of the Tivoli Workload Scheduler agent	63		50.	Calculation of internal Symphony table	162
23.	Resource advisor agent configurable properties of the Tivoli Workload Scheduler agent	64		51.	Files for Local Options	173
24.	System scanner configurable properties of the Tivoli Workload Scheduler agent	65		52.	Type of communication depending on the securitylevel value	174
25.	Default port numbers	68		53.	Changes allowed in Tivoli Workload Scheduler key and trust stores	180
26.	Tivoli Workload Scheduler predefined roles	77		54.	Algorithm for calculating the approximate size of the plan data in the Symphony file	204
27.	Tivoli Dynamic Workload Broker predefined roles	79		55.	Algorithm for calculating the approximate size of the database data in the Symphony file	205
28.	Product versions and default server names	83	1	56.	Example for the ge operator	206
			1	57.	Example for the le operator.	206
				58.	Log and trace file maintenance.	207
				59.	If and where password changes are required	242
				60.	Values of activeUserRegistry to check	243
				61.	Password change actions.	244
				62.	Options for tuning job processing on a workstation	283

About this publication

IBM® Tivoli® Workload Scheduler: Administration provides information about the administration of the main components of IBM Tivoli Workload Scheduler (often called the *engine*).

What is new in this release

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

For information about the APARs that this release addresses, see the Tivoli Workload Scheduler Download Document at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg24024804>, and Tivoli Dynamic Workload Console Download Documents at <http://www.ibm.com/support/docview.wss?rs=672&uid=swg24024805>.

What is new in this release for administration

This section describes what has changed in this release with regard to administration since version 8.5:

- A new scheduling agent with z-centric and dynamic scheduling capabilities, the Tivoli Workload Scheduler agent, is available in this release.
- Support of Federal Information Processing Standards (FIPS) compliance for connection security is provided.

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

What is new in this publication for version 8.5.1

This section describes what has changed in this publication since version 8.5.

Changed or added text with respect to the previous version is marked by a vertical bar in the left margin. If the changed or added text applies only after a particular fix pack is installed, the mark in the left margin is the number of the fix pack. For example, 1 to refer to Fix Pack 1, and so on.

- A new chapter describing the configuration of the dynamic scheduling capabilities has been provided.
- A new section describing how to configure security for Federal Information Processing Standards (FIPS) compliance has been provided.
- The possibility to disable HTTP and Administration ports has been added to the Application Server **changeHostProperties** tool.
- A new script for changing key Tivoli Workload Scheduler passwords has been provided.

1 What is new in this publication for version 8.5.1.1

1 This section describes what has changed in this publication with regard to
1 administration since version 8.5.1:

What is new

- 1 • The tcp connect timeout local option was added in “Setting local options” on
1 page 20.
- 1 • Section “Configuring the Tivoli Workload Scheduler agent” on page 59 was
1 updated to include new log and trace management information for the Tivoli
1 Workload Scheduler agent.
- 1 • Section “Specifying object attributes” on page 108 was changed to add new
1 attributes for object job.
- 1 • Section “Configuring for LDAP” on page 135 was improved.
- 1 • A new section was added to “Network links” on page 145 to describe the new
1 chkh1tst service used to verify that the workstation mailbox can be successfully
1 read, and check if there are errors in the mailbox header.
- 1 • Sections “Monitoring the status of Tivoli Workload Scheduler processes” on page
1 150 and “Monitoring the size of Tivoli Workload Scheduler message queues” on
1 page 155 were added in Chapter 6, “Network administration,” on page 143.
- 1 • Section “Monitoring the disk space used by Tivoli Workload Scheduler” on page
1 205 was added in Chapter 8, “Data maintenance,” on page 201.
- 1 • Section “Configuring user and password for running conman stopappserver” on
1 page 268 was added in Chapter 9, “Administrative tasks,” on page 233.

1 All updates are marked with a 1 (one) on the left page margin.

1 For an update on the new or changed functions in this release, see: *IBM Tivoli*
1 *Workload Automation: Overview*.

Who should read this publication

This publication provides information about the day-to-day administration of the product, and is aimed at the IT administrator or Tivoli Workload Scheduler IT administrator whose job it is to ensure that the product runs smoothly and correctly. This person will find information about making routine changes to the configuration, for example to add a user, and information about periodic procedures that ensure the integrity of the product, such as backups.

The reader of this book should be an expert systems programmer, who has a reasonable understanding of the Tivoli Workload Scheduler infrastructure and its inter-component interactions.

Publications

Full details of IBM 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.

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 Tivoli Dynamic Workload Console, see the Accessibility Appendix in the *Tivoli Workload Scheduler: User's Guide and Reference*, SC32-1274.

For full information with respect to the Job Scheduling Console, see the Accessibility Appendix in the *Tivoli Workload Scheduler: Job Scheduling Console User's Guide*, SC32-1257.

Tivoli technical training

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

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

Support information

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

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>.

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.

Chapter 1. Getting started with administration

This publication describes how to perform administrative tasks on Tivoli Workload Scheduler and Tivoli Dynamic Workload Console. Many of the procedures described in it require you to identify a file in the installation path of the product and its components. However, they can have more than one installation path, as described in “Where products and components are installed.”

Where products and components are installed

This section describes how to determine where Tivoli Workload Scheduler components are installed, as follows:

Tivoli Workload Scheduler installation path

You can install more than one Tivoli Workload Scheduler component on a system. Each master domain manager, backup master domain manager, domain manager, backup domain manager, agent, or connector is installed in a path called a *Tivoli Workload Automation instance*. Only one component can be installed in each instance. The default path for the *Tivoli Workload Automation instance* is:

UNIX /opt/ibm/TWA<n>

Windows

C:\Program Files\ibm\TWA<n>

where <n> is an integer value ranging from <null> for the first instance installed, 1 for the second, and so on.

This path is called, in the publications, *TWA_home*

The installation path of Tivoli Workload Scheduler is:

TWA_home/TWS

Dynamic workload broker installation path

The files pertinent to workload broker are installed in the following path:

TWA_home/TDWB

Tivoli Dynamic Workload Console installation path

The Tivoli Dynamic Workload Console can be installed in more than one path:

- It can be installed alongside Tivoli Workload Scheduler or alone in a *Tivoli Workload Automation instance* using the embedded version of WebSphere® Application Server. In this case its path is:

TWA_home/TDWC

- It can be installed on your own external instance of WebSphere Application Server. In this case its path depends on where your instance of WebSphere Application Server is installed (except for the uninstaller, which is installed in a path of your choice). The administrative procedures in this publication do not address problems that occur with the external version of WebSphere Application Server.

If you are using the Tivoli Dynamic Workload Console on an external version of WebSphere Application Server, and an administrative procedure refers to the path *TWA_home/TDWC*, substitute it with the

Where products and components are installed

installation path of the Tivoli Dynamic Workload Console on your external version of WebSphere Application Server

The embedded WebSphere Application Server installation path

The embedded WebSphere Application Server is automatically installed when you create a new *Tivoli Workload Automation instance*. Its installation path is:

```
TWA_home/eWAS
```

The command line client installation path

The command line client is installed outside all *Tivoli Workload Automation instances*. Its default path is:

```
UNIX /opt/ibm/TWS/CLI
```

Windows

```
C:\Program Files\IBM\TWS\CLI
```

The application server tools installation path

Because the embedded WebSphere Application Server is not supplied with an administration GUI, many of its administration tasks are performed by running tools supplied with Tivoli Workload Scheduler, that perform the required configuration changes. These tools are known as the *wastools*, and are installed in:

```
TWA_home/wastools
```

However, the information above supplies only the *default* paths. To determine the actual paths of products and components installed in Tivoli Workload Automation instances, see “Finding out what has been installed in which Tivoli Workload Automation instances”

Finding out what has been installed in which Tivoli Workload Automation instances

If you are not the installer of Tivoli Workload Scheduler and its components, you might not know what components have been installed, and in which instances of Tivoli Workload Automation. Follow this procedure to find out:

1. Access the following directory:

```
UNIX /etc/TWA
```

Windows

```
%windir%\TWA
```

2. List the contents of the directory. Each Tivoli Workload Automation instance is represented by a file called: *twainstance<instance_number>.TWA.properties*. These files are deleted when all the products or components in an instance are uninstalled, so the number of files present indicates the number of valid instances currently in use.
3. Open a file in a text viewer.

Attention: Do not edit the contents of this file, unless directed to do so by IBM Software Support. Doing so might invalidate your Tivoli Workload Scheduler environment.

The contents are similar to this:

```
#TWAInstance registry
#Mon Nov 24 15:35:02 CET 2008
TWS_version=8.5.0.00
EWas_basePath=C\:/Program Files/IBM/TWA/eWAS
TWS_counter=1
EWas_counter=2
```

Where products and components are installed

```
TWA_path=C:/Program Files/IBM/TWA
TWS_server_name=twaserver
TDWC_version=8.5.0.0
TWS_instance_type=MDM
EWas_profile_path=C:/Program Files/IBM/TWA/eWAS/profiles/twaprofile
EWas_node_name=DefaultNode
TWS_basePath=C:\\Program Files\\IBM\\TWA\\TWS
EWas_user=twuser85
EWas_cell_name=DefaultNode
TDWC_EXTERNAL_WAS_KEY=false
EWas_version=6.1.0.19
TDWC_counter=1
EWas_server_name=twaserver
EWas_update_installer_dir=C:/Program Files/IBM/WebSphere/UpdateInstaller
TDWC_basePath=C:/Program Files/IBM/TWA/TDWC
TWS_user_name=twuser85
TWS_FIX_LIST_KEY=
TDWC_FIX_LIST_KEY=
TWA_componentList=TWS,EWas,TDWC
EWas_isc_version_key=7.1.0.06
EWas_profile_name=twaprofile
EWas_service_name=twuser85
```

The important keys to interpret in this file are:

TWA_path

This is the base path, to which the installation added one or more of the following directories, depending on what was installed:

- TWS** Where the Tivoli Workload Scheduler component is installed
- TDWC** Where the Tivoli Dynamic Workload Console is installed
- eWAS** Where the embedded WebSphere Application Server is installed

wastools

Where the tools that you use to configure embedded WebSphere Application Server are installed

- ssm** Where the Netcool[®] SSM monitoring agent is installed (used in event management)

TWA_componentList

Lists the components installed in the instance of Tivoli Workload Automation

TWS_counter

Indicates if a Tivoli Workload Scheduler component is installed in this instance of Tivoli Workload Automation (when the value=1)

TWS_instance_type

Indicates which component of Tivoli Workload Scheduler is installed in this instance:

- MDM** Master domain manager
- BKM** Backup master domain manager
- FTA** Agent or domain manager

TDWC_counter

Indicates if an instance of Tivoli Dynamic Workload Console is installed in this instance of Tivoli Workload Automation (when the value=1)

Where products and components are installed

EWas_counter

Indicates how many applications are installed in this instance of Tivoli Workload Automation that access the embedded WebSphere Application Server

TWS_user_name

The ID of the *<TWS_user>* of the Tivoli Workload Scheduler component.

EWas_user

The ID of the administration user of the embedded WebSphere Application Server. For a default installation, this is the same as the *<TWS_user>*.

The only component of Tivoli Workload Scheduler which is installed in a Tivoli Workload Automation instance, but which is not explicitly indicated here, is the Connector. To determine if it has been installed, look at the following combinations of keys:

Agent installed with no Connector

```
TWS_counter=1
EWas_counter=
TWS_instance_type=FTA
TDWC_counter=
TWA_componentList=TWS
```

Agent installed with Connector

```
TWS_counter=1
EWas_counter=1
TWS_instance_type=FTA
TDWC_counter=
TWA_componentList=TWS,EWas
```

Agent installed with no Connector and Tivoli Dynamic Workload Console

```
TWS_counter=1
EWas_counter=1
TWS_instance_type=FTA
TDWC_counter=1
TWA_componentList=TWS,EWas,TDWC
```

Agent installed with Connector and Tivoli Dynamic Workload Console

```
TWS_counter=1
EWas_counter=2
TWS_instance_type=FTA
TDWC_counter=1
TWA_componentList=TWS,EWas,TDWC
```

Note: The only difference between these last two is that the *EWas_counter* is 2 instead of 1.

Chapter 2. Customizing and configuring Tivoli Workload Scheduler

After installing the product you can customize it to fit your operational requirements. You can also change the customized values at any time. This chapter describes the optional customization steps for Tivoli Workload Scheduler. It is divided into the following sections:

- Chapter 1, “Getting started with administration,” on page 1
- “Setting global options”
- “Setting local options” on page 20
- “Setting user options” on page 34
- “Configuring command-line client access authentication” on page 36
- “Tivoli Workload Scheduler console messages and prompts” on page 37
- “Enabling the time zone feature” on page 38
- “Configuring to use the report commands” on page 39

Note: For information about automating the production cycle and managing the production environment, see the *User's Guide and Reference*.

Setting global options

Set global options using the **optman** command. Note that you must have the following security permissions for the global options file in the Tivoli Workload Scheduler security file to work with this command:

- For `optman ls` or `optman show`:
FILE NAME=GLOBALOPTS ACCESS=DISPLAY
- For `optman chg`:
FILE NAME=GLOBALOPTS ACCESS=MODIFY

See Chapter 5, “Configuring user authorization (Security file),” on page 93 for more information on the security file.

To view the values of the global options, use the following command:

```
optman ls
```

Note: You must run **JnextPlan** for most of the following options to take effect. Some of these options require a restart of the WebSphere Application Server. These actions are indicated in the option descriptions. See *Tivoli Workload Scheduler: User's Guide and Reference* for more information on the **JnextPlan** command.

To display information about an option, use the following command:

```
optman show {optionName | optionShortName}
```

where **optionName** or **optionShortName** are the options you want to view.

Setting global options (optman)

To change the values of an option, use the following command:

```
optman chg {optionName | optionShortName} = value
```

where **optionName** or **optionShortName** are described in the following sections:

- “Global options - summary”
- “Global options - detailed description” on page 9

Global options - summary

This section summarizes the global options that are managed by **optman**. The columns in the tables have the following meanings:

Description

The brief description of the option

Name The **optionName** as used in the **optman** commands.

Short name

The **optionShortName** as used in the **optman** commands.

Default

The default value that is applied to the option at installation (if present).

Range The range or choice of values you can supply (where appropriate).

Units The units that apply to the default and range.

Effect How to make any changes effective. The following codes have been used:

E If you are enabling the option, start the Event Processor. If you are disabling the option, stop the Event Processor.

Imm

The change is effective immediately

Imm (DB)

The change is effective immediately in the database only.

J Run **JnextPlan**.

J (Plan)

Run **JnextPlan** - it makes the change effective in the plan only.

NSM

The change is effective on the next send mail action.

W Restart the WebSphere Application Server

Global options grouped by feature or function

The following tables summarize the options for managing the features and functions of Tivoli Workload Scheduler:

Table 1. Workload service assurance feature

Description	Name	Short name	Default	Range	Units	Effect
Enable workload service assurance	enWorkloadServiceAssurance	wa	yes	yes, no	boolean	J
Approaching late offset	approachingLateOffset	al	120	>=0	seconds	J or W
Deadline offset	deadlineOffset	do	2	>=0	minutes	J or W
Promotion offset	promotionOffset	po	120	>=0	seconds	J

Table 2. Event-driven workload automation feature - general

Description	Name	Short name	Default	Range	Units	Effect
Enable event driven workload automation	enEventDrivenWorkloadAutomation	ed	yes	yes, no	boolean	J or E
Rules deployment frequency	deploymentFrequency	df	5	0-60	minutes	Imm
Enable event processor HTTPS protocol	enEventProcessorHttpsProtocol	eh	yes	yes, no	boolean	J
Tivoli event integration facility port	eventProcessorEIFPort	ee	31131	0 - 65535	port number	W and J
Tivoli Enterprise Console® server name	TECServerName	th	localhost		name	J
Tivoli Enterprise Console server port	TECServerPort	tp	5529	0 – 65535	port number	J

Table 3. Event-driven workload automation feature - event mailing

Description	Name	Short name	Default	Range	Units	Effect
Mail sender name	mailSenderName	ms	TWS		name	NSM
SMTP server name	smtpServerName	sn	localhost		name	Imm
SMTP Server port	smtpServerPort	sp	25	0 – 65535	port number	NSM
Mail plug-in uses SMTP authentication	smtpUseAuthentication	ua	no	yes, no	boolean	Imm
SMTP user name	smtpUserName	un	TWS_user		name	Imm
SMTP user password	smtpUserPassword	up				Imm
Mail plug-in uses SSL	smtpUseSSL	us	no	yes, no	boolean	Imm
Mail plug-in uses TLS protocol	smtpUseTLS	tl	no	yes, no	boolean	Imm

Table 4. SSL

Description	Name	Short name	Default	Range	Units	Effect
Enable the SSL full connection	enSSLFullConnection	sf	no	yes, no	boolean	J
Enable strong password encryption	enStrEncrypt	se	no	yes, no	boolean	J

Setting global options (optman)

Table 5. Job management

Description	Name	Short name	Default	Range	Units	Effect
Maximum prompts after abend	baseRecPrompt	bp	1000	0 – 65535	prompts	J
Additional prompts after abend	extRecPrompt	xp	1000	0 – 65535	prompts	J
Automatically grant logon as batch	enLogonBatch	lb	no	yes, no	boolean	J
Long duration job threshold	longDurationThreshold	ld	150	100 - 1000	seconds	J or W

Table 6. Job stream management

Description	Name	Short name	Default	Range	Units	Effect
Job streams without jobs policy	enEmptySchedsAreSucc	es	no	yes, no	boolean	J
Prevent job stream without "at" dependency from starting	enPreventStart	ps	yes	yes, no	boolean	J

Table 7. Stageman

Description	Name	Short name	Default	Range	Units	Effect
Carry job states	carryStates	cs	null		list of states	J
Enable carry forward	enCarryForward	cf	all	all, no	boolean	J
Enable carry forward for internetwork dependencies	enCFinterNetworkDeps	ci	yes	yes, no	boolean	J
Enable carry forward resource quantity	enCFResourceQuantity	rq	yes	yes, no	boolean	J
Retain rerun job name	enRetainNameOnRerunFrom	rr	no	yes, no	boolean	J

Table 8. Planman

Description	Name	Short name	Default	Range	Units	Effect
Maximum preproduction plan length	maxLen	xl	8	8 - 365	days	J
Minimum preproduction plan length	minLen	ml	8	7 - 365	days	J

Table 9. Logging

Description	Name	Short name	Default	Range	Units	Effect
Log cleanup frequency	logCleanupFrequency	lc	5	0 - 60	minutes	J
Log history period	logHistory	lh	10	>=0	days	J
Logman minimum and maximum run times policy	logmanMinMaxPolicy	lm	both		literal	J
Logman normal run time calculation policy	logmanSmoothPolicy	lt	-1	0 - 100	factor	J

Table 10. General

Description	Name	Short name	Default	Range	Units	Effect
Company name	companyName	cn			name	J
Enable centralized security	enCentSec	ts	no	yes, no	boolean	J
Enable database auditing	enDbAudit	da	0	0, 1	boolean	Imm
Enable previous job stream ID	enLegacyId	li	no	yes, no	boolean	J
Evaluate start-of-day	enLegacyStartOfDayEvaluation	le	no	yes, no	boolean	J
Enable list security check	enListSecChk	sc	no	yes, no	boolean	J (Plan) Imm (DB)
Enable plan auditing	enPlanAudit	pa	0	0, 1	boolean	J
Enable the fault-tolerant switch manager	enSwfaultTol	sw	no	yes, no	boolean	J
Enable time zones	enTimeZone	tz	yes	yes, no	boolean	J (Plan) Imm (DB)
Ignore calendars	ignoreCals	ic	no	yes, no	boolean	J
Start time of processing day	startOfDay	sd	0600	0000 -2359	hhmm	J
Job statistics history period	statsHistory	sh	10	>=0	days	J (Plan) Imm (DB)

Global options - detailed description

This section gives full descriptions of the global options managed by **optman**:

approachingLateOffset | al

Approaching late offset. Used in workload service assurance. The critical start time of a job in the critical network is the latest time that the job can start without causing the critical job to finish after the deadline. In most cases, a job will start well before the critical start time so that if the job runs longer than its estimated duration, the situation does not immediately become critical. Therefore, if a job has not started and the critical start time is only a few minutes away, the timely completion of the critical job is considered to be potentially at risk.

Setting global options (optman)

The *approachingLateOffset* option allows you to determine the length of time before the critical start time of a job in the critical network at which you are to alerted to this potential risk. If a job has still not started the specified number of seconds before the critical start time, the job is added to a hot list that can be viewed on the Tivoli Dynamic Workload Console.

Note: To qualify for addition to the hot list, all time and follow dependencies must have been resolved.

This option is only active if *enWorkloadServiceAssurance* is set to *yes*.

The default is 120 seconds.

Note: Whatever value you set for this option, if Tivoli Workload Scheduler loses the connection with its database, the default value is applied to critical job processing, and the warning message AWSJCO135W is issued to tell you what has happened.

Run **JnextPlan** or restart the WebSphere Application Server (**stopappserver** and **startappserver**) to make this change effective.

baseRecPrompt | bp

Maximum prompts after abend. Specify the maximum number of prompts that can be displayed to the operator after a job abends.

The default value is 1000. Run **JnextPlan** to make this change effective.

carryStates | cs

Carry job states. A preproduction option that affects the operation of the *stageman* command. Specify the jobs, by state, to be included in job streams that are carried forward. Enclose the job states in parentheses, double quotation marks, or single quotation marks. Commas can be replaced by spaces. The valid internal job states are as follows:

<i>abend</i>	<i>abenp</i>	<i>add</i>	<i>done</i>	<i>exec</i>	<i>fail</i>
<i>hold</i>	<i>intro</i>	<i>pend</i>	<i>ready</i>	<i>rjob</i>	<i>sched</i>
<i>skel</i>	<i>succ</i>	<i>succp</i>	<i>susp</i>	<i>wait</i>	<i>waitd</i>

Some examples of the option are as follows:

```
carryStates="abend,exec,hold,intro"  
carryStates='abend,exec,hold,intro'  
carryStates="abend, exec, hold, intro"  
carryStates='abend, exec, hold, intro'
```

An empty list is entered as follows:

```
carryStates=null
```

The default value is *null*, which corresponds to selecting all states. Run **JnextPlan** to make this change effective.

companyName | cn

Company name. Specify the name of your company. The maximum length is 40 bytes. If the name contains spaces, enclose the name in quotation marks ("). If you use the Japanese-Katakana language set, enclose the name within single or double quotation marks.

Run **JnextPlan** to make this change effective.

deadlineOffset | do

Deadline offset. Used in workload service assurance. Used to calculate the critical start of a critical job in the case where a deadline has not been specified neither for the job nor its job stream. In this case the deadline is defaulted to the plan end date and time, plus this offset, expressed in minutes.

This option is only active if *enWorkloadServiceAssurance* is set to *yes*.

The default is 2 minutes.

Notes:

1. **Important:** When the plan is extended, the start time of critical jobs with a deadline calculated with this mechanism is automatically changed as a consequence of the fact that it must now match the new plan finishing time.
2. Whatever value you set for this option, if Tivoli Workload Scheduler loses the connection with its database, the default value is applied to critical job processing, and the warning message AWSJCO135W is issued to tell you what has happened.

Run **JnextPlan** or restart the WebSphere Application Server (**stopappserver** and **startappserver**) to make this change effective.

deploymentFrequency | df

Rules deployment frequency. Used in event rule management. Specify the frequency, in minutes, with which rules are to be checked to detect if there are changes to deploy. All active rules (active rules have the *isDraft* property set to *no* in their definition) that have been changed or added since the last deployment are deployed.

Valid values are in the 0-60 minutes range. If you specify 0, the changes are not deployed automatically and you must use the **planman deploy** command.

The default value is 5 minutes. The change is effective immediately.

enCarryForward | cf

Enable carry forward. A preproduction option that affects the operation of the *stageman* command. Specify if job streams that did not complete are carried forward from the old to the new production plan (Symphony). Enter *yes* to have uncompleted job streams carried forward only if the *Carry Forward* option is enabled in the job stream definition. Enter *all* to have all uncompleted job streams carried forward, regardless of the *Carry Forward* option. Enter *no* to completely disable the *Carry Forward* function. When the **stageman -carryforward** command is used, it overrides *enCarryForward*. See *Tivoli Workload Scheduler: User's Guide and Reference* for more information.

The default value is *all*. Run **JnextPlan** to make this change effective.

Setting global options (optman)

enCentSec | ts

Enable centralized security. Determine how the security file is used within the network. Centralized security is not relevant to an end-to-end scheduling environment.

If set to *yes*, the security files of all the workstations of the network can be created and modified only on the master domain manager. In this case, the Tivoli Workload Scheduler administrator is responsible for their production, maintenance, and distribution.

If set to *no*, the security file of each workstation can be managed by the root user or administrator of the system. The local user can run the *makesec* command to create or update the file.

See *Tivoli Workload Scheduler: User's Guide and Reference* for more information on centralized security.

The default value is *no*. Run **JnextPlan** to make this change effective.

enCFinterNetworkDeps | ci

Enable carry forward for internetwork dependencies. A preproduction option that affects the way **stageman** handles internetwork dependencies. It specifies if external job streams are carried forward from the old to the new production plan (Symphony file). Enter *yes* to have all external job streams carried forward. Enter *no* to have no external job streams carried forward.

The default value is *yes*. Run **JnextPlan** to make this change effective.

enCFResourceQuantity | rq

Enable carry forward resource quantity. A preproduction option that affects the way **stageman** handles resources. Enter *yes* to carry forward the resource quantity from the old production file to the new. Enter *no* to not carry forward the resource quantity. **Stageman** carries forward resource quantities only if the resource is needed by a job or job stream that is also being carried forward. Otherwise the resource quantities are set to the original value. See *Tivoli Workload Scheduler: User's Guide and Reference* for details on using this feature.

The default value is *yes*. Run **JnextPlan** to make this change effective.

enDbAudit | da

Enable database auditing. Enable or disable database auditing. To disable database auditing, specify *0*. To activate database auditing, specify *1*. Auditing information is logged to a flat file in the *TWA_home/TWS/audit/database* directory. Each Tivoli Workload Scheduler workstation maintains its own log. For the database, only actions are logged in the auditing file, not the success or failure of the action.

For more information on this feature, see *Enabling the time zone feature*.

The default value is *0*. Changes to this parameter are effective immediately.

enEmptySchedsAreSucc | es

Job streams without jobs policy. Specify the behavior of job streams without any jobs. If set to *yes*, the job streams that contain no jobs are set to SUCC after their dependencies are resolved. If set to *no*, the job streams are left in READY status.

The default value is *no*. Run **JnextPlan** to make this change effective.

enEventDrivenWorkloadAutomation | ed

Enable event driven workload automation. Enable or disable the event-driven workload automation feature. To enable, specify *yes*. To disable, specify *no*.

The default value is *yes*.

After disabling, you must run **JnextPlan** and stop the event processing server (with the **conman stopevtp** command).

After enabling, you must run **JnextPlan** and start the event processing server (with the **conman startevtp** command).

enEventProcessorHttpsProtocol | eh

Enable event processor HTTPS protocol. Used in event rule management. Enables or disables the use of the HTTPS protocol to connect to the event processor server. To enable, enter *yes*. To disable, enter *no*.

The default value is *yes*. Run **JnextPlan** to make this change effective.

enLegacyId | li

Enable previous job stream ID. Determine how job streams are to be named when operating in mixed environments with versions of Tivoli Workload Scheduler older than version 8.3, managed by a version 8.5 master domain manager. Use this option to keep consistency in identifying the job streams in the plan. The value assigned to this option is read either when the production plan is created or extended, or when submitting job streams in production using **conman**.

When the plan is created or extended, if this option is set to *no*, the job stream instance is assigned a new ID following the normal mechanism of Tivoli Workload Scheduler. In the Symphony file, the job stream name is equal to this ID. If the option is set to *yes*, the job stream instance is assigned an ID (symphony ID) equal to the job stream name. In the Symphony file the job stream name is equal to the real job stream name. If more instances of the same job stream are present, an ID is generated for every instance, with an alias that starts with the job stream name.

The default value is *no*. Run **JnextPlan** to make this change effective.

enLegacyStartOfDayEvaluation | le

Evaluate start-of-day. Specify how the *startOfDay* option is to be managed across the Tivoli Workload Scheduler network. If you set this option to *yes*, the *startOfDay* value on the master domain manager is converted to the local time zone set on each workstation across the network. If you set this option to *no*, the *startOfDay* value on the master domain manager is applied as is on

Setting global options (optman)

each workstation across the network. This option requires that the *enTimeZone* option is set to *yes* to become operational.

The default value is *no*. Run **JnextPlan** to make this change effective.

enListSecChk | sc

Enable list security check. Control the objects in the plan that a user is permitted to list when running a query on the Tivoli Dynamic Workload Console, the Job Scheduling Console, or a **conman show <object>** command. If set to *yes*, objects in the plan returned from a query or show command are shown to the user only if the user has been granted the list permission in the security file. If set to *no*, all objects are shown, regardless of the settings in the security file.

Note: Setting this option to *yes* affects how the graphical user interfaces function for the users defined in the security file.

The default value is *no*. Run **JnextPlan** to make this change effective for the plan. For the database, this option takes immediate effect.

enLogonBatch | lb

Automatically grant logon as batch. This is for Windows jobs only. If set to *yes*, the logon users for Windows jobs are automatically granted the right to *Logon as batch job*. If set to *no*, or omitted, the right must be granted manually to each user or group. The right cannot be granted automatically for users running jobs on a Backup Domain Controller, so you must grant those rights manually.

The default value is *no*. Run **JnextPlan** to make this change effective.

enPlanAudit | pa

Enable plan auditing. Enable or disable plan auditing. To enable plan auditing, specify *1*. To disable plan auditing, specify *0*. Auditing information is logged to a flat file in the *TWA_home/TWS/audit/plan* directory. Each Tivoli Workload Scheduler workstation maintains its own log. For the plan, only actions are logged in the auditing file, not the success or failure of any action. See *Tivoli Workload Scheduler: User's Guide and Reference* for more information on this feature.

The default value is *0*. Run **JnextPlan** to make this change effective.

enPreventStart | ps

Prevent job stream without "at" dependency from starting. Specify if job streams without an *at* dependency are to be prevented from starting immediately, without waiting for the run cycle specified in the job stream. Valid values are *yes* and *no*.

The default value is *yes*. Run **JnextPlan** to make this change effective.

enRetainNameOnRerunFrom | rr

Retain rerun job name. A production option that affects the operation of **Batchman**, the production control process of Tivoli Workload Scheduler. Its setting determines if jobs that are rerun

with the **Conman** *rerun* command retain their original job names. To have rerun jobs retain their original job names, enter *yes*. Enter *no* to assign the *rerun from* name to rerun jobs.

The default values is *no*. Run **JnextPlan** to make this change effective.

enSSLFullConnection | sf

Enable the SSL full connection. Specify that Tivoli Workload Scheduler uses a higher level of SSL connection than the standard level. For full details see “Configuring full SSL security” on page 175. Valid values are *yes* to enable the SSL full connection or *no* to disable the SSL full connection.

The default value is *no*. Run **JnextPlan** to make this change effective.

enStrEncrypt | se

Enable strong password encryption. Enable or disable strong encryption. Enable strong encryption by setting this option to *yes*. See “Configuring the SSL connection protocol for the network” on page 174.

The default value is *no*. Run **JnextPlan** to make this change effective.

enSwfaultTol | sw

Enable the fault-tolerant switch manager. Enable or disable the fault-tolerant switch manager feature. Valid values are *yes* to enable the fault tolerant switch manager, and *no* to disable it. See the *Tivoli Workload Scheduler: User’s Guide and Reference* for more details.

The default value is *no*. Run **JnextPlan** to make this change effective.

enTimeZone | tz

Enable time zones. Enables or disables the time zone option. To activate time zones in your network, specify *yes*. To disable time zones in the network, specify *no*. See Enabling the time zone feature.

The default value is *yes*. Run **JnextPlan** to make this change effective in the plan. For the database, this option takes effect immediately.

enWorkloadServiceAssurance | wa

Enable workload service assurance. Enables or disables workload service assurance, which is the feature that manages the privileged processing of mission critical jobs and their predecessors. Specify *yes* to enable or *no* to disable.

Note: Before starting to use workload service assurance you must set up the *TWS_user* in the security file to have the appropriate access to the objects that this feature will modify - see “The *TWS_user* - special security file considerations” on page 131

The default value is *yes*. Run **JnextPlan** to make this change effective.

eventProcessorEIFPort | ee

Tivoli event integration facility port. Used in event rule

Setting global options (optman)

management. Specify the port number where the event processor server receives events from the Tivoli Event Integration Facility (EIF). Valid values are in the 0-65535 range.

The default value is 31131. If you change the value, restart the WebSphere Application Server (**stopappserver** and **startappserver**) and run **JnextPlan** to make this change effective.

If you use a security firewall, make sure this port is open for incoming and outgoing connections.

extRecPrompt | xp

Additional prompts after abend. Specify an additional number of prompts for the value defined in *baseRecPropmt*. This applies when a job is rerun after abending and the limit specified in *baseRecPropmt* has been reached.

The default value is 1000. Run **JnextPlan** to make this change effective.

ignoreCals | ic

Ignore calendars. A preproduction option that affects the operation of the **planman** command. Its setting determines if user calendars are copied into the new production plan (Symphony) file. To prevent user calendars from being copied into the new production plan, enter *yes*.

The default value is *no*. See *Tivoli Workload Scheduler: User's Guide and Reference*. Run **JnextPlan** to make this change effective.

logCleanupFrequency | lc

Log cleanup frequency. Used in event rule management. Specify how often the automatic cleanup of log instances is run. Valid values are in the 0-60 minutes range. If you specify 0, the automatic cleanup feature is disabled.

The default value is 5 minutes. This option takes effect immediately.

logHistory | lh

Log history period. Used in event rule management. Enter the number of days for which you want to save rule instance, action run, and message log data. Log instances are discarded on a FIFO (first-in first-out) basis.

The default value is 10 days. This option takes effect immediately.

logmanMinMaxPolicy | lm

Logman minimum and maximum run times policy. Specify how the minimum and maximum job run times are logged and reported by **logman**. Possible values are:

elapsedtime The minimum and maximum elapsed runtimes are logged and reported.

cputime The minimum and maximum CPU runtimes are logged and reported.

both Both the minimum and maximum job runtimes are logged and reported.

See *Tivoli Workload Scheduler: User's Guide and Reference* for details on using this feature.

The default value is *both*. Run **JnextPlan** to make this change effective.

logmanSmoothPolicy | **lt**

Logman normal run time calculation policy. Set the weighting factor that favors the most recent job run when calculating the normal (average) run time for a job. This is expressed as a percentage. For example, specify *40* to apply a weighting factor of 40% to the most recent job run, and 60% to the existing average. See *Tivoli Workload Scheduler: User's Guide and Reference* for more information about how to use this option.

The default value is *-1*. Run **JnextPlan** to make this change effective.

longDurationThreshold | **ld**

Long duration job threshold. Specify, when comparing the actual duration of a job to the estimated duration, the threshold over which the job is considered to be of "long duration." The threshold value is expressed as a percentage with respect to the estimated duration. For example, if the threshold is set to *150*, and the actual duration is more than 150% of the estimated duration (it is 50% greater), the job is considered to be a "long duration" job.

If you have the workload service assurance feature enabled, the effect of a "critical" job satisfying the long duration criteria is that the job is inserted automatically into the hot list.

Valid values are between:

100 The minimum value. All jobs that exceed the estimated duration are considered long duration jobs

1000 The maximum value. Only those jobs that last ten times as long as their estimated duration are considered as long duration jobs

The default is *150* seconds.

Note: Whatever value you set for this option, if you have the workload service assurance feature enabled, and Tivoli Workload Scheduler loses the connection with its database, the default value is applied to critical job processing, and the warning message AWSJCO135W is issued to tell you what has happened.

Run **JnextPlan** or restart the WebSphere Application Server (**stopappserver** and **startappserver**) to make this change effective.

mailSenderName | **ms**

Mail sender name. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify a string to be used as the sender of the e-mails.

The default value is *TWS*. Changes to this parameter are effective for the next mail send action performed.

maxLen | **xl**

Maximum preproduction plan length. Specify the maximum length of the preproduction plan in days after it is automatically extended or created. The value for *maxLen* must be greater than or equal to the value for *minLen* and must be in the range of 8 to 365.

The default is *8* days. Run **JnextPlan** to make this change effective.

Setting global options (optman)

minLen | ml **Minimum preproduction plan length.** Specify the minimum length in days of the preproduction plan that can pass after the production plan is created or extended, without extending the preproduction plan. If the days left in the preproduction plan after a **JnextPlan** are less than the value of this option, the preproduction plan is automatically extended. The value for *minLen* must be less than or equal to the value for *maxLen* and must be in the range of 7 to 365.

The default is 8 days. Run **JnextPlan** to make this change effective.

promotionOffset | po

Promotion offset. Used in workload service assurance. Specify when a job become eligible for promotion in terms of the number of seconds before its critical start time is reached. Applies only to jobs that are flagged as critical in a job stream definition and to their predecessor jobs. A critical job and its predecessors make up a critical network.

When a predecessor jeopardizes the timely completion of the critical job, it is *promoted*; that is, it is assigned additional resources and its submission is prioritized with respect to other jobs that are out of the critical network. Also critical jobs might be promoted.

The scheduler calculates the critical start time of a critical job by subtracting its estimated duration from its deadline. It calculates the critical start time of a critical predecessor by subtracting its estimated duration from the critical start time of its next successor. Within a critical network the scheduler calculates the critical start time of the critical job first and then works backwards along the chain of predecessors. These calculations are reiterated as many times as necessary until the critical job has run.

This option is only active if *enWorkloadServiceAssurance* is set to *yes*.

The default is 120 seconds.

Run **JnextPlan** to make this change effective.

smtpServerName | sn

SMTP server name. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify the name of the SMTP server to be used by the mail plug-in.

The default value is *localhost*. Changes to this parameter are effective immediately.

smtpServerPort | sp

SMTP Server port. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify the port number used to connect to the SMTP server by the mail plug-in. Valid values are in the range 0–65535.

The default value is 25. Changes to this parameter are effective for the next mail send action performed.

smtpUseAuthentication | ua

Mail plug-in uses SMTP authentication. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify if the SMTP connection needs to be authenticated. Values are *yes* or *no*.

The default is *no*. Changes to this parameter are effective immediately.

smtpUserName | un

SMTP user name. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify the SMTP user name.

The default value is the name of the Tivoli Workload Scheduler user (the TWS_user) on the master domain manager. Changes to this parameter are effective immediately.

smtpUserPassword | up

SMTP user password. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify the SMTP user password. The password is stored in an encrypted form.

Changes to this parameter are effective immediately.

smtpUseSSL | us

Mail plug-in uses SSL. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify if the SMTP connection is to be authenticated via SSL. Values are *yes* or *no*.

The default is *no*. Changes to this parameter are effective immediately.

smtpUseTLS | tl

Mail plug-in uses TLS protocol. Used in event rule management. If you deploy rules implementing an action that sends e-mails via an SMTP server, specify if the SMTP connection is to be authenticated via the Transport Layer Security (TLS) protocol. Values are *yes* or *no*.

The default is *no*. Changes to this parameter are effective immediately.

startOfDay | sd

Start time of processing day. Specify the start time of the Tivoli Workload Scheduler processing day in 24-hour format: *hhmm* (0000-2359).

The default value is *0600* (6:00 a.m.). If you change this option, you must also change the launch time of the *final* job stream, which is usually set to one minute before the start time: *0559* (5:59 a.m.). Run **JnextPlan** to make the change of *startOfDay* effective.

statsHistory | sh

Job statistics history period. Specify the number of days for which you want to maintain job statistics. Statistics are discarded on a FIFO (first-in first-out) basis. For example, if you leave the default value of *10*, statistics are maintained for the last 10 days. This has no effect on job standard list files, which must be removed with

Setting global options (optman)

the *rmstdlist* command. See the *Tivoli Workload Scheduler: User's Guide and Reference* for information about the *rmstdlist* command.

The default value is *10*. Run **JnextPlan** to make this change effective in the plan. For the database, this option takes effect immediately.

TECServerName | th

Tivoli Enterprise Console server name. Used in event rule management. If you use rules implementing an action that forwards events to a Tivoli Enterprise Console server (or any other application that can process events in Tivoli Enterprise Console format), specify the Tivoli Enterprise Console server name. You can change this value when you define the action if you want to use a different Tivoli Enterprise Console server.

The default is *localhost*. Run **JnextPlan** to make this change effective.

TECServerPort | tp

Tivoli Enterprise Console server port. Used in event rule management. If you use rules implementing an action that forwards events to a Tivoli Enterprise Console (TEC) server (or any other application that can process events in TEC format), specify the port number of the TEC server. You can change this value when you define the action if you want to use a different TEC server.

The default port number is *5529*. Run **JnextPlan** to make this change effective.

Setting local options

Set local options in the `localopts` file. Changes do not take effect until **netman** is stopped (**conman shut;wait**) and restarted (**StartUp**).

A template file containing default settings is located in `TWA_home/TWS/config/localopts`.

Note: All of the SSL settings in the `localopts` file relate to the network communications and do not relate to the Tivoli Dynamic Workload Console or the Job Scheduling Console.

During the installation process, a working copy of the local options file is installed as `TWA_home/TWS/localopts`.

The options in the `localopts` file are described in the following sections:

- “Localopts summary” on page 21
- “Localopts details” on page 23
- “Local options file example” on page 32

Localopts summary

General attributes of this workstation:

thiscpu = *workstation*
merge stdlists = *yes|no*
stdlist width = *columns*
syslog local = *facility*

The attributes of this workstation for the batchman process:

bm check deadline = *seconds*
bm check file = *seconds*
bm check status = *seconds*
bm check until = *seconds*
bm look = *seconds*
bm read = *seconds*
bm stats = *on|off*
bm verbose = *on|off*

The attributes of this workstation for the jobman process:

jm job table size = *entries*
jm look = *seconds*
jm nice = *value*
jm no root = *yes|no*
jm read = *seconds*
jm promoted nice = *UNIX and Linux critical job priority*
jm promoted priority = *Windows critical job priority*

The attributes of this workstation for the mailman process:

autostart monman = *yes|no*
mm cache mailbox = *yes|no*
mm cache size = *bytes*
mm resolve master = *yes|no*
mm response = *seconds*
mm retrylink = *seconds*
mm sound off = *yes|no*
mm unlink = *seconds*

The attributes of this workstation for the netman process:

nm mortal = *yes|no*
nm port = *port number*
nm read = *seconds*
nm retry = *seconds*

The attributes of this workstation for the writer process:

wr read = *seconds*
wr unlink = *seconds*
wr enable compression = *yes|no*

Optional attributes of this workstation for remote database files

mozart directory = *mozart_share*
parameters directory = *parms_share*
unison network directory = *unison_share*

The attributes of this workstation for the custom formats

date format = *integer*
composer prompt = *key*
conman prompt = *key*
switch sym prompt = *key*

Setting local options (localopts file)

The attributes of this workstation for the customization of I/O on mailbox files
`sync level = low|medium|high`

The attributes of this workstation for networking

`tcp timeout = seconds`
`tcp connect timeout = seconds`

The attributes of this workstation for SSL

`nm SSL port = value`
`SSL key = *.key`
`SSL key pwd = *.sth`
`SSL certificate = *.crt`
`SSL certificate chain = *.crt`
`SSL random seed = *.rnd`
`SSL auth mode = caonly|string|cpu`
`SSL auth string = string`
`SSL CA certificate = *.crt`
`SSL encryption cipher = shortcut`

The attributes of this workstation for the embedded WebSphere Application Server
`local was = yes|no`

Application server check attributes

`appserver check interval = minutes`
`appserver auto restart = on|off`
`appserver min restart time = minutes`
`appserver max restarts = number`
`appserver count reset interval = hours`
`appserver service name = name`

The Tivoli Workload Scheduler instance is a command line client
`is remote cli = yes|no`

Attributes for conman connection

`host = host_name`
`protocol = protocol`
`port = port number`
`proxy = proxy server`
`proxyport = proxy server port number`
`timeout = seconds`
`clisslserverauth = yes|no`
`clisslcipher = string`
`clissservercertificate = file_name`
`clissltrusteddir = directory_name`
`defaultws = master_workstation`
`useropts = useropts_file`

Event Management parameters

`can be event processor = yes|no`

Attribute for access restriction to stdlist directory (UNIX only)

`restricted stdlists = yes|no`

Note: The localopts file syntax is not case-sensitive.

Localopts details

comment Treats everything from the indicated sign (#) to the end of the line as a comment.

appserver auto restart
Requests the appservman process to automatically start WebSphere Application Server if it is found down. The default is ON.

appserver check interval
Specifies the frequency in minutes that the appservman process is to check that WebSphere Application Server is still running. The default is 5 minutes.

appserver count reset interval
Specifies the time interval in hours after which the restart count is reset from the last WebSphere Application Server start. The default is 24 hours.

appserver max restarts
Specifies the maximum number of restarting attempts the appservman process can make before giving up and exiting without restarting WebSphere Application Server. The counter is reset if WebSphere Application Server runs for longer than the appserver count reset interval value. The default is 5.

appserver min restart time
Specifies in minutes the minimum elapsed time the appservman process must wait between each attempt to restart the WebSphere Application Server if it is down. If this value is less than the appserver check interval, the WebSphere Application Server is restarted as soon as it is found down. If it is found down before this time interval (min restart time) has elapsed, appservman exits without restarting it. The default is 10 minutes.

appserver service name
Only in Windows environments. Specifies the name of the WebSphere Application Server windows service if different from the standard name. This field is generally not used.

autostart monman
Used in event rule management. Restarts the monitoring engine automatically when the next production plan is activated (on Windows also when Tivoli Workload Scheduler is restarted). The default is Yes.

bm check deadline
Specify the minimum number of seconds Batchman waits before checking if a job has missed its deadline. Must only be used on workstations running jobs and job streams that include deadlines (with the exception of critical jobs). The default is 300 seconds. Set the **bm check deadline** option on each workstation on which you want to be notified of the deadline expiration. To disable the option and not check deadlines, enter a value of zero.

bm check file Specify the minimum number of seconds Batchman waits before checking for the existence of a file that is used as a dependency. The default is 120 seconds.

Setting local options (localopts file)

- bm check status**
Specify the number of seconds Batchman waits between checking the status of an internetwork dependency. The default is 300 seconds.
- bm check until**
Specify the maximum number of seconds Batchman waits before reporting the expiration of an Until time for job or job stream. Specifying a value below the default setting (300) might overload the system. If it is set below the value of Local Option **bm read**, the value of **bm read** is used in its place. The default is 300 seconds.
- bm look**
Specify the minimum number of seconds Batchman waits before scanning and updating its production control file. The default is 15 seconds.
- bm read**
Specify the maximum number of seconds Batchman waits for a message in the `intercom.msg` message file. If no messages are in queue, Batchman waits until the timeout expires or until a message is written to the file. The default is 10 seconds.
- bm stats**
To have Batchman send its startup and shutdown statistics to its standard list file, specify **on**. To prevent Batchman statistics from being sent to its standard list file, specify **off**. The default is **off**.
- bm verbose**
To have Batchman send all job status messages to its standard list file, specify **on**. To prevent the extended set of job status messages from being sent to the standard list file, specify **off**. The default is **off**.
- can be event processor**
Specify if this workstation can act as event processing server or not. It is set by default to **yes** for master domain managers and backup masters, otherwise it is set to **no**.
- clisslcipher**
Specify the string that is used for the SSL authentication when the command-line client and the server are using SSL authentication.
- clisslserverauth**
Specify **yes** if the command-line client and the server use SSL authentication in their communication. The default is **no**.
- clisslservercertificate**
Specify the file that contains the SSL certificate when the command-line client and the server use SSL authentication in their communication.
- clissltrusteddir**
Specify the directory that contains an SSL trusted certificate contained in files with hash naming (#) when the command-line client and the server are using SSL authentication in their communication. When the directory path contains blanks, enclose it in double quotation marks ("").
- composer prompt**
Specify the prompt for the composer command line. The prompt can be of up to 10 characters in length. The default is dash (-).

conman prompt

Specify the prompt for the conman command line. The prompt can be of up to 8 characters in length. The default is percent (%).

date format

Specify the value that corresponds to the date format you desire. The values can be:

- 0 corresponds to *yy/mm/dd*
- 1 corresponds to *mm/dd/yy*
- 2 corresponds to *dd/mm/yy*
- 3 indicates usage of Native Language Support variables

The default is 1.

defaultws

The default workstation when you are accessing using a command line client. Specify the domain manager workstation.

host

The name or IP address of the host when accessing using a command line client.

is remote cli

Specify if this instance is installed as a command line client.

jm job table size

Specify the size, in number of entries, of the job table used by Jobman. The default is 1024 entries.

jm look

Specify the minimum number of seconds Jobman waits before looking for completed jobs and performing general job management tasks. The default is 300 seconds.

jm nice

For UNIX and Linux operating systems only, specify the **nice** value to be applied to jobs launched by Jobman to change their priority in the kernel's scheduler. The **nice** boundary values vary depending upon each specific platform, but generally lower values correspond to higher priority levels and vice versa. The default depends upon the operating system.

Applies to jobs scheduled by the root user only. Jobs submitted by any other user inherit the same **nice** value of the Jobman process.

See also `jm promoted nice`.

jm no root

For UNIX and Linux operating systems only, specify **yes** to prevent Jobman from launching **root** jobs. Specify **no** to allow Jobman to launch **root** jobs. The default is **no**.

jm promoted nice

Used in workload service assurance. For UNIX and Linux operating systems only, assigns the priority value to a critical job that needs to be promoted so that the operating system processes it before others. Applies to critical jobs or predecessors that need to be promoted so that they can start at their critical start time.

Boundary values vary depending upon each specific platform, but generally lower values correspond to higher priority levels and vice versa. The default is -1. Be aware that:

- The promotion process is effective with negative values only. If you set a positive value, the system runs it with the -1 default value and logs a warning message every time Jobman starts.
- An out of range value (for example -200), prompts the operating system to automatically promote the jobs with the lowest allowed **nice** value. Note that in this case no warning is logged.

Setting local options (localopts file)

- Overusing the promotion mechanism (that is, defining an exceedingly high number of jobs as mission critical and setting the highest priority value here) might overload the operating system, negatively impacting the overall performance of the workstation.

You can use this and the `jm nice` options together. If you do, remember that, while `jm nice` applies only to jobs submitted by the root user, `jm promoted nice` applies only to jobs that have a critical start time. When a job matches both conditions, the values set for the two options add up. For example, if on a particular agent the local options file has:

```
jm nice= -2
jm promoted nice= -4
```

when a critical job submitted by the root user needs to be promoted, it is assigned a cumulative priority value of -6.

jm promoted priority

Used in workload service assurance. For Windows operating systems only, sets to this value the priority by which the operating system processes a critical job when it is promoted.

Applies to critical jobs or predecessors that need to be promoted so that they can start at their critical start time.

The possible values are:

- High
- AboveNormal
- Normal
- BelowNormal
- Low or Idle

The default is AboveNormal.

Note that if you set a lower priority value than the one non-critical jobs might be assigned, no warning is given and no mechanism like the one available for `jm promoted nice` sets it back to the default.

jm read Specify the maximum number of seconds Jobman waits for a message in the `courier.msg` message file. The default is 10 seconds.

local was For master domain managers and backup masters connected to the Tivoli Workload Scheduler database. If set to **yes**, it improves the performance of job stream and job submission from the database. The default is **no**.

merge stdlists Specify **yes** to have all of the Tivoli Workload Scheduler control processes, except Netman, send their console messages to a single standard list file. The file is given the name **TWSmerge**. Specify **no** to have the processes send messages to separate standard list files. The default is **yes**.

mm cache mailbox

Use this option to enable Mailman to use a reading cache for incoming messages. In this case, only messages considered essential for network consistency are cached. The default is **yes**.

mm cache size

Specify this option if you also use **mm cache mailbox**. The maximum value (default) is **512**.

mm resolve master

When set to **yes** the \$MASTER variable is resolved at the beginning of the production day. The host of any extended agent is switched after the next **JnextPlan** (long term switch). When it is set to **no**, the \$MASTER variable is not resolved at **JnextPlan** and the host of any extended agent can be switched after a conman **switchmgr** command (short- and long-term switch). The default is **yes**. When you switch a master domain manager and the original has **mm resolve master** set to no and the backup has **mm resolve master** set to yes, after the switch any extended agent that is hosted by \$MASTER switches to the backup domain manager. When the backup domain manager restarts, the keyword \$MASTER is locally expanded by Mailman. You should keep the **mm resolve master** value the same for master domain managers and backup domain managers.

mm response Specify the maximum number of seconds Mailman waits for a response before reporting that a workstation is not responding. The minimum wait time for a response is **90** seconds. The default is 600 seconds.

mm retrylink Specify the maximum number of seconds Mailman waits after unlinking from a non-responding workstation before it attempts to link to the workstation again. The default is 600 seconds. The **tomservers** optional mailman servers do not unlink non-responding agents. The link is repetitively checked every 60 seconds, which is the default **retrylink** for these servers.

mm sound off Specify how Mailman responds to a conman **tellop ?** command. Specify **yes** to have Mailman display information about every task it is performing. Specify **no** to have Mailman send only its own status. The default is **no**.

mm unlink Specify the maximum number of seconds Mailman waits before unlinking from a workstation that is not responding. The wait time should not be less than the response time specified for the Local Option **mm response**. The default is 960 seconds.

mozart directory

This parameter applies only to versions of Tivoli Workload Scheduler prior to version 8.3. Defines the name of the master domain manager's shared mozart directory. The default is `<TWA_home>/mozart`.

nm mortal Specify **yes** to have Netman quit when all of its child processes have stopped. Specify **no** to have Netman keep running even after its child processes have stopped. The default is **no**.

nm port Specify the TCP port number that Netman responds to on the local computer. This must match the TCP/IP port in the computer's workstation definition. It must be an unsigned 16-bit value in the range 1- 65535 (values between 0 and 1023 are reserved for services such as, FTP, TELNET, HTTP, and so on). The default is 31111.

If you run event-driven workload automation and you have a security firewall, make sure this port is open for incoming and

Setting local options (localopts file)

outgoing connections.

- nm read** Specify the maximum number of seconds Netman waits for a connection request before checking its message queue for **stop** and **start** commands. The default is 10 seconds.
- nm retry** Specify the maximum number of seconds Netman waits before retrying a connection that failed. The default is 800 seconds.
- nm SSL port** The port used to listen for incoming SSL connections. This value must match the one defined in the **secureaddr** attribute in the workstation definition in the database. It must be different from the **nm port** local option that defines the port used for normal communication.

Notes:

1. If you install multiple instances of Tivoli Workload Scheduler on the same computer, set all SSL ports to different values.
2. If you plan not to use SSL, set the value to 0.

The default is none.

parameters directory

This parameter applies only to versions of Tivoli Workload Scheduler prior to version 8.3. Defines the name of the master domain manager's shared *TWA_home* directory. The default is none.

- port** The TCP/IP port number of the protocol used when accessing using a command line client. The default is 31115.
- protocol** The protocol used to connect to the host when accessing using a command line client.
- proxy** The name of the proxy server used when accessing using a command line client.
- proxyport** The TCP/IP port number of the proxy server used when accessing using a command line client.

restricted stdlists

Use this option to set a higher degree of security to the `stdlist` directory (and to its subdirectories) allowing only selected users to create, modify, or read files.

This option is available for UNIX workstations only. You must manually add it in your local options file if you want to use it. After you define it, make sure you erase your current `stdlist` directory (and subdirectories) and that you restart Tivoli Workload Scheduler.

If the option is not present or if it is set to `no`, the newly created `stdlist` directory and its subdirectories are unaffected and their rights are as follows:

```
drwxrwxr-x 22 twsmdm staff          4096 Nov 09 12:12
drwxrwxr-x  2 twsmdm staff           256 Nov 09 11:40 2009.11.09
drwxrwxr-x  2 twsmdm staff          4096 Nov 09 11:40 logs
drwxr-xr-x  2 twsmdm staff          4096 Nov 09 11:40 traces
```

If the option is set to `yes`, these directories have the following rights:

Setting local options (localopts file)

```
drwxr-x--x 5 twsmdm staff          256 Nov 13 18:15
rwxr-x--x 2 twsmdm staff          256 Nov 13 18:15 2009.11.13
rwxr-x--x 2 twsmdm staff          256 Nov 13 18:15 logs
rwxr-x--x 2 twsmdm staff          256 Nov 13 18:15 traces
```

Do the following to define and activate this option:

1. Add the line `restricted stdlists = yes` in your local options file.
2. Stop Tivoli Workload Scheduler.
3. Stop WebSphere Application Server if present.
4. Remove the `stdlist` directory (or at least its files and subdirectories).
5. Start Tivoli Workload Scheduler.
6. Start WebSphere Application Server if present.

SSL auth mode

The behavior of Tivoli Workload Scheduler during an SSL handshake is based on the value of the `SSL auth mode` option as follows:

- caonly** Tivoli Workload Scheduler checks the validity of the certificate and verifies that the peer certificate has been issued by a recognized CA. Information contained in the certificate is not examined. The default is **caonly**.
- string** Tivoli Workload Scheduler checks the validity of the certificate and verifies that the peer certificate has been issued by a recognized CA. It also verifies that the Common Name (CN) of the Certificate Subject matches the string specified into the `SSL auth string` option. See 29.
- cpu** Tivoli Workload Scheduler checks the validity of the certificate and verifies that the peer certificate has been issued by a recognized CA. It also verifies that the Common Name (CN) of the Certificate Subject matches the name of the workstation that requested the service.

SSL auth string

Used in conjunction with the `SSL auth mode` option when the "string" value is specified. The `SSL auth string` (ranges from 1 — 64 characters) is used to verify the certificate validity. The default is "tws".

SSL CA certificate

In SSL authentication, the name of the file containing the trusted certification authority (CA) certificates required for authentication. The CAs in this file are also used to build the list of acceptable client CAs passed to the client when the server side of the connection requests a client certificate. This file is the concatenation, in order of preference, of the various PEM-encoded CA certificate files. The default is `TWA_home/TWS/ssl/TWSTrustedException`. See "Configuring the SSL connection protocol for the network" on page 174.

SSL certificate In SSL authentication, the name of the local certificate file. The default is `TWA_home/TWS/ssl/TWS.crt`. See "Configuring the SSL connection protocol for the network" on page 174.

Setting local options (localopts file)

SSL certificate chain

In SSL authentication, the name of the file that contains the concatenation of the PEM-encoded certificates of certification authorities which form the certificate chain of the workstation's certificate. This parameter is optional. The default is *TWA_home/TWS/ssl/TWScertificateChainCA.crt*. See "Configuring the SSL connection protocol for the network" on page 174.

SSL encryption cipher

The ciphers that the workstation supports during an SSL connection. Use the following shortcuts:

Table 11. Shortcuts for encryption ciphers

Shortcut	Encryption ciphers
SSLv3	SSL version 3.0
TLSv	TLS version 1.0
EXP	Export
EXPORT40	40-bit export
EXPORT56	56-bit export
LOW	Low strength (no export, single DES)
MEDIUM	Ciphers with 128 bit encryption
HIGH	Ciphers using Triple-DES
NULL	Ciphers using no encryption

The default is **SSLv3**.

SSL key

In SSL authentication, the name of the private key file. The default is *TWA_home/TWS/ssl/TWS.key*. See "Configuring the SSL connection protocol for the network" on page 174.

SSL key pwd

In SSL authentication, the name of the file containing the password for the stashed key. The default is *TWA_home/TWS/ssl/TWS.sth*. See "Configuring the SSL connection protocol for the network" on page 174.

SSL random seed

The pseudo random number file used by OpenSSL on some operating systems. Without this file, SSL authentication might not work correctly. The default is *TWA_home/TWS/ssl/TWS.rnd*. See "Configuring the SSL connection protocol for the network" on page 174.

stdlist width

Specify the maximum width of the Tivoli Workload Scheduler console messages. You can specify a column number in the range 1 to 255. Lines are wrapped at or before the specified column, depending on the presence of imbedded carriage control characters. Specify a negative number or zero to ignore line width. On UNIX and Linux operating systems, you should ignore line width if you enable system logging with the **syslog local** option. The default is 0 columns.

syslog local

Enables or disables Tivoli Workload Scheduler system logging for UNIX and Linux operating systems only. Specify **-1** to turn off system logging for Tivoli Workload Scheduler. Specify a number from 0 to 7 to turn on system logging and have Tivoli Workload Scheduler use the corresponding local facility (LOCAL0 through

LOCAL7) for its messages. Specify any other number to turn on system logging and have Tivoli Workload Scheduler use the USER facility for its messages. The default is -1. See “Tivoli Workload Scheduler console messages and prompts” on page 37.

sync level Specify the rate at which Tivoli Workload Scheduler synchronizes information written to disk. This option affects all mailbox agents and is applicable to UNIX and Linux operating systems only. Values can be:

low Allows the operating system to handle it.

medium

Flushes the updates to disk after a transaction has completed.

high Flushes the updates to disk every time data is entered.

The default is **low**.

switch sym prompt

Specify a prompt for the conman command line after you have selected a different Symphony file with the **setsym** command. The maximum length is 8 characters. The default is **<n>**.

tcp connect timeout

With this attribute, specify the maximum number of seconds that can be waited to establish a connection through non-blocking socket. The default is 15 seconds.

tcp timeout

With this attribute for the Netman process, specify the maximum number of seconds that Mailman and Conman waits for the completion of a request on a linked workstation that is not responding. The default is 300 seconds.

timeout

The timeout in seconds when accessing using a command line client. The default is 3600 seconds.

thiscpu

Specify the Tivoli Workload Scheduler name of this workstation. When a switch is made between the master domain manager and a backup domain manager, using the **switchmgr** command, the Symphony header value for **thiscpu** is overwritten by the **thiscpu** value in the localopts file. The default is **thiscpu**.

unison network directory

This parameter applies only to versions of Tivoli Workload Scheduler prior to version 8.3. Defines the name of the Unison network directory. The default is **<TWA_home>/../unison/network**.

useropts

If you have multiple instances of Tivoli Workload Scheduler on a system, use this to identify the *useropts* file that is to be used to store the connection parameters for the instance in which this *localops* file is found. See “Multiple product instances” on page 35 for more details.

wr enable compression

Use this option on fault-tolerant agents. Specify if the fault-tolerant agent can receive the Symphony file in compressed form from the master domain manager. The default is **no**.

1
1
1
1

Setting local options (localopts file)

- wr read** Specify the number of seconds the Writer process waits for an incoming message before checking for a termination request from Netman. The default is 600 seconds.
- wr unlink** Specify the number of seconds the Writer process waits before exiting if no incoming messages are received. The minimum is 120 seconds. The default is 180 seconds.

Local options file example

The following is an example of a default localopts file:

Note: Some parameters might not be present depending upon your version and configuration.

```
#####
# Licensed Materials - Property of IBM(R)
# 5698-WSH
# (C) Copyright IBM Corp. 1991, 2008. All Rights Reserved
# US Government Users Restricted Rights - Use, duplication, or disclosure
# restricted by GSA ADP Schedule Contract with IBM Corp.
# IBM is a registered trademark of International Business Machines Corporation
# in the United States, other countries, or both.
#####
#
# The Tivoli Workload Scheduler localopts file defines the attributes of this
# workstation, for various processes.
#
#-----
# General attributes of this workstation:
#
thiscpu         =$(this_cpu)
merge stdlists  =yes
stdlist width   =0
syslog local    =-1
#
#-----
# The attributes of this workstation for the batchman process:
#
bm check file    =120
bm check status  =300
bm look          =15
bm read          =10
bm stats         =off
bm verbose       =off
bm check until   =300
bm check deadline =300
#
#-----
# The attributes of this workstation for the jobman process:
#
jm job table size =1024
jm look           =300
jm nice           =0
jm promoted nice  =-1          #UNIX
jm promoted priority =AboveNormal #WINDOWS
jm no root        =no
jm read          =10
#
#-----
# The attributes of this workstation for the TWS mailman process:
#
mm response      =600
mm retrylink     =600
mm sound off     =no
mm unlink        =960
```

Setting local options (localopts file)

```
mm cache mailbox =yes
mm cache size    =512
mm resolve master =yes
autostart monman =yes
#
#-----
# The attributes of this workstation for the netman process:
#
nm mortal        =no
nm port         =$(tcp_port)
nm read          =10
nm retry         =800
#
#-----
# The attributes of this workstation for the writer process:
#
wr read          =600
wr unlink        =180
wr enable compression =no
#
#-----
# Optional attributes of this Workstation for remote database files
#
# mozart directory =      $(install_dir)/mozart
# parameters directory =  $(install_dir)
# unison network directory = $(install_dir)/../unison/network
#
#-----
# The attributes of this workstation for custom formats
#
date format      =1      # The possible values are 0-yyyy/mm/dd,
                        # 1-mm/dd/yyyy,
                        # 2-dd/mm/yyyy,
                        # 3-NLS.

composer prompt  =-
conman prompt    =%
switch sym prompt =<n>%
#
#-----
# The attributes of this workstation for the customization of I/O on mailbox
# files
#
sync level       =low
#
#-----
# The attributes of this workstation for networking
#
tcp timeout      =300
#
#-----
# The attributes of this workstation for SSL
#
nm SSL full port =0
nm SSL port      =0
SSL key          ="$(install_dir)/ssl/TWSPprivateKeyFile.pem"
SSL certificate  ="$(install_dir)/ssl/TWSPublicKeyFile.pem"
SSL key pwd      ="$(install_dir)/ssl/TWSPprivateKeyFile.sth"
SSL certificate chain ="$(install_dir)/ssl/TWSCertificateChainFile.pem"
SSL CA certificate ="$(install_dir)/ssl/TWSTrustedCA.crt"
SSL random seed  ="$(install_dir)/ssl/TWS.rnd"
SSL Encryption Cipher =SSLv3
SSL auth mode    =caonly
SSL auth string  =twS
#
#-----
# The attributes of this workstation for the embedded WebSphere Application
# Server
```

Setting local options (localopts file)

```
#
local was          =no
#
#-----
# Application server check attributes

Appserver check interval      = 5          #minutes
Appserver auto restart       = yes         #yes/no
Appserver min restart time    = 10         #minutes
Appserver max restarts       = 5          #restarts number
Appserver count reset interval = 24        #hours
Appserver service name       = "IBMWAS61Service - tws850xx"
#-----
# The TWS instance has been installed as REMOTE CLI
is remote cli = no # yes for a REMOTE CLI installation, no otherwise
#
#-----
# Attributes for conman connections
#
HOST          =          # Master hostname used when attempting a connection.
PROTOCOL      = https   # Protocol used to establish a connection with the Master.
PROTOCOL      = http    # Protocol used to establish a connection with the Master.
PORT          =          # Protocol port
PROXY         =
PROXYPORT     =
TIMEOUT       = 3600    # Timeout in seconds to wait a server response
CLISLSSERVERAUTH = yes
CLISLSCIPHER  = MD5
CLISLSSERVERCERTIFICATE =
CLISLSTRUSTEDDIR =
DEFAULTTWS   =
USEROPTS     =

#-----
# Event Management parameters
#
CAN BE EVENT PROCESSOR = no # yes for MDM and BDM, no otherwise
```

Note: The "REMOTE CLI" refers to the command line client.

Setting user options

Set the user options you require for each user on a workstation who needs them in the `useropts` file. Changes do not take effect until Tivoli Workload Scheduler is stopped and restarted.

The concept of the `useropts` file is to contain values for `localopts` parameters that must be personalized for an individual user. The files must be located in the `user_home/.TWS` directory of the user. When Tivoli Workload Scheduler needs to access data from the `localopts` file, it looks first to see if the property it requires is stored only or also in the `useropts` file for the user, always preferring the `useropts` file version of the value of the key. If a property is not specified when invoking the command that requires it, or inside the `useropts` and `localopts` files, an error is displayed.

The main use of the `useropts` file is to store the user-specific connection parameters used to access the command line client (see "Configuring command-line client access authentication" on page 36). These are the following keys, which are not stored in the `localopts` file:

username

User name used to access the master domain manager. The user must be defined in the security file on the master domain manager (see Chapter 5, “Configuring user authorization (Security file),” on page 93)

password

Password used to access the master domain manager. The presence of the ENCRYPT label in the password field indicates that the specified setting has been encrypted; if this label is not present, you must exit and access the interface program again to allow the encryption of that field.

A useropts file is created for the <TWS_user> during the installation, but you must create a separate file for each user that needs to use user-specific parameters on a workstation.

Sample useropts file

This is the sample content of a useropts file:

```
#
# Tivoli Workload Scheduler useropts file defines attributes of this Workstation.
#
#-----
# Attributes for CLI connections
USERNAME = MDMDBE4 # Username used in the connection
PASSWORD = "ENCRYPT:YEE7cEZs+HE+mEHCsdNOfg==" # Password used in the connection
#HOST = # Master hostname used when attempting a connection.
PROTOCOL = https # Protocol used to establish a connection with the Master.
#PROTOCOL = http # Protocol used to establish a connection with the Master.
PORT = 3111 # Protocol port
#PROXY =
#PROXYPORT =
TIMEOUT = 120 # Timeout in seconds to wait a server response
#DEFAULTWS =
```

Note: The # symbol is used to comment a line.

Multiple product instances

Because Tivoli Workload Scheduler supports multiple product instances installed on the same computer, there can be more than one instance of the useropts file per user. This is achieved by giving the useropts files for a user different names for each instance.

In the localopts file of each instance the option named *useropts* identifies the file name of the useropts file that has to be accessed in the *user_home/.TWS* directory to connect to that installation instance.

This means that, for example, if two Tivoli Workload Scheduler instances are installed on a computer and the user operator is a user of both instances, you could define the useropts credentials as follows:

- In the localopts file of the *first* instance the local option *useropts* = *useropts1* identifies the *operator_home/.TWS/useropts1* file containing the connection parameters settings that user operator needs to use to connect to the *first* Tivoli Workload Scheduler instance.
- In the localopts file of the *second* Tivoli Workload Scheduler instance the local option *useropts* = *useropts2* identifies the *operator_home/.TWS/useropts2* file containing the connection parameters settings that user operator needs to use to connect to the *second* Tivoli Workload Scheduler instance.

Configuring command-line client access authentication

This chapter describes how to change the configuration of the connection that is used by the command line client.

The command line client is installed automatically on the master domain manager and can be installed optionally on any other workstation. You use it to invoke either a command-line program, such as **composer**, **optman**, or **planman**, or when using the **logman** command or the **JnextPlan** scripts.

It is configured automatically by the installation wizard, but if you need to change the credentials that give access to the server on the master domain manager, or you want to use it to access a different master domain manager, modify the *connection parameters* as described here.

Notes:

1. The *connection parameters* are not required to use the local **conman** program on a fault-tolerant agent.
2. The command-line client on the master domain manager uses exactly the same mechanism to communicate with the server as it does when it is installed remotely.

Connection parameters

Provide the following setup information to connect to the master domain manager via HTTP or HTTPS.

localopts

All fields except *username* and *password*, must be stored by editing the *TWA_home/TWS/localopts* properties file on the computer from which the access is required. See “Setting local options” on page 20 for a full description of the file and the properties.

```
host = host_name
protocol = protocol
port = port number
proxy = proxy server
proxyport = proxy server port number
timeout = seconds
clisslserverauth = yes|no
clisslcipher = string
clissservercertificate = file_name
clissltrusteddir = directory_name
defaultws = master_workstation
useropts = useropts_file
```

useropts

As a minimum, the following properties are defined in the *user_home/.TWS/localopts* file. If you need to personalize for a user any of the properties found in the *localopts* file, add the properties to the *useropts* file. The values in the *useropts* file are used in preference to those in the *localopts* file. See “Setting user options” on page 34 for a full description of the file and the properties.

```
username = user_ID
password = string
```

If the *username* and *password* are not specified in this file they must be specified dynamically when invoking the command line client.

Entering passwords

You type the connection password into the `useropts` file in unencrypted form. When you access the interface for the first time it is encrypted.

Note: On Windows workstations, when you specify a password that contains double quotation marks (") or other special characters, make sure that the character is escaped. For example, if your password is `tws11"tws`, write it as `"tws11\"tws"` in `useropts`.

Tivoli Workload Scheduler console messages and prompts

The Tivoli Workload Scheduler control processes (Netman, Mailman, Batchman, Jobman, and Writer) write their status messages (referred to as console messages) to standard list files. These messages include the prompts used as job and job stream dependencies. On UNIX and Linux operating systems, the messages can also be directed to the `syslog` daemon (`syslogd`) and to a terminal running the Tivoli Workload Scheduler console manager. These features are described in the following sections.

Setting sysloglocal on UNIX

If you set `sysloglocal` in the local options file to a positive number, Tivoli Workload Scheduler's control processes send their console and prompt messages to the `syslog` daemon. Setting it to `-1` turns this feature off. If you set it to a positive number to enable system logging, you must also set the local option `stdlistwidth` to `0`, or a negative number.

Tivoli Workload Scheduler's console messages correspond to the following `syslog` levels:

<code>LOG_ERR</code>	Error messages such as control process abends and file system errors.
<code>LOG_WARNING</code>	Warning messages such as link errors and stuck job streams.
<code>LOG_NOTICE</code>	Special messages such as prompts and tellops.
<code>LOG_INFO</code>	Informative messages such as job launches and job and job stream state changes.

Setting `sysloglocal` to a positive number defines the `syslog` facility used by Tivoli Workload Scheduler. For example, specifying `4` tells Tivoli Workload Scheduler to use the local facility `LOCAL4`. After doing this, you must make the appropriate entries in the `/etc/syslog.conf` file, and reconfigure the `syslog` daemon. To use `LOCAL4` and have the Tivoli Workload Scheduler messages sent to the system console, enter the following line in `/etc/syslog.conf`:

```
local4    /dev/console
```

To have the Tivoli Workload Scheduler error messages sent to the `maestro` and `root` users, enter the following command:

```
local4.err    maestro,root
```

The selector and action fields must be separated by at least one tab. After modifying `/etc/syslog.conf`, you can configure the `syslog` daemon by entering the following command:

Tivoli Workload Scheduler console messages and prompt

```
kill -HUP `cat /etc/syslog.pid`
```

console command

You can use the conman **console** command to set the Tivoli Workload Scheduler message level and to direct the messages to your terminal. The message level setting affects only Batchman and Mailman messages, which are the most numerous. It also sets the level of messages written to the standard list file or files and the **syslog** daemon. The following command, for example, sets the level of Batchman and Mailman messages to 2 and sends the messages to your computer:

```
console sess;level=2
```

Messages are sent to your computer until you either run another **console** command, or exit conman. To stop sending messages to your terminal, enter the following conman command:

```
console sys
```

Enabling the time zone feature

Time zones are enabled by default on installation of the product.

When you upgrade, the time zone feature inherits the setting of the previous installation. You can enable the time zone using the **enTimeZone** option of the **optman** command, as follows:

```
optman chg enTimeZone = yes
```

The following steps outline the method of implementing the time zone feature:

1. Load Tivoli Workload Scheduler.

The database allows time zones to be specified for workstations, but not on *start* and *deadline* times within job streams in the database. The plan creation (JnextPlan) ignores any time zones that are present in the database. You will not be able to specify time zones anywhere in the plan.

2. Define workstation time zones.

Set the time zone of the master domain manager workstation, of the backup master domain manager, and of any agents that are in a different time zone than the master domain manager. No time zones are allowed in the database for **Start**, **Latest Start Time**, and **Termination Deadline** times. No time zones are allowed anywhere in the plan at this point, because **enTimeZone** is set to **no**.

3. When workstation time zones have been set correctly, enable the time zone feature.

All users are able to use time zones anywhere in the database, although they should wait for the next run of JnextPlan to use them on **Start**, **Latest Start Time**, and **Termination Deadline** times. The next time JnextPlan runs, time zones are carried over to the plan, the Tivoli Dynamic Workload Console, and the Job Scheduling Console, and the back end allows specification of time zones anywhere in the plan.

4. Start using time zones on *start* and *until* times where needed.

You can now use all time zone references in the database and in the plan with the Tivoli Dynamic Workload Console, the Job Scheduling Console, and the command-line interface.

Configuring to use the report commands

You use the Tivoli Workload Scheduler report commands to obtain summary or detailed information about your workload scheduling. Before using these commands, however, they must be configured for your environment. This process is described in the chapter on getting reports and statistics in the *Tivoli Workload Scheduler: User's Guide and Reference*.

Modifying jobmon service rights for Windows

On Windows systems, the Tivoli Workload Scheduler jobmon service runs in the SYSTEM account with the right **Allow Service to Interact with Desktop** granted to it. You can remove this right for security reasons. However, if you do so, it prevents the service from launching interactive jobs that run in a window on the user's desktop. These jobs will be run, but are not accessible from the desktop or from Tivoli Workload Scheduler and do not have access to desktop resources. As a result, they may run forever or abend due to lack of resources.

Modifying jobmon service rights

Chapter 3. Configuring for dynamic scheduling

This chapter describes the configuration parameters that you can customize in your dynamic scheduling environment. It is divided into two main sections:

- “Configuring the dynamic workload broker instance on the master domain manager.”
- “Configuring the Tivoli Workload Scheduler agent” on page 59.

For all other configuration options, see Chapter 2, “Customizing and configuring Tivoli Workload Scheduler,” on page 5.

You can perform these configuration tasks after completing the installation of your master domain manager and agents with the dynamic scheduling capability, and any time that you want to change or tune specific parameters in the dynamic workload broker environment. For information about dynamic workload broker architecture and components, see *Tivoli Workload Scheduler: Scheduling Workload Dynamically*.

Configuring the dynamic workload broker instance on the master domain manager

The configuration parameters for dynamic workload broker are defined by default at installation time. You can modify a subset of these parameters using the files that are created when you install dynamic workload broker. The following files are created in the path:

TWA_home/TDWB/config

ResourceAdvisorConfig.properties

Contains configuration information about the **Resource Advisor**. For more information, see “ResourceAdvisorConfig.properties file” on page 42.

JobDispatcherConfig.properties

Contains configuration information about the **Job Dispatcher**. For more information, see “JobDispatcherConfig.properties file” on page 44.

CLConfig.properties

Contains configuration information for the dynamic workload broker command line. This file is described in *IBM Tivoli Workload Scheduler: Scheduling Workload Dynamically*.

audit.properties

Contains options for configuring the auditing of events. This file is documented in the *IBM Tivoli Workload Scheduler: Troubleshooting Guide*.

You can modify a subset of the parameters in these files to change the following settings:

- Heartbeat signal from the workload broker agents
- Time interval for job allocation to resources
- Time interval for notifications on resources
- Maximum number of results when allocating jobs to global resources
- Encryption of any passwords sent in the JSDL definitions
- Time interval for retrying the operation after a **Job Dispatcher** failure

Configuring the dynamic workload broker instance

- Time interval for retrying the operation after a client notification failure
- Archive settings for job data
- Job history settings
- Command line properties (see *IBM Tivoli Workload Scheduler: Scheduling Workload Dynamically*)

The editable parameters are listed in the following sections. If you edit any parameters that are not listed, the product might not work. After modifying the files, you must stop and restart the IBM WebSphere server.

Because one instance of dynamic workload broker is installed with your master domain manager and one instance with every backup master, if you use dynamic scheduling, you have at least two instances present in your Tivoli Workload Scheduler network. The instance running with the master domain manager is the only active one at any time. The instances installed in the backup masters are idle until you switch masters, and the one in the new master becomes the active instance (see “Starting, stopping and displaying dynamic workload broker status” on page 264 for important information about this scenario). To have a smooth transition from one instance to the other when you switch managers, it is important that you keep the same configuration settings in the `ResourceAdvisorConfig.properties` and `JobDispatcherConfig.properties` files in all the instances that you have. When you make a change in any of these files of your running dynamic workload broker, remember to apply the same change also in the dynamic workload broker idling in your backup manager.

ResourceAdvisorConfig.properties file

The parameters in this file affect the following dynamic workload broker server settings:

- Heartbeat signal from the workload broker agents
- Time interval for job allocation to resources
- Time interval for notifications on resources
- Maximum number of results when allocating jobs to global resources

You can modify the following parameters in the `ResourceAdvisorConfig.properties` file:

ResourceAdvisorURL

Specifies the URL of the **Resource Advisor**.

RaaHeartBeatInterval

Specifies the time interval within which the **Resource Advisor** expects a heartbeat signal from the workload broker agents. The default value is 367 seconds. After the maximum number of retries (specified in the **MissedHeartBeatCount** parameter) is exceeded, the **Resource Advisor** reports the related computer as unavailable. In a slow network, you might want to set this parameter to a higher value. However, defining a higher value might delay the updates on the availability status of computer systems. If, instead, you decrease this value together with the value defined for the **NotifyToResourceAdvisorIntervalSecs** parameter, this might generate network traffic and increase CPU usage when updating cached data. The value defined in this parameter must be consistent with the **NotifyToResourceAdvisorIntervalSecs** parameter defined in the `ResourceAdvisorAgentConfig.properties` file, which defines the time interval for each workload broker agent to send the heartbeat signal to the **Resource Advisor**.

Configuring the dynamic workload broker instance

MissedHeartBeatCount

Specifies the number of missed heartbeat signals after which the computer is listed as not available. The default value is 2. In a slow network, you might want to set this parameter to a higher value.

MaxWaitingTime

Specifies the maximum time interval that a job must wait for a resource to become available. If the interval expires before a resource becomes available, the job status changes to Resource Allocation Failure. The default value is 600 seconds. You can override this value for each specific job by using the **Maximum Resource Waiting Time** parameter defined in the Job Brokering Definition Console. For more information about the **Maximum Resource Waiting Time** parameter, see the Job Brokering Definition Console online help. If you set this parameter to -1, no waiting interval is applied for the jobs. If you set this parameter to 0, the **Resource Advisor** tries once to find the matching resources and, if it does not find any resource, the job changes to the ALLOCATION FAILED status. If you increase this value, all submitted jobs remain in WAITING status for a longer time and the **Resource Advisor** tries to find matching resources according to the value defined for the **CheckInterval** parameter.

CheckInterval

Specifies how long the **Resource Advisor** waits before retrying to find matching resources for a job that did not find any resource in the previous time slot. The default value is 60 seconds.

TimeSlotLength

Specifies the time slot interval during which the **Resource Advisor** allocates resources to each job. Jobs submitted after this interval has expired are considered in a new time slot. The default value is 15 seconds. The default value is adequate for most environments and should not be modified. Setting this parameter to a higher value, causes the **Resource Advisor** to assign resources to higher priority jobs rather than to lower priority jobs when all jobs are trying to obtain the same resource. It might also, however, cause the job resource matching processing to take longer and the resource state updates from agents to be slowed down. Setting this parameter to a lower value, causes the **Resource Advisor** to process the resource matching faster and, if you have a high number of agents with frequent updates, to update the resource repository immediately. If job requirements match many resources, lower values ensure a better load balancing. If most jobs use resource allocation, do not lower this value because the allocation evaluation requires many processing resources.

NotifyTimeInterval

Specifies the interval within which the **Resource Advisor** retries to send notifications on the job status to the **Job Dispatcher** after a notification failed. The default value is 15 seconds. The default value is adequate for most environments and should not be modified.

MaxNotificationCount

Specifies the maximum number of attempts for the **Resource Advisor** to send notifications to the **Job Dispatcher**. The default value is 100. The default value is adequate for most environments and should not be modified.

ServersCacheRefreshInterval

Specifies with what frequency (in seconds) the Resource Advisor checks the list of active and backup workload broker servers for updates. This list

Configuring the dynamic workload broker instance

is initially created when the master domain manager is installed, and after that it is updated every time a new backup master is installed and connected to the master domain manager database (the master domain manager and every backup master include also a workload broker server). When the Resource Advisor agents send their data about the resources discovered in each computer, they are able to automatically switch between the servers of this list, so that the workload broker server that is currently active can store this data in its Resource Repository. For this reason, the Resource Advisor agents must know at all times the list of all workload broker servers. The possible values range between 300 (5 minutes) and 43200 (12 hours). The default value is 3600 seconds.

After modifying the file, you must stop and restart the WebSphere Application Server.

JobDispatcherConfig.properties file

The parameters in this file affect the following dynamic workload broker server settings:

- Encryption of any passwords sent in the JSDL definitions
- Time interval for retrying the operation after a **Job Dispatcher** failure
- Time interval for retrying the operation after a client notification failure
- Archive settings for job data
- Job history settings

In the JobDispatcherConfig.properties file, the following parameters are available:

EnablePasswordEncryption

Specifies that any user passwords contained in the JSDL definitions are to be encrypted when the definitions are sent to the workload broker agents. The default is true. Setting this property to false forces the workload broker server to send the passwords in plain text. This applies to any password field.

RAEndpointAddress

Specifies the URL of the **Resource Advisor**.

JDURL

Specifies the URL of the **Job Dispatcher**.

FailQInterval

Specifies the numbers of seconds for retrying the operation after the following failures:

- Client notification.
- Allocation, Reallocate, Cancel Allocation requests to **Resource Advisor**.
- Any database operation failed for connectivity reasons.

The default value is 30 seconds. Increasing this value improves recovery speed after a failure but can use many system resources if the recovery operation is complex. For example, if the workload broker workstation is processing a new Symphony file, this operation might require some time, so you should set this parameter to a high value. If you are not using workload broker workstation, this parameter can be set to a lower value.

MaxNotificationCount

Specifies the maximum number of retries after a client notification failure.

Configuring the dynamic workload broker instance

The default value is 1440. For example, if the workload broker workstation is processing a new Symphony file, this operation might require some time, so you should set this parameter to a high value. If you are not using the workload broker workstation, this parameter can be set to a lower value.

MoveHistoryDataFrequencyInMins

Specifies how often job data must be moved to the archive tables in the **Job Repository** database and the tables in the archive database must be dropped. The unit of measurement is minutes. The default value is 60 minutes. Increasing this value causes the **Job Dispatcher** to check less frequently for jobs to be moved. Therefore, the volume of jobs in the **Job Repository** might increase and all queries might take longer to complete. Dynamic workload broker servers with high job throughput might require lower values, while low job throughputs might require higher values.

SuccessfulJobsMaxAge

Specifies how long successfully completed or canceled jobs must be kept in the **Job Repository** database before being archived. The unit of measurement is hours. The default value is 24 hours.

UnsuccessfulJobsMaxAge

Specifies how long unsuccessfully completed jobs or jobs in unknown status must be kept in the **Job Repository** database before being archived. The unit of measurement is hours. The default value is 72 hours.

ArchivedJobsMaxAge

Specifies how long jobs must be kept in the archive database before being deleted. The unit of measurement is hours. The default value is 168 hours.

AgentConnectTimeout

Specifies the number of minutes that the workload broker server waits for a scheduling agent response after it first attempts to establish a connection with that agent. If the agent does not reply within this time, the server does not open the connection. Values range from 0 to 60 (use 0 to wait indefinitely). The default is 3.

AgentReadTimeout

Specifies the number of minutes that the workload broker server waits to receive data from established connections with a scheduling agent. If no data arrives within the specified time, the server closes the connection with the agent. Values range from 0 to 60 (use 0 to wait indefinitely). The default is 3.

You can use this file to configure the product behavior when archiving job data. For more information about archive tables, see “Historical database tables created during installation” on page 46.

If an unexpected job workload peak occurs and a cleanup of the database is required earlier than the value you specified in the `MoveHistoryDataFrequencyInMins` parameter, you can use the `movehistorydata` command to perform a cleanup before the scheduled cleanup is performed.

After modifying the file, you must stop and restart the IBM WebSphere server.

Archiving job data

Job definitions created using the Job Brokering Definition Console or the Tivoli Dynamic Workload Console are stored in the **Job Repository** database. The **Job**

Configuring the dynamic workload broker instance

Repository database stores also the jobs created when the job definitions are submitted to the dynamic workload broker.

Job information is archived on a regular basis. By default, successful jobs are archived every 24 hours. Jobs in failed or unknown status are archived by default every 72 hours. Archived jobs are moved to historical tables in the **Job Repository**.

You can configure the time interval after which job data is archived using the following parameters:

- **MoveHistoryDataFrequencyInMins**
- **SuccessfulJobsMaxAge**
- **UnsuccessfulJobsMaxAge**
- **ArchivedJobsMaxAge**

These parameters are available in the `JobDispatcherConfig.properties` file, as described in “`JobDispatcherConfig.properties` file” on page 44. You can also use the **movehistorydata** command to perform a cleanup before the scheduled cleanup is performed.

Historical database tables created during installation

Database creation differs depending on the database vendor you are using. If you are using DB2®, two databases are created by default when the dynamic workload broker server is installed. If you are using Oracle, two schemas are created in the same database. The names for both the databases and the schemas are as follows:

IBMCDB (DB2) / CDB (Oracle)

Contains Agent manager data. The name is fixed and cannot be changed.

TDWB

Contains dynamic workload broker data. You can change the name.

The following three historical tables are created during the installation process in the **TDWB** database. These tables are used to contain historical data about job instances.

JOA_JOB_ARCHIVES

Contains archived job instances. See Table 12.

JRA_JOB_RESOURCE_ARCHIVES

Contains resource information related to a job. See Table 13 on page 47.

MEA_METRIC_ARCHIVES

Contains metrics collected for a job. See Table 14 on page 47.

To improve RDBMS performance, you can move data from the standard tables to historical tables on a regular basis. You can configure RDBMS maintenance using the `JobDispatcherConfig.properties` file. For more information, see “`JobDispatcherConfig.properties` file” on page 44. You can also use the **movehistorydata** command to move data to the historical tables and delete archived data.

Table 12. JOA_JOB_ARCHIVES database table

Column Name	DB2 Data Type	Oracle Data Type	Length	Nullable	Description
JOA_ID	CHAR () FOR BIT DATA	RAW	16	No	Contains the unique identifier of the job

Configuring the dynamic workload broker instance

Table 12. JOA_JOB_ARCHIVES database table (continued)

Column Name	DB2 Data Type	Oracle Data Type	Length	Nullable	Description
JOA_START_TIME	TIMESTAMP	TIMESTAMP	26	Yes	The start time of the job, if started
JOA_END_TIME	TIMESTAMP	TIMESTAMP	26	Yes	The end time of the job, if ended
JOA_JSDL_INSTANCE	CLOB	CLOB		No	The JSDL (job definition), stored in binary format
JOA_SUBMIT_USERNAME	VARCHAR	VARCHAR2	120	No	The submitter
JOA_TIMEZONE	VARCHAR	VARCHAR2	40	Yes	Not used in this release
JOA_STATE	DECIMAL	NUMBER	2	No	The job state code
JOA_RETURN_CODE	DECIMAL	NUMBER	10	No	The job return code
JOA_SUBMIT_TIME	TIMESTAMP	TIMESTAMP	26	No	The submit time
JOA_NAME	VARCHAR	VARCHAR2	250	No	The job definition name
JOA_NAMESPACE	VARCHAR	VARCHAR2	250	Yes	The job definition namespace
JOA_ALIAS_NAME	VARCHAR	VARCHAR2	250	Yes	The job definition alias
JOA_SUBMITTER_TYPE	VARCHAR	VARCHAR2	80	Yes	The submitter type (for example, TDWB CLI, TDWB UI)
JOA_UPDATE_TIME	TIMESTAMP	TIMESTAMP	26	No	The last update timestamp of actual row

Table 13. JRA_JOB_RESOURCE_ARCHIVES database table

Column Name	DB2 Data Type	Oracle Data type	Length	Nullable	Description
JOA_ID	CHAR () FOR BIT DATA	RAW	16	No	Contains the unique identifier of the job
JRA_RESOURCE_NAME	VARCHAR	VARCHAR2	250	No	The resource internal name
JRA_RESOURCE_TYPE	VARCHAR	VARCHAR2	30	No	The resource type (for example, ComputerSystem, FileSystem, ...)
JRA_RESOURCE_GROUP	DECIMAL	NUMBER	5	No	The group code (grouping an allocation for actual job)
JRA_DISPLAY_NAME	VARCHAR	VARCHAR2	250	Yes	The displayed name
JRA_IS_TARGET	DECIMAL	NUMBER	1	No	A flag indicating the root resource (typically the ComputerSystem)

Table 14. MEA_METRIC_ARCHIVES database table

Column Name	DB2 Data Type	Oracle Data Type	Length	Nullable	Description
JOA_ID	CHAR () FOR BIT DATA	RAW	16	No	Contains the unique identifier of the job
MEA_NAME	VARCHAR	VARCHAR2	80	No	The metric name (for example, JOB_MEMORY_USAGE, JOB_CPU_USAGE, ...)
MEA_TYPE	CHAR	CHARACTER	10	No	The metric datatype (for example, DECIMAL, ...)

Configuring the dynamic workload broker instance

Table 14. MEA_METRIC_ARCHIVES database table (continued)

Column Name	DB2 Data Type	Oracle Data Type	Length	Nullable	Description
MEA_VALUE	VARCHAR	VARCHAR2	250	No	The metric value

Table 15 lists the status of jobs as stored in the historical tables in association with the job status available in the Tivoli Dynamic Workload Console and in the command line, mapping them with the related options in the **movehistorydata** command.

Table 15. Job statuses in the historical tables

movehistorydata option	Job status in tables	Tivoli Dynamic Workload Console status	Command line status
SuccessfulJobsMaxAge	43	Completed successfully	SUCCEEDED_EXECUTION
	44	Canceled	CANCELED
UnsuccessfulJobMaxAge	41	Resource allocation failed	RESOURCE_ALLOCATION_FAILED
	42	Run failed	FAILED_EXECUTION
	45	Unknown	UNKNOWN
	46	Unable to start	NOT_EXECUTED

Configuring to schedule J2EE jobs

Using the dynamic workload broker component you can schedule J2EE jobs. To do this you must complete the following configuration tasks:

- Configure the J2EE executor on every agent on which you submit J2EE jobs.
- Configure the J2EE Job Executor Agent on an external WebSphere Application Server

Configuring the J2EE executor

To dynamically schedule J2EE jobs, you must configure the following property files on every agent on which you submit J2EE jobs:

- J2EEJobExecutorConfig.properties
- logging.properties
- soap.client.props

These files are configured with default values at installation time. The values that you can customize are indicated within the description of each file.

J2EEJobExecutorConfig.properties file: The path to this file is *TWA_home/TWS/JavaExt/cfg/J2EEJobExecutorConfig.properties* (*TWA_home\TWS\JavaExt\cfg\J2EEJobExecutorConfig.properties*) on the agent.

The keywords of this file are described in the following table:

Configuring the dynamic workload broker instance

Table 16. *J2EEJobExecutorConfig.properties* file keywords

Keyword	Specifies...	Default value	Must be customized
wasjaas.default	The path to the IBM WebSphere configuration file (<code>wsjaas_client.conf</code>) used to authenticate on the external WebSphere Application Server using JAAS security.	<code>TWA_home/TWS/JavaExt/cfg/wsjaas_client.conf</code> or <code>TWA_home\TWS\JavaExt\cfg\wsjaas_client.conf</code>	Optionally yes, if you move the file to the path you specify.
credentials.mycred	The credentials (ID and password) used to establish the SOAP connection to the external WebSphere Application Server when using indirect scheduling (the password must be {xor} encrypted)	<code>wasadmin,{xor}KD4sPj syNjE\=</code> (ID= <code>wasadmin</code> and password= <code>wasadmin</code> in {xor} encrypted format)	Yes, see “Running {xor} encryption on your password” on page 50 to learn how to encrypt your password.
connector.indirect	The name of the communication channel with WebSphere Application Server. Selecting an indirect invoker means that dynamic workload broker uses an existing WebSphere Application Server scheduling infrastructure that is already configured on a target external WebSphere Application Server. When creating the job definition, you can specify if you want to use a direct or indirect connector in the J2EE Application pane in the Application page in the Job Brokering Definition Console, or in the invoker element in the JSDL file. For more information about the Job Brokering Definition Console, see the online help.	A single line with the following values separated by commas: <ul style="list-style-type: none"> indirect keyword Name of the scheduler: <code>sch/MyScheduler</code> soap keyword Host name of the external WebSphere Application Server instance: <code>washost.mydomain.com</code> SOAP port of the WebSphere Application Server instance: <code>8880</code> Path to the <code>soap_client.props</code> file: <code>TWA_home/TWS/JavaExt/cfg/soap_client.props</code> Credentials keyword: <code>mycred</code> 	You must customize the following: <ul style="list-style-type: none"> The scheduler name. Replace the <code>sch/MyScheduler</code> string with the JNDI name of the IBM WebSphere scheduler that you plan to use. The host name of the external WebSphere Application Server instance. The SOAP port of the external WebSphere Application Server instance.

Configuring the dynamic workload broker instance

Table 16. *J2EEJobExecutorConfig.properties* file keywords (continued)

Keyword	Specifies...	Default value	Must be customized
connector.direct	The name of the direct communication channel without using the WebSphere Application Server scheduler. Select a direct invoker to have dynamic workload broker immediately forward the job to the external WebSphere Application Server instance components (EJB or JMS). When creating the job definition, you can specify if you want to use a direct or indirect connector in the J2EE Application pane in the Application page in the Job Brokering Definition Console, or in the invoker element in the JSDL file. For more information about the Job Brokering Definition Console, see the online help.	A single line with the following values separated by commas: <ul style="list-style-type: none"> • direct keyword • The following string: com.ibm.websphere.naming.WsnInitialContextFactory • The following string: corbaloc:iiop:washost.mydomain.com:2809 	You must customize the following: <ul style="list-style-type: none"> • The host name of the external WebSphere Application Server instance: washost.mydomain.com • The RMI port of the external WebSphere Application Server instance: 2809
trustStore.path	The path to the WebSphere Application Server trustStore file (this file must be copied to this local path from the WebSphere Application Server instance).	<i>TWA_home</i> /TWS/JavaExt/cfg/DummyClientTrustFile.jks	You can change the path (<i>TWA_home</i> /TWS/JavaExt/cfg), if you copy the trustStore path from the external WebSphere Application Server to this path.
trustStore.password	The password for the WebSphere Application Server trustStore file.	WebAs	Yes

Running {xor} encryption on your password: To {xor} encrypt your password, use the PropFilePasswordEncoder command located in the *WAS_home*/bin directory of the external WebSphere Application Server.

Follow these steps:

1. Open a new text file and write the following line:

```
string=your_password_in_plain_text
```

2. Save the file with a *file_name* of your choice.
3. Run PropFilePasswordEncoder as follows:

```
PropFilePasswordEncoder file_name string
```

where:

file_name

Is the name of the file with your password.

string Is the *string* you used in the text file. This can be any word that you choose, for example, password, mypwd, joe, and so on.

4. When the command completes, open the text file again. The content has changed to:

Configuring the dynamic workload broker instance

```
string={xor}your_encrypted_password
```

5. Copy your encrypted password, inclusive of the {xor} characters, and paste it where required into your property files.

For example, if you want to encrypt your password catamaran. Proceed as follows:

1. Open a text file and write the following:

```
mypasswd=catamaran
```

2. Save the file with name encrfile.txt.

3. Run:

```
PropFilePasswordEncoder encrfile.txt mypasswd
```

4. Open encrfile.txt. You find:

```
mypasswd={xor}PD4rPjI+LT4x
```

5. Copy {xor}PD4rPjI+LT4x and paste it where you need to.

The logging.properties file: The path to this file is *TWA_home*/TWS/JavaExt/cfg/logging.properties (*TWA_home*\TWS\JavaExt\cfg\logging.properties) on the agent.

After installation, this file is as follows:

```
# Specify the handlers to create in the root logger
# (all loggers are children of the root logger)
# The following creates two handlers
handlers = java.util.logging.ConsoleHandler, java.util.logging.FileHandler

# Set the default logging level for the root logger
.level = INFO

# Set the default logging level for new ConsoleHandler instances
java.util.logging.ConsoleHandler.level = INFO

# Set the default logging level for new FileHandler instances
java.util.logging.FileHandler.level = ALL
java.util.logging.FileHandler.pattern = C:\TWA_home\TWS\JavaExt\logs\javaExecutor%g.log
java.util.logging.FileHandler.limit = 1000000
java.util.logging.FileHandler.count = 10

# Set the default formatter for new ConsoleHandler instances
java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter
java.util.logging.FileHandler.formatter = java.util.logging.SimpleFormatter

# Set the default logging level for the logger named com.mycompany
com.ibm.scheduling = INFO
```

You can customize:

- The logging level (from INFO to WARNING, ERROR, or ALL) in the following keywords:
 - .level
 - java.util.logging.ConsoleHandler.level
 - com.ibm.scheduling
- The path where the logs are written, specified by the following keyword:
 - java.util.logging.FileHandler.pattern

The soap.client.props file: The path to this file is *TWA_home*/TWS/JavaExt/cfg/soap.client.props (*TWA_home*\TWS\JavaExt\cfg\soap.client.props) on the agent.

After installation, this file is as follows:

Configuring the dynamic workload broker instance

```
#-----  
# SOAP Client Security Enablement  
#  
# - security enabled status ( false[default], true )  
#-----  
com.ibm.SOAP.securityEnabled=false  
  
com.ibm.SOAP.loginUserId=wasadmin  
com.ibm.SOAP.loginPassword={xor}KD4sPjsyNjE\  
  
#-----  
# SOAP Login Prompt  
#  
# The auto prompting will happen only if all of the following are met:  
#  
# - Running from a SOAP client  
# - Server is reachable and server security is enabled  
# - Username and password are not provided either on command line or in this  
#   file  
# - com.ibm.SOAP.loginSource below is set to either "stdin" or "prompt"  
#  
#   stdin: prompt in command window  
#   prompt: GUI dialog box; falls back to stdin if GUI not allowed  
#  
# (So to disable auto prompting, set loginSource to nothing)  
#-----  
com.ibm.SOAP.loginSource=prompt  
  
#-----  
# SOAP Request Timeout  
#  
# - timeout (specified in seconds [default 180], 0 implies no timeout)  
#-----  
com.ibm.SOAP.requestTimeout=180  
  
#-----  
# SSL configuration alias referenced in ssl.client.props  
#-----  
com.ibm.ssl.alias=DefaultSSLSettings
```

If you want to enable SOAP client security, you must:

1. Change `com.ibm.SOAP.securityEnabled` to `true`
2. Customize:
 - `com.ibm.SOAP.loginUserId` with the true WebSphere Application Server administrator user ID.
 - `com.ibm.SOAP.loginPassword` with the true WebSphere Application Server administrator password in {xor} encrypted format. See “Running {xor} encryption on your password” on page 50.

Configuring the J2EE Job Executor Agent

To set up the environment on the external WebSphere Application Server (Versions 6.1 or 7.0) for the J2EE Job Executor Agent, do the following:

Create a Service Integration Bus

1. Open the WebSphere Administrative Console (for example, `http://localhost:9060/admin`, depending on the admin port you configured).
2. Expand **Service Integration** and select **Buses**. The Buses window is displayed.
3. Click **New** to display the Buses configuration window.
4. Type a name for the new bus, for example **MyBus** and click **Next** and then **Finish** to confirm.

Configuring the dynamic workload broker instance

5. Click the MyBus name and the MyBus properties are displayed.
6. Under Topology, click **Bus Members**. The Buses>MyBus>Bus members window is displayed.
7. Click **Add**, select the **Server** radio button, choose `<your_application_server_name>`, click **Next**, and then click **Finish**.
8. When the Confirm the addition of a new bus member panel is displayed, click **Finish**.
9. Select **Service Integration** → **Buses** → **MyBus** → **Destinations** → **New**.
10. Select **Queue** as the type and click **Next**
11. Type **BusQueue** as the identifier and assign the queue to a bus member. Click **Next**. In the confirmation panel click **Finish**.

Configure the Default Messaging Service

1. From the left panel of the WebSphere Administrative Console, expand **Resources** → **JMS** → **JMS Providers**, then click **Default messaging** at the server level as scope.
2. In the **Connection Factories** section, click **New**.
3. On the New JMS connection factory window, fill in the following fields:
Name MyCF
JNDI name
jms/MyCF
Bus name
MyBus
Provider endpoints
<hostname>:<Basic SIB port number>:BootstrapBasicMessaging;<hostname>:<Secure SIB port number>:BootstrapSecureMessaging,
where, <Basic SIB port number> and <Secure SIB port number> can be found by expanding **Servers**, selecting `<your_application_server_name>`, and then selecting **Messaging engine inbound transports** under **Server Messaging**.
4. Select again **Resources** → **JMS** → **JMS Providers** → **Default Messaging** at the server level as scope, locate the section **Destinations**, and click **Queues**. Click **New** and fill in the following fields as shown:
Name=MyQueue
JNDI name=jms/MyQueue
Bus name=MyBus
Queue name=BusQueue
Click **Ok**.
5. Select again **Resources** → **JMS** → **JMS Providers** → **Default Messaging** at the server level as scope, and locate the section **Activation Specifications**.
6. Click **JMS activation specification**. Click **New** and fill in the following fields as shown:
Name=MyActSpec
JNDI name=eis/MyActSpec
Bus name=MyBus
Destination type=Queue
Destination JNDI name=jms/MyQueue
Click **Ok**.

Configuring the dynamic workload broker instance

Configure the Java security

1. Select **Security** → **Secure Administration, applications and infrastructure**.
2. Locate the **Authentication** section, expand the **Java Authentication and Authorization Service**, and click **J2C authentication data**.
3. Click **New** and fill in the following fields as shown:

Alias=*usr*
User ID=*usr*
Password=*pwd*

where *usr* is the user ID authenticated when using connector security and *pwd* is the related password.

4. Click **Ok**.

Create an XA DataSource

1. In the left pane, go to **Resources** → **JDBC..** → **JDBCProviders**. In the resulting right pane, check that the scope is pointing to *<your_application_server_name>*.
2. Locate the **DERBY JDBC Provider (XA)** entry and click it.
3. Locate the **Additional Properties** section and click **Data Sources**.
4. Click **New** and fill in the following fields as shown:
Name = MyScheduler XA DataSource
JNDI name = jdbc/SchedulerXADS
Database name = \${USER_INSTALL_ROOT}/databases/Schedulers/
\${SERVER}/SchedulerDB;create=true
5. At the top of the page, click **Test connection button**.
6. Even if you get a negative result, modify the **Database name** field, deleting the part `;create=true`. Click **Ok**.

Create a WorkManager

1. In the left pane, go to **Resources** → **Asynchronous beans** → **Work managers** and click **New**.
2. Fill in the following fields as shown:
Name=SchedulerWM
JNDI name=wm/SchedulerWM
3. Click **Ok**.

Create and configure a scheduler

1. In the left pane, go to **Resources** → **Schedulers** and click **New**.
2. Fill in the following fields as shown:
Name=MyScheduler
JNDI name=sch/MyScheduler
Data source JNDI name=jdbc/SchedulerXADS
Table prefix=MYSCHED
Work managers JNDI name=wm/SchedulerWM
3. Click **Ok**.
4. Select **MyScheduler** and click **Create tables**.
5. Deploy the test application.

Security order of precedence used for running J2EE tasks

There are three ways of verifying that a task runs with the correct user credentials. Tasks run with specified security credentials using the following methods:

1. Java Authentication and Authorization Service (JAAS) security context on the thread when the task was created.

Configuring the dynamic workload broker instance

2. `setAuthenticationAlias` method on the `TaskInfo` object.
3. A specified security identity on a `BeanTaskInfo` task `TaskHandler` EJB method.

The authentication methods are performed in the order listed above, so that if an authentication method succeeds, the following checks are ignored. This means that the `usr` and `pwd` credentials defined in **Configure the Java security** take precedence over any credentials specified in the tasks themselves.

Configuring the workload broker workstation

The workload broker workstation transfers jobs submitted in Tivoli Workload Scheduler to the workload broker. The workload broker dynamically selects the agents on which to run the jobs depending on resource availability and optimization policies. The advantage of using the workload broker and the workload broker workstation together with Tivoli Workload Scheduler is that, when you schedule workload on Tivoli Workload Scheduler, you specify the name of the workload broker workstation as the destination CPU and at submission time the workload broker selects the best matching workstation.

The workload broker workstation works as a Tivoli Workload Scheduler standard agent and is configured with the dynamic brokering feature of the domain manager as part of the Tivoli Workload Scheduler installation procedure.

If you need to make configuration changes after the installation has completed, you can edit the `BrokerWorkstation.properties` file. The `BrokerWorkstation.properties` file contains the configuration properties shown in Table 17.

Table 17. Workload broker workstation configuration properties

Property	Values
<code>Broker.Workstation.Enable</code>	<p>A switch that enables or disables the workload broker workstation. The value can be true or false. The default value is true.</p> <p>Set this value to false if you decide not to use a workload broker workstation. Not using the workload broker workstation means that you can submit jobs dynamically on the workload broker directly (using either the Tivoli Dynamic Workload Console or the workload broker command line) without using the scheduling facilities of Tivoli Workload Scheduler.</p>
<code>Broker.Workstation.Name</code>	<p>The name of the workload broker workstation in the Tivoli Workload Scheduler production plan. This name is first assigned at installation time.</p>
<code>Broker.Workstation.Port</code>	<p>The port used by the workload broker workstation to listen to the incoming traffic (equivalent to the Netman port). It is first assigned at installation time. This port number must always be the same for all the workload broker workstations that you define in your Tivoli Workload Scheduler network (one with the master domain manager and one with every backup master you install) to ensure consistency when you switch masters.</p>

Configuring the dynamic workload broker instance

Table 17. Workload broker workstation configuration properties (continued)

Property	Values
Broker.Workstation.RetryLink	The number of seconds between consecutive attempts to link to the workload broker workstation. The default is 600.
MasterDomainManager.Name	The name of the master domain manager workstation.

Note that no SSL security is available for the connection between the master domain manager and the workload broker workstation. All the data between the two workstations is sent unencrypted. If this might cause a security risk in your environment, you can choose not to use the workload broker workstation functions, by setting `Broker.Workstation.Enable` to false.

Configuring security roles for users and groups

At Dynamic Workload Console installation time, new predefined roles and groups are created in Integrated Solutions Console. These roles determine which Dynamic Workload Console windows are available to a user, and therefore which activities the user is authorized to perform from the Dynamic Workload Console. If you do not assign a role to an Integrated Solutions Console user, that user does not see any entry for workload broker in the navigation tree. Access to the entries in the navigation tree does not mean that the user can access product functions. There is a second level of authorization, which is determined by a set of security roles created at master domain manager installation time in WebSphere Application Server. These roles define the levels of authorization needed to perform product functions regardless of the interface used.

You must therefore map the users and roles in WebSphere Application Server to match the users and roles defined in the Integrated Solutions Console so that communication between the Dynamic Workload Console and the workload broker instance is ensured. This procedure is described in “Mapping security roles to users and groups in WebSphere Application Server”

Mapping security roles to users and groups in WebSphere Application Server

When the workload broker instance is installed on your master domain manager, corresponding roles are set up in WebSphere Application Server. By default, these roles are not used. However, if you enable global security in your environment, the authorization required to perform any tasks is always validated by WebSphere Application Server. Users are required to provide credentials for accessing dynamic scheduling tasks. These credentials correspond to existing users defined in the domain user registry or the LDAP server.

To allow users and groups to access the workload broker functions when global security is enabled, they must be mapped to the security roles in WebSphere Application Server. This mapping allows those users and groups to access applications defined by the role. At installation time, the following actor roles are created in the WebSphere Application Server:

Table 18. Actors and their roles

Actor	Description
Operator	Monitors and controls the jobs submitted
Administrator	Manages the scheduling infrastructure

Configuring the dynamic workload broker instance

Table 18. Actors and their roles (continued)

Developer	Defines the jobs to be run specifying the job parameters, resource requirements, and so on.
Submitter	Manages the submission of their own jobs and monitors and controls the job lifecycle. This is the typical role for a Tivoli Workload Scheduler user.
	Tivoli Workload Scheduler acts as submitter of jobs to the Tivoli Workload Scheduler agent with dynamic scheduling capabilities
Configurator	Is the entity responsible for running the jobs on a local environment

To map security roles to users and groups on the WebSphere Application Server you must modify the **BrokerSecurityProps.properties** file using the **changeBrokerSecurityProperties** script.

To avoid the risk of changing a configuration value inadvertently or of overwriting the latest changes, you should always first create a file containing the current properties, edit it to the values you require, and apply the changes. Proceed as follows:

1. Log on to the computer where Tivoli Workload Scheduler is installed as the following user:

UNIX root

Windows Any user in the *Administrators* group.

2. Access the directory: <TWA_home>/wastools
3. Stop the WebSphere Application Server using the **conman stopappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
4. From that same directory run the following script to create a file containing the current broker security properties:

UNIX **showBrokerSecurityProperties.sh** > *my_file_name*

Windows **showBrokerSecurityProperties.bat** > *my_file_name*

5. Edit *my_file_name* with a text editor.
6. Edit the properties as you require. For each of the roles in the file, you can set the following properties:

Everyone?

Possible values:

- **Yes:** Every user is authorized to perform tasks for the role. No check is performed on the WebSphere Application Server user registry.
- **No :** Access is denied to users not defined in the WebSphere Application Server user registry.

All authenticated?

Possible values:

- **Yes:** All users belonging to the current WebSphere Application Server user registry who have been authenticated can access resources and tasks for the role. This is the default value.
- **No :** Access is granted only to those users and groups defined in the WebSphere Application Server user registry and listed in the mapped user and mapped group properties.

Configuring the dynamic workload broker instance

Mapped users

If specified, one or more users separated by the vertical bar symbol (|). This field can be left blank.

Mapped groups

If specified, one or more groups separated by the vertical bar symbol (|). This field can be left blank.

7. Save the file *my_file_name*.

8. Run the script:

Windows **changeBrokerSecurityProperties.bat** *my_file_name*

UNIX **changeBrokerSecurityProperties.sh** *my_file_name*

where *my_file_name* is the *fully qualified path* of the file containing the new parameters.

The properties are updated, according to the rules given in the descriptions of each property type.

9. Start the WebSphere Application Server using the **conman startappserver** command (see "Starting and stopping the application server and **appservman**" on page 268)

10. Check that the change has been implemented.

Notes:

1. If the mapped user or group names contain blanks, the entire user or group list must be specified between double quotes (""). For example, if you want to add the users John Smith, MaryWhite and DavidC to the developer role, you specify them as follows:

```
Role: Developer
Everyone?: No
All authenticated?: No
Mapped users:"John Smith|MaryWhite|DavidC"
Mapped groups:
```

2. In the file there is an additional default role named **WSClient** which you must leave as is.

Examples: To assign the Operator role to users Susanna and Ann belonging to the current WebSphere Application Server user registry:

```
Role: Operator
Everyone?: No
All authenticated?: No
Mapped users:Susanna|Ann
Mapped groups:
```

To assign the Administrator role to user Tom and the Developer role to the user group MyGroup defined in the current WebSphere Application Server user registry:

```
Role: Administrator
Everyone?: No
All authenticated?: No
Mapped users:Tom
Mapped groups:
```

```
Role: Developer
Everyone?: No
All authenticated?: No
Mapped users:
Mapped groups:MyGroup
```

BrokerSecurityProps.properties file

```
#####  
# Broker Security Properties  
#####
```

```
Role: WSClient  
Everyone?: No  
All authenticated?: Yes  
Mapped users:  
Mapped groups:
```

```
Role: Administrator  
Everyone?: No  
All authenticated?: Yes  
Mapped users:  
Mapped groups:
```

```
Role: Operator  
Everyone?: No  
All authenticated?: Yes  
Mapped users:  
Mapped groups:
```

```
Role: Submitter  
Everyone?: No  
All authenticated?: Yes  
Mapped users:  
Mapped groups:
```

```
Role: Configurator  
Everyone?: No  
All authenticated?: Yes  
Mapped users:  
Mapped groups:
```

```
Role: Developer  
Everyone?: No  
All authenticated?: Yes  
Mapped users:  
Mapped groups:
```

Configuring the Tivoli Workload Scheduler agent

The configuration settings of the Tivoli Workload Scheduler agent are contained in the `JobManager.ini` file. The file is made up of many different sections. Each section name is enclosed between square brackets and each section includes a sequence of `variable = value` statements.

You can customize properties for the following:

- Log and trace
- Native job executor
- Java job executor
- Resource advisor agent
- System scanner

Not all the properties in the `JobManager.ini` file can be customized. For a list of the configurable properties, see the tables in the following sections.

Configuring log and trace properties

You have the following options when configuring logs and traces:

Configuring logs

Edit the [JobManager.Logging] section in the JobManager.ini file. This procedure requires that you stop and restart the Tivoli Workload Scheduler agent.

Configuring traces

- Edit the [JobManager.Logging] section in the JobManager.ini file. This procedure requires that you stop and restart the Tivoli Workload Scheduler agent.
- Use one or more of the following command line commands, without stopping and restarting the Tivoli Workload Scheduler agent:
 - enableTrace
 - disableTrace
 - showTrace
 - changeTrace

For more information, see “Configuring traces without stopping the agent” on page 61.

In the JobManager.ini file, the section containing the log and trace properties is named:

```
[JobManager.Logging]
```

You can change the configurable properties listed in Table 19.

Table 19. Trace and log configurable properties of the Tivoli Workload Scheduler agent

	Property	Value
Log messages	JobManager.loggerhd.fileName	The name of the file where messages are to be logged.
	JobManager.loggerhd.maxFileBytes	The maximum size that the log file can reach. The default is 1024000 bytes.
	JobManager.loggerhd.maxFiles	The maximum number of log files that can be stored. The default is 3.
	JobManager.loggerfl.level	The amount of information to be provided in the logs. The value ranges from 4000 to 7000. Smaller numbers correspond to more detailed logs. The default is 4000.

Configuring the dynamic workload broker instance

Table 19. Trace and log configurable properties of the Tivoli Workload Scheduler agent (continued)

	Property	Value
Traces The output trace (JobManager_trace.log) is provided in XML format.	JobManager.trhd.fileName	The name of the trace file.
	JobManager.trhd.maxFileBytes	The maximum size that the trace file can reach. The default is 1024000 bytes.
	JobManager.trhd.maxFiles	The maximum number of trace files that can be stored. The default is 3.
	JobManager.trfl.level	The quantity of information to be provided in the traces. The value ranges from 0 to 3000. Smaller numbers correspond to more detailed tracing. The default is 3000.

Configuring traces without stopping the agent

Trace files are enabled by default for the Tivoli Workload Scheduler agent. You can use the following commands to configure traces without stopping and restarting the agent:

enableTrace

Sets the trace to the maximum level, producing a verbose result.

disableTrace

Sets the traces to the lowest level.

showTrace [>trace_file_name.xml]

Displays the current trace settings defined in the [JobManager.Logging] section of the JobManager.ini file. You can also redirect the [JobManager.Logging] section to a file to modify it. Save the modified file and use the **changeTrace** command to make the changes effective immediately.

changeTrace [trace_file_name.xml]

Reads the file containing the modified trace settings and implements the changes immediately and permanently, without stopping and restarting the Tivoli Workload Scheduler agent.

The following file is an example of the file created by the **showTrace** command:

```
<?xml version="1.0" encoding="UTF-8"?><jmgr:updateConfigurationResponse
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:jmgr="http://www.ibm.com/xmlns/prod/scheduling
/1.0/JobManager">
  <jmgr:Section name="JobManager.Logging.cclog">
    <jmgr:Property>
      <jmgr:Name>JobManager.trfl.level
    </jmgr:Name>
      <jmgr:Value>1011</jmgr:Value>
    </jmgr:Property>
    <jmgr:Property>
      <jmgr:Name>JobManager.trhd.maxFileBytes
```

Configuring the dynamic workload broker instance

```
1             <jmgr:Value>1024000</jmgr:Value>
1             </jmgr:Property>
1             <jmgr:Property>
1             <jmgr:Name>JobManager.trhd.maxFiles
1             </jmgr:Name>
1             <jmgr:Value>4</jmgr:Value>
1             </jmgr:Property>
1         </jmgr:updateConfigurationResponse>
```

where:

JobManager.trfl.level

Defines the quantity of information to be provided in the traces. The value ranges from 0 to 3000. Smaller numbers correspond to more detailed tracing. The default is 3000.

JobManager.trhd.maxFileBytes

Defines the maximum size that the trace file can reach. The default is 1024000 bytes.

JobManager.trhd.maxFiles

Defines the maximum number of trace files that can be stored. The default is 3.

Configuring common launchers properties

In the `JobManager.ini` file, the section containing the properties common to the different launchers (or executors) is named:

```
[Launchers]
```

You can change the configurable properties listed in Table 20:

Table 20. Common launchers configurable properties of the Tivoli Workload Scheduler agent

Property	Value
BaseDir	The installation path of the Tivoli Workload Scheduler agent.
SpoolDir	The path to the folder containing the jobstore and outputs. The default is: <i>value of BaseDir/stdl1dst/JM</i>
MaxAge	The number of days that job logs are kept (in path <i>TWA_home/TWS/stdl1dst/JM</i>) before being deleted. The default is 2. Possible values range from a minimum of 1 day.
CommandHandlerMinThreads	The default is 20.
CommandHandlerMaxThreads	The default is 100.
ExecutorsMinThreads	The default is 38.
ExecutorsMaxThreads	The default is 400.
NotifierMinThreads	The default is 3.
NotifierMaxThreads	The default is 5.

Configuring properties of the native job launcher

In the `JobManager.ini` file, the section containing the properties of the native job launcher is named:

```
[NativeJobLauncher]
```

Configuring the dynamic workload broker instance

You can change the configurable properties listed in Table 21.

Table 21. Native job launcher configurable properties of the Tivoli Workload Scheduler agent

Property	Value
AllowRoot	Applies to UNIX systems only. Specifies if the root user can run jobs on the agent. It can be true or false. The default is true.
CheckExec	If true, before launching the job, the agent checks both the availability and the execution rights of the binary file. The default is true.
LoadProfile	Specifies if the user profile is to be loaded. It can be true or false. The default is true.
PortMax	The maximum range of the port numbers used by the task launcher to communicate with the Job Manager. The default is 0, meaning that the operating system assigns the port automatically.
PortMin	The minimum range of the port numbers used by the task launcher to communicate with the Job Manager. The default is 0, meaning that the operating system assigns the port automatically.
RequireUserName	When true, requires that you add the user name in the JSDL job definition. When false, runs with the user name used by job manager, that is: <ul style="list-style-type: none">• <i>TWS_user</i> on UNIX and Linux systems• The local system account on Windows systems The default is false.
ScriptSuffix	The suffix to be used when creating the script files. It is: .cmd For Windows .sh For UNIX
VerboseTracing	Enables verbose tracing. It is set to true by default.

Configuring properties of the Java job launcher

In the `JobManager.ini` file, the section containing the properties of the Java job launcher is named:

```
[JavaJobLauncher]
```

You can change the configurable properties listed in Table 22:

Table 22. Java job launcher configurable properties of the Tivoli Workload Scheduler agent

Property	Value
JVMDir	The path to the virtual machine used to launch Java jobs. You can change the path to another compatible Java virtual machine.

Configuring properties of the Resource advisor agent

In the `JobManager.ini` file, the section containing the properties of the Resource advisor agent is named:

Configuring the dynamic workload broker instance

[ResourceAdvisorAgent]

You can change the configurable properties listed in Table 23.

Table 23. Resource advisor agent configurable properties of the Tivoli Workload Scheduler agent

Property	Value
FullyQualifiedHostname	The fully qualified host name of the Tivoli Workload Scheduler agent. It is configured automatically at installation time and is used to connect with the workload broker. Edit only if the host name is changed after installation.
ResourceAdvisorUrl	The value is <code>https://\$(tdwb_server):\$(tdwb_port)/JobManagerRESTWeb/JobScheduler/resource</code> , where: <ul style="list-style-type: none">• <code>\$(tdwb_server)</code> is the fully qualified host name of the workload broker• <code>\$(tdwb_port)</code> is the workload broker port number. It is configured automatically at installation time. Edit only if the host name or the port number are changed after installation, or if you do not use secure connection (set to <code>http</code>). If you set the port number to zero, the resource advisor agent does not start. The port is set to zero if at installation time you specify that you will not be using the workload broker.
CPUScannerPeriodSeconds	The time interval that the Resource advisor agent collects resource information about the local CPU. The default value is every 10 seconds.
ScannerPeriodSeconds	The time interval that the Resource advisor agent collects information about all the resources in the local system other than CPU resources. The default value is every 120 seconds.
NotifyToResourceAdvisorPeriodSeconds	The time interval that the Resource advisor agent forwards the collected resource information to the Resource advisor. The default (and maximum value) is every 180 seconds.

The resource advisor agent, intermittently scans the resources of the machine (computer system, operating system, file systems and networks) and periodically sends an update of their status to the workload broker.

The CPU is scanned every `CPUScannerPeriodSeconds` seconds, while all the other resources are scanned every `ScannerPeriodSeconds` seconds. As soon as one of the scans shows a significant change in the status of a resource, the resources are synchronized with the workload broker. The following is the policy followed by the agent to tell if a resource attribute has significantly changed:

- A resource is added or deleted
- A string attribute changes its value
- A CPU value changes by more than `DeltaForCPU`
- A file system value changes by more than `DeltaForDiskMB` megabytes
- A Memory value changes by more than `DeltaForMemoryMB` megabytes

Configuring the dynamic workload broker instance

If there are no significant changes, the resources are synchronized with the workload broker every `NotifyToResourceAdvisorPeriodSeconds` seconds.

Configuring properties of the System scanner

In the `JobManager.ini` file, the section containing the properties of the System scanner is named:

```
[SystemScanner]
```

You can change the configurable properties listed in Table 24.

Table 24. System scanner configurable properties of the Tivoli Workload Scheduler agent

Property	Value
CPUSamples	The number of samples used to calculate the average CPU usage. The default value is 3.
DeltaForCPU	The change in CPU usage considered to be significant when it becomes higher than this percentage (for example, DeltaForCPU is 20 if the CPU usage changes from 10 percent to 30 percent). The default value is 20 percent.
DeltaForDiskMB	The change in use of all file system resources that is considered significant when it becomes higher than this value. The default value is 100 MB.
DeltaForMemoryMB	The change in use of all system memory that is considered significant when it becomes higher than this value. The default value is 100 MB.

Configuring the dynamic workload broker instance

Chapter 4. Configuring the Tivoli Dynamic Workload Console

This chapter describes how to configure Tivoli Dynamic Workload Console. It is divided into the following sections:

- “Launching in context with the Tivoli Dynamic Workload Console”
- “Configuring roles to access to the Tivoli Dynamic Workload Console” on page 77
- “Configuring Tivoli Dynamic Workload Console to use Single Sign-On” on page 79
- “Configuring the use of Lightweight Third-Party Authentication” on page 80
- “Configuring Tivoli Dynamic Workload Console to use SSL” on page 84
- “Customizing Tivoli Dynamic Workload Console (Advanced configuration)” on page 84
- “Configuring Tivoli Dynamic Workload Console to view reports” on page 89
- “Preventing a connection to specific Tivoli Workload Scheduler Version 8.3 engines” on page 92

Launching in context with the Tivoli Dynamic Workload Console

This chapter describes how to create a URL to launch the Tivoli Dynamic Workload Console and have it directly open the results of a particular query.

You can then include this URL in an external application, for example, to monitor jobs and job streams that are critical to your business, and to quickly and easily manage them. You can access specific job or job stream details without having to create customized queries and monitor the state and health of the workstations that are critical in your environment so that, when unavailability or malfunctioning impacts job scheduling, you are alerted.

Scenarios

The following main scenarios can be identified:

- Obtain the result of a Monitor task on:
 - Jobs
 - Critical jobs
 - Job streams
- Obtain the result of a Monitor task on workstations
- Obtain the result of a saved task.

For all the scenarios, you must create a basic URL as described in “Creating a basic URL.”

Creating a basic URL

To create a basic URL, perform the following steps:

1. Define the URL to access the Tivoli Dynamic Workload Console:

```
https://{WebUIHostname:adminSecurePort}  
/ibm/console/xLaunch.do?pageID=com.ibm.tws.  
WebUI.External.navigation&showNavArea=false
```

where:

| **WebUIHostname**

| It is the fully qualified hostname or the IP address of the computer
| where the Tivoli Dynamic Workload Console is installed.

| **adminSecurePort**

| It is the number of the port on which the Tivoli Dynamic Workload
| Console is listening.

| **Example**

| `https://mypc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.
| External.navigation&showNavArea=false`

- | 2. Specify the action that you want to run, by specifying the corresponding
| parameter:

| **&action**

| It indicates the action that you want to perform and can have one of
| the following values:

- | • BrowseJobs
- | • ZBrowseJobs
- | • BrowseJobStreams
- | • BrowseCriticalJobs
- | • BrowseWorkstation
- | • InternalTask

- | 3. Specify the engine on which you want to run the query, by entering its
| parameters:

| **&hostname**

| For distributed environments, it is the host name or TCP/IP address of
| the computer on which the Tivoli Workload Scheduler engine is
| installed. For z/OS environments, it is the host name or TCP/IP
| address of the computer on which the z/OS connector is installed.

| **&port** The port number that is used to connect to the computer on which the
| Tivoli Workload Scheduler engine or the z/OS connector is installed.
| Typically, the default port numbers are:

| *Table 25. Default port numbers*

Port number	Engine
31117	Tivoli Workload Scheduler distributed engine
31127	Tivoli Workload Scheduler for z/OS engine with z/OS connector V.8.3
31217	Tivoli Workload Scheduler for z/OS engine with z/OS connector V.8.5 or higher

| **&server**

| It applies to z/OS systems only and is mandatory. It is the name of the
| remote server of the engine as it was specified in the z/OS connector.

| **Example**

| `&hostname = webuidev&port = 31217&server = C851`

| **Example of a complete URL:**

| `https://mypc:29043/ibm/console/xLaunch.do?pageID=
| com.ibm.tws.WebUI.External.navigation&showNavArea=false
| &action=BrowseJobs&hostname=webuidev&port=31117`

Advanced optional parameters

Depending on the query whose results you want to view, you can complete your URL with the following parameters.

Monitor Jobs on distributed systems

Create a URL by specifying the **BrowseJobs** action, as described in “Creating a basic URL” on page 67.

You can also specify any of the following filters:

&workstation

Filter by the workstation on which the jobs runs.

&jobstream

Filter by the job stream that contains the jobs.

&job

Filter by the job name.

&schedtime

Filter by the job scheduled time.

&status

Filter by the job status. You can filter by one or more statuses. Possible values are:

W	Waiting
O	Successful
H	Held
R	Ready
E	Error
U	Undecided
S	Running
C	Canceled
B	Blocked

&columns

Specify the number of columns that you want to display in your result table.

If not specified, the default number of columns for this query is shown.

Supported values are:

Min	Display a minimum set of columns
All	Display all columns

Example:

```
https://mypc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.  
External.navigation&showNavArea=false&action=BrowseJobs  
&hostname=webuidev&port=31117  
&workstation=my_ws&jobstream=my_js_name&job=my_job_name&status=ESB&columns=ALL
```

Monitor Jobs on z/OS systems

Create a URL by specifying the **ZBrowseJobs** action, as described in “Creating a basic URL” on page 67.

You can also specify any of the following filters:

&workstation

Filter by the workstation on which the job runs.

&jobstream

Filter by the job stream that contains the jobs.

&job Filter by the job name.

&schedtime

Filter by the job scheduled time.

&columns

Specify the number of columns that you want to display in your result table. If not specified, the default number of columns for this query is shown. Supported values are:

Min display a minimum set of columns

All display all columns

Example:

```
https://mypc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.  
External.navigation&showNavArea=false&action=ZBrowseJobs  
&hostname=webuidev&port=31117  
&server=C851&workstation=my_ws&jobstream=my_js_name  
&job=my_job_name&schedtime=200812081100
```

Monitor Critical Jobs

Create a URL by specifying the **BrowseCriticalJobs** action, as described in “Creating a basic URL” on page 67.

You can also specify any of the following filters:

&workstation

Filter by the workstation on which the job runs.

&jobstream

Filter by the job stream that contains the jobs.

&job Filter by the job name.

&schedtime

Filter by the job scheduled time.

&columns

Specify the number of columns that you want to display in your result table. If not specified, the default number of columns for this query is shown. Supported values are:

Min Display a minimum set of columns

All Display all columns

Example:

```
https://mypc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.  
External.navigation&showNavArea=false&action=BrowseCriticalJobs  
&hostname=webuidev&port=31117  
&workstation=my_ws&jobstream=my_js_name  
&job=my_job_name&server=C851&columns=Min
```

where **&server** is a parameter used for z/OS only.

Monitor Job Streams

Create a URL by specifying the **BrowseJobStreams** action, as described in “Creating a basic URL” on page 67.

You can also specify any of the following filters:

&workstation

Filter by the workstation on which the job stream runs.

&jobstream

Filter by the job stream name.

&columns

Specify the number of columns that you want to display in your result table. If not specified, the default number of columns for this query is shown. Supported values are:

Min Display a minimum set of columns

All Display all columns

Example:

```
https://mypc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.  
External.navigation&showNavArea=false&action=BrowseJobStreams  
&hostname=webuidev&port=31117  
&workstation=my_ws&jobstream=my_js_name
```

Monitor Workstations

Create a URL specifying the **BrowseWorkstation** action, as described in “Creating a basic URL” on page 67.

You can also specify any of the following filters:

&workstation

Filter by the workstation on which the job stream runs.

&columns

Specify the number of columns that you want to display in your result table. If not specified, the default number of columns for this query is shown. Supported values are:

Min Display a minimum set of columns

All Display all columns

Example:

```
https://mypc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.  
External.navigation&showNavArea=false&action=BrowseJobStreams  
&hostname=webuidev&port=31117  
&workstation=my_ws&jobstream=my_js_name  
&server=C851&columns=ALL
```

where **&server** is a parameter used for z/OS only.

Existing task

Create a URL by specifying the **InternalTask** action, as described in “Creating a basic URL” on page 67.

You can save this URL can be saved as a bookmark in your browser, so that, by clicking the bookmark, you can directly open the results of a previously created task.

To save a task URL, perform the following steps:

1. Create a task with the Tivoli Dynamic Workload Console:

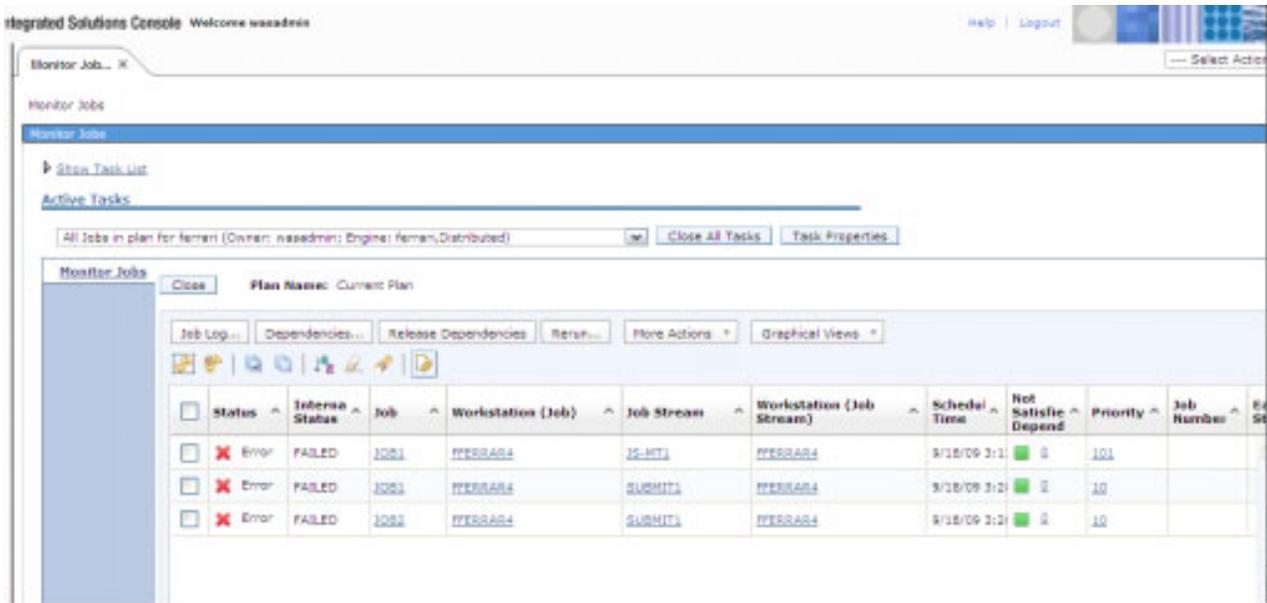


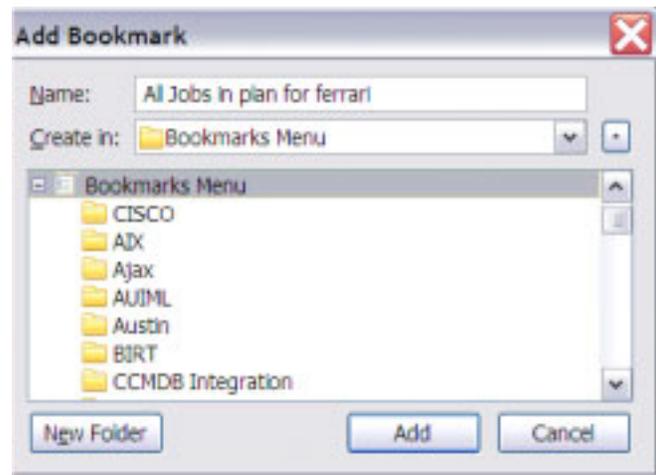
Figure 1. List of tasks

- From the displayed panel, click the **Add bookmark icon**



to save this link in your bookmarks.

- Specify a name for the new bookmark. By default the task name is used. Organize your bookmarks for your convenience, for example, you might organize your saved tasks in a different folder for each engine.



Example of a saved bookmark:

```
https://cairapc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.
External.navigation&showNavArea=false
&action=InternalTask&hostname=fferrar4&port=31117
&taskname=All%20Jobs%20in%20plan%20for%20ferrari
```

Starting from this bookmark you can manually create a URL as follows:

```
https://mypc:29043/ibm/console/xLaunch.do?pageID=com.ibm.tws.WebUI.  
External.navigation&showNavArea=false&action= InternalTask  
&hostname=webuidev&port=31117  
&server=C851 &taskname=myTask
```

where `&server` parameter is used for z/OS engines only.

Configuring access to the Tivoli Dynamic Workload Console

As soon as you finish installing the Tivoli Dynamic Workload Console, you can launch it by using the link provided in the final installation panel.

However, after the installation, the administrator is the only user who can log into the console, using the credentials specified during the installation.

This is the user defined in the Local OS user registry (for Windows) or in the custom user registry (for UNIX). For further details on custom user registry see, “Modify or display security settings - custom user registry” on page 263.

There are two important steps an administrator must perform before other users can log into and use the Tivoli Dynamic Workload Console

- “Configuring a User Registry”
- “Configuring roles to access to the Tivoli Dynamic Workload Console” on page 77

If WebSphere Application Server is configured to use the local operating system user registry, users and groups must be created into the local operating system. However, you can change the user registry to use the LDAP or federated user registry, which is embedded with WebSphere Application Server.

Users defined in the user registry can log in to the Tivoli Dynamic Workload Console; then they need to be associated to a role to be able access the Tivoli Dynamic Workload Console features (see “Configuring roles to access to the Tivoli Dynamic Workload Console” on page 77.)

Note: Before implementing any change in WebSphere Application Server configuration, save a backup version of the existing configuration by running the command `TWA_HOME/wastools/backupConfig.sh` (for UNIX) or `TWA_HOME/wastools/backupConfig.bat` (for Windows)

Configuring a User Registry

Configuring an LDAP User Registry

If WebSphere Application Server is configured to use LDAP user registry, users and groups must be created by the system administrator into the LDAP.

Choose one of the following options to configure the Tivoli Dynamic Workload Console to use an LDAP user registry:

- Use the `showSecurityProperties` and `changeSecurityProperties` scripts to update the security configuration, as described in “Changing security properties” on page 259. You can customize the following templates and run them it as input for the `changeSecurityProperties.sh` (for UNIX) and `changeSecurityProperties.bat` (for Windows):

Note: After the WebSphere Application Server has been modified to use LDAP, it is important to update it by using the **updateWas.sh** script (see “Updating associated WebSphere Application Server properties” on page 138 for full details).

Active Directory

```
#####  
Global Security Panel  
#####  
enabled=true  
enforceJava2Security=false  
useDomainQualifiedUserNames=false  
cacheTimeout=600  
ltpaTimeOut=720  
issuePermissionWarning=true  
activeProtocol=CSI  
useFIPS=false  
activeAuthMechanism=LTPA  
activeUserRegistry=LDAP  
  
#####  
LDAP Panel  
#####  
LDAPServerId=TWSAdmin  
LDAPPassword=mypassword  
LDAPPrimaryAdminId=  
LDAPUseRegistryServerId=true  
LDAPServerType=ACTIVE_DIRECTORY  
LDAPHostName=ldapservr.mycompany.com  
LDAPPort=389  
LDAPBaseDN=dc=mycompany,dc=com  
LDAPBindDN=CN=ldap bind,DC=mycompany,DC=com  
LDAPBindPassword=bindpassword  
LDAPsearchTimeout=120  
LDAPPreuseConnection=true  
LDAPIgnoreCase=true  
LDAPsslEnabled=false  
LDAPsslConfig=DefaultNode/DefaultSSLSettings  
  
#####  
Advanced LDAP Panel  
#####  
LDAPUserFilter=(&(sAMAccountName=%v)(objectCategory=user))  
LDAPGroupFilter=(&(cn=%v)(objectCategory=group))  
LDAPUserIdMap=*:sAMAccountName  
LDAPGroupIdMap=*:cn  
LDAPGroupMemberIdMap=memberof:member  
LDAPCertificateFilter=  
LDAPCertificateMapMode=EXACT_DN
```

SunOne

```
#####  
Global Security Panel  
#####  
enabled=true  
enforceJava2Security=false  
useDomainQualifiedUserNames=false  
cacheTimeout=600  
ltpaTimeOut=720  
issuePermissionWarning=true  
activeProtocol=CSI  
useFIPS=false  
activeAuthMechanism=LTPA  
activeUserRegistry=LDAP  
  
#####
```

```

LDAP Panel
#####
LDAPServerId=TWSAdmin
LDAPPassword=mypassword
LDAPPrimaryAdminId=
LDAPUseRegistryServerId=true
LDAPServerType=IPLANET
LDAPHostName=ldapsrvr.mycompany.com
LDAPPort=389
LDAPBaseDN=dc=mycompany,dc=com
LDAPBindDN=CN=ldap bind,DC=mycompany,DC=com
LDAPBindPassword=bindpassword
LDAPsearchTimeout=120
LDAPPreuseConnection=true
LDAPIgnoreCase=true
LDAPsslEnabled=false
LDAPsslConfig=DefaultNode/DefaultSSLSettings

#####
Advanced LDAP Panel
#####
LDAPUserFilter=(uid=%v)(objectclass=inetOrgPerson)
LDAPGroupFilter=(cn=%v)(objectclass=groupofuniquenames))
LDAPUserIdMap=*.uid
LDAPGroupIdMap=*.cn
LDAPGroupMemberIdMap=*.uniqueMember
LDAPCertificateFilter=
LDAPCertificateMapMode=EXACT_DN

```

IBM Tivoli Directory Server

```

#####
Global Security Panel
#####
enabled=true
enforceJava2Security=false
useDomainQualifiedUserNames=false
cacheTimeout=600
ltpaTimeOut=720
issuePermissionWarning=true
activeProtocol=CSI
useFIPS=false
activeAuthMechanism=LTPA
activeUserRegistry=LDAP

#####
LDAP Panel
#####
LDAPServerId=TWSAdmin
LDAPPassword=mypassword
LDAPPrimaryAdminId=
LDAPUseRegistryServerId=true
LDAPServerType=IBM_DIRECTORY_SERVER
LDAPHostName=ldapsrvr.mycompany.com
LDAPPort=389
LDAPBaseDN=dc=mycompany,dc=com
LDAPBindDN=CN=ldap bind,DC=mycompany,DC=com
LDAPBindPassword=bindpassword
LDAPsearchTimeout=120
LDAPPreuseConnection=true
LDAPIgnoreCase=true
LDAPsslEnabled=false
LDAPsslConfig=DefaultNode/DefaultSSLSettings

#####
Advanced LDAP Panel
#####
LDAPUserFilter=(uid=%v)(objectclass=person))

```

```
LDAPGroupFilter=(amp(cn=%v)(objectclass=groupOfNames))
LDAPUserIdMap=:uid
LDAPGroupIdMap=:cn
LDAPGroupMemberIdMap=ibm-allGroups:member
LDAPCertificateFilter=
LDAPCertificateMapMode=EXACT_DN
```

Alternatively

- 1. Login to the Tivoli Dynamic Workload Console with current WebSphere Application Server administration credentials.
 2. Configure the LDAP as described in “Configuring for LDAP” on page 135
 3. Stop the server using the commands `stopappserver`, as described in the *Tivoli Workload Scheduler: User’s Guide and Reference*. To stop the server, use the original WebSphere administrator credentials.
 4. Restart the server using the commands `startappserver`, as described in the *Tivoli Workload Scheduler: User’s Guide and Reference*.
 5. After the WebSphere Application Server has been modified to use LDAP, it is important to update it by using the `updateWas.sh` script (see “Updating associated WebSphere Application Server properties” on page 138 for full details).

Configuring a Federated User Registry

Follow this steps to configure the Tivoli Dynamic Workload Console to use a federated user registry.

1. Login to the Tivoli Dynamic Workload Console with current WebSphere Application Server administration credentials.
2. Expand the **Security** menu in the navigation tree on the left and click **Secure administration, applications, and infrastructure**.
3. Click the **Security Configuration Wizard** button and follow the displayed steps:
 - a. **Step 1:** Leave the default selection
 - b. **Step 2:** Select **Federated repositories**
 - c. **Step 3:** Specify a **Primary administrative user name**. This will be the new WebSphere administrator.
 - d. **Step 4:** Leave the default selection
4. Finish the wizard and click **Apply** on the parent panel to set as current user registry the **Federated repositories**.
5. Finally, click the link: **Save directly to the master configuration**.
6. Stop the server using the commands `stopappserver`, as described in the *Tivoli Workload Scheduler: User’s Guide and Reference*. To stop the server, use the original WebSphere administrator credentials.
7. Restart the server using the commands `startappserver`, as described in the *Tivoli Workload Scheduler: User’s Guide and Reference*.
8. Login to WebSphere using the newly changed user and password.
9. Assign to the new administrative user the TWSWEBUIAdministrator role.
10. Create new users and groups as explained in “Creating users and groups with the Federated User Registry” on page 77 and assign roles to them, as explained in “Configuring roles to access to the Tivoli Dynamic Workload Console” on page 77.

11. After the WebSphere Application Server has been modified, it is important to update it by using the `updateWas.sh` script (see “Updating associated WebSphere Application Server properties” on page 138 for full details).

Creating users and groups with the Federated User Registry: You can create one or more users. The users are added to the registry and a login account for each new user is automatically created. When creating the new user, you can also add the user as a member of one or more groups. The way you can add users and groups to the Tivoli Dynamic Workload Console varies, depending on the user registry used by WebSphere Application Server.

1. Launch the Tivoli Dynamic Workload Console by using the link provided in the final installation panel.
2. Login providing the WebSphere Application Server credentials.
3. Expand the portfolio on your left by clicking **Users and Groups** -> **Manage Users**
4. Click **Create** to create a new user.
5. In the **User ID** field, type a unique name to identify the user. This user ID will be added to the user registry and also will be used as the login account name.
6. Optionally, Click **Group Membership** and then follow the wizard to add the user as a member of one or more existing groups.
7. Enter the remaining information for the new user and, when you are finished, click **Create**
8. If successful, a message indicates that the user has been created. Also, the user ID and other user information will be added to the user registry, and a new login account will be created for the user.
9. To create another user, click **Create Another**.

Configuring roles to access to the Tivoli Dynamic Workload Console

During the Tivoli Dynamic Workload Console installation, new predefined roles are created in the Integrated Solutions Console. They determine which console panels are available to a user, and therefore which activities that user can perform from Tivoli Dynamic Workload Console.

If you do not assign any of the predefined roles to an Integrated Solutions Console user, that user, after having logged in, will not see any entry for Tivoli Workload Scheduler, and dynamic workload broker in the navigation tree.

Tip It is not necessary to assign a role to every single user. If the user registry already contains groups of users that are properly defined for using the console, it is possible to assign roles to groups too. If groups are not available in the user registry, then the special group **ALL_AUTHENTICATED_USERS** can be used to assign roles to *all* the users at once.

Table 26 lists the predefined roles created in the Integrated Solutions Console for accessing the Tivoli Workload Scheduler environments using Tivoli Dynamic Workload Console, and the panels they can access:

Table 26. Tivoli Workload Scheduler predefined roles

Roles	Tivoli Workload Scheduler panels accessible from the Navigation Tree
TWSWEBUIAdministrator	All panels

Configuring new users

Table 26. Tivoli Workload Scheduler predefined roles (continued)

Roles	Tivoli Workload Scheduler panels accessible from the Navigation Tree
TWSWEBUIOperator	All Configured Tasks Dashboard Workload <ul style="list-style-type: none"> Forecast <ul style="list-style-type: none"> Generate Trial Plan Generate Forecast Plan List Available Plans Submit <ul style="list-style-type: none"> Submit Predefined Job Streams Submit Predefined Jobs Submit Ad Hoc Jobs Monitor <ul style="list-style-type: none"> Monitor Jobs Monitor Critical Jobs Monitor Job Streams Show Plan View Workload Dependencies <ul style="list-style-type: none"> Monitor Files Monitor Resources Monitor Prompts Workload Events <ul style="list-style-type: none"> Monitor Event Rules Monitor Operator Messages Monitor Triggered Actions Scheduling Environment <ul style="list-style-type: none"> Monitor <ul style="list-style-type: none"> Monitor Workstations Monitor Domains Settings <ul style="list-style-type: none"> Manage User Preferences
TWSWEBUIDeveloper	Workload <ul style="list-style-type: none"> Design <ul style="list-style-type: none"> Create Workload Definitions Create Event Rules List Workload Definitions List Event Rules List Jobs on SAP Settings <ul style="list-style-type: none"> Manage User Preferences
TWSWEBUIAnalyst	All Configured Reports Reporting <ul style="list-style-type: none"> Generate Historical reports Generate Plan Reports Generate Custom SQL Reports Settings <ul style="list-style-type: none"> Manage User Preferences
TWSWEBUIConfigurator	Scheduling Environment <ul style="list-style-type: none"> - Design <ul style="list-style-type: none"> - Create Workstations - List Workstations Settings <ul style="list-style-type: none"> Manage Engines Manage User Preferences

Assigning a predefined role to an Integrated Solutions Console user allows that user to access the Tivoli Dynamic Workload Console panels. The Tivoli Workload Scheduler user specified in the engine connection, instead, determines which operations can be run locally on the connected Tivoli Workload Scheduler engine. For example, if the user specified in a Tivoli Workload Scheduler engine connection is not authorized to run reporting in the Tivoli Workload Scheduler *Security file*, then, even though the Integrated Solutions Console user logged in to Tivoli Dynamic Workload Console can access the reporting panels, he or she cannot perform reporting operations on that specific Tivoli Workload Scheduler engine. For more information about how to configure the security file, see “Configuring the security file” on page 99

Table 27 lists the predefined roles created in the Integrated Solutions Console for accessing the dynamic workload broker environments using Tivoli Dynamic Workload Console, and the panels they can access:

Table 27. Tivoli Dynamic Workload Broker predefined roles

Roles	dynamic workload broker panels accessible from the Navigation Tree
TDWBAdministrator	All panels
TDWBOperator	Scheduling Environment Configuration Definitions, except Define a New Job Tracking Preferences
TDWBDeveloper	Configuration Definitions Preferences
TDWBConfigurator	Scheduling Environment Configuration Tracking, except Job Instances Preferences

Configuring Tivoli Dynamic Workload Console to use Single Sign-On

Single Sign-On (SSO) is a method of access control that allows a user to authenticate once and gain access the resources of multiple applications sharing the same user registry.

This means that using SSO you can run queries on the plan or manage object definitions on the database accessing the engine without authenticating, automatically using the same credentials you used to login to the Tivoli Dynamic Workload Console.

After the installation completes you can configure Tivoli Dynamic Workload Console and the Tivoli Workload Scheduler engine, to use SSO. To do this they must share the same LDAP user registry. See “Configuring an LDAP User Registry” on page 73, for more details.)

The Lightweight Directory Access Protocol (LDAP) is an application protocol for querying and modifying directory services running over TCP/IP.

If you configured Tivoli Dynamic Workload Console to use Single Sign-On with an engine, then, the following behavior is applied:

Configuring Tivoli Dynamic Workload Console to use Single Sign-On

If engine connection has the user credentials specified in its definitions

These credentials are used. This behavior regards also engine connections that are shared along with their user credentials.

If the user credentials are not specified in the engine connection

The credentials you specified when logging in to Tivoli Dynamic Workload Console are used. This behavior regards also shared engine connections having unshared user credentials.

LTPA token-keys

In addition to sharing the same LDAP user registry, the instance of WebSphere Application Server that supports the Tivoli Dynamic Workload Console and also the instance which supports the engine where the Single Sign-On is required, must both be configured to use the same Lightweight Third-Party Authentication token-keys. See “Configuring the use of Lightweight Third-Party Authentication”

Configuring the use of Lightweight Third-Party Authentication

The WebSphere Application Server uses the Lightweight Third-Party Authentication (LTPA) mechanism to propagate user credentials.

Depending on your circumstances, you might need to configure the use of the same LTPA token_keys between Tivoli Dynamic Workload Console and the engine, or disable the automatic generation of the LTPA token_keys, or both:

Configuring for Single Sign-On

If you are configuring for Single Sign-On, between any version of Tivoli Dynamic Workload Console and any engine, whether or not they are installed on the same system, you must configure both instances of WebSphere Application Server involved to use the same LTPA token_keys, and disable their automatic regeneration on expiry, following the procedures described in:

- “Configuring to use the same LTPA token_keys” on page 81
- “Disabling the automatic generation of LTPA token_keys” on page 83

More than one instance of WebSphere Application Server on one system

In this case you *must* use the same LTPA token_keys on all the applications on the system that use a version of WebSphere Application Server (for example, the Tivoli Dynamic Workload Console and the Tivoli Workload Scheduler engine, regardless of their versions), even if you are not implementing Single Sign-On. You must also disable their automatic regeneration on expiry. Whether you need to take any action depends on which multiple instances of WebSphere Application Server are involved:

Multiple instances of the embedded WebSphere Application Server freshly installed in multiple Tivoli Workload Automation instances

In any system, if you have installed more than one instance of the embedded WebSphere Application Server (for example, Tivoli Dynamic Workload Console and Tivoli Workload Scheduler in different instances of Tivoli Workload Automation) they will by default use the same keys, so you only need to take an action to use the same keys if you need to change them for any reason. However, you must disable the automatic regeneration on expiry, following the procedure described in:

- “Disabling the automatic generation of LTPA token_keys” on page 83

Configuring Tivoli Dynamic Workload Console to use LTPA

Other multiple instances of the WebSphere Application Server

In any other circumstances where two instances of WebSphere Application Server are installed on the same system (for example, you have installed Tivoli Dynamic Workload Console on an external WebSphere Application Server and a Tivoli Workload Scheduler component in a Tivoli Workload Automation instance on the same system, or you have installed a new instance of Tivoli Dynamic Workload Console to work with a previous version of Tivoli Workload Scheduler, or vice versa), you must yourself ensure that all instances use the same keys, and disable their automatic regeneration on expiry, following the procedures described in:

- “Configuring to use the same LTPA token_keys”
- “Disabling the automatic generation of LTPA token_keys” on page 83

No Single Sign-On, and only one instance of WebSphere Application Server on a system

No action need to be taken.

Configuring to use the same LTPA token_keys

To use the same LTPA token_keys between WebSphere Application Servers, you must run this procedure between Tivoli Dynamic Workload Console and each engine you want to connect to.

The LTPA token_keys can be either exported from Tivoli Dynamic Workload Console and imported into the engine, or exported from the engine and imported into Tivoli Dynamic Workload Console.

1. Use the following script to export the LTPA token_keys from the WebSphere Application Server where the Tivoli Dynamic Workload Console is installed, and to import them into the other instance of WebSphere Application Server:

Tivoli Workload Scheduler and Tivoli Dynamic Workload Console, Version 8.5 and 8.5.1

```
<TWA_home>/wastools/manage_ltpa.sh or ... \manage_ltpa.bat
```

Tivoli Workload Scheduler, Version 8.4

```
<TWA_home>/wastools/manage_ltpa.sh or ... \manage_ltpa.bat
```

Tivoli Dynamic Workload Console, Version 8.4

```
tdwc_install_dir\tdwcutils\scripts\manage_ltpa.sh or  
... \manage_ltpa.cmd
```

Tivoli Workload Scheduler and Tivoli Dynamic Workload Console, Version 8.3

For information relating to these releases see the relevant product documentation.

There is also a copy of manage_ltpa.sh and manage_ltpa.bat on each installation CD.

Make sure that the user who runs this script is allowed to access the WebSphere Application Server profile that hosts the Tivoli Dynamic Workload Console or the engine.

The syntax used to run the script is the following:

```
manage_ltpa -operation import|export -profilepath profile_path  
-ltpafile LTPA_file_path -ltpapassword LTPA_file_password  
-user username -password password  
-port SOAP_port -server server_name
```

Configuring Tivoli Dynamic Workload Console to use LTPA

where:

-operation

Select *export* to read the LTPA token_keys from the profile and save it to a file. Select *import* to update the profile with the LTPA token_keys stored in a file.

-profilepath

Specify the path to the profile on top of which the application, either Tivoli Dynamic Workload Console or Tivoli Workload Scheduler is installed.

-ltpafile

Specify the fully qualified path name of the file that contains, if you import, or where to encrypt, if you export, the LTPA token_keys.

-ltpapassword

Specify a password of your choice to encrypt the file that contains the LTPA keys when exporting them, or, when importing them, the password that was used to encrypt them when they were exported. This password is used only when importing and exporting that LTPA token_keys. It does not need to match the administrator password.

-user The administrator of the server hosting the Tivoli Dynamic Workload Console or the engine. In the case of Tivoli Workload Scheduler, the administrator is, by default, the owner of the instance (*TWS_user*).

-password

The password of the administrator of the server defined in the selected profile.

-port Specify the SOAP port used by the profile. By default the SOAP port is 28880 for Tivoli Dynamic Workload Console installed on the embedded WebSphere Application Server, and 31118 for Tivoli Workload Scheduler installed on the embedded WebSphere Application Server.

-server

Specify the name of the server of the profile on which to import or export the LTPA tokens. The default server name varies, depending on how it was installed. See Table 28 on page 83.

Notes:

- a. The server and path might have been modified from the default value after installation.
- b. This keyword is mandatory if the Tivoli Workload Scheduler server name is different from the Tivoli Dynamic Workload Console server name.

Configuring Tivoli Dynamic Workload Console to use LTPA

Table 28. Product versions and default server names

Product version	WebSphere Application Server version	Default server name
Tivoli Workload Scheduler, V8.5 and 8.5.1:	The embedded WebSphere Application Server installed in an instance of Tivoli Workload Automation (on which any Tivoli Workload Scheduler component, including the Tivoli Dynamic Workload Console is installed).	twaserver<n>, found in the following path: <TWA_home>/eWAS/profiles/twaprofile/config/cells/DefaultNode/nodes/DefaultNode/servers
	Your external version of the WebSphere Application Server, on which the Tivoli Dynamic Workload Console is installed.	server1, found in the appropriate path of your external version of WebSphere Application Server
Tivoli Workload Scheduler, V8.4 and before:	The embedded WebSphere Application Server on which Tivoli Workload Scheduler is installed.	server1, found in the following path: <TWS_home>/appserver/profiles/twsprofile/config/cells/DefaultNode/nodes/DefaultNode/servers
	The embedded WebSphere Application Server on which the Tivoli Dynamic Workload Console is installed.	tdwserver, found in the following path: <tdwc_install_dir>/AppServer/profiles/tdwcprofile/servers
	Your external version of the WebSphere Application Server, on which the Tivoli Dynamic Workload Console is installed.	server1, found in the appropriate path of your external version of WebSphere Application Server

2. Stop and start each server involved in this activity to enable it.
3. If you are configuring Single Sign-On, test that the configuration is correctly set between Tivoli Dynamic Workload Console and the engine by performing the following steps:
 - a. Log in to Tivoli Dynamic Workload Console.
 - b. Create an engine connection without specifying User ID and password.
 - c. Perform a test connection.

The next step is to disable the automatic generation of the LTPA token_keys, for which see: "Disabling the automatic generation of LTPA token_keys"

Disabling the automatic generation of LTPA token_keys

Disable the automatic generation of LTPA token_keys, in either of the following circumstances:

- You are enabling Single Sign-On
- You have more than one instance of WebSphere Application Server on the same system

You must disable the generation of the keys at both ends of the communication, in other words, at the Tivoli Dynamic Workload Console, and at the engine of Tivoli Workload Scheduler or dynamic workload broker, as appropriate:

Configuring Tivoli Dynamic Workload Console to use LTPA

At the Tivoli Dynamic Workload Console

1. Log in to Tivoli Dynamic Workload Console.
2. Click **Security**, and then **Secure administration, applications, and infrastructure**.
3. Click **Authentication mechanisms and expiration**.
4. Click the **Key set groups** link.
5. Click on the name of the key set group displayed in the list.
6. Clear the **Automatically generate keys** check box.
7. Click **OK**.
8. Check in the list that the field **Automatically generate keys** beside the available key set group is set to *false*.

At Tivoli Workload Scheduler

The implementation of the embedded WebSphere Application Server on the Tivoli Workload Scheduler engine includes a limited functionality version of the Integrated Solutions Console portal. Use this portal to disable the automatic generation of LTPA token_keys, as follows:

1. Connect to the Integrated Solutions Console portal from an Internet browser, using the following URL:

```
http://TWS_server_hostname:WAS_admin_host_(secure_)port/ibm/console
```

Use the **showHostProperties** tool to identify the *WAS_admin_host_port* (default 31123) or *WAS_admin_host_secure_port* (default 31124), as appropriate. For more information about this tool, refer to “Application server - using the utilities that change the properties - reference” on page 277.

2. Perform the procedure you used above to disable the token-keys generation for the Tivoli Dynamic Workload Console, starting from step 2.

Configuring Tivoli Dynamic Workload Console to use SSL

The Secure Sockets Layer (SSL) protocol provides secure communications between remote server processes or applications. SSL security can be used to establish communications inbound to, and outbound from, an application. To establish secure communications, a certificate, and an SSL configuration must be specified for the application.

Full details are supplied in Chapter 7, “Setting connection security,” on page 169.

1 Customizing Tivoli Dynamic Workload Console (Advanced configuration)

1 Optionally, you can configure some advanced settings of Tivoli Dynamic Workload
1 Console to customize its behavior. To do it, you must specify your settings in a
1 customizable file named *TdwcGlobalSettings.xml*, which you must then copy it in
1 the Tivoli Dynamic Workload Console installation structure.

1 When you modify this file, stop and restart the Tivoli Dynamic Workload Console
1 to make the changes effective in your environment.

Customizing your global settings

Users with Administrator privileges can use a configuration file, named `TdwcGlobalSettings.xml`, to add and modify some customizable information.

A template of this file is located on the installation DVD under `WEBUI/platform_name/utlils`. You can modify it, replacing default values with customized ones and enabling commented sections. After customizing, you must copy it under the following path: `Installation_dir/TWA/eWAS/profiles/TIPProfile/registry` directory.

This file is accessed at each login, and all configurations specified in the file are immediately applied, except for **precannedTaskCreation** property. This property is read only when a user logs in for the first time and then is used whenever this user logs in again.

You can use any text or XML editor to edit this file, but ensure that you save it is as a valid XML file.

The file is organized into the following sections that group similar properties:

graphViews

The `graphViews` section contains the configuration parameters that apply to the graphical views in the plan, such as the maximum number of objects shown in each view.

planViewMaxJobstreams

The maximum number of job streams displayed in the Plan View. Default value is **1000**.

jobstreamViewLimit

The maximum number of objects displayed in the Job Stream View. Default value is **1000**.

impactViewLimit

The maximum number of job streams displayed in the Impact View. Default value is **2000**.

planViewNewWindow

Set it to **TRUE** if you want the plan view to be displayed in a new window each time it is launched. Default value is **FALSE**.

NewsFeed

The `NewsFeed` section contains the configuration details to be constantly up-to-date with product information.

FeedURL

Contains the URL from which you receive news and updates. Default value is the product's Information Center at: `http://publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/com.ibm.tivoli.itws.doc_8.5.1.1/ic-homepage.html`

PollInterval

The interval in seconds between two checks for updates. Default value is **3600**.

FeedType

A string that identifies the type of update. Default value is **JSONP**.

Configuring Tivoli Dynamic Workload Console to use SSL

1 **application**
1 The application section defines the environment for which predefined tasks are
1 created

1 **precannedTaskCreation**
1 Some predefined tasks are created by default and are available when you
1 log in to the console. There is a predefined Monitor task for every object,
1 for both z/OS and distributed engines. Default value is **all**. To change this
1 setting, use one of the following values:
1

- 1 • all
- 1 • distributed
- 1 • zos
- 1 • none

1 **help**
1 The help section indicates the address of the current version of the Information
1 Center.

1 **infocenterURL**
1 The URL of the Information Center. Default value is: `http://`
1 `publib.boulder.ibm.com/infocenter/tivihelp/v3r1/topic/`
1 `com.ibm.tivoli.itws.doc_8.5.1.1/ic-homepage.html`.

1 **security**
1 Use this section to configure some properties related to the User Registry in use.

1 **groupIdMap**
1 The property is related to the groups of User Registry, and can be modified
1 to map and display the specified value of each group. The default is the
1 common name of the group.

1 **Examples:**
1

```
<settings>  
1            <security>  
1              <property name="groupIdMap" value="cn"></property>  
1            </security>  
1           </settings>
```

1 Therefore, if you need to change the default value "cn" to "racfid", you can define
1 this property as follows:
1

```
<property name="groupIdMap" value="racfid"></property>
```

1 **twisObjectDoc**
1 The twisObjectDoc section contains URLs where you can store customized
1 documentation about your jobs or job streams. By default this setting is not
1 specified. If you want to associate customized documentation to a job or job
1 stream, use this setting to specify the external address where this information is
1 located. When you specify this setting, an action to access the relevant
1 documentation becomes available from **More Actions** menus in Monitor Jobs and
1 Monitor Job Streams tasks as well as in the graphical views in the plan (in the
1 object's tooltips, context menus and properties) making it possible to open the
1 documentation while monitoring your job or job stream in the plan.

1 To change this setting, use one of the following values:

Configuring Tivoli Dynamic Workload Console to use SSL

customActionLabel

The name of the action displayed in menus, object properties, and tooltips to access customized documentation about your jobs or job streams.

jobUrlTemplate

The address of your job documentation. No default value available.

jobstreamUrlTemplate

The address of your job stream documentation. No default value available.

These properties must be valid URLs, containing one or more of the variables listed in the table below, as shown in the example:

```
<settings>
  <twsobjectDoc>
    <property
      name="jobstreamUrlTemplate"
      value="http://www.yourhost.com/tws/docs/jobstream/${js_name_w}" />
    <property
      name="jobUrlTemplate"
      value="http://www.yourhost.com/docs/jobs/${job_name_w}" />
  </twsobjectDoc>
</settings>
```

If you use any of the following special characters in the URL, you must write them as follows:

Table 29. Syntax for special characters

Special characters	Write them as...
quote (")	\"
apostrophe (')	'
ampersand (&)	&
less than (<)	<
greater than (>)	>
backslash (\)	\\

Multiple variables can be included in a URL and must be specified using the following syntax: `${variable}`:

Table 30. Variables used in the URL definition

Name	Object	Description
job_number_w	Job z/OS	The number of the job
job_wkst_w	Job	The name of the workstation on which the job runs
job_jsname_w	Job	The name of the job stream that contains the job
job_jswkst_w	Job	The name of the workstation on which the job stream runs
job_actualarrival_w	Job z/OS	The actual start time of the job (date format: YYYY-MM-DDThh:mm:ss)
job_actualend_w	Job z/OS	When the job actually completed (date format: YYYY-MM-DDThh:mm:ss)

Configuring Tivoli Dynamic Workload Console to use SSL

Table 30. Variables used in the URL definition (continued)

Name	Object	Description
job_starttime_w	Job	The start time of the job (date format: YYYY-MM-DDThh:mm:ss)
job_id_w	Job	The ID of the job
job_returncode_w	Job	The return code of the job
js_name_w	Job stream	The name of the job stream that contains the job
js_wkst_w	Job stream	The name of the workstation on which the job stream runs
js_id_w	Job stream	The job stream ID
js_latest_start_w	Job stream	The latest time at which a job stream can start (date format: YYYY-MM-DDThh:mm:ss)
engine_name_w	Engine	The name of the engine connection
engine_host_w	Engine	The hostname of the engine connection
engine_port_w	Engine	The port number of the engine connection
engine_plan_w	Engine	The ID of selected plan
engine_serv_w	Engine	The remote server name of the engine connection

Sections can also be repeated multiple times in the same file and applied differently to different user roles. To apply a section only to the users belonging to a role, the section must be included within the following setting:

settings role

The user for which the following configuration must be applied. Default value: all users, unless otherwise specified.

Example:

```
<settings>
  <graphViews>
    <property name="planViewNewWindow" value="true"/>
  </graphViews>
</settings>

<settings role="TWSWEBUIOperator">
  <graphViews>
    <property name="planViewNewWindow" value="false"/>
  </graphViews>
</settings>
```

TdwcGlobalSettings.xml sample

The following is a sample of the file:

```
<settings role="TWSWEBUIOperator">
  <graphViews>
    <property name="planViewMaxJobstreams" value="1000"></property>
    <property name="jobstreamViewLimit" value="1000"></property>
    <property name="impactViewLimit" value="1000"></property>
```

```
1      </graphViews>
1      </settings>
1
1      <settings role="TWSWEBUIAdministrator">
1      <application>
1      <property name="precannedTaskCreation" value="all"/>
1      </application>
1      </settings>
1
1      <settings>
1      <twsoObjectDoc>
1      <property
1      name="jobstreamUrlTemplate"
1      value="http://www.yourhost.com/tws/docs/jobstream/${js_name_w}" />
1      <property
1      name="jobUrlTemplate"
1      value="http://www.yourhost.com/docs/jobs/${job_name_w}" />
1      </twsoObjectDoc>
1      </settings>
```

Configuring Tivoli Dynamic Workload Console to view reports

This topic describes the configuration steps that you perform to be able to see the reports from the Tivoli Dynamic Workload Console.

To access the databases where reports are stored, you must have the following prerequisites:

- A user ID and password to access the database
- A working connection between the Tivoli Dynamic Workload Console and the database

Perform the following steps on the system where the Tivoli Workload Scheduler engine is running:

- “Configuring for a DB2 database”
- “Configuring for an Oracle database” on page 90
- “Configuring for an Informix Dynamic Server database” on page 91

Configuring for a DB2 database

For DB2, the IT administrator, or the Tivoli Workload Scheduler IT administrator, or both working together, do the following:

1. Create an operating system user and specify a password.
2. Launch the following script:

```
<TWA_home>/TWS/dbtools/DB2/scripts/dbgrant.bat/.sh
<ID_of_user_to_be_granted>
<database_name>
[<database_admin_user> <password>]
```

where the variables are as follows:

<TWA_home>

The Tivoli Workload Automation instance directory

<ID_of_user_to_be_granted>

The ID of the user created in step 1, who is going to be granted the access to the reports

Configuring Tivoli Dynamic Workload Console to view reports

| *<database_name>*

| The name of the database, as created when the master domain manager
| was installed

| [*<database_admin_user> <password>*]

| The user ID and password of the database administration user. If you
| are running this command as the database administration user, you can
| omit these parameters.

- | 3. Log on to the Tivoli Dynamic Workload Console.
- | 4. In the Portfolio, select **Manage Engines**. The Manage Engines panels is
| displayed.
- | 5. Select the engine you defined or create another engine. The Engine Connection
| properties panel is displayed.
- | 6. In Database Configuration for Reporting, do the following:
 - | a. Check **Enable Reporting** to enable the engine connection you selected to
| run reports.
 - | b. In **Database User ID and Password**, specify the database user and
| password that you authorized to access reports.

Configuring for an Oracle database

| For Oracle, the IT administrator, or the Tivoli Workload Scheduler IT administrator,
| or both working together, do the following:

- | 1. Create a database user authorized to access the database and specify a
| password.
- | 2. Launch the following script:

| *<TWA_home>/TWS/dbtools/Oracle/scripts/dbgrant.bat/.sh*
| *<ID_of_user_to_be_granted>*
| *<database_name>*
| *<database_admin_user> <password>*

| where the variables are as follows:

| *<TWA_home>*

| The Tivoli Workload Automation instance directory

| *<ID_of_user_to_be_granted>*

| The ID of the user created in step 1, who is going to be granted the
| access to the reports

| *<database_name>*

| The name of the database, as created when the master domain manager
| was installed

| *<database_schema_owner> <password>*

| The user ID and password of the database schema owner.

- | 3. Define a valid connection string to the database:
 - | a. Ensure that the following property is set in the *<TWA_home>/eWAS/profiles/
| twaprofile/properties/TWSConfig.properties* file to point to the Oracle
| JDBC URL:
| `com.ibm.tws.webui.oracleJdbcURL`

| For example:

| `com.ibm.tws.webui.oracleJdbcURL=
| jdbc:oracle:thin:@//9.132.235.7:1521/orcl`

- | b. Restart the embedded WebSphere Application Server.

Configuring Tivoli Dynamic Workload Console to view reports

4. Log on to the Tivoli Dynamic Workload Console.
5. In the Portfolio, select **Manage Engines**. The Manage Engines panels is displayed.
6. Select the engine you defined or create another engine. The Engine Connection properties panel is displayed.
7. In Database Configuration for Reporting, do the following:
 - a. Check **Enable Reporting** to enable the engine connection you selected to run reports.
 - b. In **Database User ID and Password**, specify the database user and password that you authorized to access reports.

Configuring for an Informix Dynamic Server database

For Informix Dynamic Server, the IT administrator, or the Tivoli Workload Scheduler IT administrator, or both working together, do the following:

1. Create an operating system user and specify a password.
2. Launch the following script:

```
<TWA_home>/TWS/dbtools/ids/scripts/dbgrant.sh
<host>
<port>
<TWS_user>
<TWS_password>
<database>
<ID_of_user_to_be_granted>
```

where the variables are as follows:

<TWA_home>

The Tivoli Workload Automation instance directory.

<host>

The physical address of the workstation where you installed the Informix Dynamic Server and the Tivoli Workload Scheduler database.

<port>

The number of the port used by the Informix Dynamic Server to accept requests from the Data Server Clients. You specified it during the installation of the Tivoli Workload Scheduler master domain manager. This number is also specified in the *informix_dynamic_server_installation_directory\etc\sqlhosts* file in the **drsotcp** communication protocol.

<TWS_user>

The user name of the administrator of the Informix Dynamic Server. You specified it during the installation of the Tivoli Workload Scheduler master domain manager.

<TWS_password>

The password of the administrator of the Informix Dynamic Server.

<database>

The name of the database, as created when you installed the master domain manager.

<ID_of_user_to_be_granted>

The ID of the user created in step 1, who is going to be granted the access to the reports.

3. Log on to the Tivoli Dynamic Workload Console.

Configuring Tivoli Dynamic Workload Console to view reports

4. In the Portfolio, select **Manage Engines**. The Manage Engines panels is displayed.
5. Select the engine you defined or create another engine. The Engine Connection properties panel is displayed.
6. In Database Configuration for Reporting, do the following:
 - a. Check **Enable Reporting** to enable the engine connection you selected to run reports.
 - b. In **Database User ID and Password**, specify the database user and password that you authorized to access reports.

Preventing a connection to specific Tivoli Workload Scheduler Version 8.3 engines

Run the following script on Tivoli Workload Scheduler side if you want to disable the ability to establish engine connections from Tivoli Dynamic Workload Console to a Tivoli Workload Scheduler Version 8.3 engine

On Windows:

```
webui -operation disable
```

Run the script as Tivoli Workload Scheduler administrator, from the directory *TWS_home*\wastools:

On UNIX

```
./webui.sh -operation disable
```

Run the script as root, from the directory *TWS_home*/wastools:

Restart the WebSphere Application Server on the Tivoli Workload Scheduler engine where you run the script.

Chapter 5. Configuring user authorization (Security file)

This chapter describes how to manage the authorizations to access scheduling objects assigned to Tivoli Workload Scheduler users.

The chapter is divided into the following sections:

- “Security management overview”
- “Getting started” on page 94
- “Updating the security file” on page 94
- “Centralized security management” on page 98
- “Configuring the security file” on page 99
- “Sample security file” on page 131
- “Configuring for LDAP” on page 135

Security management overview

The way Tivoli Workload Scheduler manages security is controlled by a configuration file named **security file**. This file controls activities such as:

- Linking workstations.
- Accessing command-line interface programs, the Job Scheduling Console, and the Tivoli Dynamic Workload Console.
- Performing operations on scheduling objects in the database or in the plan.

In the file you specify for each user what scheduling objects the user is allowed to access, and what actions the user is allowed to perform on those objects. You can determine access by object type (for example, workstations or resources) and, within an object type, by selected attributes, such as the object's name or the workstation in the object's definition. You can use wildcards to select related sets of objects. Access rights can be granted on an "included" or an "excluded" basis, or a combination of both.

Whenever you need to change access permissions you modify the configuration file and convert it into an encrypted format (for performance and security), replacing the previous file. The system uses this encrypted *security file* from that point onwards.

Each time a user runs Tivoli Workload Scheduler programs, commands, and user interfaces, the product compares the name of the user with the user definitions in the *security file* to determine if the user has permission to perform those activities on the specified scheduling objects.

By default, the security on scheduling objects is managed locally on each workstation. This means that the system administrator or the *TWS_user* who installed the product on that system can decide which Tivoli Workload Scheduler users defined on that system can access which scheduling resources in the Tivoli Workload Scheduler network and what actions they can perform.

Alternatively, you can centralize control of how objects are managed on each workstation. This can be configured by setting a global option. In this scenario, you configure all user permissions in the *security file* on the master domain

Security management overview

manager. The encrypted version of the file is distributed automatically every time you run **JnextPlan**, so that all workstations have the file locally to determine the permissions of the users on that workstation.

Getting started

This section describes how to get started with defining authorizations after installation.

A template file named *TWA_home/TWS/config/Security.conf* is provided with the product. During installation, a copy of the template file is installed as *TWA_home/TWS/Security.conf*, and a compiled, operational copy is installed as *TWA_home/TWS/Security*.

This version of the file contains a full access definition for the user who installed the product, *TWS_user*, and the system administrator (root on UNIX or Administrator on Windows), who are the only users defined and allowed to connect to the user interfaces and to perform all operations on all scheduling resources.

Within the Tivoli Workload Scheduler network, using the security file you can make a distinction between local **root** users and the **root** user on the master domain manager by allowing local **root** users to perform operations affecting only their login workstations and providing the master domain manager **root** user the authorizations to perform operations affecting any workstation across the network.

As you continue to work with the product you might want to add more users with different roles and authorization to perform specific operations on a defined set of objects.

Do not edit the original *TWA_home/TWS/config/Security.conf* template, but follow the steps described in “Updating the security file” to make your modifications on the operational copy of the file.

Updating the security file

By default, every workstation in a Tivoli Workload Scheduler network (domain managers, fault-tolerant agents, and standard agents) has its own security file. You can maintain that file on each workstation, or, if you enable centralized security management you can create a security file on the master domain manager and copy it to each domain manager and agent, ensuring that all Tivoli Workload Scheduler users are assigned the required authorization in the file (see “Centralized security management” on page 98). Whether working on an agent workstation for an individual security file, or on the master domain manager to modify a centralized file, the steps are just the same; all that changes are the number of users you are defining - just those on the local system or all in the Tivoli Workload Scheduler network.

Neither the Tivoli Workload Scheduler processes nor the WebSphere Application Server infrastructure needs to be stopped or restarted to update the security file. You just need to close any open **conman** user interfaces before running **makesec**.

To modify the security file, perform the following steps:

1. Navigate to the *TWA_home/TWS* directory from where the **dumpsec** and **makesec** commands *must* be run.

2. Run the **dumpsec** command to decrypt the current security file into an editable configuration file. See “dumpsec” on page 96.
3. Modify the contents of the editable security configuration file using the syntax described in “Configuring the security file” on page 99.
4. Close any open **conman** user interfaces using the **exit** command.
5. Stop any connectors on systems running Windows operating systems.
6. Run the **makesec** command to encrypt the security file and apply the modifications. See “makesec” on page 97.
7. If you are using local security, the file will be immediately available on the workstation where it has been updated.

If you are using centralized security (see “Centralized security management” on page 98) you must now do the following:

- a. If you are using a backup master domain manager, copy the file to it
- b. Distribute the centralized file manually to all fault-tolerant agents in the network (not standard, extended, or broker agents), and store it in the *TWA_home/TWS* directory
- c. Run **JnextPlan** to distribute the Symphony file that corresponds to the new Security file

See the next pages for a full description of **dumpsec** and **makesec**.

dumpsec

Writes in an editable format the information contained in the compiled and encrypted security file. The output file can be edited and then used as input for the **makesec** command which compiles and activates the modified security settings.

Authorization

You must have **display** access to the security file and write permission in the *TWA_home/TWS* directory from where the command *must* be run.

Syntax

dumpsec **-v** | **-u**

dumpsec *security_file* [**>** *output_file*]

Comments

If no arguments are specified, the operational security file is sent to stdout. To create an editable copy of the security file, redirect the output of the command to an output file, using the redirect symbol.

Arguments

-v Displays command version information only.

-u Displays command usage information only.

security_file

Specifies the name of the security file to dump.

[**>** *output_file*]

Specifies the name of the output file, If omitted, the security file is output to the stdout.

Examples

The following command dumps the operational security file (*TWA_home/TWS/Security*) to a file named **mysec**:

```
dumpsec > mysec
```

The following command dumps a security file named **sectemp** to **stdout**:

```
dumpsec sectemp
```

makesec

Compiles security definitions and installs the security file. Changes to the security file are recognized as soon as **makesec** has completed, or, in the case of centralized security, after **JnextPlan** has distributed it.

Note: Before running the **makesec** command, stop **conman**, and on systems running Windows operating systems any connectors.

Authorization

You must have **modify** access to the security file and read permission in the *TWA_home/TWS* directory from where the command *must* be run.

Syntax

makesec **-v** | **-u**

makesec [**-verify**] *in_file*

Comments

The **makesec** command compiles the specified file and installs it as the operational security file (*../TWA_home/TWS/Security*). If the **-verify** argument is specified, the file is checked for correct syntax, but it is not compiled and installed.

Arguments

-v	Displays command version information only.
-u	Displays command usage information only.
-verify	Checks the syntax of the user definitions in <i>in_file</i> . The file is not compiled and installed as the security file.
<i>in_file</i>	Specifies the name of a file or set of files containing user definitions. Syntax checking is performed automatically when the security file is installed.

Examples

Example 1: Modifying the security file definitions - full scenario

The following example shows how to modify the security file definitions:

1. An editable copy of the operational security file is created in a file named *tempsec* with the **dumpsec** command:

```
dumpsec > tempsec
```
2. The user definitions are modified with a text editor:

```
edit tempsec
```
3. The file is then compiled and installed with the **makesec** command:

```
makesec tempsec
```

Example 2: Compiling user definitions from multiple files

The following command compiles user definitions from the fileset *userdef** and replaces the operational security file:

```
makesec userdef*
```

Centralized security management

A Tivoli Workload Scheduler environment where centralized security management is enabled is an environment where all workstations share the same security file information contained in the security file stored on the master domain manager and the Tivoli Workload Scheduler administrator on the master domain manager is the only one who can add, modify, and delete entries in the security file valid for the entire Tivoli Workload Scheduler environment.

This is configured with the **enCentSec** global option. By default the value assigned to the **enCentSec** option is **no**.

To set central security management, the Tivoli Workload Scheduler administrator must run the following steps on the master domain manager:

1. Use the **optman** command line program, to set the value assigned to the *enCentSec* global property to **yes**. For information on how to manage the global properties using **optman**, see “Setting global options” on page 5.
2. Save the information in the security file into an editable configuration file using the **dumpsec** command.
3. Set the required authorizations for all Tivoli Workload Scheduler users, as described in “Configuring the security file” on page 99
4. Close any open **conman** user interfaces using the **exit** command.
5. Stop any connectors on systems running Windows operating systems.
6. Compile the security file using the **makesec** command.
7. If you are using a backup master domain manager, copy the compiled security file to it as soon as possible.
8. Distribute the compiled security file to all the workstations in the environment and store it in their *TWA_home/TWS* directories.
9. Run **JnextPlan** to update the security information distributed with the Symphony file.

The value of the checksum of the newly compiled security file is encrypted and loaded into the Symphony file and distributed to all the workstations in the Tivoli Workload Scheduler network.

On each workstation, when a link is established or when a user connects to a user interface or attempts to issue commands on the plan, either with **conman**, the Tivoli Dynamic Workload Console, or the Job Scheduling Console, Tivoli Workload Scheduler compares the value of the checksum in the security file delivered with the Symphony file with the value of the checksum of the security file stored on the workstation. If the values are equal, the operation is allowed. If the values are different, the operation fails and a security violation message is issued.

Centralized security usage notes

In a network with centralized security management, two workstations are unable to establish a connection if one of them has **enCentSec** turned off in its Symphony file or if their security file information does not match.

The only exception to the security file matching criteria introduced by the centralized security management mechanism is that a workstation must always accept incoming connections from its domain manager, regardless of the result of the security file matching process.

Centralized security does not apply to Tivoli Workload Scheduler operations for which the Symphony file is not required. Commands that do not require the Symphony file to run use the local security file. For example, the **parms** command, used to modify or display the local parameters database, continues to work according to the local security file, even if centralized security is active and the local security file differs from the centralized security rules.

If a workstation's security file is deleted and recreated, the checksum of the new security file will not match the value in the Symphony file. In addition, a run-number mechanism associated with the creation process of the Symphony file ensures prevention from tampering with the file.

Configuring the security file

In the security file you can specify which scheduling objects a user can manage and how. You define these settings by writing user definitions. A user definition is an association between a name and a set of users, the objects they can access, and the actions they can perform on the specified objects.

When defining user authorization consider that:

- When commands are issued from the **composer** command line program, the user authorizations are checked in the security file of the master domain manager since the methods used to manage the entries in the database are invoked on the master domain manager. Therefore the user must be defined:
 - As system user on the system where the master domain manager is installed.
 - In the security file on the master domain manager with the authorizations needed to run the allowed commands on the specific objects.
- When commands are issued from the **conman** command line program, the user must be authorized to run the specific commands in the security file both on the connecting workstation and on the master domain manager where the command actually runs.

The configuration is described in these sections:

- “Security file syntax”
- “Specifying user attributes” on page 102
- “Specifying object types” on page 107
- “Specifying object attributes” on page 108
- “Specifying access” on page 112
- “The *TWS_user* - special security file considerations” on page 131

Security file syntax

The syntax of the security file is as follows:

Security file syntax

Security file

Syntax: [# *comment*]

user *definition_name* *user_attributes*

begin [* *comment*]

object_type [*object_attributes*]. **access**[=*keyword*[,*keyword*]...]

[*object_type* [*object_attributes*]. **access**[=*keyword*[,*keyword*]...]]...

end

Arguments:

[# | *] *comment*

All text following a pound sign or an asterisk and at least one space is treated as a comment. Comments are not copied into the operational security file installed by the **makesec** command.

user *definition_name*

Specifies the name of the user definition. The name can contain up to 36 alphanumeric characters and must start with an alphabetic character.

user_attributes

Contains one or more attributes that identify the user or users to whom the definition applies. For details of how to define user attributes, see "Specifying user attributes" on page 102.

begin Begin the part containing object statements and accesses within the user definition.

object_type

Identifies the type of object (for example: workstation, resource, or prompt) to which access is to be given for the specified user or users. All object types that the specified user or users needs to access must be explicitly defined. If they are not, no access will be given. For details of how to define object types, see "Specifying object types" on page 107.

object_attributes

Contains one or more attributes that identify the specific objects of the defined object type to which the same access is to be given. If no object attributes are defined, access is given to all objects of the defined object type. For details of how to define object attributes, see "Specifying object attributes" on page 108.

access[=*keyword*[,*keyword*]...]

Describes the access to the specified objects given to the selected users. If none is specified (by specifying just the keyword "access") no access is given to the associated objects. If **access=@** then all access rights are assigned to the specified users. For details of how to define access, see "Specifying access" on page 112.

end Terminates the part containing object statements and accesses within the user definition.

Wildcards: The following wildcard characters are permitted in user definition syntax:

- ? Replaces one alphanumeric character.
- @ Replaces zero or more alphanumeric characters.

For information about variables supplied with the product that can be used in object attributes, refer to “Using variables in object attribute definitions” on page 111. Refer to “Sample security file” on page 131 for an example on how to use variables.

Specifying user attributes

The user attributes define *who* has the access that is going to be subsequently defined. They can identify one user, a selection of users, a group of users, a selection of groups of users, or all users. You can also exclude one or more specific users or groups from a selection. As well as being identified by logon ID and group name, users can also be described by the workstation from which they log on. And finally, you can mix selection criteria, for example selecting all users in a named group that can access from a set of workstations identified by a wildcard, but excluding a specific set of users identified by their logon IDs.

The general syntax

You make this selection by specifying one or more user attributes. Each user attribute is specified as follows:

user_attribute_type=value

user_attribute_type

Can be *cpu* (workstation), *group*, or *logon*

value Identifies an individual *cpu* (workstation), *group*, or *logon*, or, by using wildcards, can identify a set of any of these.

Including or excluding

Each attribute can be *included* or *excluded* from the selection.

Thus, for each *attribute type*, your options are one of the following:

Include all

This is the default. Thus, for example, if you want to include all *groups*, you need add no user attribute with respect to any group.

Include a selection

This can be defined in one of these ways:

- By specifically including users you want to select (individuals or one or more sets)
- By specifically excluding (from the *include all* default) all users you do *not* want to select
- By specifically including a set of users and then excluding some of those contained in the set

Which of these options you choose is determined by which is easier to specify.

Using the include or exclude symbols:

Include

Precede the user attribute expression by a plus (+) sign. All users identified by the expression will be selected, unless they are also selected by an *exclude* expression. If the first attribute in your definition is an *include*, it does not need to have a (+) sign defined, because the sign is implicit.

The default (if you specify no user attributes) is to include all users, on all workstations, in all groups, so if you want to define, for example, all users except one named user, you would just supply the *exclude* definition for the one user.

Exclude

Precede the user attribute expression by a tilde (~) sign. All users identified by the expression will *never* be selected, regardless of if they are identified by any *include* expressions.

Selection expressions

The following are the different types of selection expression you can use:

Basis selection expressions

Include only one attribute

user_attribute_type=value

For example, to include one named user logon ID, and exclude all other users:

`logon=jsmith1`

Exclude one attribute

~user_attribute_type=value

For example, to exclude one set of logon IDs identified by a wildcard (those that start with the letter "j"), but include all others:

`~logon=j@`

Include only several attributes of the same type

user_attribute_type=value[,value]...

For example, to include three specific users and exclude all others:

`logon=jsmith1,jbrown1,jjones1`

Exclude several attributes of the same type

~user_attribute_type=value[,value]...

For example, to exclude three specific users and include all others:

`~logon=jsmith1,jbrown1,jjones1`

Complex selection expressions

Include users identified by different selection expressions

basic_selection_expression[+basic_selection_expression]...

The selection expressions can be of the same or a different attribute type:

Same attribute type

An example of the same attribute type is the following, which selects all the groups beginning with the letter "j", as well as those with the letter "z":

`group=j@+group=z@`

If the first selection identifies 200 users, and the second 300, the total users selected is 500.

Different attribute type

An example of selection expressions of a different attribute type is the following, which selects all the groups beginning with the letter "j", as well as all users with IDs beginning with a "6":

`group=j@+logon=6@`

Specifying user attributes

If the first selection identifies 200 users, and the second 20, of whom 5 are also in the first group, the total users selected is 215.

Exclude users identified in one selection expressions from those identified in another

basic_selection_expression[~basic_selection_expression]...

Same attribute type

The selection expressions can be of the same attribute type, provided that the second is a subset of the first. An example of the same attribute type is the following, which selects all the workstations beginning with the letter "j", but excludes those with a "z" as a second letter:

```
group=j@~group=jz@
```

If the first selection identifies 200 users, and the second 20, the total users selected is 180. Note that if the second expression had not been a subset of the first, the second expression would have been ignored.

Different attribute type

Selection expressions of a different attribute type do not have to have a subset relationship, an example being the following, which selects the group "mygroup", but excludes from the selection all users in the group with IDs beginning with a "6":

```
group=mygroup~logon=6@
```

If the first selection identifies 200 users, and the second 20, of whom 5 are also in the first group, the total users selected is 195.

Multiple includes and excludes

You can link together as many include and exclude expressions as you need to identify the precise subset of users who require the same access. The overall syntax is thus:

```
[~]user_attribute_type=value[,value]...  
[+|~]user_attribute_type=value[,value]...
```

Note: Making your *first* user attribute an *exclude* means that *all* user attributes of that type are selected *except* the indicated *value*. Thus, *~user_attribute_type=value* equates to the following:

```
user_attribute_type=@~same_user_attribute_type=value
```

However, if you use this syntax, you cannot, and do not need to, specifically add "+user_attribute_type=@", after the negated item, so you do not define:

```
~user_attribute_type=value+same_user_attribute_type=@
```

Order of user definition

You must order user definitions from most specific to least specific. Tivoli Workload Scheduler scans the security file from top-down, with each user ID being tested against each definition in turn. If the user ID is satisfied by the definition, it is selected, and the matching stops.

For example:

Incorrect:

```
#First User Definition in the Security File
USER TwsUser
CPU=@+LOGON=TWS_user
Begin
job name=@ access=modify
End

#Second User Definition in the Security File
USER Twsdomain:TwsUser
CPU=@+LOGON=TWSDomain\\TWS_user
Begin
job name=@ access=display
End
```

Note: The domain name is actually "TWSDomain\TWS_user", but it has been "escaped", as described in "Escaping special characters in user attribute values" on page 107.

The definitions are intended to determine the following:

1. Users on all workstations with a logon of "TWS_user" will be given "modify" access to all jobs
2. Users on all workstations with a logon of "TWSDomain\TWS_user" will be given "display" access to all jobs

However, all users with a logon of "TWS_user" will satisfy the first rule, regardless of their domain, and will be given "modify" access to all jobs. This is because defining a user without its domain is a shorthand way of defining that user ID in *any* domain; it is the equivalent of "@\TWSUser". So the second rule will never be satisfied, for any user, because the matching for the "TWS_user" stops after a successful match is made.

Correct

```
#First User Definition in the Security File
USER Twsdomain:TwsUser
CPU=@+LOGON="TWSDomain\\TWS_user"
Begin
job name=@ access=display
End

#Second User Definition in the Security File
USER TwsUser
CPU=@+LOGON=TWS_user
Begin
job name=@ access=modify
End
```

By putting the more specific definition first, both object access definitions are applied correctly.

See "Sample security file" on page 131 for a practical example.

User attribute types - detailed description

The *user_attribute_types* and their associated *values* can be any of the following:

cpu={workstation | @}

where:

Specifying user attributes

workstation

Specifies the workstation on which the user is logged in. Wildcard characters are permitted. The following Tivoli Workload Scheduler variables can be used:

\$master	Means that the user is logged in on the Tivoli Workload Scheduler master domain manager.
\$remotes	Means that the user is logged in on any Tivoli Workload Scheduler standard agent.
\$slaves	Means that the user is logged in on any Tivoli Workload Scheduler fault-tolerant agent.
\$thiscpu	Means that the user is logged in on the Tivoli Workload Scheduler workstation on which the security check is running.
@	Specifies that the user is accessing Tivoli Workload Scheduler with the Tivoli Dynamic Workload Console or the Job Scheduling Console, or is logged in on any Tivoli Workload Scheduler workstation.

group=*groupname*

Specifies the name of the group of which the user is a member. Available for both UNIX and Windows users. Wildcard characters are permitted. Do not use this argument for Tivoli Dynamic Workload Console or Job Scheduling Console users.

logon={*username***|@}**

where:

username

Specifies the user ID with which the user is logged in on a Tivoli Workload Scheduler workstation. Wildcard characters are permitted. The **cpu=** attribute must be set to a specific workstation name (no wildcards) or **@**.

Notes:

1. If the WebSphere Application Server security configuration option **useDomainQualifiedUserNames** is set to *true*, each user ID defined in the security file must have the format *domain/username* to use the product from one of the following:
 - **composer**
 - Tivoli Dynamic Workload Console
 - Job Scheduling Console
 - **logman**
 - **optman**
 - **planman**For more information on WebSphere Application Server security configuration, see “Changing the security settings” on page 258.
2. If the user is defined on a Windows 2000 Service Pack 4 (SP4), or Windows XP Service Pack 2 (SP2), or Windows 2003 system, or when upgrading the Windows operating system from an

older version to one of those mentioned above, make sure you add the **Impersonate a client after authentication** right to the user settings.

- @ Specifies any user logged in with any name or being a member of any Tivoli administrators group.

Escaping special characters in user attribute values

If you need to use a special character in the value field of a user attribute, you must enclose the value in double quotes and escape the special character with a back slash. These are the special characters supported in this way:

```
'
"
\
space
```

This, if you want to include a group called wingrp\sales, you should enter:

```
+group="wingrp\\sales"
```

Similarly, Wingroup Sales would be:

```
+group="Wingroup\ Sales"
```

Specifying object types

Specify one or more object types that the user or users in the associated user definition is authorized to access. If you specify the object type but no attributes, the authorized actions defined for the user with the **access** keyword apply to all the objects of that type defined in the Tivoli Workload Scheduler domain. If an object type from the following list is omitted for a user or users, no objects of that type can be accessed.

The object types are the following:

action	Actions defined in scheduling event rules
calendars	User calendars
cpu	Workstations, domains and workstation classes
event	Event conditions in scheduling event rules
eventrule	Scheduling event rule definitions
file	Tivoli Workload Scheduler database file
job	Scheduled jobs and job definitions
parameter	Local parameters. See note below.
prompt	Global prompts
report	The reports on the Tivoli Dynamic Workload Console that have the following <i>names</i> :
	RUNHIST Job Run History
	RUNSTATS Job Run Statistics

Specifying object types

WWS Workstation Workload Summary

WWR Workstation Workload Runtimes

SQL Custom SQL

ACTPROD

Actual production details (for current and archived plans)

PLAPROD

Planned production details (for trial and forecast plans)

Permission to use these reports is granted by default to the *TWS_user* on fresh installations.

resource	Scheduling resources
schedule	Job streams
userobj	User objects
variable	Variable tables. This includes authorization to the variable definitions in the tables. See the note below.

Note: Starting from version 8.5, the **parameter** object type is reserved for parameters created and managed in a local parameter database with the `parms` utility command, while authorization to act on global variables is managed using the **variable** object type. For this reason, when the security file is migrated from previous versions to 8.5, a **variable** security definition for the default variable table is added to match each parameter definition found, as part of the upgrade process documented in the *Tivoli Workload Scheduler: Planning and Installation Guide*.

Specifying object attributes

Specify one or more attributes that identify a set of objects that the user of the user definition is authorized to access. If you specify the object type but no object sets, the authorized actions defined for the user with the **access** keyword apply to all the objects of that type defined in the Tivoli Workload Scheduler domain.

The general syntax

Each object attribute is specified as follows:

object_attribute=value

object_attribute

Object attributes differ according to the object. All objects can be selected by *name*, but some, *jobs*, for example, can be selected by the *workstation* on which they run. See “Object attribute” for full details of which attributes are available for each object type.

value Identifies an individual object, or, by using wildcards, a set of objects. See “Specifying object attributes” for full details of which attributes are available for each object type.

Object attribute

Table 31 on page 109 lists object attributes that are used to identify a specific set of object within all objects of the same type. For example, access can be restricted to a set of resource objects having the same name or being defined on the same workstation, or both.

Table 31. Object attribute types for each object type

Attribute		name	cpu	custom	jcl	jcltype	logon	provider	type	host	port
1	Object										
	action							✓	✓	✓	✓
	calendar	✓									
	cpu (workstation)		✓								
	event			✓				✓	✓		
	eventrule	✓									
	file	✓									
	job	✓	✓		✓	✓	✓				
	parameter	✓	✓								
	prompt	✓									
	report	✓									
	resource	✓	✓								
	schedule (job stream)	✓	✓								
	userobj		✓				✓				
variable	✓										

Including or excluding

Each attribute can be *included* or *excluded* from the selection using the plus (+) and tilde (~) symbols, in the same way as for the user attributes.

Selection expressions

The detailed syntax and use of the selection expressions for objects is the same as that used to select users:

```
[~]object_attribute=value[,value]...[+|~]object_attribute=value[,value]...
```

Order of object definition

You must order object definitions from most specific to least specific, in the same way as for user attributes. For example,

Incorrect

```
job name=@ access=display
job name=ar@ access=@
```

In this case, a job with the name beginning with "ar" would satisfy the first definition, and so would be given the display access, not all access.

Correct

```
job name=ar@ access=@
job name=@ access=display
```

Specifying object attribute values

The following describes the values allowed for each object attribute type:

name=name[,name]...

Specifies one or more names for the object type. Wildcard characters are permitted. Multiple names must be separated by commas.

- The following values apply to the **file** object type:

Specifying object attributes

globalopts	Allows the user to set global options with the <code>optman</code> command. Gives the following access types: <ul style="list-style-type: none">– Display access for <code>optman ls</code> and <code>optman show</code>– Modify access for <code>optman chg</code>
prodsked	Allows the user to create, extend, or reset the production plan.
security	Allows the user to manage the security file.
Symphony	Allows the user to run stageman and JnextPlan .
trialsked	Allows the user to create trial and forecast plans or to extend trial plans.

Note: Users who have restricted access to files should be given at least the following privilege to be able to display other objects (ie. calendars and cpus):

```
file name=globalopts access=display
```

- For the **event** object type use one or more of the event type names listed in the *TWSObjectsMonitor events* table or the *FileMonitor events* table in the *Tivoli Workload Scheduler: User's Guide and Reference*.
- For the **action** object type use one or more of the action type names listed in the table *Action types by action provider* in the *Tivoli Workload Scheduler: User's Guide and Reference*.
- For the **variable** object type, you can use the \$DEFAULT value for the **name** attribute to indicate the default variable table. This selects the table defined with the `isdefault` attribute.

cpu=workstation[,workstation]...

Specifies one or more workstation, domain, or workstation class names. Wildcard characters are permitted. Multiple names must be separated by commas. If this attribute is not specified, all defined workstations and domains can be accessed. Workstation variables can be used - see "Using variables in object attribute definitions" on page 111.

custom=value[,value]...

Use this attribute to assign access rights to events defined in event plug-ins. The precise syntax of the value will depend on the plug-in. For example:

- Specify different rights for different users based on SAP R/3 event names when defining event rules for SAP R/3 events.
- Define your own security attribute for your custom-made event providers.
- Specify the type of event that is to be monitored. Every event can be referred to an event provider.

jcl="path" | "command"

Specifies the command or the path name of a job object's executable file. The command or path must be enclosed in double quotes ("). Wildcard characters are permitted. If omitted, all defined job files and commands qualify.

jcltype=[scriptname | docommand]

Specifies that the user is allowed to act on the definitions of jobs that run only scripts (if set to **scriptname**) or commands (if set to **docommand**). Use this optional attribute to restrict user authorization to actions on the definitions of jobs of one type or the other only. Actions are granted for both scripts and commands when **jcltype** is missing.

1
1
1
1
1
1

1 A user who is not granted authorization to work on job definitions that
 1 run either a command or a script is returned a security error message
 1 when attempting to run an action on them.

logon=*username*[,...]

Specifies the user IDs. Wildcard characters are permitted. Multiple names must be separated by commas. If omitted, all user IDs qualify. For the **job** type object, the following variables are permitted: **\$jclowner**, **\$owner**, and **\$user** (see “Using variables in object attribute definitions”).

provider=*provider_name*[,...]

For **action** object types, specifies the name of the action provider.

For **event** object types, specifies the name of the event provider.

Wildcard characters are permitted. Multiple names must be separated by commas. If provider is not specified, no defined objects can be accessed.

type=*type*[,...]

For **action** object types, is the `actionType`.

For **event** object types, is the `eventType`.

Wildcard characters are permitted. Multiple names must be separated by commas. If **type** is not specified, all defined objects are accessed for the specified providers.

host=*host_name*

For **action** object types, specifies the TEC or SNMP host name (used for some types of actions, such as sending TEC events, or sending SNMP). If it does not apply, this field must be empty.

port=*port_number*

For **action** object types, specifies the TEC or SNMP port number (used for some types of actions, such as sending TEC events, or sending SNMP). If it does not apply, this field must be empty.

Using variables in object attribute definitions

The following variables supplied with the product can be used in object attributes:

User identifiers

\$jclgroup	The group name of a job's executable file.
\$jclowner	The owner of a job's executable file.
\$owner	The creator of a job stream and its jobs.
\$user	The user running the Tivoli Workload Scheduler command or program.

Note: The variables **\$jclgroup** and **\$jclowner** can only be verified if the user is running a Tivoli Workload Scheduler program on the workstation where the job's executable file is present. If the program is being run on a different workstation, the user is denied access.

Workstation identifiers

Specifying object attributes

\$master	The Tivoli Workload Scheduler master domain manager.
\$remotes	All standard agent workstations.
\$slaves	All fault-tolerant agent workstations.
\$thiscpu	The workstation on which the user is running the Tivoli Workload Scheduler command or program.

Variable table identifiers

\$default	The name of the current default variable table.
------------------	---

Specifying access

Specify the type of access the selected users are allowed to have to the specified objects as follows:

access[=*keyword*[,*keyword*]...]

- To specify that no actions are permitted, use **access=**
- To specify that all actions are permitted, use **access=@**
- To specify any other access, consult the access tables, by object type, below.

How the access tables are organized

The access tables for object types are as follows:

“Object types - calendar, cpu, eventrule, job, prompt, resource, schedule, userobj - using in composer” on page 114

Most of the **composer** and GUI database maintenance actions are common to most objects, so they are listed in a table of common object access keywords.

“Object type - action” on page 116

This gives the access rights for action objects, which are not included in the common table.

“Object type - calendar” on page 117

This gives the access rights for calendars, which are different or additional to those in the common table.

“Object type - cpu” on page 118

This gives the access rights for workstations (cpus), which are different or additional to those in the common table.

“Object type - event” on page 119

This gives the access rights for events, which are different or additional to those in the common table.

“Object type - file” on page 120

This gives the access rights for files, which are different or additional to those in the common table.

“Object type - job” on page 121

This gives the access rights for jobs, which are different or additional to those in the common table.

“Object type - parameter” on page 124

This gives the access rights for local parameters, which are not included in the common table.

“Object type - prompt” on page 125

This gives the access rights for prompts, which are different or additional to those in the common table.

“Object type - report” on page 126

This gives the access rights for reports, which are different or additional to those in the common table.

“Object type - resource” on page 127

This gives the access rights for resources, which are different or additional to those in the common table.

“Object type - schedule” on page 128

This gives the access rights for job streams (schedules), which are different or additional to those in the common table.

“Object type - userobj” on page 129

This gives the access rights for userobj, which are different or additional to those in the common table.

“Object type - vartable” on page 130

This gives the access rights for variable tables, which are not included in the common table.

Specifying access

Object types - calendar, cpu, eventrule, job, prompt, resource, schedule, userobj - using in composer

The following table gives the access keywords required to use composer to work with objects of the following types:

- calendar
- cpu
- eventrule
- job
- prompt
- resource
- schedule
- userobj

Note: Starting from version 8.5, the parameter keyword is reserved for parameters created and managed in a local parameter database with the parms utility command. See the *Tivoli Workload Scheduler: User's Guide and Reference* for details on parms.

Table 32. Access keywords for composer actions

Activity			Access keywords required
Composer	add	Add new object definitions in the database from a file of object definitions. Unlock access is needed to use the ;unlock attribute.	add, modify, unlock
	create	Create a text file of object definitions in the database. Modify access is need to use the ;lock attribute.	display, modify
	delete	Delete object definitions from the database.	delete
	display	Display object definitions in the database.	display
	extract	Extract a text file of object definitions from the database.	display
	list	List object definitions in the database.	display
	lock	Lock object definitions in the database.	modify
	modify	Modify object definitions in the database. Definitions are extracted into a file. After you have edited the file the definitions are used to replace the existing ones.	add, modify
	new	Create object definitions in the database from a template.	add, modify
	print	Print object definitions in the database.	display
	rename	Rename object definitions in the database. The user needs add access to the new object and delete and display access to the old object.	add, delete, display
	replace	Replace object definitions in the database. Unlock access is needed to use the ;unlock attribute.	add, modify, unlock
	unlock	Unlock object definitions in the database.	unlock

Table 32. Access keywords for composer actions (continued)

Activity			Access keywords required
Tivoli Dynamic Workload Console	Create object in database	Add new object definitions in the database.	add
Job Scheduling Console ¹	Delete object in database	Delete object definitions from the database. Unlock access is needed to use the ;unlock option.	delete
	Display object in database	Display object definitions in the database.	display
	List object in database	List object definitions in the database.	display
	Modify object in database	Modify object definitions in the database. Unlock access is needed to use the ;unlock option.	modify
	Unlock object in database	Unlock object definitions in the database locked by another user.	unlock
Using the workload service assurance feature	All activities	For any user to perform any workload service assurance activities, the <i>TWS_user</i> must have the following access keywords for all <i>cpu</i> , <i>job</i> , and <i>schedule</i> objects:	display, modify, list

Notes:

1. The Job Scheduling Console does not manage event rules.

Example: To allow a user to use the composer list, display, and modify actions on event rules, specify:

```
eventrule          access=add,display,modify
```

Specifying access

Object type - action

The following table gives the access keywords required for actions:

Table 33. Actions - access keywords

Activity		Access keywords required
Tivoli Dynamic Workload Console	Display action instances	display
Job Scheduling Console	List action instances.	list
Tivoli Dynamic Workload Console Job Scheduling Console conman	Use these specific action types in event rule definitions. <ul style="list-style-type: none">• For actions with provider TWSAction and types sbj, sbd, or sbs, you must set this keyword in combination with the submit access keyword for the specific jobs and job streams specified in the action.• For actions with provider TWSAction and type reply, you must set this keyword in combination with the reply access keyword set for the specific prompts specified in the action. The <i>TWS_user</i> of the workstation running the event processing server must have these submit and reply authorizations, otherwise the event processing server will not be able to run this type of actions.	use

Example: To allow a user to use the Tivoli Dynamic Workload Console to list action instances, specify:

```
action          access=list
```

Object type - calendar

The following table gives the additional access keywords required to work with calendars, other than those described in Table 32 on page 114:

Table 34. Calendar - additional access keywords

Activity		Access keywords required
Composer Tivoli Dynamic Workload Console Job Scheduling Console	Use calendars in job streams	use

Example 1: To allow a user to only use calendars when working with job streams in any of the interfaces, specify:

```
calendar          access=use
```

Example 2: To allow a user to display, list, and print calendars, and use them when working with job streams in any of the interfaces, specify:

```
calendar          access=display,use
```

Specifying access

Object type - cpu

The following table gives the additional access keywords required to work with cpus (includes workstations, domains, and workstation classes), other than those described in Table 32 on page 114:

Table 35. Cpus - additional access keywords

Activity			Access keywords required
Conman Tivoli Dynamic Workload Console Job Scheduling Console	console	View and send messages to the Tivoli Workload Scheduler conman console.	console
	deployconf	Force update the monitoring configuration file for the event monitoring engine.	start
	fence	Alter workstation job fences in the production plan.	fence
	limit cpu	Alter workstation job limits in the production plan.	limit
	link	Open workstation links.	link
	showcpus	Display workstations, domains and links in the plan.	list
	shutdown	Shut down Tivoli Workload Scheduler processing.	shutdown
	start	Start Tivoli Workload Scheduler processing.	start
	startappserver	Start the application server.	start
	starteventprocessor	Start the event processor server.	start
	startmon	Start the event monitoring engine.	start
	stop	Stop Tivoli Workload Scheduler processing.	stop
	stop;progressive	Stop Tivoli Workload Scheduler processing progressively.	stop
	stopappserver	Stop the application server.	stop
	stopeventprocessor	Stop the event processor server.	stop
	stopmon	Stop the event monitoring engine.	stop
	switcheventprocessor	Switch the event processor server from the master domain manager to the backup master domain manager or vice versa.	start, stop
	switchmgr	Switch the domain manager functionality to a workstation.	start, stop
	unlink	Close workstation links.	unlink
	Startup	Start Tivoli Workload Scheduler processing.	start
Using the workload service assurance feature	All activities	For any user to perform any workload service assurance activities, the <i>TWS_user</i> must have the following access keywords:	display, modify, list

Example: To allow a user to display, list, and print workstation, workstation class, and domain definitions, and link and unlink workstations, specify:

```
cpu          access=display,link,unlink
```

Object type - event

The following table gives the access keywords required to work with events:

Table 36. Events - access keywords

Activity		Access keywords required
Composer Tivoli Dynamic Workload Console	Use an event in an event rule definition.	use

Example: To allow a user to use an event in an event rule definition, specify:
event access=use

Specifying access

Object type - file

The following table gives the access keywords required to work with files (valid only for the command line).

You must specify the file names to which the type of access applies.

Table 37. Files - access keywords

Activity		Access keywords required	
Delete objects from the database.		delete	
dumpsec	Create a text file of the settings contained in the compiled security file.	display	
JnextPlan	Generate the production plan.	build	
makesec	Compile the security file from a text file of the settings.	modify	
optman	ls	List all global options.	display
	show	Show the details of a global option.	display
	change	Change the details of a global option.	modify
planman	deploy	Manually deploy event rules.	build
prodsked	Work with the production plan.	build	
stageman	Carry forward uncompleted job streams, archive the old production plan, and install the new production plan.	build	

Example 1: To allow a user to manage the Security file, specify:

```
file          access=display,modify
```

Example 2: To allow a user to run **JnextPlan**, specify:

```
file          access=build
```

Note: The user will also be able to run **planman deploy**, **prodsked**, and **stageman**.

Object type - job

The following table gives the additional access keywords required to work with jobs, other than those described in Table 32 on page 114:

Table 38. Jobs - additional access keywords

Activity			Access keywords required
Composer Tivoli Dynamic Workload Console Job Scheduling Console	Use jobs in job streams. Also, if a job is used as a recovery job in a job definition, the user must have "use" access to the definition of the job identified as the recovery job.		use
Conman Tivoli Dynamic Workload Console Job Scheduling Console	adddep	Add dependencies to jobs in the production plan. Not valid for workstations in end-to-end environment.	adddep
	altpri	Alter the priority of jobs in the production plan. Not valid for workstations in end-to-end environment.	altpri
	cancel job	Cancel jobs in the production plan. Not valid for workstations in end-to-end environment.	cancel
	confirm	Confirm completion of jobs in the production plan. Not valid for workstations in end-to-end environment.	confirm
	deldep job	Delete dependencies from jobs in the production plan. Not valid for workstations in end-to-end environment.	deldep
	display	Display jobs in the plan.	display
	kill	Kill running jobs.	kill
	release job	Release jobs from dependencies in the production plan. Not valid for workstations in end-to-end environment.	release
	reply	Reply to job prompts in the production plan.	reply
	rerun	Rerun jobs in the production plan. Not valid for workstations in end-to-end environment.	rerun
showjobs	Display information about jobs in the production plan.	list	

Specifying access

Table 38. Jobs - additional access keywords (continued)

Activity			Access keywords required
<p>Conman</p> <p>Tivoli Dynamic Workload Console</p> <p>Job Scheduling Console</p> <p>1</p>	submit docommand	<p>Submit commands as jobs or recovery jobs into the production plan.</p> <p>If the submit also identifies a second job with the "ALIAS" or "RECOVERYJOB" arguments, the user must have "submit" access to that other job, as well</p> <p>Not valid for workstations in end-to-end environment.</p>	submit
	submit file	<p>Submit files as jobs or recovery jobs into the production plan.</p> <p>If the submit also identifies a second job with the "ALIAS" or "RECOVERYJOB" arguments, the user must have "submit" access to that other job, as well</p> <p>Not valid for workstations in end-to-end environment.</p>	submit
	submit job	<p>Submit jobs or recovery jobs into the production plan.</p> <p>If the submit also identifies a second job with the "ALIAS" or "RECOVERYJOB" arguments, the user must have "submit" access to that other job, as well</p> <p>Not valid for workstations in end-to-end environment.</p> <p>Restricts the submission action to jobs defined in the database. With this authorization level a user cannot submit ad-hoc jobs. Use this keyword to allow a user to submit only jobs defined in the database. Use the submit keyword to allow a user to submit both defined and ad-hoc jobs.</p> <p>Users granted only submitdb rights:</p> <ul style="list-style-type: none"> • Cannot run submit docommand and submit file successfully • Are displayed tasks related to ad-hoc job submission on the graphical user interfaces, but if they run them, are returned error messages for lacking the submit access right. 	submitdb
	submit sched	Submit job streams into the production plan. Not valid for workstations in end-to-end environment.	submit
	Tivoli Dynamic Workload Console	<p>For critical jobs on which you run any of the following actions:</p> <ul style="list-style-type: none"> • Display hot list • Display critical path • Display uncompleted predecessors • Display completed predecessors 	The predecessors are listed regardless of the fact that this authorization might not be extended to them. However, if you want to run any further action on any of the listed predecessors, this will require that you have the proper authorization.
Using the workload service assurance feature	All activities	For any user to perform any workload service assurance activities, the <i>TWS_user</i> must have the following access keywords:	display, modify, list

Specifying access

Object type - parameter

The following table gives the access keywords required to work with parameters:

Note: Starting from version 8.5, the parameter keyword is reserved for parameters created and managed in a local parameter database with the parms utility command. See the *Tivoli Workload Scheduler: User's Guide and Reference* for details on parms.

Table 39. Parameters - additional access keywords

Activity		Access keywords required
parms	Manage local parameter definitions.	display

Example: To allow a user to perform all activities on parameters, specify:

```
parameter      access=@
```

Object type - prompt

The following table gives the additional access keywords required to work with prompts, other than those described in Table 32 on page 114:

Table 40. Prompts - additional access keywords

Activity			Access keywords required
Composer Tivoli Dynamic Workload Console Job Scheduling Console	Use prompts when defining or submitting jobs and job streams		use
Conman Tivoli Dynamic Workload Console Job Scheduling Console	adddep	Use prompts when adding dependencies to jobs in the production plan. Not valid for workstations in end-to-end environment.	use
	recall	Display prompts waiting for a response.	display
	reply	Reply to a job or job stream prompt.	reply
	showprompts	Display information about prompts.	list
	submit docommand	Use prompts when submitting commands as jobs into the production plan. Not valid for workstations in end-to-end environment.	use
	submit file	Use prompts when submitting files as jobs into the production plan. Not valid for workstations in end-to-end environment.	use
	submit job	Use prompts when submitting jobs into the production plan. Not valid for workstations in end-to-end environment.	use
	submit sched	Use prompts when submitting job streams into the production plan. Not valid for workstations in end-to-end environment.	use

Example: To allow a user to perform all activities on prompts except reply to them, specify:

```
prompt          access=use,display,list
```

Specifying access

Object type - report

The following table gives the access keywords required to work with reports.

Table 41. Files- access keywords

Activity		Access keywords required
Tivoli Dynamic Workload Console	Display reports on Tivoli Dynamic Workload Console.	display

Example: To allow a user to display reports on the Tivoli Dynamic Workload Console, specify:

```
report          access=display
```

Object type - resource

The following table gives the additional access keywords required to work with resources, other than those described in Table 32 on page 114:

Table 42. Resources - additional access keywords

Activity		Access keywords required	
Composer Tivoli Dynamic Workload Console Job Scheduling Console	Use resources when defining or submitting jobs and job streams	use	
Conman Tivoli Dynamic Workload Console Job Scheduling Console	adddep	Use resources when adding dependencies to jobs in the production plan. Not valid for workstations in end-to-end environment.	use
	resource	Change the number of units of a resource on a workstation.	resource
	showresources	Display information about resources.	list
	submit docommand	Use resources when submitting commands as jobs into the production plan. Not valid for workstations in end-to-end environment.	use
	submit file	Use resources when submitting files as jobs into the production plan. Not valid for workstations in end-to-end environment.	use
	submit job	Use resources when submitting jobs into the production plan. Not valid for workstations in end-to-end environment.	use
	submit sched	Use resources when submitting job streams into the production plan. Not valid for workstations in end-to-end environment.	use

Example: To allow a user to display information about resources and change the units of a resource on a workstation, but not to use them in any other scheduling objects or actions, specify:

resource access=list,resource

Specifying access

Object type - schedule

The following table gives the additional access keywords required to work with job streams, other than those described in Table 32 on page 114:

Table 43. Job streams - additional access keywords

Activity			Access keywords required
Conman Tivoli Dynamic Workload Console Job Scheduling Console	adddep	Add dependencies to job streams in the production plan. Not valid for workstations in end-to-end environment.	adddep
	altpri	Alter the priority of job streams in the production plan. Not valid for workstations in end-to-end environment.	altpri
	cancel sched	Cancel job streams in the production plan. Not valid for workstations in end-to-end environment.	cancel
	deldep sched	Delete dependencies from job streams in the production plan. Not valid for workstations in end-to-end environment.	deldep
	display	Display job streams in the plan. .	display
	limit sched	Modify the limit for jobs concurrently running within a job stream.	limit
	release sched	Release job streams from dependencies in the production plan. Not valid for workstations in end-to-end environment.	release
	reply	Reply to job stream prompts in the production plan.	reply
	showschedules	Display information about job streams in the production plan.	list
submit sched	Submit job streams into the production plan. If the submit also identifies a second job stream with the "ALIAS" argument, the user must have "submit" access to that other job stream, as well Not valid for workstations in end-to-end environment.	submit	
Using the workload service assurance feature	All activities	For any user to perform any workload service assurance activities, the <i>TWS_user</i> must have the following access keywords:	display, modify, list

Example: To allow a user to perform all actions on a job stream except submit it and release it, specify:

```
schedule          access=adddep,altpri,cancel,deldep,display,limit,reply,list
```

Object type - userobj

The following table gives the additional access keywords required to work with users, other than those described in Table 32 on page 114:

Table 44. Users - additional access keywords

Activity			Access keywords required
Conman	altpass	Alter user passwords in the plan.	altpass
Tivoli Dynamic Workload Console			
Job Scheduling Console			

Example: To allow a user to list and modify user information, including passwords in the database, specify:

```
userobj      access=display,modify,altpass
```

Specifying access

Object type - variable

The following table gives the access keywords for using variable tables and the variables they contain (this includes the global variables)

Table 45. Variable tables - access keywords

Activity			Access keywords required
Composer Tivoli Dynamic Workload Console	add	Add new variable table (vartable) definitions in the database from a file of object definitions. Unlock access is needed to use the ;unlock attribute. To <i>add</i> individual variable entries within a table, the table must have <i>modify</i> access.	add, modify, unlock
	create	Create a text file of variable table (vartable) definitions in the database. Modify access is need to use the ;lock attribute. Create individual variable entries within the table.	display, modify
	delete	Delete variable table (vartable) definitions from the database. To <i>delete</i> individual variable entries within a table, the table must have <i>modify</i> access.	delete
	display	Display variable table (vartable) definitions in the database. Display individual variable entries within the table.	display
	extract	Extract a text file of variable table (vartable) definitions from the database. Extract individual variable entries within the table.	display
	list	List variable table (vartable) definitions in the database. List individual variable entries within the table.	display
	lock	Lock variable table (vartable) definitions in the database.	modify
	modify	Modify variable table (vartable) definitions in the database. Definitions are extracted into a file. After you have edited the file the definitions are used to replace the existing ones. To <i>modify</i> individual variable entries within a table, the table must have <i>modify</i> access.	add, modify
	new	Create variable table (vartable) definitions in the database from a template.	add, modify
	print	Print variable table (vartable) definitions in the database. Print individual variable entries within the table.	display
	rename	Rename variable table (vartable) definitions in the database. The user needs add access to the new object and delete and display access to the old object.	add, delete, display
	replace	Replace variable table (vartable) definitions in the database. Unlock access is needed to use the ;unlock attribute.	add, modify, unlock
	unlock	Unlock variable table (vartable) definitions in the database. Unlocking a table unlocks all the variables contained therein. Unlocking a variable unlocks the entire table where it is defined.	unlock
	Use variable tables in run cycles, job streams, and workstations		use

Example: To allow a user only to use variable tables when defining other scheduling objects, specify:

```
vartable          access=use
```

The *TWS_user* - special security file considerations

The *TWS_user* is a special user, and requires special consideration for the security file.

Required access for the *TWS_user* for workload service assurance

For any user to perform Workload service Assurance activities, the *TWS_user* must have *display*, *modify* and *list* access keywords assigned for all *job*, *schedule* and *cpu* objects.

New *TWS_user* in migrated Security file

If you change the *TWS_user* of your environment, for example, as you might do when performing a parallel upgrade, and then you migrate the Security file (to preserve your settings) you must set up the new *TWS_user* in the Security file in advance, with all its required access rights, before attempting to start Tivoli Workload Scheduler.

Sample security file

The following is a sample security file. A description of the file follows the listing.

```
#####
#   Sample Security File
#####
# (1) APPLIES TO TWS_users AND ROOT USERS LOGGED IN ON THE
#   MASTER DOMAIN MANAGER.
user mastersm cpu=$master + logon=TWS_user,root
begin
# OBJECT      ATTRIBUTES      ACCESS CAPABILITIES
# -----
job           access=@
schedule     access=@
resource     access=@
prompt       access=@
file         access=@
calendar     access=@
cpu          access=@
parameter    name=@ ~ name=r@  access=@
userobj      cpu=@ + logon=@    access=@
eventrule    name=@            access=add,delete,display,modify,list,unlock
action       provider=@        access=display,submit,use,list
event        provider=@        access=use
report       name=@            access=display
variable     name=a@,$default   access=add,delete,display,modify,use,list,unlock
end

#####
# (2) APPLIES TO TWS_users AND ROOT USERS LOGGED IN ON ANY
#   WORKSTATION OTHER THAN THE MASTER DOMAIN MANAGER.
user sm logon=TWS_user,root
begin
# OBJECT      ATTRIBUTES      ACCESS CAPABILITIES
# -----
job           cpu=$thiscpu      access=@
schedule     cpu=$thiscpu      access=@
resource     cpu=$thiscpu      access=@
prompt       access=@
calendar     access=@
cpu          cpu=$thiscpu      access=@
parameter    cpu=$thiscpu
              ~ name=r@    access=@
action       provider=@        access=display,submit,use,list
event        provider=@        access=use
report       name=RUNHIST,RUNSTATS access=display
file         name=globalopts    access=display
end
```

Sample security file

```
#####
# (3) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON THE
#     MASTER DOMAIN MANAGER.
user masterop cpu=$master + group=sys
begin
# OBJECT      ATTRIBUTES      ACCESS CAPABILITIES
# -----      -
job           cpu=@
              + logon="TWS_domain\\TWS_user" access=@
job           cpu=@
              + logon=root      access=adddep,altpri,cancel,
                              confirm,deldep,release,
                              reply,rerun,submit,use
job           cpu=@
              + logon=$user,$jclowner
              ~ logon=root      access=add,adddep,altpri,
                              cancel,confirm,
                              deldep,release,reply,
                              rerun,submit,use
schedule     cpu=$thiscpu  access=@
schedule     cpu=@        access=adddep,altpri,cancel,
                              deldep,limit,release,
                              submit
resource     access=add,display,
              resource,use
file         name=globalopts access=display
file         name=prodsked  access=display
file         name=symphony  access=display
file         name=trialsked access=build, display
calendar    access=display,use
cpu         cpu=@        access=@
parameter   name=@ ~ name=r@ access=@
report      name=RUNHIST,RUNSTATS access=display
end

#####
# (4) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON ANY
#     WORKSTATION OTHER THAN THE MASTER DOMAIN MANAGER
user op group=sys
begin
# OBJECT      ATTRIBUTES      ACCESS CAPABILITIES
# -----      -
job           cpu=$thiscpu
              + logon=$user  access=@
job           cpu=$thiscpu
              + logon=root    access=adddep,altpri,cancel,
                              confirm,deldep,release,
                              reply,rerun,submit,use
job           cpu=$thiscpu
              ~ logon=root    access=adddep,altpri,cancel,
                              confirm,deldep,release,
                              reply,rerun,submit,use
schedule     cpu=$thiscpu  access=@
resource     access=add,display,resource,use
prompt      access=add,display,reply,use
calendar    access=use
cpu         cpu=$thiscpu  access=console,fence,limit,
                              link,start,stop,unlink
parameter   name=@ ~ name=r@ access=@
end

#####
#####
# (5) APPLIES TO USERS LOGGED INTO THE MIS GROUP ON
#     ANY WORKSTATION.
user misusers group=mis
begin
```

```

# OBJECT      ATTRIBUTES      ACCESS CAPABILITIES
# -----      -
job           cpu=$thiscpu
              + logon=$user   access=@
job           cpu=$thiscpu
              + logon=$jclowner
              ~ logon=root   access=submit,use
schedule     cpu=$thiscpu   access=add,submit,
              modify,display
parameter    name=r@        access=@
parameter    name=@        access=display
end
#####
#####
# (6) APPLIES TO ALL OTHER USERS LOGGED IN ON ANY
#      WORKSTATION.
user default logon=@
begin
# OBJECT      ATTRIBUTES      ACCESS CAPABILITIES
# -----      -
job           cpu=$thiscpu
              + logon=$user   access=@
job           cpu=$thiscpu
              + logon=$jclowner
              ~ logon=root   access=submit,use
schedule     cpu=$thiscpu   access=add,submit,
              modify,display
parameter    name=u@        access=@
parameter    name=@ ~ name=r@ access=display
end
#####

```

Note that the order of definitions is from most to least specific. Because of the order, *TWS_users* and **root** users are matched first, followed by users in the **sys** group, and then users in the **mis** group. All other users are matched with the last definition, which is the least specific.

(1) APPLIES TO *TWS_users* AND ROOT USERS LOGGED IN ON THE MASTER DOMAIN MANAGER. user mastersm cpu=\$master + logon=*TWS_user*,root

This user definition applies to GUI and CLI access for *TWS_users* and **root** users logged into a master domain manager. They are given unrestricted access to all objects, except parameters that have names beginning with **r**. Access to the **r** parameters is given only to users in the **mis** group. They are the only ones who can generate all kinds of plans and who can create, update, and delete event rule definitions.

All users have access to all variable tables beginning with "a" and to the default table, irrespective of the default variable table name.

(2) APPLIES TO *TWS_users* AND ROOT USERS LOGGED IN ON ANY WORKSTATION OTHER THAN THE MASTER DOMAIN MANAGER. user sm logon=maestro,root

This user definition applies to *TWS_users* and **root** users to whom definition (1) does not apply, which are those who are logged in on any workstation other than the master domain manager. They are given unrestricted access to all objects on their login workstation. Note that prompts, files, and calendars are global in nature and are not associated with a workstation.

Sample security file

They can use event rules, but are not allowed to create, update, or delete event rule definitions.

(3) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON THE MASTER DOMAIN MANAGER. user masterop cpu=\$master + group=sys

This user definition applies to users logged into the **sys** group on the master domain manager. They are given a unique set of access capabilities. Multiple object statements are used to give these users specific types of access to different sets of objects. For example, there are three job statements:

- The first job statement permits unrestricted access to jobs that run on any workstation (@) under the user's name (**\$user**).
- The second job statement permits specific types of access to jobs that run on any workstation and that run as **root**.
- The third job statement permits specific types of access to jobs that run on any workstation. The jobs must run under the user's name (**\$user**) or under the name of the owner of the job file (**\$jclowner**). Jobs that run as root are excluded.

They are the only users defined on the master domain manager, different from maestro or root, who can generate trial and forecast plans.

(4) APPLIES TO USERS LOGGED INTO THE SYS GROUP ON ANY WORKSTATION OTHER THAN THE MASTER DOMAIN MANAGER. user op group=sys

This user definition applies to **sys** group users to whom definition (3) does not apply, which are those who are logged in on any workstation other than the master domain manager. They are given a set of access capabilities similar to those in definition (3). The exception is that access is restricted to objects on the user's login workstation (**\$thiscpu**).

(5) APPLIES TO USERS LOGGED INTO THE MIS GROUP ON ANY WORKSTATION. user misusers group=mis

This user definition applies to users logged into the **mis** group on any workstation. They are given a limited set of access capabilities. Resources, prompts, files, calendars, and workstations are omitted, which prevents access to these objects. These users are given unrestricted access to parameters with names that begin with **r**, but can only display other parameters.

(6) APPLIES TO ALL OTHER USERS LOGGED IN ON ANY WORKSTATION.

user default logon=@

This user definition gives a set of default capabilities to users other than those covered by the preceding definitions (1 to 5). These users are given unrestricted access to parameters with names that begin with **u**, but can only display other parameters. No access is permitted to parameters with names that begin with **r**.

Configuring for LDAP

This section describes how to configure LDAP (Lightweight Directory Access Protocol) for Tivoli Workload Scheduler. Only master domain managers, backup master domain managers, and agents that have a connector installed can be configured to use LDAP. For examples of how to configure LDAP for Tivoli Workload Scheduler, see “Example Configuration of an LDAP server for Microsoft Active Directory” on page 136.

Tivoli Workload Scheduler uses LDAP to authenticate users to post-installation tasks using Websphere in the following environments:

- Microsoft Active Directory
- IBM Tivoli Directory Server
- Sun ONE DS (formerly known as IPLanet)
- RACF®

LDAP authenticated users can run configuration tasks from the following interfaces:

- **composer** command interface
- Tivoli Dynamic Workload Console
- Job Scheduling Console
- Command-line client
- Command-line as clients connected to the master domain manager using HTTP or HTTPS

If you have implemented LDAP security, you need to configure the embedded WebSphere Application Server for LDAP support. For full information about each of the properties you need to configure you should refer to the documentation of WebSphere Application Server. However, in this section are provided examples of LDAP configurations for Microsoft Active Directory and RACF, and also information about the Pluggable Authentication Module:

- “Example Configuration of an LDAP server for Microsoft Active Directory” on page 136
- “Configuring RACF LDAP servers” on page 139
- “Using the Pluggable Authentication Module” on page 141

To configure LDAP for Tivoli Workload Scheduler, perform the following steps:

1. Change WebSphere Application Server security.
 - a. Stop WebSphere Application Server using the command **StopWAS**.
 - b. Backup WebSphere Application Server configuration using the command **backupConfig**.
 - c. Dump your current security properties using the command **showSecurityProperties**.
 - d. Customize security properties. See Chapter 7, “Setting connection security,” on page 169.
 - e. Load new properties using the command **changeSecurityProperties**.
 - f. Restart WebSphere Application Server using the command **StartWas**.
2. Change the security file.
 - a. Dump the security file using the command **dumpsec > my_sec.txt** where *my_sec.txt* is the name of the security file.
 - b. Add the new LDAP username to the security file.

Configuring for LDAP

- c. Load the new security file using the command **makesec my_sec.txt**.
3. Update the properties of the WebSphere Application Server services as follows:

UNIX

On UNIX, after you modified the WebSphere Application Server security properties to use LDAP, it is important to change the SOAP client properties using the **updateWas.sh** script (see “Application server - updating the SOAP properties after changing the WebSphere Application Server user or its password” on page 271 for full details).

Windows

On Windows, after you modified the WebSphere Application Server security properties to use LDAP, it is important to change the properties of the Windows Services using the **updateWasService.bat** script (see “Application server - updating the Windows services after modifications” on page 270 for full details).

4. Propagate changes.
 - a. Update the useropts file on every command-line client that points to your workstation.
 - b. Update the useropts file on every fault-tolerant agent in your environment that has an HTTP/HTTPS connection defined in localopts that points to your workstation. The HTTP/HTTPS connection is used to submit a predefined job or jobstream.
 - c. Update the engine connection parameters on every connected Tivoli Dynamic Workload Console or Job Scheduling Console.
5. Change the useropts file.

Change the username in the useropts file and insert the new password in plain text between quotes. The password will be encrypted the first time you log in.

Note: Tivoli Dynamic Workload Console version 8.4 and higher can be configured with an LDAP login.

For more information about WebSphere Application Server configuration, see: Standalone LDAP registry settings

Example Configuration of an LDAP server for Microsoft Active Directory

This section is an example of how to configure an LDAP server for Microsoft Active Directory. This section is divided in the following subsections:

- “Creating an Active Directory account”
- “Example properties file for LDAP servers” on page 137
- “Updating the Tivoli Workload Scheduler security file” on page 138
- “Using the Tivoli Workload Scheduler command-line utilities” on page 138
- “Updating associated WebSphere Application Server properties” on page 138

Creating an Active Directory account

For this example an Active Directory account as a supported LDAP server using TEST_LDAP as the user in the twscvt.it domain was created.

The **useDomainQualifiedUserNames** security settings is set to **true**, so each user ID must be provided in the format realm/username. For example:

```
1ab235.acmelab.it.acme.com:389/TEST_LDAP
```

Example configuration of Microsoft Active Directory LDAP servers

Example properties file for LDAP servers

You can use the `changeSecurityProperties` script to create the LDAP properties (see “Changing the security settings” on page 258 for full details). The following is a sample properties file:

```
#####
Global Security Panel
#####
enabled=true
enforceJava2Security=false
useDomainQualifiedUserNames=false
cacheTimeout=600
ltpaTimeOut=720
issuePermissionWarning=true
activeProtocol=CSI
useFIPS=false
activeAuthMechanism=LTPA
activeUserRegistry=LDAP

#####
Local OS Registry
#####
LocalOSServerID=tw8511
LocalOSServerpassword=*****
LocalOSServerREALM=myhost.mydomain.com
#####
Custom User Registry
#####
CustomServerID=
CustomServerpassword=*****
CustomServerREALM=
#####
WIM Registry
#####
#WIMServerID=<Unknown>
#WIMServerpassword=*****
#WIMServerREALM=defaultWIMFileBasedRealm
#####
LDAP Panel
#####
LDAPServerId=tw8511
LDAPPassword=*****
LDAPPrimaryAdminId=
LDAPUseRegistryServerId=true
LDAPServerType=IBM_DIRECTORY_SERVER
LDAPHostName=
LDAPPort=389
LDAPBaseDN=
LDAPBindDN=
LDAPBindPassword=*****
LDAPsearchTimeout=120
LDAPreuseConnection=true
LDAPIgnoreCase=true
LDAPsslEnabled=false
LDAPsslConfig=DefaultNode/DefaultSSLSettings
```

Update the `LDAPServerId` and `LDAPPassword` properties with actual values.

Because the server type is `ACTIVE_DIRECTORY` some advanced LDAP properties are also needed. The following advanced properties are required:

```
#####
Advanced LDAP Panel
#####
LDAPUserFilter=(&(uid=%v)(objectclass=person))
LDAPGroupFilter=(&(cn=%v)(|(objectclass=groupOfNames)
(objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))
LDAPUserIdMap=*:uid
```

Example configuration of Microsoft Active Directory LDAP servers

```
LDAPGroupIdMap=:cn
LDAPGroupMemberIdMap=ibm-allGroups:member;ibm-allGroups:uniqueMember
LDAPCertificateFilter=
LDAPCertificateMapMode=EXACT_DN
```

Updating the Tivoli Workload Scheduler security file

The Tivoli Workload Scheduler security file needs to be updated to allow Active Directory users to access Tivoli Workload Scheduler objects (see “Updating the security file” on page 94 for full details). The following is a brief example of security file updates:

```
USER MAESTRO
  CPU=@+LOGON=tw83,Administrator,administrator,TEST_LDAP
BEGIN
USEROBJ CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,ALTPASS,UNLOCK
JOB CPU=@ ACCESS=ADD,ADDDEP,ALTPRI,CANCEL,CONFIRM,DELDEP,DELETE,DISPLAY,KILL,
  MODIFY,RELEASE,REPLY,RERUN,SUBMIT,USE,LIST,UNLOCK
SCHEDULE CPU=@ ACCESS=ADD,ADDDEP,ALTPRI,CANCEL,DELDEP,DELETE,
  DISPLAY,LIMIT,MODIFY,RELEASE,REPLY,SUBMIT,LIST,UNLOCK
RESOURCE CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,RESOURCE,USE,LIST,UNLOCK
PROMPT ACCESS=ADD,DELETE,DISPLAY,MODIFY,REPLY,USE,LIST,UNLOCK
FILE NAME=@ ACCESS=CLEAN,DELETE,DISPLAY,MODIFY,UNLOCK
CPU CPU=@ ACCESS=ADD,CONSOLE,DELETE,DISPLAY,FENCE,LIMIT,LINK,MODIFY,
  SHUTDOWN,START,STOP,UNLINK,LIST,UNLOCK
PARAMETER CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,UNLOCK
CALENDAR ACCESS=ADD,DELETE,DISPLAY,MODIFY,USE,UNLOCK
END
```

In the example, the `useDomainQualifiedUserNames` security property is set to `false`.

Using the Tivoli Workload Scheduler command-line utilities

Composer and other Tivoli Workload Scheduler command lines can use LDAP users when connecting to WebSphere Application Server. You can provide user and password either in the `useropts` file (see “Setting user options” on page 34 for full details) or as command line parameters. For example:

```
composer -user TEST_LDAP -password xxxxxxxx list ws=@
```

The following shows the contents of a `useropts` file with a BluePages user:

```
USERNAME = TEST_LDAP
PASSWORD = "ENCRYPT:0sRfeHjUtG5ZsyLgc0kGMq=="
```

In both the examples, the `useDomainQualifiedUserNames` security property is set to `false`.

Updating associated WebSphere Application Server properties

On both Windows and UNIX you must also update associated configuration properties after configuring to use LDAP:

UNIX

On UNIX, after the WebSphere Application Server has been modified to use LDAP, it is important to change the SOAP client properties using the `updateWas.sh` script (see “Application server - updating the SOAP properties after changing the WebSphere Application Server user or its password” on page 271 for full details). For example:

```
updateWas.sh -user john.smith@domain.com -password zzzz
```

where `theuser` and `password` options are the new logical user that is authorized to stop the WebSphere Application Server. The user must be defined in the user registry

Example configuration of Microsoft Active Directory LDAP servers

Windows

On Windows, after WebSphere Application Server has been modified to use LDAP, it is important to change the properties of the Windows Services using the **updateWasService.bat** script (see “Application server - updating the Windows services after modifications” on page 270 for full details). For example:

```
updateWasService -userid tws83 -password zzzz  
-wasuser TEST_LDAP -waspassword xxxxxx
```

where the **-userid** and **-password** options are the operating system user ID and password of the user that is running the WebSphere Application Server process, and **-wasuser** and **-waspassword** are the new logical user that is authorized to stop the WebSphere Application Server. The **-wasuser** must be defined in the user registry

Configuring RACF LDAP servers

This section is an example of how to configure RACF and IBM Tivoli Directory Server as an LDAP user registry for WebSphere. This section is divided in the following subsections:

- “Creating the LDAP properties file”
- “Updating the Tivoli Workload Scheduler security file” on page 140
- “Using the Tivoli Workload Scheduler command-line utilities” on page 140
- “Updating associated WebSphere Application Server properties” on page 141

When you use an LDAP server, the WAS realm name is no longer TWSREALM but follows the format `ldaphost:port`, where `ldaphost` and `port` are the fully qualified host name and port of the LDAP server. In this example, the host name of the LDAP server is `9.87.130.11` and the port is `3389`. If **useDomainQualifiedUserNames** security settings is set to true, each user ID must have the format `realm/username`. For example:

```
9.87.130.11:3389/brutus
```

Creating the LDAP properties file

You can use the **changeSecurityProperties** script to create the LDAP properties (see “Changing the security settings” on page 258 for full details). The following is a sample properties file:

```
#####  
Global Security Panel  
#####  
enabled=true  
enforceJava2Security=false  
useDomainQualifiedUserNames=false  
cacheTimeout=600  
ltpaTimeOut=720  
issuePermissionWarning=true  
activeProtocol=CSI  
useFIPS=false  
activeAuthMechanism=LTPA  
activeUserRegistry=LDAP  
  
#####  
LDAP Panel  
#####  
LDAPServerId=TWSAdmin  
LDAPPassword=mypassword  
LDAPPrimaryAdminId=  
LDAPUseRegistryServerId=true
```

Configuring RACF LDAP servers

```
1          LDAPServerType=IBM_DIRECTORY_SERVER
1          LDAPHostName=ldapservr.mycompany.com
1          LDAPPort=389
1          LDAPBaseDN=dc=mycompany,dc=com
1          LDAPBindDN=CN=ldap bind,DC=mycompany,DC=com
1          LDAPBindPassword=bindpassword
1          LDAPsearchTimeout=120
1          LDAPPreuseConnection=true
1          LDAPIgnoreCase=true
1          LDAPsslEnabled=false
1          LDAPsslConfig=
1
1          #####
1          Advanced LDAP Panel
1          #####
1          LDAPUserFilter=(&(uid=%v)(objectclass=person))
1          LDAPGroupFilter=(&(cn=%v)(objectclass=groupOfNames))
1          LDAPUserIdMap=*:uid
1          LDAPGroupIdMap=*:cn
1          LDAPGroupMemberIdMap=ibm-allGroups:member
1          LDAPCertificateFilter=
1          LDAPCertificateMapMode=EXACT_DN
```

1 Update the **LDAPServerId** and **LDAPPASSWORD** properties with actual values.
1 Because the server type is **IBM_DIRECTORY_SERVER** advanced LDAP properties
1 are needed.

Updating the Tivoli Workload Scheduler security file

1 The Tivoli Workload Scheduler security file needs to be updated to allow LDAP
1 users to access Tivoli Workload Scheduler objects (see “Updating the security file”
1 on page 94 for full details). The following is a brief example of security file
1 updates:

```
1          USER MAESTRO
1          CPU=@+LOGON=brutus,u0001,Administrator,administrator,
1          BEGIN
1          USEROBJ CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,ALTPASS,UNLOCK
1          JOB CPU=@ ACCESS=ADD,ADDDEP,ALTPRI,CANCEL,CONFIRM,DELDEP,DELETE,DISPLAY,KILL,
1          MODIFY,RELEASE,REPLY,RERUN,SUBMIT,USE,LIST,UNLOCK
1          SCHEDULE CPU=@ ACCESS=ADD,ADDDEP,ALTPRI,CANCEL,DELDEP,DELETE,DISPLAY,LIMIT,
1          MODIFY,RELEASE,REPLY,SUBMIT,LIST,UNLOCK
1          RESOURCE CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,RESOURCE,USE,LIST,UNLOCK
1          PROMPT ACCESS=ADD,DELETE,DISPLAY,MODIFY,REPLY,USE,LIST,UNLOCK
1          FILE NAME=@ ACCESS=CLEAN,DELETE,DISPLAY,MODIFY,UNLOCK
1          CPU CPU=@ ACCESS=ADD,CONSOLE,DELETE,DISPLAY,FENCE,LIMIT,LINK,MODIFY,
1          SHUTDOWN,START,STOP,UNLINK,LIST,UNLOCK
1          PARAMETER CPU=@ ACCESS=ADD,DELETE,DISPLAY,MODIFY,UNLOCK
1          CALENDAR ACCESS=ADD,DELETE,DISPLAY,MODIFY,USE,UNLOCK
1          END
```

1 In the example, the **useDomainQualifiedUserNames** security property is set to
1 **false**.

Using the Tivoli Workload Scheduler command-line utilities

1 Composer and other Tivoli Workload Scheduler command lines can use LDAP
1 users when connecting to WebSphere Application Server. You can provide user and
1 password either in the **useropts** file (see “Setting user options” on page 34 for full
1 details) or as command line parameters. For example:

```
1          composer -user brutus -password changeit list ws=@
```

1 The following shows the contents of a **useropts** file with a **BluePages** user:

```
1          USERNAME = brutus
1          PASSWORD = "ENCRYPT:K1YreGjWsP7NqyLXzHq3Ab=="
```

In both the examples, the `useDomainQualifiedUserNames` security property is set to `false`.

Updating associated WebSphere Application Server properties

On both Windows and UNIX you must also update associated configuration properties after configuring to use LDAP:

UNIX

On UNIX, after the WebSphere Application Server has been modified to use LDAP, it is important to change the SOAP client properties using the `updateWas.sh` script (see “Application server - updating the SOAP properties after changing the WebSphere Application Server user or its password” on page 271 for full details). For example:

```
updateWas.sh -user brutus -password xxxxxx
```

where the `user` and `password` options are the new logical user that is authorized to stop the WebSphere Application Server. The user must be defined in the user registry

Windows

On Windows, after WebSphere Application Server has been modified to use LDAP, it is important to change the properties of the Windows Services using the `updateWasService.bat` script (see “Application server - updating the Windows services after modifications” on page 270 for full details). For example:

```
updateWasService -userid tws83 -password zzzz  
-wasuser brutus -waspassword xxxxxx
```

where the `-userid` and `-password` options are the operating system user ID and password of the user that is running the WebSphere Application Server process, and `-wasuser` and `-waspassword` are the new logical user that is authorized to stop the WebSphere Application Server. The `-wasuser` must be defined in the user registry

Using the Pluggable Authentication Module

Tivoli Workload Scheduler enhances the embedded WebSphere Application Server by supporting a user authentication mechanism based on the Pluggable Authentication Module.

This enhancement provides a single authentication mechanism that is capable of authenticating users no matter what their user registry implementations are based on, local OS or LDAP.

Tivoli Workload Scheduler automatically installs the plug-in that enables WebSphere Application Server to use Pluggable Authentication Module-enabled authentication. The plug-in uses the service with name `other`. Ordinarily, you need do nothing to configure the Pluggable Authentication Module. However, if the level of your authorizations inhibits you from using `other`, you should add the service with name `checkpassword` in the `/etc/pam.conf` file.

The use of the Pluggable Authentication Module also extends the WebSphere Application Server’s capabilities to include support for authentication in HP Trusted Mode environments.

Using the Pluggable Authentication Module

Tivoli Workload Scheduler is set by default to use a Pluggable Authentication Module user registry called "custom". If the Pluggable Authentication Module is not configured with this registry, WebSphere Application Server looks in the local user registry on the master domain manager.

Chapter 6. Network administration

This chapter describes how to administer the Tivoli Workload Scheduler network. It has the following topics:

- “Network overview”
- “Network definitions” on page 144
- “Network communications” on page 145
- “Network operation” on page 146
- “Support for Internet Protocol version 6” on page 165
- “Optimizing the network” on page 151
- “Netman configuration file” on page 160
- “Extended agents” on page 162
- “IP address validation” on page 165
- “Impact of network changes” on page 167

Network overview

A Tivoli Workload Scheduler network consists of one or more domains arranged hierarchically. A Tivoli Workload Scheduler domain is a logical grouping of workstations, consisting of a domain manager and a number of agents.

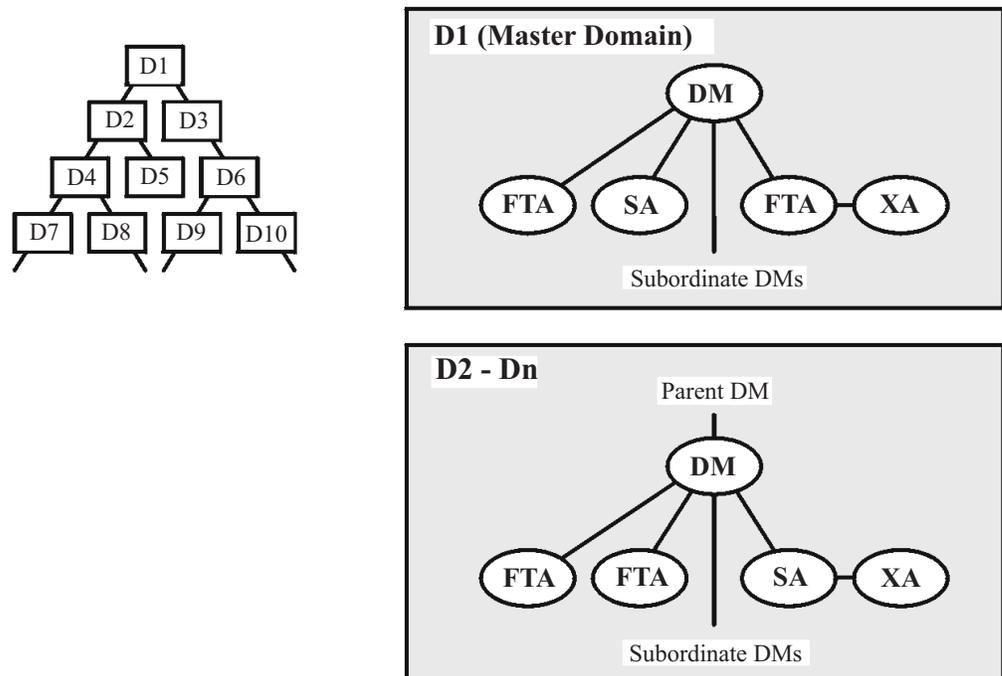


Figure 2. Tivoli Workload Scheduler network domain structure

Network definitions

Domain

A named group of Tivoli Workload Scheduler workstations consisting of one or more agents and a domain manager. All domains have a parent, except the master domain.

Master domain

The topmost domain in an Tivoli Workload Scheduler network.

Master domain manager

The domain manager in the topmost domain of an Tivoli Workload Scheduler network. It contains the centralized master files used to document scheduling objects. It creates the Production Control file (Symphony) at the start of each production period and performs all logging and reporting for the network. See also Domain Manager.

Backup master domain manager

A fault-tolerant agent capable of assuming the responsibilities of the master domain manager.

Parent domain

The domain directly above the current domain. All domains, except the master domain, have a parent domain. All communications to/from a domain is routed through the parent domain manager.

Domain Manager

The management hub in a domain. All communications in and from the agents in a domain is routed through the domain manager. See also Master Domain Manager.

Backup domain manager

A fault-tolerant agent capable of assuming the responsibilities of its domain manager.

Fault-tolerant agent

An agent workstation capable of resolving local dependencies and launching its jobs in the absence of a domain manager.

Standard agent

An agent workstation that launches jobs only under the direction of its domain manager.

Extended agent

An agent workstation that launches jobs only under the direction of its host. Extended agents can be used to interface Tivoli Workload Scheduler with non-Tivoli Workload Scheduler systems and applications

Host The scheduling function required by extended agents. It can be performed by any Tivoli Workload Scheduler workstation, except another extended agent.

Network communications

In a Tivoli Workload Scheduler network, agents communicate with their domain managers, and domain managers communicate with their parent domain managers. There are basically two types of communications that take place:

- Start-of-production period initialization (distribution of new Symphony file)
- Scheduling events in the form of change-of-state messages during the production period

Before the start of each new production period, the master domain manager creates a production control file called Symphony. Then, Tivoli Workload Scheduler is restarted in the network, and the master domain manager sends a copy of the new Symphony file to each of its automatically-linked agents and subordinate domain managers. The domain managers, in turn, send copies to their automatically-linked agents and subordinate domain managers. Agents and domain managers that are not set up to link automatically are initialized with a copy of Symphony as soon as a link operation is run in Tivoli Workload Scheduler.

Once the network is started, scheduling messages, like job starts and completions, are passed from the agents to their domain managers, through parent domain managers to the master domain manager. The master domain manager then broadcasts the messages throughout the hierarchical tree to update the Symphony files of all domain managers and the domain managers forward the messages to all fault-tolerant agents in their domain running in *FullStatus* mode.

Network links

Links provide communications between Tivoli Workload Scheduler workstations in a network. Links are controlled by the AUTO Link flag, and the Console Manager **link** and **unlink** commands. When a link is open, messages are passed between two workstations. When a link is closed, the sending workstation stores messages in a local pobox file and sends them to the destination workstation when the link is reopened.

This means that when links are closed, the message queues fill up with messages for the inaccessible workstations. To maximize the performance of Tivoli Workload Scheduler, monitor workstations for closed links and attempt to reopen them as soon as possible.

Note: Extended agents do not have links. They communicate with their domain managers through their hosts.

To have a workstation link opened automatically, turn on the AUTO Link flag in the workstation's definition. The link is first opened when Tivoli Workload Scheduler is started on the Master Domain workstation. If the subdomain manager and workstations are not initialized and their AUTO Link flag is on, the master domain manager attempts to link to its subordinates and begin the initialization processes. If the AUTO Link flag is turned off, the workstation is only initialized by running a **link** command from the master domain manager. After the workstation is initialized, it automatically starts and issues a link back to its domain manager.

If you stop a workstation, the links from it to other workstations are closed. However, the links from the other workstations to it remain open until either one of the following situations occurs:

Network communications

- The stopped workstation is restarted and a **link** command is issued
- The other workstations' **mailman** processes time out, and perform an **unlink** for the workstation

1 When the **link** command is issued and the connection has been established, if the
1 domain manager does not receive any reply within the timeout period, the
1 chkhltst service is automatically invoked by **mailman**.

1 This service verifies that the workstation mailbox can be successfully read, and
1 checks if there are errors in the mailbox header. Resulting information is logged in
1 the TWSMERGE.log file of the domain manager as follows:

- If a file system error occurs while opening the mailbox, the following message is reported: AWSBDY126E An error occurred opening the Mailbox.msg file in *CPU_NAME*.
- If an error occurs while opening the mailbox because **mailman** is reading the mailbox, the following message is reported: AWSBDY123I The Mailbox.msg file in *CPU_NAME* is correctly read by Mailman.
- If the mailbox is correctly opened, but an error occurs while reading the header, the following message is reported: AWSBDY125E An error occurred reading the header of the Mailbox.msg file in *CPU_NAME*.
- If the mailbox is correctly opened and no error occurs while reading the header, the following message is reported: AWSBDY124W The Mailbox.msg file in *CPU_NAME* is not read by Mailman.

1 This service can also be launched manually by using the **conman** command. See
1 the *IBM Tivoli Workload Scheduler User's Guide and Reference* for more details.

To be certain that inter-workstation communication is correctly restored, you can issue a **link** command after restarting a workstation.

Network operation

The batchman process on each domain manager and fault-tolerant agent workstation operates autonomously, scanning its Symphony file to resolve dependencies and launch jobs. Batchman launches jobs via the jobman process. On a standard agent, the jobman process responds to launch requests from the domain manager's batchman.

The master domain manager is continuously informed of job launches and completions and is responsible for broadcasting the information to domain managers and fault-tolerant agents so they can resolve any inter-workstation dependencies.

The degree of synchronization among the Symphony files depends on the setting of the *FullStatus* mode in a workstation's definition. Assuming that these modes are turned on, a fault-tolerant agent's Symphony file contains the same information as the master domain manager's (see the section that explains how to manage workstations in the database in the *Tivoli Workload Scheduler: Job Scheduling Console User's Guide*).

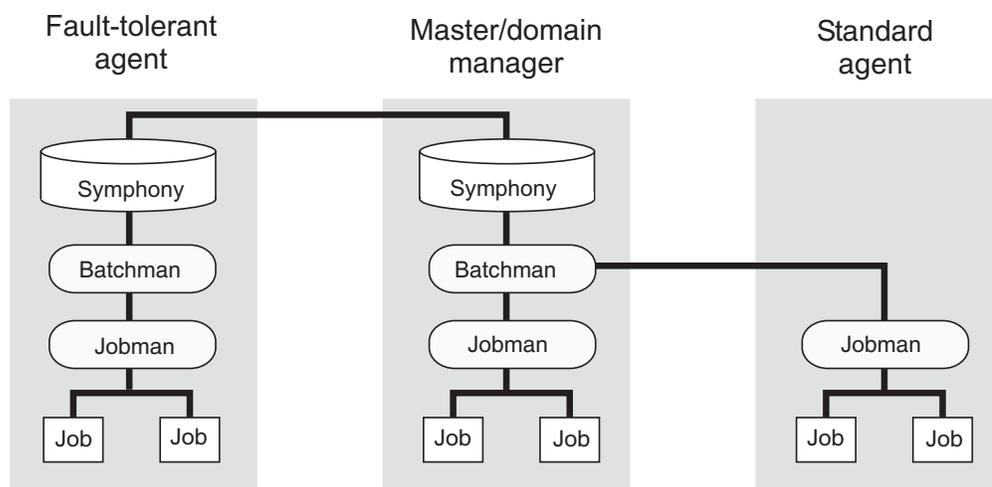


Figure 3. Symphony file synchronization

Network processes

Netman is started by the **StartUp** script (command). The order of process creation is netman, mailman, batchman, and jobman. On standard agent workstations, batchman does not run. All processes, except jobman, run as the **TWS** user. Jobman runs as **root**.

When network activity begins, netman receives requests from remote mailman processes. Upon receiving a request, netman spawns a writer process and passes the connection off to it. Writer receives the message and passes it to the local mailman. The writer processes (there might be more than one on a domain manager) are started by link requests and are stopped by unlink requests (or when the communicating mailman terminates).

Domain managers, including the master domain manager, can communicate with a large number of agents and subordinate domain managers. For improved efficiency, you can define mailman servers on a domain manager to distribute the communications load (see the section that explains how to manage workstations in the database in the *Tivoli Workload Scheduler: Job Scheduling Console User's Guide*).

Network operation

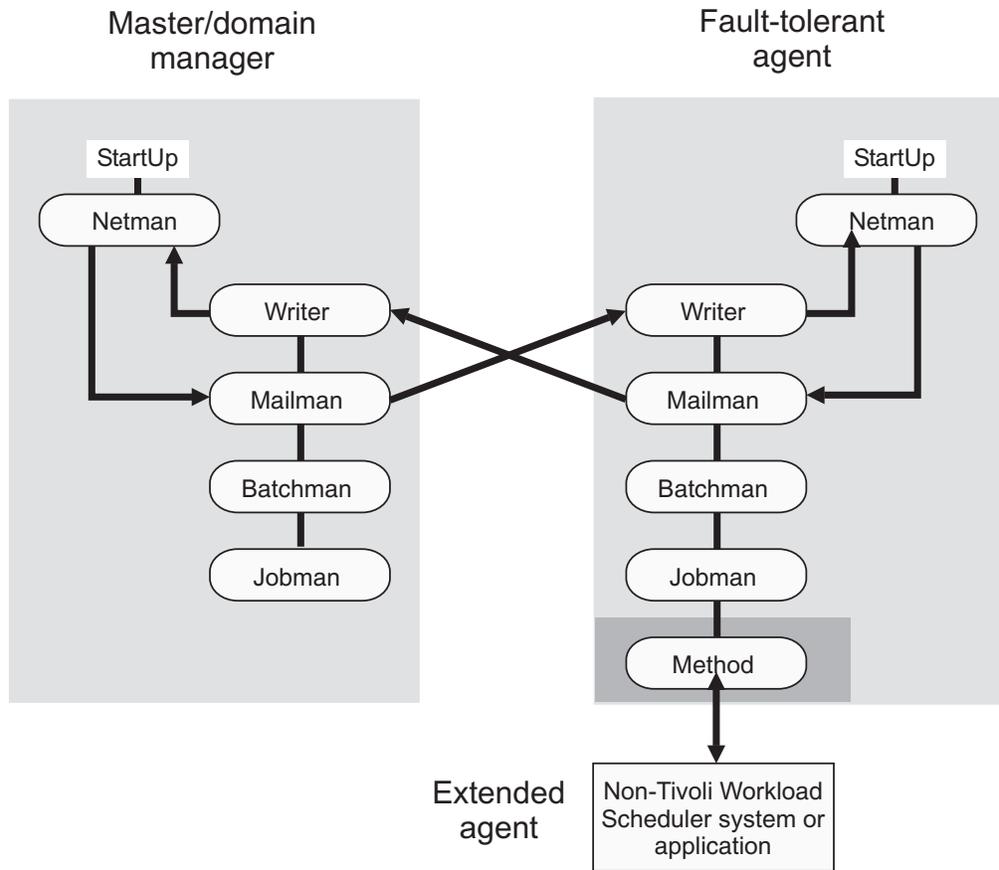


Figure 4. Process creation on domain manager and fault-tolerant agent

The **StartUp** command is normally run automatically, but can also be run manually, as follows:

StartUp

Starts **netman**, the Tivoli Workload Scheduler network management process.

In Windows, the **netman** service is started automatically when a computer is restarted. **StartUp** can be used to restart the service if it is stopped for any reason.

In UNIX, the **StartUp** command can be run automatically by invoking it from the `/etc/inittab` file, so that WebSphere Application Server infrastructure and **netman** is started each time a computer is rebooted. **StartUp** can be used to restart **netman** if it is stopped for any reason.

The remainder of the process tree can be restarted with the

```
conman start  
conman startmon
```

commands. See the documentation about **conman** in the *Tivoli Workload Scheduler: User's Guide and Reference* for more information.

Authorization: You must have **start** access to the workstation.

Syntax: **StartUp** [-v | -u]

Arguments:

- v Displays the command version and exits.
- u Displays command usage information and exits.

Examples: To display the command name and version, run the following command:

```
StartUp -v
```

To start the **netman** process, run the following command:

```
StartUp
```

Monitoring the status of Tivoli Workload Scheduler processes

You can use event-driven workload automation (EDWA) to monitor the status of network processes and to start a predefined set of actions when one or more specific events take place. For more information about event-driven workload automation, see *Tivoli Workload Scheduler: User's Guide and Reference*.

You can monitor the following processes:

- agent
- appservman
- batchman
- jobman
- mailman
- monman
- netman

The .XML file contains the definition of a sample event rule to monitor the status of the specified processes on the specified workstation. This event rule calls the MessageLogger action provider to write a message in a log file in an internal auditing database. If the condition described in the rule is already existing when you deploy the rule, the related event is not generated. For more information about the MessageLogger action provider, see *Tivoli Workload Scheduler: User's Guide and Reference*:

```
<eventRule name="PROCESSES" ruleType="filter" isDraft="no">
  <eventCondition name="twProcMonEvt1" eventProvider="TWSApplicationMonitor"
eventType="TWSProcessMonitor">
    <scope>
      AGENT, BATCHMAN DOWN
    </scope>
    <filteringPredicate>
      <attributeFilter name="ProcessName" operator="eq">
        <value>process_name1</value>
      </attributeFilter>
      <attributeFilter name="TWSPath" operator="eq">
        <value>TWS_path</value>
      </attributeFilter>
      <attributeFilter name="Workstation" operator="eq">
        <value>workstation_name</value>
      </attributeFilter>
      <attributeFilter name="SampleInterval" operator="eq">
        <value>sample_interval</value>
      </attributeFilter>
    </filteringPredicate>
  </eventCondition>
  <action actionProvider="MessageLogger" actionType="MSGLOG" responseType="onDetection">
    <scope>
      OBJECT=AAAAAAA MESSAGE=TWS PROCESS DOWN: %TWSPROCONEVT1.PROCESSNAME}
ON %TWSPROCONEVT1.TWSPATH}
    </scope>
    <parameter name="ObjectKey">
      <value>object_key</value>
    </parameter>
    <parameter name="Severity">
      <value>message_severity</value>
    </parameter>
    <parameter name="Message">
      <value>log_message</value>
    </parameter>
  </action>
</eventRule>
</eventRuleSet>
```

where:

```

1      process_name
1          Is the name of the process to be monitored. You can insert more than one
1          process name, as follows:
1          <attributeFilter name="ProcessName" operator="eq">
1              <value>agent</value>
1              <value>batchman</value>
1          </attributeFilter>
1
1      TWS_path
1          Is the directory containing the Symphony file and the bin directory.
1
1      workstation_name
1          Is the workstation on which the event is generated.
1
1      sample_interval
1          Is the interval, expressed in seconds, for monitoring the process status.
1
1      object_key
1          Is a key identifying the object to which the message pertains.
1
1      message_severity
1          Is the severity of the message.
1
1      log_message
1          Is the message to be logged.

```

Optimizing the network

The structure of a Tivoli Workload Scheduler network goes hand in hand with the structure of your enterprise's network. The structure of the domains must reflect the topology of the network in order to best use the available communication channels.

But when planning the Tivoli Workload Scheduler network, the following must be taken into consideration:

- Data volumes
- Connectivity

Data volumes

Network capacity must be planned to adapt to the amount of data that is circulating. Particularly high transmission volumes might be caused by the following:

- Transfer of large Symphony files.
- Message traffic between the master domain manager and a *FullStatus* agent.
- Message traffic from a domain manager when the domain has many agents.
- Heavy use of internetwork dependencies, which extends traffic to the entire network.

Connectivity

For the more critical agents in your network, you need to consider their position in the network. The reliability of workload execution on a particular agent depends on its capacity to receive a fresh Symphony file at the start of the production period. If the workload contains many dependencies, a reliable connection to the rest of the network is also required. These factors suggest that the best place for critical agents is in the master domain, or to be set up as domain managers immediately under the master domain manager, possibly receiving their Symphony files through a set of dedicated mailman servers. Further, it is important for critical agents that any domain manager above them in the tree structure must be hosted on powerful systems and must have an adequate backup system to ensure continuity of operation in the event of problems.

Tivoli Workload Scheduler provides two mechanisms to accommodate a particular network situation: the domain structure and mailman servers. Whereas domain structure establishes a hierarchy among Tivoli Workload Scheduler agents, mailman servers are used to tune the resources dedicated to the connection between two agents.

Domain

Use the Tivoli Workload Scheduler domain structure mechanism to create a tree-shaped structure for the network, where all communications between two points use the unique path defined by the tree (climb to the common ancestor and go down to the target, as opposed to direct TCP communication). As a consequence, the domain structure separates the network into more-manageable pieces. This is for easier filtering, overview, action, and monitoring. However, it does also introduce some delay in the workload processing. For instance when distributing the Symphony file, a fault-tolerant agent inside a domain needs to wait for two steps of Symphony distribution to be completed (from master domain manager to domain manager and from domain manager to fault-tolerant agent). The same is valid for every other type of communication that comes from the master domain manager.

This has the following implications:

- Critical business activities must be as close as possible to the master domain manager
- The domain manager must be installed on as powerful a workstation as possible
- A similarly powerful backup domain manager must be included in the network
- The network link between the domain manager and its backup must be as fast as possible to pass all the updates received from the subtree
- If intervention is needed directly on the domain, either give shell access to the operators to use the Tivoli Workload Scheduler command line, or install a connector so that the Tivoli Dynamic Workload Console and the Job Scheduling Console can be used.

Mailman servers

Mailman servers allocate separate processes dedicated to the communication with other workstations. The main mailman is dedicated to the transfer and network hub activities. The use of mailman servers on the domain manager must be carefully planned. The main parameter is the number of downstream connections at each level of the tree. This number describes the number of mailman servers that a main mailman is connected to, or the number of agents a mailman server is connected to.

The maximum number of downstream connections is about 20 for Solaris, 50 for Windows and about 100 for other UNIX workstations, depending on their power. Typical downstream connections is about 10 for Solaris, about 15 for Windows and about 20 for other UNIX workstations. However, you must also take into consideration the link speed and the queue sizes, discussed below.

Planning space for queues

To plan space for event queues, and possible alert levels and reactions, you must model the flows passing through the agents and, in particular, the domain managers.

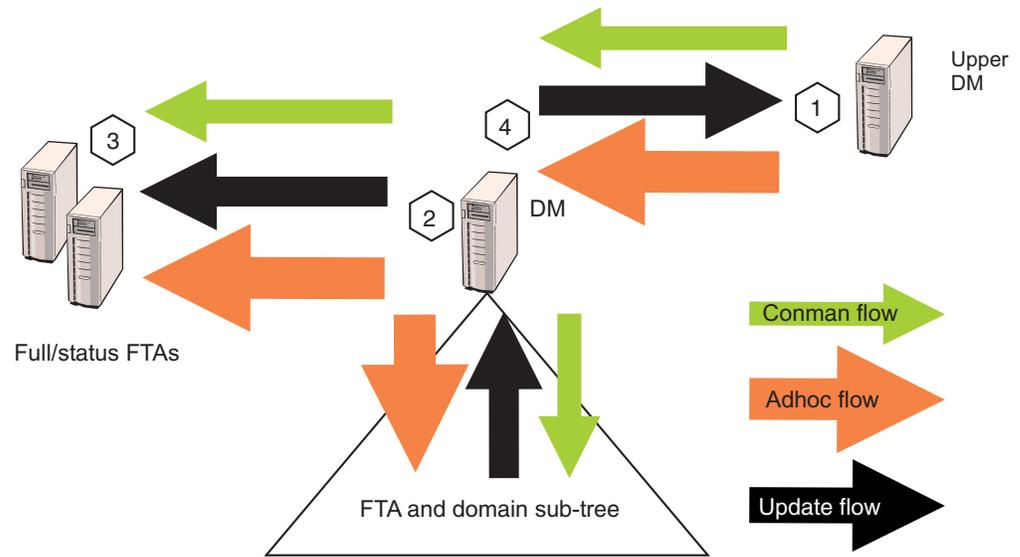


Figure 5. Typical Tivoli Workload Scheduler network flows.

For a typical domain manager, the main flow comes from update activity reported by the subtree and from ad hoc submissions arriving from the master domain manager and propagating to the entire network. Under these conditions, the most critical errors are listed by order of importance in Table 46 on page 154:

Optimizing the network

Table 46. Critical flow errors.

Flow no.	Location	Queue	Risk	Impact
1	Upper domain manager	dm.msg	The queue fills up because of too many unlinked workstations in the domain or a downstream domain manager has failed.	The upper domain manager fails and propagates the error.
2	Domain manager	<i>FullStatus</i> fta.msg	The queue fills up because of too many unlinked workstations in the domain or because the <i>FullStatus</i> fault-tolerant agent is not coping with the flow.	The domain manager fails and favors the occurrence of #1.
3	Domain manager and <i>FullStatus</i> fault-tolerant agent	Mailbox.msg or Intercom.msg	The queue fills up because the <i>FullStatus</i> fault-tolerant agent cannot cope with the flow.	The <i>FullStatus</i> fault-tolerant agent fails and favors the occurrence of #2.
4	Domain manager	tomaster.msg	The queue fills up because of too many unlinked workstations in the domain.	The domain manager starts to unlink the subtree and accumulates messages in the structure.
5	Fault-tolerant agents - only when <code>enSwfaultTo1</code> global option is set to <i>yes</i>	deadletter.msg	The queue fills up because of too many unlinked workstations in the domain.	The agent stops.
6	Fault-tolerant agents - only when <code>enSwfaultTo1</code> global option is set to <i>yes</i>	ftbox.msg	The queue is circular. The rate of messages entering the queue exceeds the rate of messages being processed, because of too many unlinked workstations in the domain.	Events are lost.

Notes:

1. Flows are greater at the master domain manager and at any *FullStatus* fault-tolerant agents in the master domain than at subordinate domain managers or *FullStatus* fault-tolerant agents.
2. Use `evtsize -show` to monitor queue sizes.
3. The amount of update flow is related to the amount of workload running in a particular subtree and is unavoidable.
4. The amount of ad hoc flow is related to the amount of additional workload at any point in the network. It can be reduced by planning more workload even if it is inactive. Simple reruns (not rerun from) do not create an ad hoc flow.

The planning, alert and recovery strategy must take into account the following points:

- Queue files are created with a fixed size and messages are added and removed in a cyclical fashion. A queue reaches capacity when the flow of incoming messages exceeds the outgoing flow for a sufficient length of time to use up the available space. For example, if messages are being added to a queue at a rate of 1MB per time unit and are being processed and removed at a rate of 0.5 MB per time unit, a queue sized at 10 MB (the default) is at capacity after 20 time units. But if the inward flow rate descends to be the same as the outward flow rate after 19 time units, the queue does not reach capacity.

- The risk of the domain manager failing can be reduced by switching to the backup domain manager. In this case, the contents of the queues on the domain manager are unavailable until the domain manager backup is started. In all cases, the size of the queue on the upper domain manager towards any other domain manager must respect condition A of Table 47.
- The risk that fault-switching fault-tolerant agents might be unable to cope with the flow must be planned beforehand. The specifications for fault-switching fault-tolerant agents must be similar to those of the domain manager, to avoid that an agent receives a load that is not appropriate for its capacity. Check if a queue is forming at the *FullStatus* fault-tolerant agents, both in ordinary and peak operation situations.
- When risk #2 has been dealt with, the possibility of a network link failure can be reduced by sizing the queue from a domain manager to the *FullStatus* fault-tolerant agents appropriately as a function of the average network outage duration, and by increasing the size of the mailbox in case of unexpected long outage (see condition B of Table 47).
- The same condition applies for avoiding an overflow of the domain manager's tomaster.msg queue with respect to network outages (see condition C) of Table 47.

Table 47. Queue sizing conditions.

A	$\text{MaxAlertTime} \leq \text{size}(\text{UpperDM}\#\text{queueToDM}) / \text{averageAdhocFlow}$
B	$\text{MaxNetOutage} \leq \text{size}(\text{DM}\#\text{queueToFSFTA}) / (\text{averageAdhocFlow} + \text{averageUpdateFlow})$
C	$\text{MaxNetOutage} \leq \text{size}(\text{DM}\#\text{queueToUpperDM}) / (\text{averageUpdateFlow})$

Monitoring the size of Tivoli Workload Scheduler message queues

You can use event-driven workload automation (EDWA) to monitor the size of message queues and to start a predefined set of actions when one or more specific events occur. For more information about event-driven workload automation, see *Tivoli Workload Scheduler: User's Guide and Reference*.

You can monitor the following message queues:

- appserverbox
- clbox
- courier
- intercom
- mailbox
- monbox
- moncmd
- pobox
- server
- tomaster

The following .XML file contains the definition of a sample event rule to monitor the mailbox queue on the specified workstation and send an e-mail when the filling percentage is greater than the specified value. If the condition described in the rule already exists when you deploy the rule, the related event is not generated. This event rule calls the MailSender action provider to send an e-mail to

Optimizing the network

the receivers that you specify. For more information about the MailSender action provider, see *Tivoli Workload Scheduler: User's Guide and Reference*:

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
  xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/event-management
  /rules http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules/EventRules.xsd">
  <eventRule name="MONITORQUEUE" ruleType="filter" isDraft="no">
    <eventCondition name="twsMesQueEvt1" eventProvider="TWSApplicationMonitor"
      eventType="TWSMessageQueues">
      <scope>
        MAILBOX FILLED UP 80% ON FTA
      </scope>
      <filteringPredicate>
        <attributeFilter name="MailboxName" operator="eq">
          <value>mailbox_name</value>
        </attributeFilter>
        <attributeFilter name="FillingPercentage" operator="ge">
          <value>filling_percentage</value>
        </attributeFilter>
        <attributeFilter name="Workstation" operator="eq">
          <value>workstation_name</value>
        </attributeFilter>
        <attributeFilter name="SampleInterval" operator="eq">
          <value>sample_interval</value>
        </attributeFilter>
      </filteringPredicate>
    </eventCondition>
    <action actionProvider="MailSender" actionType="SendMail" responseType=
    "onDetection">
      <scope>
        TWSUSER@TWS : THE MAILBOX ON workstation_name...
      </scope>
      <parameter name="To">
        <value>main_receiver_list</value>
      </parameter>
      <parameter name="Subject">
        <value>mail_subject</value>
      </parameter>
    </action>
  </eventRule>
</eventRuleSet>
```

where:

mailbox_name

Is the name of the mailbox to monitor.

filling_percentage

Is the filling percentage. The following operators are supported:

ge Causes the event generation when the mailbox filling percentage increases above the threshold value. The event is generated only the first time that the specified mailbox filling percentage is reached. If you restart the SSM agent and the filling percentage is higher than the threshold value, the event is generated again. Table 48 provides an example in which the **ge** operator is set to 70%.

Table 48. Example for the *ge* operator

Mailbox name	Filling percentage	Action
Sample (0)	>= 70%	event not generated
Sample (0)	< 70%	event not generated
Sample (n-1)	< 70%	event not generated
Sample (n)	>= 70%	event generated
Sample (n+1)	>= 70%	event not generated

le Causes the event generation when the mailbox filling percentage decreases below the threshold value. The event is generated only the first time that the specified mailbox filling percentage is reached. If you restart the SSM agent and the filling percentage is lower than the threshold value, the event is not generated until the filling percentage increases above the threshold value and then decreases below it again. Table 49 provides an example in which the **le** operator is set to 50%:

Table 49. Example for the *le* operator

Mailbox name	Filling percentage	Action
Sample (0)	<= 50%	event not generated
Sample (0)	> 50%	event not generated
Sample (n-1)	> 50%	event not generated
Sample (n)	<= 50%	event generated
Sample (n+1)	<= 50%	event not generated

workstation_name

Is the workstation on which the event is generated.

sample_interval

Is the interval, expressed in seconds, for monitoring the mailbox filling percentage.

main_receiver_list

Is the main receiver list.

mail_subject

Is the subject of the mail.

Changing a queue size

Use the **evtsize** command to resize a queue.

When you use **evtsize** to resize a queue, the queue remains at that size until the next time you use **evtsize**. It only reverts to the default size of 10MB if you delete it and Tivoli Workload Scheduler recreates it with the default size.

evtsize

evtsize:

Defines the size of the Tivoli Workload Scheduler message files. This command is used by the Tivoli Workload Scheduler administrator either to increase the size of a message file after receiving the message, "End of file on events file.", or to monitor the size of the queue of messages contained in the message file.

Authorization: You must be **maestro** or **root** in UNIX, or **Administrator** in Windows to run **evtsize**. Stop the IBM Tivoli Workload Scheduler engine before running this command.

Syntax: **evtsize -V | -U**

evtsize *file_name* *size*

evtsize -compact *file_name* [*size*]

evtsize -show *file_name*

Arguments:

-V Displays the command version and exits.

-U Displays command usage information and exits.

-compact *file_name* [*size*]

Reduces the size of the specified message file to the size occupied by the messages in the file when you run the command. You can optionally use this keyword to specify also a new file size.

-show *file_name*

Displays the size of the queue of messages contained in the message file

file_name

The name of the event file. Specify one of the following:

Courier.msg
Intercom.msg
Mailbox.msg
pobox/*workstation*.msg

size The maximum size of the event file in bytes. When first created by Tivoli Workload Scheduler, the maximum size is set to 10 MB.

Note: The size of the message file is equal to or greater than the real size of the queue of messages it contains and it progressively increases until the queue of messages becomes empty; when this occurs the message file is emptied.

Examples: To set the maximum size of the Intercom.msg file to 20 MB, run the following command:

```
evtsize Intercom.msg 20000000
```

To set the maximum size of the pobox file for workstation chicago to 15 MB, run the following command:

```
evtsize pobox\chicago.msg 15000000
```

The following command:

```
evtsize -show Intercom.msg
```

returns the following output:

```
Tivoli Workload Scheduler (UNIX)/EVTSIZE 8.3 (1.2.2.4) Licensed Materials -  
Property of IBM(R)  
5698-WSH  
(C) Copyright IBM Corp 1998, 2006 All rights reserved.  
US Government User Restricted Rights  
Use, duplication or disclosure restricted by  
GSA ADP Schedule Contract with IBM Corp.  
IBM is a registered trademark of International Business Machines  
Corporation in the United States, other countries, or both.  
AWSDEK703I Queue size current 240, maximum 10000000 bytes (read 48, write 288)
```

where:

880	Is the size of the current queue of the Intercom.msg file
10000000	Is the maximum size of the Intercom.msg file
read 48	Is the pointer position to read records
write 928	Is the pointer position to write records

Tuning mailman servers

Once the distribution of agents to mailman servers has been established, all the groups of agents attached to the same server must respect the link condition.

The link condition relates the number of agents connected to a mailman process and the tuning parameters for unlink on the mailman and writer side.

No_agents(i)

The number of agents connected to a given mailman server *i*

Mm_unlink

A parameter set in the localopts of both domain manager and agent. Specifies the maximum number of seconds mailman waits before unlinking from a workstation that is not responding.

Wr_unlink

A parameter set in the localopts of both domain manager and agent. Specifies the number of seconds the writer process waits before exiting if no incoming messages are received.

Max_down_agents

The maximum probable number of agents that are unavailable without having the ignore flag set in the database and having the autolink flag on.

Avg_connect_timeout

The system timeout (not set by Tivoli Workload Scheduler) for a TCP connection. This can be computed by the command, which times out because no process is listening on that port.

```
telnet agent inactive port
```

The condition is:

$\text{Wr_unlink} = \text{Mm_unlink} > 1.2 * \text{Max_down_agents} * \text{Avg_connect_timeout}$

This condition expresses that if the time before unlink is smaller than the probable time of idle waiting of the mailman process (waiting connect timeout for each agent that is currently down) in its loop to reactivate the connections, the agents unlink constantly when some agents are down.

Netman configuration file

The netman configuration file exists on all Tivoli Workload Scheduler workstations to define the services provided by netman. It is called `<TWA_home>/TWS/network/Netconf`. The NetConf file includes comments describing each service. The services are:

- 2001** Start a writer process to handle incoming messages from a remote mailman.
- 2002** Start the mailman process. Mailman, in turn, starts the rest of the process tree (batchman, jobman).
- 2003** Stop the Tivoli Workload Scheduler process to handle incoming messages from a remote mailman.

- 2004 Find and return a stdlist file to the requesting Conman process.
- 2005 Switch the domain manager in a domain.
- 2006 Locally download scripts scheduled by an Tivoli Workload Scheduler for z/OS[®] master domain manager.
- 2007 Required to bypass a firewall.
- 2008 Stop Tivoli Workload Scheduler workstations in a hierarchical fashion
- 2009 Runs the *switchmgr* script to stop and restart a manager in such a way that it does not open any links to other workstations until it receives the *switchmgr* event. Can only be used when the *enSwfaultTo1* global option is set to *yes*.
- 2010 Starts mailman with the parameter *demgr*. It is used by the service 2009. Can only be used when the *enSwfaultTo1* global option is set to *yes*.
- 2011 Runs **monman** as a child process (son bin/monman.exe)
- 2012 Runs **conman** to stop the event monitoring engine (command bin/conman.exe stopmon).
- 2013 Runs **conman** to switch event processors (command bin/conman.exe switchevtproc -this)
- 2014 Runs **conman** to start event processing (command bin/conman.exe startevtproc -this)
- 2015 Runs **conman** to stop event processing (command bin/conman.exe stopevtproc -this)
- 2016 Runs **conman** to force the update of the monitoring configuration file for the event monitoring engine (command bin/conman.exe deployconf)
- 2017 Runs **conman** to stop event processing on a client (client bin/conman.exe synchronizedcmd -stopevtproc)
- 2018 Runs **conman** to check event processing on a client (client bin/conman.exe synchronizedcmd -checkevtproc)
- 2021 Runs **conman** to start appservman
- 2022 Runs **conman** to run the subcommand **stopappserver** that stops the application server
- 2023 Runs **conman** to run the subcommand **startappserver** that starts the application server
- 2501 Check the status of a remote job.
- 2502 Start the Console Manager – a service requested by the client side of the Remote Console. See the *IBM Tivoli Remote Control: User's Guide* for more information.
- 2503 Used by the connector to interact with r3batch extended agent.

Determining internal Symphony table size

The mailman service (2002) can optionally take a parameter that determines the initial size of the internal Symphony table. If you do not supply this parameter, mailman calculates the initial table size based on the number of records in the file.

Note: Mailman expands the table if it needs to, even if this parameter is not supplied.

In normal circumstances, leave mailman to take the default value in the NetConf file as supplied (32000). However, if you are experiencing problems with memory, you can allocate a table that is initially smaller. To do this you change the parameter to the service 2002 in the NetConf file. The syntax for the entry is:

```
2002    son    bin/mailman [ -parm <number> ]
```

where, <number> is used to calculate the initial Symphony table size based on the number of records in the Symphony file.

If r is the number of records in the Symphony file when batchman starts, Table 50 shows how the size of the internal Symphony table is calculated, depending on the value of <number>:

Table 50. Calculation of internal Symphony table

Value of <number>	Table size
0	$(4/3r) + 512$
n	if $n > r$, n if $n \leq r$, $(4/3r) + 512$
-1	65535
$-n$	if $+n \Rightarrow r$, n if $+n < r$, $r + 512$

If during the production period you add more jobs, the maximum internal Symphony table size is increased dynamically, up to the maximum number of records allowed in the Symphony file, which is 2,000,000,000.

Extended agents

An extended agent is a logical workstation related to an access method hosted by a physical Tivoli Workload Scheduler workstation (not another extended agent). More than one extended agent workstation can be hosted by the same Tivoli Workload Scheduler workstation and use the same access method. The extended agent is defined using a standard Tivoli Workload Scheduler workstation definition, which gives the extended agent a name and identifies the access method. The access method is a program that is run by the hosting workstation whenever Tivoli Workload Scheduler submits a job to an external system.

Jobs are defined for an extended agent in the same manner as for other Tivoli Workload Scheduler workstations, except that job attributes are dictated by the external system or application.

Extended agent software is available for several systems and applications. The UNIX extended agents, included with Tivoli Workload Scheduler, are described in the following section. Please contact your sales representative for information

about other extended agents. For information on defining Tivoli Workload Scheduler workstations, see the section that explains how to manage workstations in the database in the *Tivoli Workload Scheduler: Job Scheduling Console User's Guide*. For information on writing access methods, see the *Tivoli Workload Scheduler: User's Guide and Reference*.

UNIX extended agents

Tivoli Workload Scheduler includes access methods for two types of UNIX extended agents. Use the Local UNIX method to enable a single UNIX workstation to operate as two Tivoli Workload Scheduler workstations, both of which can run Tivoli Workload Scheduler scheduled jobs. Use the Remote UNIX access method to designate a remote UNIX workstation to run Tivoli Workload Scheduler scheduled jobs without having Tivoli Workload Scheduler installed on it.

Information about a job's execution is sent to Tivoli Workload Scheduler from an extended agent using the job's `stdlist` file. A method options file can specify alternate logins to launch jobs and check *opens* file dependencies. For more information, see the *Tivoli Workload Scheduler: User's Guide and Reference*.

Local UNIX access method

The Local UNIX method can be used to define multiple Tivoli Workload Scheduler workstations on one workstation: the host workstation and one or more extended agents. When Tivoli Workload Scheduler sends a job to a local UNIX extended agent, the access method, `unixlocl`, is invoked by the host to run the job. The method starts by running the standard configuration script on the host workstation (`<TWA_home>/TWS/jobmanrc`). If the job's logon user is permitted to use a local configuration script and the script exists as `$HOME/TWS/.jobmanrc`, the local configuration script is also run. The job itself is then run either by the standard or the local configuration script. If neither configuration script exists, the method starts the job.

The launching of the configuration scripts, `jobmanrc` and `.jobmanrc` is configurable in the method script. The method runs the configuration scripts by default, if they exist. To disable this feature, you must comment out a set of lines in the method script. For more information, examine the script file `<TWA_home>/TWS/methods/unixlocl` on the extended agent's host.

Remote UNIX access method

The Remote UNIX access method can be used to designate a non-Tivoli Workload Scheduler workstation to run jobs scheduled by Tivoli Workload Scheduler. You can use `unixrsh` or `unixssh`:

The `unixrsh` method

When Tivoli Workload Scheduler sends a job to a remote UNIX extended agent, the access method, `unixrsh`, creates a `/tmp/maestro` directory on the non-Tivoli Workload Scheduler workstation. It then transfers a wrapper script to the directory and runs it. The wrapper then runs the scheduled job. The wrapper is created only once, unless it is deleted, moved, or is outdated.

To run jobs using the extended agent, the job logon users must be given appropriate access on the non-Tivoli Workload Scheduler UNIX workstation. To do this, a `.rhost`, `/etc/host.equiv`, or equivalent file must be set up on the workstation. If *opens* file dependencies are to be checked, *root* access must also be permitted. Contact your system administrator for help. For more information about the access method, examine the script file `<TWA_home>/TWS/methods/unixrsh` on an extended agent's host.

The unixssh method

The `unixssh` method works like `unixrsh` but uses a secure remote shell to connect to the remote host. The files used by this method are:

```
methods/unixssh  
methods/unixssh.wrp
```

The `unixssh` method uses the `ssh` keyword. You can generate this keyword with any tools that are compatible with the secure remote shell.

Note: The passphrase must be blank.

The following scenario gives an example of how to set up the method:

You have installed a Tivoli Workload Scheduler, version 8.4 fault-tolerant agent with the `<TWS_user>`: `tw830`. You want to run a remote shell in the remote host "REMOTE_HOST" with the user "guest". The procedure is as follows:

1. Create the public and private key for the user `tw830`, The following is an example using `rsa`:

- a. Log on as `tw830`

- b. Run

```
ssh-keygen -t rsa
```

- c. When the tool asks for the passphrase, press Enter (leaving the passphrase blank.) The keys are saved as follows:

Key	Location	Comment
Public	<code><TWA_home>/TWS/.ssh/id_rsa.pub</code>	
Private	<code><TWA_home>/TWS/.ssh/id_rsa</code>	Do not send this file!

Note: Different tools store the key in different places.

2. At the remote host, do the following:

- a. Telnet to the remote host.

- b. Log on as "guest".

- c. Change to the `.ssh` directory in the user's home directory, or create it if it does not exist (the directory permissions must be adequate: for example, 600 for the directory and 700 for its contents.).

- d. Append the *public* key you created in step 1. to the `authorized_keys` file (create the file if it does not exist), using the command:

```
cat id_rsa.pub >> authorized_keys
```

3. At the fault-tolerant agent, make the remote host "known" before attempting to let Tivoli Workload Scheduler processes use the connection. This can be achieved in one of two ways:

- Log on as `tw830` and connect to the host using the command:

```
ssh -l guest <remote_host_name> ls
```

A prompt will be displayed saying that the host is not known, and asking permission to access it. Give permission, and the host will be added to the list of known hosts.

- Alternatively, use the `ssh` documentation to add the remote host to the file of known hosts.

Managing production for extended agents

In general, jobs that run on extended agents behave like other Tivoli Workload Scheduler jobs. Tivoli Workload Scheduler tracks a job's status and records output in the job's `stdlist` files. These files are stored on the extended agent's *host* workstation. For more information on managing jobs, see the section that describes Tivoli Workload Scheduler plan tasks in the *Tivoli Workload Scheduler: Job Scheduling Console User's Guide*.

Failure launching jobs on an extended-agent

If the access method is not located in the proper directory on the extended agent's host, or the method cannot be accessed by Tivoli Workload Scheduler, jobs fail to launch or a file dependency is not checked. For a job, the Tivoli Workload Scheduler job's logon or the logon specified in the method options file must have read and execute permissions for the access method. When checking a file to satisfy an *opens* dependency, root is used as the login unless another login is specified in the method options file. For more information on method options, see the *Tivoli Workload Scheduler: User's Guide and Reference*.

IP address validation

When a TCP/IP connection is established, netman reads the requester's node name and IP address from the socket. The IP address and node name are used to search the Symphony file for a known Tivoli Workload Scheduler workstation with one of the following possible results:

- If an IP address match is found the validation is considered successful.
- If a node name match is found, the validation is considered successful.
- If no match is found in the Symphony file or the IP address returned does not match the one read from the socket, the validation is considered unsuccessful.

The local option, `nm ipvalidate`, determines the action to be taken if IP validation is unsuccessful. If the option is set to `full`, unsuccessful validation causes Tivoli Workload Scheduler to close the connection and generate an error message. If the option is set to `none` (default), Tivoli Workload Scheduler permits all connections, but generates a warning message for unsuccessful validation checks.

Support for Internet Protocol version 6

Tivoli Workload Scheduler supports Internet Protocol version 6 (IPv6) in addition to the legacy IPv4. To assist customers in staging the transition from an IPv4 environment to a complete IPv6 environment, Tivoli Workload Scheduler provides IP dual-stack support. In other terms, the product is designed to communicate using both IPv4 and IPv6 protocols simultaneously with other applications using IPv4 or IPv6.

To this end, the IPv4-specific `gethostbyname` and `gethostbyaddr` functions have been replaced by the new `getaddrinfo` API that makes the client-server mechanism entirely protocol independent.

The `getaddrinfo` function handles both name-to-address and service-to-port translation, and returns `sockaddr` structures instead of a list of addresses. These `sockaddr` structures can then be used by the socket functions directly. In this way, `getaddrinfo` hides all the protocol dependencies in the library function, which is where they belong. The application deals only with the socket address structures that are filled in by `getaddrinfo`.

Operating system configuration (UNIX only)

IP validation depends on the system call `getaddrinfo()` to look up all the valid addresses for a host. The behavior of this routine varies, depending on the system configuration. When `getaddrinfo()` uses the file `/etc/hosts`, it returns the first matching entry. If the connection is initiated on an address which appears after the first matching entry, IP validation fails. To resolve the problem, place the entry used to initiate the connection before any other matching entries in the `/etc/hosts` file. If `getaddrinfo()` uses the "named" name server or the Network Information Service server and `getaddrinfo()` fails, contact your system administrator for assistance.

IP address validation messages

Following is a list of the messages for IP validation. If the Local Option `nm ipvalidate` is set to `none` (default), the errors appear as warnings.

See the end of the list of conditions for the key to the variables:

- Tivoli Workload Scheduler workstation name is not found in the Symphony file

```
Ip address validation failed for request:
Service <num> for <program> on <workstation>(<operating_system_type>).
Connection received from IP address:
<c_ipaddr>. MAESTRO CPU <workstation> not found in
Symphony file.
```

- Call to `getaddrinfo()` fails:

```
IP address validation failed for request:
Service num for <program> on cpu(<operating_system_type>).
Connection received from IP address:
<c_ipaddr>. getaddrinfo() failed, unable to
retrieve IP address of connecting node: <node>.
```

- IP Addresses returned by `getaddrinfo()` do not match the IP address of connection workstation:

```
IP address validation failed for request:
Service <num> for <program> on <workstation>(<operating_system_type>).
Connection received from IP address:
<c_ipaddr>. System known IP addresses for node
name node: <k_ipaddr>.
```

- The IP address specified in the workstation definition for the Tivoli Workload Scheduler workstation indicated in the service request packet does not match the IP address of connecting workstation:

```
IP address validation failed for request:
Service <num> for <program> on <workstation>(<operating_system_type>).
Connection received from IP address:
<c_ipaddr>. TWS known IP addresses for cpu
<k_ipaddr>.
```

- Regardless of the state of `nm ipvalidate`, the following information message is displayed when IP validation cannot be performed because the Symphony file does not exist or an error occurs when reading it:

```
IP address validation not performed for
request: Service <num> for <program> on
<workstation>(<operating_system_type>). Connection received from IP
address: <c_ipaddr>. Cannot open or read
Symphony file. Service request accepted.
```

Where:

<num>	Service number (2001- writer , 2002- mailman ...)
<program>	Program requesting service
<workstation>	Tivoli Workload Scheduler workstation name of connecting workstation
<operating_system_type>	Operating system of connecting workstation
<node>	Node name or IP address of connecting workstation
<c_ipaddr>	IP address of connecting workstation
<k_ipaddr>	Known IP address for connecting workstation

IP validation is always successful in the absence of a Symphony file. In communications from a domain manager to an agent it is normally successful because a Symphony file does not yet exist. However, if the agent has a Symphony file from a previous run, the initial link request might fail if the Symphony file does not include the name of the domain manager.

Impact of network changes

Any changes that you make to your network might have an impact on Tivoli Workload Scheduler. Workstations can be identified within Tivoli Workload Scheduler by host name or IP address. Any changes to host names or IP addresses of specific workstations must obviously be also implemented in the Tivoli Workload Scheduler database. However, remember that if those workstations are involved in jobs that are currently scheduled in the Symphony file, those jobs are looking for the old workstation identity.

Changes to host names or IP addresses of specific workstations can be activated immediately by running **JnextPlan -for 0000**. A new production plan is created (containing the updated IP addresses and host names), but the plan time span is not extended.

Thus, plan any network changes with the job schedules in mind, and for major changes you are advised to suspend Tivoli Workload Scheduler activities until the changes have been completed in the network and also implemented in the Tivoli Workload Scheduler database.

Network changes also have a specific impact on the connection parameters used by the application server and the command line client:

Application server

If you change the network you will need to change the communication parameters specified in the application server's configuration files. How to do this is described in the appendix on the utilities supplied with the embedded WebSphere Application Server in the *Tivoli Workload Scheduler: Planning and Installation Guide*.

Impact of network changes

Command line client

When you connect from the command line client you supply a set of connection parameters. This is done in one of these ways:

From the `localopts` file

The default method is that the connection parameters in the `localopts` file are customized when the command line client is installed.

From the `useropts` file

A `useropts` file might have been created for the user in question, containing a version of the connection parameters personalized for the user.

In the command line, individually

When you invoke one of the command line programs, you can optionally include the parameters as arguments to the command. These override the values in the `localopts` or `useropts` files.

In the command line, in a file

When you invoke one of the command line programs, you can optionally include the parameters in a file, the name of which is identified as the `-file` argument to the command. These override the values in the `localopts` or `useropts` files.

Modify whichever method you are using to incorporate the new network connection details.

Chapter 7. Setting connection security

This chapter describes connection security. It is divided into the following sections:

- “Connection security overview”
- “Using an SSL connection for communication across the network by netman and conman”
- “Interface communication” on page 178
- “Working across firewalls” on page 186
- “LDAP connection authentication” on page 187
- “Configuring the Job Scheduling Console to work in a Network Address Translation (NAT) environment” on page 187
- “Federal Information Processing Standards” on page 188

Connection security overview

Tivoli Workload Scheduler uses the following protocols:

- SSL for communication across the network by **netman** and **conman**
- HTTP/HTTPS over SSL for the command-line client
- RMI/IIOP over SSL for the Tivoli Dynamic Workload Console and the Job Scheduling Console

The Tivoli Workload Scheduler password encryption algorithm uses 3DES, which is also known as Triple-DES, or DES-FDE3. The algorithm is run in Cipher Block Chaining Mode, which uses padding according to the PKCS#5 standard.

The product is installed with default settings that you customize according to your security requirements.

Before you perform any customization of the SSL connections, stop the WebSphere Application server. Depending on the security settings you implement in your network, you will need to reconfigure the security settings of the WebSphere Application Server. This is described generically in “Changing the security settings” on page 258, and the detail of the procedure described therein is not repeated in this chapter.

Using an SSL connection for communication across the network by netman and conman

Tivoli Workload Scheduler provides a secure, authenticated, and encrypted connection mechanism for communication across the network topology. This mechanism is based on the Secure Sockets Layer (SSL) protocol and uses the OpenSSL Toolkit, which is automatically installed with Tivoli Workload Scheduler.

The SSL protocol is based on a private and public key methodology. SSL provides the following authentication methods:

CA trusting only

Two workstations trust each other if each receives from the other a certificate that is signed or is trusted. That is, if the CA certificate is in the list of trusted CAs on each workstation. With this authentication level, a workstation does not perform any additional checks on certificate content, such as the distinguished name. Any signed or trusted certificate can be used to establish an SSL session. See “Setting local options” on page 20 for a definition of the **caonly** option.

Check if the distinguished name matches a defined string

Two workstations trust each other if, after receiving a trusted or signed certificate, each performs a further check by extracting the distinguished name from the certificate and comparing it with a string that was defined in its local options file. See “Setting local options” on page 20 for a definition of the **string** option.

Check if the distinguished name matches the workstation name

Two workstations trust each other if, after receiving a signed or trusted certificate, each performs a further check by extracting the distinguished name from the certificate and comparing it with the name of the workstation that sent the certificate. See “Setting local options” on page 20 for a definition of the **cpu** option.

To provide SSL security for a domain manager attached to z/OS in an end-to-end connection, configure the OS/390[®] Cryptographic Services System SSL in the Tivoli Workload Scheduler code that runs in the OS/390 USS UNIX shell in the IBM Tivoli Workload Scheduler for z/OS server address space. See the Tivoli Workload Scheduler z/OS documentation.

When configuring SSL you can:

Use the same certificate for the entire network

If the workstations are configured with CA trusting only, they accept connections with any other workstation that sends a signed or trusted certificate. To enforce the authentication you define a name or a list of names that must match the contents of the certificate distinguished name (DN) field in the `localopts` file of each workstation.

Use a certificate for each domain

Install private keys and signed certificates for each domain in the network. Then, configure each workstation to accept a connection only with partners that have a particular string of the certificate DN field in the `localopts` file of each workstation.

Use a certificate for each workstation

Install a different key and a signed certificate on each workstation and add a Trusted CA list containing the CA that signed the certificate. Then, configure each workstation to accept a connection only with partners that have their workstation name specified in the Symphony file recorded in the DN field of the certificate.

Setting up private keys and certificates

To use SSL authentication on a workstation, you need to create and install the following:

- The private key and the corresponding certificate that identify the workstation in an SSL session.

- The list of certificate authorities that can be trusted by the workstation.

Use the **openssl** command line utility to:

- Create a file containing pseudo random generated bytes (TWS.rnd). This file is needed on some operating systems for SSL to function correctly.
- Create a private key.
- Save the password you used to create the key into a file.
- Create a Certificate Signing Request.
- Send this Certificate Signing Request (CSR) to a Certifying Authority (CA) for signing, or:
 - Create your own Certificate Authority (CA)
 - Create a self-signed CA Certificate (X.509 structure) with the RSA key of your own CA
 - Use your own Certificate Authority (CA) to sign and create real certificates

These actions will produce the following files that you will install on the workstation(s):

- A private key file (for example, TWS.key). This file should be protected, so that it is not stolen to use the workstation's identity. You should save it in a directory that allows read access to the TWS user of the workstation, such as *TWA_home/TWS/ssl/TWS.key*.
- The corresponding certificate file (for example, TWS.crt). You should save it in a directory that allows read access to the TWS user of the workstation, such as *TWA_home/TWS/ssl/TWS.crt*.
- A file containing a pseudo-random generated sequence of bytes. You can save it in any directory that allows read access to the TWS user of the workstation, such as *TWA_home/TWS/ssl/TWS.rnd*.

In addition, you should create the following:

- A file containing the password used to encrypt the private key. You should save it in a directory that allows read access to the TWS user of the workstation, such as *TWA_home/TWS/ssl/TWS.sth*.
- The certificate chain file. It contains the concatenation of the PEM-encoded certificates of certification authorities which form the certificate chain of the workstation's certificate. This starts with the issuing CA certificate of the workstation's certificate and can range up to the root CA certificate. Such a file is simply the concatenation of the various PEM-encoded CA certificate files, usually in certificate chain order.
- The trusted CAs file. It contains the trusted CA certificates to use during authentication. The CAs in this file are also used to build the list of acceptable client CAs passed to the client when the server side of the connection requests a client certificate. This file is simply the concatenation of the various PEM-encoded CA certificate files, in order of preference.

Creating Your Own Certification Authority

If you are going to use SSL authentication within your company's boundaries and not for outside internet commerce, you might find it simpler to create your own certification authority (CA) to trust all your IBM Tivoli Workload Scheduler installations. To do so, follow the steps listed below.

Using SSL for netman and conman

Note: In the following steps, the names of the files created during the process TWS and TWSca are sample names. You can use your own names, but keep the same file extensions.

1. Choose a workstation as your CA root installation.
2. Type the following command from the SSL directory to initialize the pseudo random number generator, otherwise subsequent commands might not work.
 - On UNIX:

```
$ openssl rand -out TWS.rnd -rand ./openssl 8192
```
 - On Windows:

```
$ openssl rand -out TWS.rnd -rand ./openssl.exe 8192
```
3. Type the following command to create the CA private key:

```
$ openssl genrsa -out TWSca.key 1024
```
4. Type the following command to create a self-signed CA Certificate (X.509 structure):

```
$ openssl req -new -x509 -days 365 -key TWSca.key -out TWSca.crt -config ./openssl.cnf
```

Now you have a certification authority that you can use to trust all of your installations. If you wish, you can create more than one CA.

Creating private keys and certificates

The following steps explain how to create one key and one certificate. You can decide to use one key and certificate pair for the entire network, one for each domain, or one for each workstation. The steps below assume that you will be creating a key and certificate pair for each workstation and thus the name of the output files created during the process has been generalized to *workstationname*.

On each workstation, perform the following steps to create a private key and a certificate:

1. Enter the following command from the SSL directory to initialize the pseudo random number generator, otherwise subsequent commands might not work.
 - On Windows operating systems:

```
$ openssl rand -out workstationname.rnd -rand ./openssl.exe 8192
```
 - On UNIX and Linux operating systems :

```
$ openssl rand -out workstationname.rnd -rand ./openssl 8192
```
2. Enter the following command to create the private key (this example shows triple-DES encryption):

```
$ openssl genrsa -des3 -out workstationname.key 1024
```

Then, save the password that was requested to encrypt the key in a file named *workstationname.pwd*.

Note: Verify that file *workstationname.pwd* contains just the characters in the password. For instance, if you specified the word *maestro* as the password, your *workstationname.pwd* file should not contain any CR or LF characters at the end (it should be 7 bytes long).
3. Enter the following command to save your password, encoding it in base64 into the appropriate stash file:

```
$ openssl base64 -in workstationname.pwd -out workstationname.sth
```

You can then delete file *workstationname.pwd*.
4. Enter the following command to create a certificate signing request (CSR):

```
$ openssl req -new -key workstationname.key -out workstationname.csr -config ./openssl.cnf
```

Some values—such as company name, personal name, and more—will be requested at screen. For future compatibility, you might specify the workstation name as the distinguished name.

- Send the *workstationname.csr* file to your CA in order to get the matching certificate for this private key.

Using its private key (*TWSca.key*) and certificate (*TWSca.crt*), the CA will sign the CSR (*workstationname.csr*) and create a signed certificate (*workstationname.crt*) with the following command:

```
$ openssl x509 -req -CA TWSca.crt -CAkey TWSca.key -days 365
  -in workstationname.csr -out workstationname.crt -CAcreateserial
```

- Distribute to the workstation the new certificate *workstationname.crt* and the public CA certificate *TWSca.crt*.

The table below summarizes which of the files created during the process have to be set as values for the workstation's local options.

Table 51. Files for Local Options

Local option	File
SSL key	<i>workstationname.key</i>
SSL certificate	<i>workstationname.crt</i>
SSL key pwd	<i>workstationname.sth</i>
SSL ca certificate	<i>TWSca.crt</i>
SSL random seed	<i>workstationname.rnd</i>

Configuring SSL attributes

Use the **composer** command line, the Tivoli Dynamic Workload Console, or the Job Scheduling Console to update the workstation definition in the database. See the *Tivoli Workload Scheduler: User's Guide and Reference* or the *Tivoli Workload Scheduler: Job Scheduling Console User's Guide* for further reference.

Configure the following attributes:

secureaddr

Defines the port used to listen for incoming SSL connections. This value must match the one defined in the **nm SSL port** local option of the workstation. It must be different from the **nm port** local option that defines the port used for normal communications. If **securitylevel** is specified but this attribute is missing, 31113 is used as the default value.

securitylevel

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

enabled

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

on

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

Using SSL for netman and conman

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

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

Table 52. Type of communication depending on the securitylevel value

Fault-tolerant agent (domain manager)	Domain manager (parent domain manager)	Connection type
-	-	TCP/IP
Enabled	-	TCP/IP
On	-	No connection
Force	-	No connection
-	On	TCP/IP
Enabled	On	TCP/IP
On	On	SSL
Force	On	SSL
-	Enabled	TCP/IP
Enabled	Enabled	TCP/IP
On	Enabled	SSL
Force	Enabled	SSL
-	Force	No connection
Enabled	Force	SSL
On	Force	SSL
Force	Force	SSL

The following example shows a workstation definition that includes the security attributes:

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

Configuring the SSL connection protocol for the network

To configure SSL for your network, perform the following steps:

1. Create an SSL directory under the *TWA_home/TWS* directory. By default, the path *TWA_home/TWS/ssl* is registered in the *localopts* file. If you create a directory with a name different from *ssl* in the *TWA_home/TWS* directory, then update the *localopts* file accordingly.

2. Copy openssl.cnf and openssl.exe to the SSL directory
3. Create as many private keys, certificates, and Trusted CA lists as you plan to use in your network.
4. For each workstation that will use SSL authentication:
 - Update its definition in the Tivoli Workload Scheduler database with the SSL attributes.
 - Add the SSL local options in the localopts file.

Although you are not required to follow a particular sequence, these tasks must all be completed to activate SSL support.

In Tivoli Workload Scheduler, SSL support is available for the fault-tolerant agents only (including the master domain manager and the domain managers), but not for the extended agents. If you want to use SSL authentication for a workstation that runs an extended agent, you must specify this parameter in the definition of the host workstation of the extended agent.

Configuring full SSL security

This section describes how to implement full SSL security when using an SSL connection for communication across the network by **netman** and **conman**. It contains the following topics:

- “Overview”
- “Setting up full SSL security” on page 176
- “Migrating a network to full SSL connection security” on page 176
- “Configuring full SSL support for internetwork dependencies” on page 177

Overview

This feature provides the option to set a higher degree of SSL-based connection security on Tivoli Workload Scheduler networks in addition to the already available level of SSL security.

If you require a more complete degree of SSL protection, this enhancement supplies new configuration options to setup advanced connection security.

If you do not require more SSL security than Tivoli Workload Scheduler has provided prior to the release of this feature, you can use the standard settings documented above in this chapter.

The Full SSL security enhancements: Full SSL security support provides the following enhancements:

- TCP ports that can become security breaches are no longer left open.
- Travelling data, including communication headers and trailers, is now *totally* encrypted.

Compatibility between SSL support levels: The non-full and the full SSL support levels are mutually exclusive. That is, they cannot be configured simultaneously and cannot be enabled at the same time. If you enable full SSL support for a Tivoli Workload Scheduler network, any connection attempts by agents that are not configured for full SSL will be rejected by agents with full SSL support enabled. Vice-versa, agents configured for full SSL support cannot communicate with the rest of a network set up for non-full SSL support.

Configuring full SSL security

Setting up full SSL security

To set full SSL connection security for your network, you must, *in addition to all the steps described above in Chapter 7, "Setting connection security," on page 169*) configure the following options:

enSSLFullConnection (or sf)

Use `optman` on the master domain manager to set this global option to Yes to enable full SSL support for the network.

nm SSL full port

Edit the `localopts` file on every agent of the network (including the master domain manager) to set this local option to the port number used to listen for incoming SSL connections. Take note of the following:

- This port number is to be defined also for the `SECUREADDR` parameter in the workstation definition of the agent.
- In a full SSL security setup, the `nm SSL port` and `nm port` local options are to be set to zero.
- You must stop `netman` (`conman shut;wait`) and restart it (`StartUp`) after making the changes in `localopts`.
- Check that the `securitylevel` parameter in the workstation definition of each workstation using SSL is set at least to *enabled*.

Other than the changed value for `secureaddr`, no other changes are required in the workstation definitions to set up this feature.

Migrating a network to full SSL connection security

Run the following steps to migrate your Tivoli Workload Scheduler version 8.3 production environment to full SSL connection security support. The scenario assumes that the network already runs on non-full SSL; that is, that the master and all the agents have:

- The `securitylevel` attribute set to *enabled*, *on*, or *force* in their workstation definition. On the master it is set to *enabled*.
- Either the `nm port` or the `nm SSL port` local option configured and the port number set as the value of the `secureaddr` attribute in their workstation definition.
- Group or individual private keys and certificates.

Proceed as follows:

1. Upgrade all the agents. The objective is to upgrade locally every agent in the network (including the master domain manager). You can perform this step over several days. On the master and on every agent:
 - a. Install the fix containing the full SSL support feature.
 - b. Add the `nm SSL full port` local option and set it to a port number.At this stage, the network is still operating on non-full SSL connection security.
2. Enable full SSL support in the network. Perform this step in one single time slot. To do this:
 - a. Check that no firewall blocks the connection between the agents and their domain manager (and, optionally, the master domain manager).
 - b. In the workstation definition of the master and of every agent set the value of the `secureaddr` attribute to the port number you configured for the `nm SSL full port` local option.
 - c. Use `Optman` to set the `enSSLFullConnection` global option to *yes* in the database.

d. Run `JnextPlan -for 0000` to make these settings operational.

At this stage, the network is operating on full SSL connection security. Any agents left on SSL security can no longer communicate with the rest of the full SSL security network.

The upgraded workstations still have the old SSL and TCP ports open in listening mode. The aim of the final step is to close them down.

3. Disable the old SSL and TCP ports on the master and on every agent. You can perform this step over several days. To do this, edit the local options file of every workstation as follows:

- On the workstations that have the `securitylevel` attribute set to `enabled` or `on`, set the `nm SSL port` local option to 0.
- On the workstations that have the `securitylevel` attribute set to `force`, set both `nm port` and `nm SSL port` local options to 0.

At this stage, all the agents operate with the new SSL connections and all agents set on `securitylevel=force` listen only on the new SSL full port. From now on:

- No bytes are sent in clear.
- No active services are left in clear.
- No TCP ports are left in listening mode on agents with `securitylevel=force`.

Configuring full SSL support for internetwork dependencies

The network agent that resolves internetwork dependencies requires a particular setup for full SSL support.

To enable a network agent for full SSL support:

1. Configure both the hosting and the remote fault-tolerant agents for full SSL support.
2. On the hosting fault-tolerant agent copy or move the `netmth.opts` file from the `TWA_home/TWS/config` to the `TWA_home/TWS/methods` directories and add (and configure) the following options:

SSL remote CPU

The workstation name of the remote master or fault-tolerant agent.

SSL remote full port

The port number defined for full SSL support on the remote master or fault-tolerant agent.

The local options that specify the private key and certificate on the hosting fault-tolerant agent

These are documented in the “Setting local options” on page 20).

Note that if the hosting fault-tolerant agent hosts more than one network agent, the `TWA_home/TWS/methods` directory contains one `netmth.opts` file for every defined network agent. In this case the complete name of each `netmth.opts` file must become:

`network-agent-name_netmth.opts`

If the `TWA_home/TWS/methods` directory contains both `network-agent-name_netmth.opts` and `netmth.opts` files, only `network-agent-name_netmth.opts` is used. If multiple agents are defined and the directory contains only `netmth.opts`, this file is used for all the network agents.

The following example adds full SSL support to the example described in *A sample network agent definition* in the *Tivoli Workload Scheduler: User's Guide and Reference*:

Configuring full SSL security

- This is the workstation definition for the NETAGT network agent:

```
CPUNAME NETAGT
DESCRIPTION "NETWORK AGENT"
OS OTHER
NODE MASTERA.ROME.TIVOLI.COM
TCPADDR 31117
FOR maestro
HOST MASTERB
ACCESS NETMTH
END
```

- These are the full SSL security options in the netmeth.opts file of NETAGT:

```
#####
# Remote cpu parameters
#####

SSL remote full port = 31119
SSL remote CPU = MASTERA

#####
# Configuration Certificate
#####

SSL key                = "C:\TWS\installations\SSL\XA.key"
SSL certificate        = "C:\TWS\installations\SSL\XA.crt"
SSL CA certificate     = "C:\TWS\installations\SSL\VeriSte.crt"
SSL key pwd           = "C:\TWS\installations\SSL\XA.sth"
SSL certificate chain  = "C:\TWS\installations\SSL\TWSCertificateChain.crt"
SSL random seed       = "C:\TWS\installations\SSL\random_file.rnd"
SSL auth mode         = cpu
SSL auth string       = tws
```

Note: The SSL configuration certificate options must refer to the private key and certificate defined on the hosting fault-tolerant agent.

- This is the workstation definition for MASTERA (the remote workstation):

```
CPUNAME MASTERA
OS WNT
NODE 9.168.68.55 TCPADDR 31117
SECUREADDR 31119
DOMAIN NTWKA
FOR MAESTRO
TYPE MANAGER
AUTOLINK ON
BEHINDFIREWALL OFF
SECURITYLEVEL enabled
FULLSTATUS ON
SERVER H
END
```

Interface communication

This section describes how to configure SSL communication for the Tivoli Workload Scheduler interfaces. It has the following topics:

- “Overview” on page 179
- “Customizing the connector configuration files” on page 180
- “Changing a server key” on page 182
- “Customizing the SSL connection for the Job Scheduling Console” on page 183
- “Customizing the SSL connection for a command-line client” on page 184
- “Command-line SSL connection settings” on page 186

Overview

The following interfaces use SSL encryption and server authentication to communicate:

Tivoli Dynamic Workload Console and SOAP internal communication on master domain manager

The Tivoli Dynamic Workload Console and the SOAP internal communication on the master domain manager use RMI/IIOP over SSL. They use both the server and client security features of WebSphere Application Server to implement it.

Job Scheduling Console

The Job Scheduling Console also uses RMI/IIOP over SSL, but only the client security feature of WebSphere Application Server is used.

The command-line client

The command-line client communicates using HTTP/HTTPS over SSL, and does not use WebSphere Application Server

The command line

The command-line on the master domain manager communicates using HTTP/HTTPS over SSL, and does not use WebSphere Application Server

The Tivoli Workload Scheduler interfaces use default certificates that are installed into default keystores. To set SSL, you customize the defaults to the required security level. The default certificates are not used for the Tivoli Dynamic Workload Console or Job Scheduling Console client authentication, which is obtained using a user ID and password. The default passwords associated with each of the default keystores is **default**.

The SSL security paradigm implemented in the WebSphere Application Server requires two stores to be present on the clients and the server: a keystore containing the private key and a trust store containing the certificates of the trusted counterparts.

Figure 6 on page 180 shows the server and client keys, and to where they must be exported for the Tivoli Dynamic Workload Console and the Job Scheduling Console:

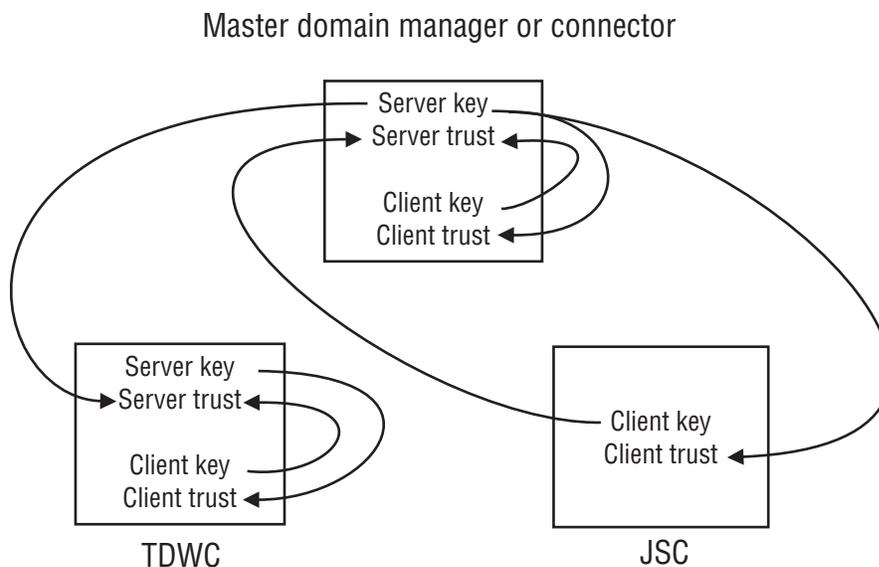


Figure 6. SSL server and client keys

The diagram shows the keys that must be extracted and distributed to enable SSL between the master domain manager, the Tivoli Dynamic Workload Console and the Job Scheduling Console, and within the master domain manager for the internal SOAP connection. Each arrow in the diagram includes the following activities performed using an appropriate key management tool on each keystore:

1. Create a self-signed certificate or import a third party certificate
2. Extract a new key
3. Open the appropriate trust store
4. Use the new key to add a signed certificate to the trust store

In addition to creating a new key, for some of these key and trust stores you can also customize the name, location, and password of the store. Table 53 details the possibilities:

Table 53. Changes allowed in Tivoli Workload Scheduler key and trust stores

File	Name	Path	Password	New key
TWS server key store	✓	✓	✓	✓
TWS server trust store	✓	✓	✓	✓
TWS client key store				✓
TWS client trust store				✓
JSC client key store				✓
JSC client trust store				✓

The following sections tell you how to customize the configuration files for the key and trust stores (where you need to make changes), and how to create a new key.

Customizing the connector configuration files

To make any changes to the name, location, password of the Tivoli Workload Scheduler server key or trust stores, you must modify the configuration files which describe them. The configuration files are as follows:

Configuration files for server key files

The name and location of the server key and trust stores are contained in the `security.xml` file located in the `TWA_home/eWAS/profiles/twaprofile/config/cells/DefaultNode/` directory. Alternatively, use the `showSecurityProperties` tool.

The default names and locations are: `TWSServerKeyFile.jks` and `TWSServerTrustFile.jks`, located in `TWA_home/eWAS/profiles/twaprofile/etc`

The following is an example of the `security.xml` that contains those settings:

```
<repertoire xmi:id="SSLConfig_1" alias="DefaultNode/DefaultSSLSettings">
  <setting xmi:id="SecureSocketLayer_1"
    keyFileName="/opt/ibm/TWA0/eWAS/profiles/twaprofile/etc/
                TWSServerKeyFile.jks"
    keyFilePassword="{xor}Ozo5PiozKw==" keyFileFormat="JKS"
    trustFileName="/opt/ibm/TWA0/eWAS/profiles/twaprofile/etc/
                TWSServerTrustFile.jks"
    trustFilePassword="{xor}Ozo5PiozKw==" trustFileFormat="JKS"
    clientAuthentication="false" securityLevel="HIGH"
    enableCryptoHardwareSupport="false">
  <cryptoHardware xmi:id="CryptoHardwareToken_1"
    tokenType="" libraryFile="" password="{xor}"/>
  <properties xmi:id="Property_6" name="com.ibm.ssl.protocol" value="SSL"/>
  <properties xmi:id="Property_7" name="com.ibm.ssl.contextProvider"
    value="IBMJSSE2"/>
</setting>
</repertoire>
```

Client key files for all components other than the Job Scheduling Console

The client key files are described in the file: `TWA_home/eWAS/profiles/twaprofile/properties/ssl.client.props`

The information in this file must not be modified. An example of it is as follows:

```
# KeyStore information
com.ibm.ssl.keyStoreName=ClientDefaultKeyStore
com.ibm.ssl.keyStore=/opt/ibm/TWA0/eWAS/profiles/twaprofile/etc/
                TWSCClientKeyFile.jks
com.ibm.ssl.keyStorePassword={xor}Ozo5PiozKw\=\=
com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.keyStoreProvider=IBMJCE
com.ibm.ssl.keyStoreFileBased=true

# TrustStore information
com.ibm.ssl.trustStoreName=ClientDefaultTrustStore
com.ibm.ssl.trustStore=/opt/ibm/TWA0/eWAS/profiles/twaprofile/etc/
                TWSCClientTrustFile.jks
com.ibm.ssl.trustStorePassword={xor}Ozo5PiozKw\=\=
com.ibm.ssl.trustStoreType=JKS
com.ibm.ssl.trustStoreProvider=IBMJCE
com.ibm.ssl.trustStoreFileBased=true
```

Client key files for the Job Scheduling Console

The default files are `JSCDefaultKeyFile.jks` and `JSCDefaultTrustFile.jks` located in `JSC_install_dir/keys`

To modify the server key file names, paths, or passwords, modify the configuration files using the script `changeSecurityProperties` located in the `TWA_home/TWS/wastool` directory. For instructions on how to do this see “Changing the security settings” on page 258. The following is a sample of the input:

Interface communication

```
#####  
SSL Panel  
#####  
alias=DefaultSSLSettings  
keyFileName=${USER_INSTALL_ROOT}/etc/TWSServerKeyFile.jks  
keyFilePassword=*****  
keyFileFormat=JKS  
trustFileName=${USER_INSTALL_ROOT}/etc/TWSServerTrustFile.jks  
trustFilePassword=*****  
trustFileFormat=JKS  
clientAuthentication=false  
securityLevel=HIGH  
enableCryptoHardwareSupport=false
```

Changing a server key

You change a server key when you want the console or command-line client to safely authenticate with their server. Changing a connector private key means that the trusted server certificates for the consoles and command-line client must also be updated.

This section describes how to apply changes to the connector side. “Customizing the SSL connection for the Job Scheduling Console” on page 183 and “Customizing the SSL connection for a command-line client” on page 184 describe configuration for the consoles and command-line client.

You can customize certificates and update server keystores using **ikeman** that is provided with the embedded WebSphere Application Server, or using the **keytool** utility provided with the Java runtime environment. There are also other tools distributed with other products or available in the Internet. For the connector **ikeman** is located in the directory *TWA_home/eWAS/bin*. The **keytool** utility is located in *TWA_home/eWAS/java/jre/bin*.

You can change the server key with a new self signed certificate that you generate directly, or with a certificate signed by a Certificate Authority.

If you use a self signed certificate for the server, replace the server private key in the server Key keystore and the server public key in the Trust keystores for all the consoles and command-line clients that connect to it.

If you use a certificate signed by a Certificate Authority, replace the server private key in the server Key keystore and make sure you have the certificate of the Certificate Authority in the Trust keystores for all the consoles and command-line clients that connect to it.

The following procedure is an example of how to create a new server key pair using the **keytool** utility:

1. Generate an RSA key pair for the connector in the keystore:

```
keytool -genkey  
-alias tws  
-dname "CN=TWS, OU=Test, O=IBM, C=US"  
-keystore TWSServerKeyFile.jks  
-storepass default  
-keypass default  
-validity 3000  
-keyalg RSA
```

2. Export the certificate to PEM format and import it into the keystore:

```
keytool -export
        -alias tws
        -rfc
        -file server.crt
        -keystore TWSServerKeyFile.jks
        -storepass default
```

When creating self signed certificates, do not use the DSA algorithm as the Tivoli Workload Scheduler command-line utilities cannot use it to establish SSL connection to the server. Make sure that the keys have the same password as the keystore where they are contained. Make sure you update the Trust keystores for the consoles and command-line client.

Customizing the SSL connection for the Job Scheduling Console

When you change the connector private key you need to update the Trust keystore of the Job Scheduling Console with the public key pair when you use self signed certificates, or make sure that it contains the certificate of the Certificate Authority.

When you configure the Job Scheduling Console to connect to different connectors it must have a certificate in its Trust keystore that enables trust for each connector.

You can customize certificates and update the Job Scheduling Console Trust keystore using **ikeman** that is provided with the embedded WebSphere Application Server, or using the **keytool** utility provided with the Java runtime environment. For the Job Scheduling Console **ikeman** and the **keytool** utility are located in the directory *TWA_home/TWS/tools/_jvm/jre/bin*.

The following is an example of how to import the connector certificate in PEM format:

```
keytool -import
        -alias tws
        -file server.crt
        -trustcacerts
        -noprompt
        -keystore JSCTrustFile.jks
        -storepass default
```

When you are customizing the Job Scheduling Console settings, make sure the keys have the same password as the keystore where they are contained. The keystore password must correspond between the connector server and the Job Scheduling Console client. The keys in the Job Scheduling Console Trust and Key keystores must be paired with the keys in the connector Key and Trust keystores respectively.

If you changed the Job Scheduling Console keystore password, name, or location, modify the following Job Scheduling Console configuration files when customizing SSL:

NTConsole.bat

Specifies the keystores and connection configuration files for the SSL connections with Java properties:

com.ibm.ssl.keyStore

For the key keystore name.

Interface communication

com.ibm.ssl.trustStore

For the trusted keystore name.

com.ibm.CORBA.ConfigURL

For the connection properties file name.

NTConsole.bat is located in the *TWA_home/TWS/bin/java/* directory. For UNIX and Linux operating systems the corresponding scripts assume the operating system name, such as *LINUXconsole.sh*, *SUNconsole.sh* and so on.

sas.client.props

Specifies the default properties for SSL connection and contains the keystore names, types, and encoded keystore passwords. The following is an example:

```
#
# Keystores
#
com.ibm.ssl.keyStorePassword={xor}0zo5Pioz\=\=
com.ibm.ssl.contextProvider=IBMJSSE
com.ibm.ssl.keyStoreType=JKS
com.ibm.ssl.trustStorePassword={xor}0zo5Pioz\=\=
com.ibm.ssl.protocol=
com.ibm.ssl.trustStoreType=JKS
```

Encrypt the password using the **encryptProfileProperties** utility. See “Application server - encrypting the profile properties files” on page 269 for details on how to encrypt profile properties.

Customizing the SSL connection for a command-line client

Tivoli Workload Scheduler command-line clients connect to the connector through HTTP or HTTPS. The default connection type is HTTPS. If the command-line connects through a proxy, use the HTTP connection protocol as HTTPS is not supported for this type of configuration.

You configure connection protocol modifying the following options in the *localopts* file:

- | | |
|------------------|---|
| PROXY | Specify the IP address or the server name for the proxy server. |
| PROTOCOL | Specify the protocol type as either HTTP or HTTPS. HTTPS is the default. |
| PORT | Specify the port required by the protocol you set in the PROTOCOL option. The defaults are 31115 for HTTP and 31116 for HTTPS. |
| PROXYPORT | Specify the port required by the proxy server, when you specified PROXY as the server type. |

Alternatively, you can specify the entries in the *useropts* file. When you specify the options in the *useropts* file, they overwrite the options specified in the *localopts* file. See “Setting local options” on page 20 for an explanation of these local options.

The HTTPS connection protocol offers the following additional security features:

- Data encryption between the command-line utility and the connector
- Optional server authentication by validating the server certificates

You can activate optional server authentication in one of the following ways:

- “Configuring SSL using the predefined certificate”
- “Configuring multiple SSL communication instances”
- “Using a customized certificate”

Configuring SSL using the predefined certificate

To customize the SSL connection for the command-line client using the predefined certificate, perform the following steps:

1. Stop the embedded WebSphere Application Server using the **conman stopappserver** command.
2. Extract the certificate from the TWSServerKeyFile.jks keystore:


```
keytool -export
        -alias tws
        -rfc
        -file server.crt
        -keystore TWSServerKeyFile.jks
        -storepass default
```
3. Copy the .crt certificate to each workstation that has a command-line client installed.
4. Set the **clisslserverauth** and **clisslservercertificate** command-line client options in the localopts file. See “Setting local options” on page 20.
5. Start the embedded WebSphere Application Server using the **conman startappserver** command.

Configuring multiple SSL communication instances

To customize the SSL connection for the command-line client for multiple connections to WebSphere Application Server, perform the following steps:

1. Stop WebSphere Application Server using the **conman stopappserver** command.
2. Extract a certificate from TWSServerKeyFile.jks keystore for each instance.


```
keytool -export
        -alias tws
        -rfc
        -file server.crt
        -keystore ServerKeyFile.jks
        -storepass default
```
3. Extract the hash number for each exported certificate:


```
openssl x509
        -hash
        -noout
        -in keyname
```
4. Rename each certificate file with the exported key.
5. Copy the renamed certificates to each workstation that has a command-line client installed.
6. Set the **clisslserverauth** and **clissltrusteddir** command-line client options in the localopts file. See “Setting local options” on page 20.
7. Start the WebSphere Application Server using the **conman startappserver** command.

Using a customized certificate

To customize the SSL certificate and keystore, perform the following steps:

Interface communication

1. Create a new RSA and extract the key for the server keystore TWSServerKeyFile.jks.
2. Follow the steps in “Customizing the SSL connection for the Job Scheduling Console” on page 183.
3. Follow the steps in “Configuring SSL using the predefined certificate” on page 185.

Note: When you want to customize certificates for multiple instances, perform these steps for each instance.

Command-line SSL connection settings

The Tivoli Workload Scheduler command-line utilities connect to the connector through HTTP or HTTPS connections. The default connection type is HTTPS, but you can configure the connection as HTTP modifying entries in the `localopts` file. See “Setting local options” on page 20.

The secure connection protocol (HTTPS) offers the following additional security:

- Data encryption between the command-line utility and the connector
- Optional server authentication by validating the server certificates

When configuring HTTP or HTTPS for the command-line utilities, you need to provide user credentials for connection to the connector. You can do this in one of the following ways:

Using the command-line

You can provide the user credentials using the command-line parameters `-username` and `-password`.

Using the useropts file

You can specify the `USERNAME` and `PASSWORD` parameters in the `useropts` file. For example:

```
# USERNAME and PASSWORD
USERNAME = tws84
PASSWORD = "ENCRYPTMEPLEASE"
```

The password must be enclosed in double-quotes. For a complete definition of the user options, refer to *Tivoli Workload Scheduler: User's Guide and Reference*.

At run time When the user credentials are not specified using command-line parameters, or in the `useropts` file, the command-line requests them and automatically adds them to the `useropts` file.

If the command-line connects through a proxy, use the HTTP connection protocol. The HTTPs protocol is not supported in this type of configuration. Customize the following options in the `localopts` file if you connect through a proxy:

PROXY Specify the IP address or the server name for the proxy server.

PROXYPORT Specify the port number used by the proxy server.

Working across firewalls

In the design phase of a Tivoli Workload Scheduler network, the administrator must know where the firewalls are positioned in the network, which fault-tolerant agents and which domain managers belong to a particular firewall, and which are the entry points into the firewalls. When this has been clearly understood, the

administrator should define the **behindfirewall** attribute for some of the workstation definitions in the Tivoli Workload Scheduler database. In particular, if a workstation definition is set with the **behindfirewall** attribute to ON, this means that there is a firewall between that workstation and the Tivoli Workload Scheduler master domain manager. In this case, the workstation-domain manager link is the only link allowed between the workstation and its domain manager.

All Tivoli Workload Scheduler workstations should be defined with the **behindfirewall** attribute if the link with the corresponding domain manager, or with any domain manager in the Tivoli Workload Scheduler hierarchy right up to the master domain manager, is across a firewall.

When mapping an IBM Tivoli Workload Scheduler network over an existing firewall structure, it does not matter which fault-tolerant agents and which domain managers are on the secure side of the firewall and which ones are on the non secure side. Firewall boundaries should be the only concern. For example, if the master domain manager is in a non secure zone and some of the domain managers are in secured zones, or vice versa, does not make any difference. The firewall structure must always be considered starting from the master domain manager and following the Tivoli Workload Scheduler hierarchy, marking all the workstations that have a firewall between them and their corresponding domain manager.

For all workstations with **behindfirewall** set to ON, the **conman start** and **stop** commands on the workstation, and the **showjobs** commands are sent following the domain hierarchy, instead of making the master domain manager or the domain manager open a direct connection to the workstation. This makes a significant improvement in security.

This attribute works for multiple nested firewalls as well. For extended agents, you can specify that an extended agent workstation is behind a firewall by setting the **behindfirewall** attribute to ON, on the host workstation. The attribute is read-only in the plan; to change it in the plan, the administrator must update it in the database and then recreate the plan.

See the *Tivoli Workload Scheduler: User's Guide and Reference* for details on how to set this attribute.

LDAP connection authentication

Tivoli Workload Scheduler uses LDAP (Lightweight Directory Access Protocol) to authenticate users to post-installation tasks. See "Configuring for LDAP" on page 135 for details of LDAP configuration using the WebSphere Application Server.

Configuring the Job Scheduling Console to work in a Network Address Translation (NAT) environment

The Job Scheduling Console and the connector use the RMI/IIOP protocol to communicate. Some of the connections between the Job Scheduling Console and the connector use JNDI and ORB Services. You can use the **showHostProperties** and **changeHostProperties** utilities to manage the parameters used in this phase of communication. See "Application server - changing the host name or TCP/IP ports" on page 273.

You can set the following parameters:

Configuring the Job Scheduling Console to work in a NAT environment

bootHost

Specify the IP address or host name that is returned to the Job Scheduling Console and used to establish connections to the JNDI server. This address must be visible from the workstation where the Job Scheduling Console is running. This value is set during installation. Change the value to the public IP address or host name when you are running in a NAT.

orbPort

The default value is 0. Set the **orbPort** to a controlled port when running in a NAT environment.

csiServerAuthPort

The default value is 0. Set the **csiServerAuthPort** to a controlled port when running in a NAT environment.

The **bootHost**, **orbPort**, and **csiServerAuthPort** must be opened on firewalls and forwarded to NATs.

Federal Information Processing Standards

This section describes Federal Information Processing Standards (FIPS) compliance. It is divided into the following topics:

- “FIPS overview”
- “Configuring SSL to be FIPS-compliant” on page 193
- “Using FIPS certificates” on page 190
- “Configuring DB2 for FIPS” on page 196
- “Using Tivoli Dynamic Workload Console and FIPS” on page 198
- “Configuring dynamic workload broker for FIPS” on page 199
- “Configuring LDAP for FIPS” on page 199

FIPS overview

Federal Information Processing Standards (FIPS) are standards and guidelines issued by the National Institute of Standards and Technology (NIST) for federal government computer systems. FIPS are developed when there are compelling federal government requirements for standards, such as for security and interoperability, but acceptable industry standards or solutions do not exist. Government agencies and financial institutions use these standards to ensure that the products conform to specified security requirements.

Tivoli Workload Automation uses cryptographic modules that are compliant with the Federal Information Processing Standard FIPS-140-2. Certificates used internally are encrypted using FIPS-approved cryptography algorithms. FIPS-approved modules can optionally be used for the transmission of data.

To satisfy the FIPS 140-2 requirement, you must use IBM Global Security Kit (GSKit) version 7d runtime dynamic libraries instead of OpenSSL. GSKit uses IBM Crypto for C version 1.4.5 which is FIPS 140-2 level 1 certified by the certificate number 755. See <http://csrc.nist.gov/groups/STM/cmvp/documents/140-1/1401val2007.htm>. IBM Java JSSE FIPS 140-2 Cryptographic is another module used by Tivoli Workload Automation. It has the certificate number 409.

If you are currently using SSL for secure connections across the network, to ensure FIPS compliance, you must use GSKit for secure connections instead of OpenSSL Toolkit. GSKit is automatically installed with Tivoli Workload Scheduler. It is based on dynamic libraries and offers several utilities for certificate management.

Federal Information Processing Standards

| To comply with FIPS, all components of Tivoli Workload Automation must be
| FIPS-compliant. You must use Tivoli Dynamic Workload Console or the Tivoli
| Workload Scheduler command-line as the interface to Tivoli Workload Scheduler.
| The Job Scheduling Console is not FIPS-compliant. Additionally, you must use DB2
| as your Tivoli Workload Scheduler database.

| If FIPS compliance is not of concern to your organization, you can continue to use
| SSL for secure connections across your network.

To set FIPS compliance for your network, perform the procedures described in the following sections:

- To create FIPS certificates, see “Using FIPS certificates.”
- To configure SSL for FIPS-compliance, see “Configuring SSL to be FIPS-compliant” on page 193.
- To configure your DB2 database for FIPS-compliance, see “Configuring DB2 for FIPS” on page 196.

Using FIPS certificates

To ensure your network is FIPS-compliant, create FIPS certificates as follows:

- If you do not already have SSL certificates, see “Using fresh FIPS certificates.”
- If you already have SSL certificates but are switching to GSKit, see “Switching from OpenSSL to GSKit” on page 191.

If you are using FIPS certificates, you must use SSL parameters for communication over the network. During the installation or upgrade to Tivoli Workload Scheduler version 8.5.1, note that default SSL certificates are located in the following directories:

```
TWS_InstallDir\TWS\ssl\GSKit  
TWS_InstallDir\TWS\ssl\OpenSSL
```

Using fresh FIPS certificates

Create FIPS certificates for communication between workstations by using the `-fips` option in the GSKit command line utility. You can create FIPS certificates in the following ways:

- Use the default FIPS certificates existing on each Tivoli Workload Scheduler agent in the network. Note that the default FIPS certificates are not secure.
- Create your own secure FIPS certificates. See “Creating your own FIPS certificates.”

Creating your own FIPS certificates: Use the `gsk7capicmd` command line utility to:

- Create your own Certificate Authority (CA).
- Create a self-signed CA certificate (x.509 structure) for your CA.
- Export the CA certificate in PEM format.

Creating your own Certificate Authority: Create the CA on any workstation in your network. Run the following steps only once to create a CA that will be used each time a new certificate needs to be created and signed.

1. Enter the following command to create the CMS key database “ca.kdb” with password “password00” that expires after 1000 days.

```
gsk7capicmd -keydb -create -db ca.kdb -pw password00 -stash -expire 1000 -fips
```
2. Enter the following command to create the self-signed certificate with label “CA certificate” using the distinguish name “CN=CA certificate,O=IBM,OU=TWS,C=IT”. The certificate expires after 1000 days.

```
gsk7capicmd -cert -create -db ca.kdb -pw password00 -label "CA certificate"  
-size 1024 -expire 1000 -dn "CN=CA certificate,O=IBM,OU=TWS,C=IT"
```
3. Enter the following command to extract the CA certificate into external file “ca.crt”. The certificate is addressed by the corresponding label.

```
gsk7capicmd -cert -extract -db ca.kdb -pw password00 -label "CA certificate"  
-target CA.crt
```

This file will contain the public certificate of the certificate authority.

Creating a certificate for the Tivoli Workload Scheduler agent: Perform the following steps to create certificates that are signed by a local common trusted CA on every Tivoli Workload Scheduler agent in your network.

1. Enter the following command to create a default CMS key database client.kdb with password "password02" that expires after 1000 days. The password is also stored in stash file "client.sth".

```
gsk7capicmd -keydb -create -db client.kdb -pw password02
            -stash -expire 1000 -fips
```

2. Enter the following command to add the CA certificate as trusted in the CMS key database. The label "CA certificate client" is used to address that certificate.

```
gsk7capicmd -cert -add -db client.kdb -pw password02
            -label "CA certificate client" -trust enable -file CA.crt
            -format ascii -fips
```

3. Enter the following command to create the client certificate request based on 1024 bits key, with label "Client TWS85 Certificate" and distinguish name "CN=Client TWS85,O=IBM,OU=TWS,C=IT". The certificate request "client.csr" is generated and the private key is created in the key database client.kdb.

```
gsk7capicmd -certreq -create -db client.kdb -pw password02
            -label "Client TWS85 Certificate" -size 1024 -file client.csr
            -dn "CN=Client TWS85,O=IBM,OU=TWS,C=IT" -fips
```

4. Enter the following command so that the CA signs the client's certificate request and generates a new signed in file "client.crt".

```
gsk7capicmd -cert -sign -db ca.kdb -pw password00 -label "CA certificate"
            -target client.crt -expire 365 -file client.csr -fips
```

5. Enter the following command to import the signed certificate "client.crt" in the CMS key database "client.kdb".

```
gsk7capicmd -cert -receive -db client.kdb -pw password02 -file client.crt -fips
```

You can repeat these steps above for all agents or you can use the same certificate for all agents, depending on your security policies and Tivoli Workload Automation localopts configurations.

Switching from OpenSSL to GSKit

This section describes how to migrate your OpenSSL certificates to GSKit certificates.

The following is a list of certificate formats that can be migrated to the GSKit format, **KDB**:

- **PEM**: Used by OpenSSL
- **JKS**: Used by Java and WebSphere Application Server
- **PKCS12**: Used by Microsoft applications and Internet Explorer

To migrate certificates, you may use one or more of the following tools:

- **gsk7cmd**: Java command line provided by GSKit
- **gsk7capicmd**: Native command line provided by GSKit
- **openssl**: Native command line provided by OpenSSL
- **keyman**: Optional graphical interface provided by GSKit
- **keytool**: Optional graphical interface provided by Java Virtual Machine (JVM)

Note: Be sure to backup your original certificates before migrating them to GSKit format.

Federal Information Processing Standards

To migrate your certificates, perform the following steps:

1. "Configuring the tool environment"
2. "Migrating the certificates"

Configuring the tool environment: This section describes the commands you must run to configure `gsk7cmd`, `gsk7capicmd`, and `openssl`.

Configuring gsk7cmd:

UNIX

```
export JAVA_HOME=/opt/IBM/TWA/eWAS/java/jre
export CLASSPATH= /opt/IBM/TWA/eWAS/java/bin:
/opt/IBM/TWA/eWAS/java/lib
```

Windows

```
set JAVA_HOME=C:\Program Files\IBM\TWA\ewAS\java\jre
set CLASSPATH= C:\Program Files\IBM\TWA \ewAS\java\bin; C:\Program
Files\IBM\TWA \ewAS\java\lib
```

Configuring gsk7capicmd: `set PATH=C:\Program Files\IBM\TWA\TWS\GSKit\7d\lib; C:\Program Files\IBM\TWA\TWS\GSKit\7d\bin;%PATH%`

Configuring openssl:

UNIX `twS_env.sh`

Windows

`twS_env.cmd`

Migrating the certificates: This section describes the commands you must run to migrate certificates to the FIPS-compliant format, KDB.

Note that PEM format cannot be directly converted to KDB format; you must first convert PEM to PKCS12 and then to KDB.

The following list describes the command you must run to convert from one format to another:

JKS format to KDB format

```
gsk7cmd -keydb -convert -db TWSClntKeyFile.jks -pw default
-old_format jks -new_format cms

gsk7cmd -keydb -convert -db TWSClntTrustFile.kdb -pw default
-old_format cms -new_format jks
```

PKCS12 format to KDB format

```
gsk7capicmd -cert -export -target TWSClntKeyFile_new.kdb -db
TWSClntKeyFileP12.P12 -fips -target_type cms -type pkcs12
```

PKCS12 format to PEM format

```
openssl pkcs12 -in TWSClntKeyFileP12.P12 -out TWSClntKeyFile.pem
```

PEM format to PKCS12 format

```
openssl pkcs12 -export -in TWSClntKeyFile.pem -out cred.p12
```

KDB format to PKCS12 format

```
gsk7capicmd -cert -export -db TWSClntKeyFile.kdb -target
TWSClntKeyFileP12.P12 -fips -target_type pkcs12 -type cms
```

Converting PEM certificates to CMS certificates: This section describes the procedure to convert PEM (OpenSSL) certificates to CMS (GSKit) certificates. The examples in this section use the following input and output files.

Input files

Personal certificate file: *CPU1.crt*
 Personal key of certificate file: *CPU1.key*
 Certificate of CA file: *TWSca.crt*
 Stash file: *CPU1.sth*

Output files

Keystore database file: *TWS.kdb*
 Stash file: *TWS.sth*
 Label of your certificate: *CPU1*

To migrate OpenSSL certificates to GSKit certificates, perform the following procedure:

1. Merge the public and private keys in a new temporary file called **all.pem** by running the following commands:

```
UNIX cat CPU2.crt CPU2.key > all.pem
```

Windows

```
type CPU1.crt CPU1.key > all.pem
```

2. If you do not already know the password, extract it from the stash file by running `openssl base64 -d -in CPU1.sth`.
3. Choose a password for the new keystore database. You can reuse the old password.
4. Choose a label for your personal certificate and personal key (in this example, CPU1) and create the PKCS12 database that contains the labels. You use the name, CPU1, as the label of the new keystore database. To create the PKCS12 database, run the following:

```
openssl pkcs12 -export -in all.pem -out TWS.p12 -name CPU1 -passin pass:  
password1 -passout pass:password2
```

where *password1* is the password extracted from the stash file and *password2* is the new password to manage the new keystore database.

5. Convert the PKCS12 database from TWS.p12 to the CMS database, TWS.kdb by running the following:

```
gsk7capicmd -cert -import -target TWS.kdb -db TWS.p12 -target_type cms  
-type pkcs12 -label CPU1 -target_pw "password2" -pw "password3"
```

where *password2* is the old password that you extracted from the stash file, CPU1.sth and *password3* is the new password.

6. Choose a label for your Certification Authority contained in TWSca.crt. For this example, it is *TWSca*.
7. Add the certificate of the Certification Authority into your TWS.kdb file by running:

```
gsk7capicmd -cert -add -db TWS.kdb -label TWSca -trust -file TWSca.crt  
-format ascii -pw "password"
```

8. Delete all .pem files.

Configuring SSL to be FIPS-compliant

To configure SSL to be FIPS-compliant, perform the following procedures:

Federal Information Processing Standards

- Set localopts parameters. See “Setting localopts parameters for FIPS.”
- Configure embedded WebSphere Application Server. See “Configuring embedded WebSphere Application Server for FIPS.”
- Configure the Tivoli event integration facility port. See “Configuring the Tivoli event integration facility port” on page 196.

Note:

If you are using dynamic workload broker for dynamic scheduling in your network, note that the workstation of type **BROKER** does not support SSL. All Tivoli Workload Scheduler workstations must communicate with the workstation of type **BROKER** using TCP/IP protocol.

Setting localopts parameters for FIPS

To set your environment for FIPS, set the following local option on every Tivoli Workload Scheduler agent in the network.

SSL Fips enabled = *yes*

Set the following local options for the engine:

```
SSL keystore file = "/opt/ibm/TWA/TWS/ssl/TWSKeyRing.kdb"  
SSL certificate keystore label = "IBM TWS 8.5 workstation"  
SSL keystore pwd = "/opt/ibm/TWA/TWS/ssl/TWSKeyRing.sth"
```

Set the following local options for the CLI:

```
CLI SSL keystore file =  
    "C:\TWS\installations\tws850cli\TWS\ssl\gskit\TWSClientKeyStore.kdb"  
CLI SSL certificate keystore label = "label.txt"  
CLI SSL keystore pwd =  
    "C:\TWS\installations\tws850cli\TWS\ssl\gskit\ TWSClientKeyStore.sth"
```

Note: On Windows workstations, the user, **SYSTEM**, must have read-permissions to read the GSKit FIPS certificates.

Configuring embedded WebSphere Application Server for FIPS

To be FIPS-compliant, you must configure embedded WebSphere Application Server for Tivoli Workload Scheduler.

The section describes how to:

- Configure embedded WebSphere Application Server for Tivoli Workload Scheduler. See “Configuring embedded WebSphere Application Server for Tivoli Workload Scheduler.”
- Configure the Tivoli event integration facility port. See “Configuring the Tivoli event integration facility port” on page 196.

Configuring embedded WebSphere Application Server for Tivoli Workload Scheduler: To configure embedded WebSphere Application Server for FIPS compliance, perform the following steps:

1. In the WebSphere Application Server administration interface, click **Security** > **SSL certificate and key management**. Select **Use the United States Federal**

Information Processing Standard (FIPS) algorithms and click **Apply**.
Alternatively, you can use wastools, running `changeSecurityProperties` to change the following parameter:

```
useFIPS=true
```

2. In the `profile_root/properties/ssl.client.props` file, set the following parameters:

- `com.ibm.security.useFIPS=true`
- `com.ibm.ssl.protocol=SSL_TLS`

3. If you have an administrative client that uses a SOAP connector, add the following line to the `profile_root/properties/soap.client.props` file:

```
com.ibm.ssl.contextProvider=IBMJSEFIPS
```

4. Edit the SDK `java.security` file located in the `WASHOME/java/jre/lib/security` directory to insert the **IBMJCEFIPS** provider (`com.ibm.crypto.fips.provider.IBMJCEFIPS`). **IBMJCEFIPS** must precede the **IBMJCE** provider in the provider list.

The following is an example of the edited SDK `java.security` file:

```
security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.2=com.ibm.crypto.provider.IBMJCE
security.provider.3=com.ibm.jsse.IBMJSSEProvider
security.provider.4=com.ibm.jsse2.IBMJSSEProvider2
security.provider.5=com.ibm.security.jgss.IBMJGSSProvider
security.provider.6=com.ibm.security.cert.IBMCertPath
security.provider.7=com.ibm.crypto.pkcs11.provider.IBMPKCS11
security.provider.8=com.ibm.security.cmskeystore.CMSProvider
security.provider.9=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
```

The following is an example of the edited `java.security` file if you are using the Sun Java SE Development Kit:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.jsse.IBMJSSEProvider
security.provider.5=com.ibm.jsse2.IBMJSSEProvider2
security.provider.6=com.ibm.security.jgss.IBMJGSSProvider
security.provider.7=com.ibm.security.cert.IBMCertPath
security.provider.8=com.ibm.i5os.jsse.JSSEProvider
#security.provider.8=com.ibm.crypto.pkcs11.provider.IBMPKCS11
security.provider.9=com.ibm.security.jgss.mech.spnego.IBMSPNEGO
```

5. Restart the WebSphere Application Server.

Note: For additional information about WebSphere Application Server and FIPS, see the WebSphere Application Server documentation.

Unconfiguring the FIPS provider: To unconfigure the FIPS provider, reverse the changes that you made in “Configuring embedded WebSphere Application Server for FIPS” on page 194. After you reverse the changes, verify that you have made the following changes to the `ssl.client.props`, `soap.client.props`, and `java.security` files:

- In the `ssl.client.props` file, change the `com.ibm.security.useFIPS` value to `false`.
- In the `java.security` file, change the FIPS provider to a non-FIPS provider.
- If you are using the SDK `java.security` file, change the first provider to a non-FIPS provider as shown in the following example:

```
#security.provider.1=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.1=com.ibm.crypto.provider.IBMJCE
security.provider.2=com.ibm.jsse.IBMJSSEProvider
security.provider.3=com.ibm.jsse2.IBMJSSEProvider2
```

Federal Information Processing Standards

```
security.provider.4=com.ibm.security.jgss.IBMJGSSProvider
security.provider.5=com.ibm.security.cert.IBMCertPath
#security.provider.6=com.ibm.crypto.pkcs11.provider.IBMPKCS11
```

- If you are using the Sun JDK java.security file, change the third provider to a non-FIPS provider as shown in the following example:

```
security.provider.1=sun.security.provider.Sun
security.provider.2=com.ibm.security.jgss.IBMJGSSProvider
#security.provider.3=com.ibm.crypto.fips.provider.IBMJCEFIPS
security.provider.3=com.ibm.crypto.provider.IBMJCE
security.provider.4=com.ibm.jsse.IBMSJSEProvider
security.provider.5=com.ibm.jsse2.IBMSJSEProvider2
security.provider.6=com.ibm.security.cert.IBMCertPath
#security.provider.7=com.ibm.crypto.pkcs11.provider.IBMPKCS11
```

- This step applies only if you added the default JSSE socket factories parameters to the SDK java.security file in step 4 on page 197 of “Configuring DB2.” If you added them, remove the following parameters:

```
ssl.SocketFactory.provider=com.ibm.jsse2.SSLSocketFactoryImpl
ssl.ServerSocketFactory.provider=com.ibm.jsse2.SSLServerSocketFactoryImpl
```

Configuring the Tivoli event integration facility port

The Tivoli event integration facility port for SSL, `eventProcessorEIRSSLPort`, is used in event management. For the Tivoli event integration facility port to communicate in FIPS mode, you must first configure embedded WebSphere Application Server for FIPS. See “Configuring embedded WebSphere Application Server for Tivoli Workload Scheduler” on page 194.

To configure the Tivoli event integration facility port for SSL, perform the following steps:

1. Set the global option for the port using `optman`. Set the port as follows:
`eventProcessorEIFSSLPort / ef = portnumber`
where `portnumber` is the port number of any free port on your network.
2. To update the Symphony file, run `JnextPlan -for 0000`.
3. Restart the EventProcessor using the `conman stopevtpt` and `conman startevtpt` commands.
4. Restart the Tivoli Workload Scheduler monitoring engine with the `conman` commands, `stopmon` and `startmon`.

Configuring DB2 for FIPS

To configure DB2 for FIPS compliance, perform the following procedures:

- “Configuring DB2”
- “Configuring the DB2 connection to Tivoli Workload Scheduler” on page 198

Note: DB2 version 9.1 (HP-UX for PA-RISC) does not support FIPS.

Configuring DB2

To configure DB2 for FIPS compliance, perform the following procedure:

Note: If you want to create your own DB2 certificates, see the DB2 documentation.

1. Ensure that the path to the GSKit libraries appear as follows:

UNIX and Linux

The `LIBPATH`, `LD_LIBRARY_PATH`, or `SHLIB_PATH` variables must contain the path to the GSKit libraries.

Windows

The system path variable must contain the path to the GSKit libraries, for example, c:\Program Files\IBM\gsk7\lib.

2. Configure SSL with the command **db2set DB2COMM=SSL**.

Note: During the installation of a Tivoli Workload Scheduler master domain manager or backup master domain manager, it is necessary to enable the DB2 TCPIP port. DB2 can support both TCP/IP and SSL communications protocols at the same time. The DB2 administrator can set the TCPIP port with the command **db2set DB2COMM=TCPIP, SSL**. Use this command if you are installing a Tivoli Workload Scheduler master domain manager or backup master domain manager and already have a FIPS-enabled instance of DB2. After installation, you can choose to reset DB2COMM with only SSL.

3. In **INSTHOME**, create the file **SSLconfig.ini** as follows:

UNIX and Linux

INSTHOME/cfg/SSLconfig.ini

Windows

INSTHOME/SSLconfig.ini

where **INSTHOME** is the home directory of the instance. It is recommended that file permission is set to limit access to the **SSLconfig.ini** file because the file contains sensitive data, for example, password information.

The following is an example of an **SSLconfig.ini** file:

```
DB2_SSL_KEYSTORE_FILE=/tools/keystores/DB2/TWSSClientKeyStore.kdb
DB2_SSL_KEYSTORE_PW=yyyyy
DB2_SSL_LISTENER=nnnnn
DB2_SSL_KEYSTORE_LABEL=zzzzz
```

where

- *TWSSClientKeyStore.kdb* is fully-qualified file name of the KeyStore that stores the Server Certificate. This KeyStore can be the same as you specified in the **localopts** parameters. See “Setting **localopts** parameters for FIPS” on page 194. Note that it must be recognized by the JKS WebSphere Application Server certificate.
- *yyyyy* is the password of the KeyStore that stores the Server Certificate.
- *nnnnn* is the port number used by DB2 . This port must be the same port as the SSL port.
- *zzzzz* is the label for the Server Certificate.

4. Insert the following default JSSE socket factories parameters in the **java.security** file of the Tivoli Workload Scheduler WebSphere Application Server:

```
ssl.SocketFactory.provider=com.ibm.jsse2.SSLSocketFactoryImpl
ssl.ServerSocketFactory.provider=com.ibm.jsse2.SSLServerSocketFactoryImpl
```

5. Restart DB2.

Note: It is not necessary to update the DB2 JVM. This is because you already updated the JVM of the embedded WebSphere Application Server in the procedure described in “Configuring embedded WebSphere Application Server for FIPS” on page 194.

For more information about how to configure DB2 to be FIPS compliant, see the DB2 documentation that describes how to configure Secure Sockets Layer (SSL) support in a DB2 instance.

Configuring the DB2 connection to Tivoli Workload Scheduler

After configuring DB2, you must configure Tivoli Workload Scheduler to communicate with the new settings of DB2. Perform the following procedure:

1. Modify the DB2 DataSource properties in wastools by running **showDataSourceProperties** and **changeDataSourceProperties** to include the following parameters:

```
DB2Type4PortNumber=nnnnn  
DB2Type4SslConnection=true
```

where *nnnnn* is the SSL DB2 port number.

2. Restart the WebSphere Application Server.

Using Tivoli Dynamic Workload Console and FIPS

To ensure that you connect to Tivoli Dynamic Workload Console using FIPS, perform the following:

1. Enable Transport Layer Security (TLS) in your browser as follows:
 - To enable TLS in Internet Explorer, open the browser and click **Tools > Internet Options**. On the Advanced tab, select **Use TLS 1.0**.
 - To enable TLS in Mozilla Firefox, open the browser and click **Tools > Options > Advanced**. On the Encryption tab, select **Use TLS 1.0**.
 - To enable TLS on other internet browsers, see the product documentation for that browser.
2. Depending on your configuration, perform one of the following procedures:

If you have a standalone instance of Tivoli Dynamic Workload Console on embedded WebSphere Application Server:

- a. Configure FIPS on the embedded WebSphere Application Server of Tivoli Dynamic Workload Console. See “Configuring embedded WebSphere Application Server for FIPS” on page 194.
- b. In the Tivoli Dynamic Workload Console wastools, run **changeDataSourceProperties** to change the following parameters:

```
DB2Type4PortNumber=nnnnn  
DB2Type4SslConnection=true
```

where *nnnnn* is the SSL DB2 port number.

- c. Restart the embedded WebSphere Application Server.

If you have an instance of Tivoli Dynamic Workload Console that shares an instance of Tivoli Workload Scheduler embedded WebSphere Application Server:

- a. Ensure that the Tivoli Workload Scheduler embedded WebSphere Application Server is FIPS-compliant. See “Configuring embedded WebSphere Application Server for FIPS” on page 194.
- b. In the Tivoli Workload Scheduler wastools, run **showDataSourceProperties** to ensure the following parameters are set:

```
DB2Type4PortNumber=nnnnn  
DB2Type4SslConnection=true
```

where *nnnnn* is the SSL DB2 port number.

- c. Restart the embedded WebSphere Application Server.

If you have Tivoli Dynamic Workload Console on an external WebSphere Application Server:

Ensure that the WebSphere Application Server is FIPS-compliant. See “Configuring embedded WebSphere Application Server for FIPS” on page 194.

3. If you are using Dynamic Workload Broker, set a secure connection by performing the following:
 - a. In Tivoli Dynamic Workload Console, access Tivoli Dynamic Workload Broker and expand the **Configuration** menu.
 - b. Click **Server Connections**.
 - c. In the Server Connections screen, select **Use Secure Connection**.
 - d. Click **OK**.

Note: To enable communication between Tivoli Dynamic Workload Console and DB2, configure the Java system properties in Tivoli Dynamic Workload Console to use the trustStore. To do this, set the following Java system properties:

```
javax.net.ssl.trustStore
javax.net.ssl.trustStorePassword
```

For more information, see the DB2 documentation.

Configuring dynamic workload broker for FIPS

If you are using the dynamic workload broker component in your network, perform the following configurations:

- Configure the ita.ini file of every agent that will communicate with the dynamic workload broker component. Ensure that the ssl_port is set and set fips_enable = 1.
- If you are using Tivoli Dynamic Workload Console, set a secure connection by performing the following:
 1. In Tivoli Dynamic Workload Console, access dynamic workload broker and expand the **Configuration** menu.
 2. Click **Server Connections**.
 3. In the Server Connections screen, select **Use Secure Connection**.
 4. Click **OK**.

Configuring LDAP for FIPS

To be FIPS-compliant if you are using an LDAP server, before configuring LDAP, edit the security.xml file. Edit the following value:

```
"com.ibm.ssl.contextProvider" value="IBMJSSEFIPS"
```

Finding the GSKit version on agents running on UNIX and Linux platforms

To find which version of GSKit runs on your agent, run the gsk7ver command.

On UNIX and Linux, you can optionally run the ita_props.sh script to set the environment to /usr/Tivoli/TWS/GSKit/7d/bin, so that you can run this command directly without having to specify the relative path.

Chapter 8. Data maintenance

This chapter describes how to maintain your Tivoli Workload Scheduler database and other data files. The database is hosted on either the DB2 or Oracle RDBMS infrastructure, as you determined when you installed it. You should use the documentation of DB2 or Oracle for general instructions on database maintenance. This chapter describes the maintenance activities that are specific to Tivoli Workload Scheduler.

It comprises the following sections:

- “Maintaining the database”
- “Maintaining the file system” on page 204
- “Administrative tasks - DB2” on page 209
- “Administrative tasks - Oracle” on page 215
- “Migrating data from DB2 to Oracle and *vice versa*” on page 217
- “Upgrading your database” on page 231

Maintaining the database

This section discusses the following:

- Backing up and restoring files in the Tivoli Workload Scheduler databases. See “Backing up and restoring.”
- Ensuring that a backup master domain manager is as up-to-date as possible. See “Using a backup master domain manager with a backup database” on page 202.
- Maintaining the performance level of the Tivoli Workload Scheduler databases. See “Reorganizing the database” on page 203.

Backing up and restoring

To minimize downtime during disaster recovery, back up your master data files frequently to either offline storage or a backup master domain manager.

Backing up the database to offline storage

Run a frequent backup of the database to offline storage. Follow the instructions in the DB2 or Oracle documentation, as appropriate.

1 Tivoli Workload Scheduler is supplied with a utility that can be used for backup. It
1 is called **twins_inst_pull_info**. Its primary use is as a tool to gather Tivoli Workload
1 Scheduler information for IBM Software Support in the event of any problems
1 arising. However, it can equally be used as a backup tool. It backs up the database
1 (DB2 only), the configuration files and the log files.

This tool is described in *Tivoli Workload Scheduler: Troubleshooting Guide* and gives full details of what files are backed up, how to take a backup, and how to restore from one.

Using a backup master domain manager with a backup database

Set up a backup master domain manager that accesses a different database than the master domain manager, and get your database administrator to set up a mirror of the master domain manager's database onto the backup master domain manager's database. In this way your backup master domain manager not only receives copies of all the processing messages, as is provided for by the setting of the *FullStatus* attribute on the backup master domain manager, but is also able to access the mirrored database. The mirror frequency must be set high enough to match the frequency with which you change the database.

For more information about how to use a backup master domain manager, see "Changing a domain manager" on page 234.

Backing up the configuration files

The configuration files used by Tivoli Workload Scheduler are found in the following places:

<TWA_home>/TWS

For the user options file, useropts.

<TWA_home>/TWS/*.*

For the localopts, Sfinal, Security and *.msg files

<TWA_home>/TWS/mozart/*.*

This directory contains the following files:

runmsgno

This is used for the allocation of unique prompt numbers. On the master domain manager this file should not be edited manually. On other workstations it can be edited only in the circumstances described in the *Tivoli Workload Scheduler: Troubleshooting Guide*. This file does not need to be backed up.

globalopts

This is used to store a copy of three of the global properties stored in the database. If you have used versions of Tivoli Workload Scheduler prior to version 8.3 you will probably remember that it was an editable file that contained the global options. It is no longer used for this purpose. It must be edited only in the circumstances described in "Changing a master domain manager" on page 236. This file should be backed up if it is edited.

<TWA_home>/eWAS/profiles/twaprofile/properties

For the application server configuration file, TWSConfig.properties

<TWA_home>/eWAS/profiles/twaprofile/config

This contains the other configuration files for the embedded WebSphere Application Server. Do not back them up manually. A utility to back them up is described in "Application server - configuration files backup and restore" on page 272.

<TWA_home>/TWS/schedForecast

For forecast plan files.

<TWA_home>/TWS/schedlog

For archived plan files.

<TWA_home>/TWS/schedTrial

For trial plan files.

A detailed list of all files is not supplied, as there are too many files. Back up all the files in these directories.

1 **Note:** The **twc_inst_pull_info** tool (described in the *Tivoli Workload Scheduler: Troubleshooting Guide*) is provided for sending information to support, but
 1 can also be used to perform a backup of a DB2 database and some of the
 1 configuration files.
 1

Backing up log files

Make a regular offline backup of all log files, identifying them from the information given in the section on log and trace files in the *Tivoli Workload Scheduler: Troubleshooting Guide*.

1 If you use **twc_inst_pull_info** for backup (see the documentation in the same
 1 guide), you do not need to separately backup these files.

Reorganizing the database

The database requires routine maintenance, as follows:

DB2 The DB2 database has been set up to maintain itself, so there is little user maintenance to do. Periodically, DB2 checks the database by running an internal routine. DB2 determines when this routine must be run using a default policy. This policy can be modified, if need be, or can be switched off so that DB2 does not perform internal automatic maintenance. Using the statistical information that DB2 discovers by running this routine, it adjusts its internal processing parameters to maximize its performance.

This routine has also been made available for you to run manually in the case either where you feel that the performance of DB2 has degraded, or because you have just added a large amount of data, and anticipate performance problems. The routine is imbedded in a tool called **dbbrunstats**, which can be run to improve performance while DB2 is processing data without causing any interruption.

It is also possible to physically and logically reorganize the database using the **dbreorg** script. This effectively recreates the *tablespace* using its internal algorithms to determine the best way to physically and logically organize the tables and indexes on disk. This process is time-consuming, and requires that Tivoli Workload Scheduler is down while it is run, but it does provide you with a freshly reorganized database after major changes.

The use of these tools is described in “Administrative tasks - DB2” on page 209.

These tools are implementations of standard DB2 facilities. If you are an expert user of DB2 you can use the standard facilities of DB2 to achieve the same results. For details go to the Information Center for DB2, version 9.5, at: <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5//index.jsp>.

Oracle For Oracle databases see the Oracle maintenance documentation.

Oracle 10g by default has an internally scheduled procedure to collect database statistics: if the default schedule is not changed, Oracle 10g will

Database maintenance

automatically optimize its performance by running this procedure daily. Oracle 9i does not have the same schedule by default, but could be set up to do so.

Maintaining the file system

Some of the file systems and directories need periodic maintenance. The details are given under the following topics:

- “Avoiding full file systems”
- “Log files and archived files” on page 207
- “Temporary files” on page 209
- “Managing event message queue file sizes” on page 209

Avoiding full file systems

An important maintenance task to perform is to regularly control the file system or systems where Tivoli Workload Scheduler is installed, particularly on the master domain manager.

Tivoli Workload Scheduler has a number of files that can increase in size, either with more extensive use, such as the Symphony file, or because of network problems, such as the message files. If the Symphony file cannot be expanded to contain all the required records, it might become corrupted. If this happens on a fault-tolerant agent or on a domain manager other than the master domain manager, there is a recovery procedure (see the *Tivoli Workload Scheduler: Troubleshooting Guide*). If the Symphony file on the master domain manager is corrupted, you must restart Tivoli Workload Scheduler, losing the current plan's workload.

It is thus *most important* that you monitor the available space on the file system of the master domain manager where the Symphony file is generated, to ensure that there is always sufficient space for it to expand to cover any workload peaks, and also that there is sufficient space for message files to expand due to network problems. Your experience with your workload and your network will guide you to determine what are the acceptable limits of available disk space.

The approximate size of the Symphony file can be estimated in advance. It contains items related both to the plan (see Table 54) and to the database (see Table 55 on page 205). Estimate how many items, that you have in each category, multiply them by the indicated size in bytes, and sum them to find the approximate Symphony file size:

Table 54. Algorithm for calculating the approximate size of the plan data in the Symphony file

Data in Symphony file from the current plan	Bytes per instance
Per job stream instance:	512
Per job instance:	512
Per job "docommand" string > 40 bytes:	The length of the "docommand" string
Per ad-hoc prompt:	512
Per file dependency:	512
Per recovery prompt:	512
Per recovery job:	512

Table 55. Algorithm for calculating the approximate size of the database data in the Symphony file

Data in Symphony file from the database (on the master domain manager)	Bytes per instance
Per workstation:	512
Per resource:	512
Per Windows user:	256
Per prompt:	512
If the global option ignoreCalendars is set to <i>off</i> , per calendar:	512

If you find that disk space is becoming too limited, and you cannot dynamically extend it, you must create a backup master domain manager with much more space on its file system and then use the **switchmgr** command to have the backup become your new domain manager. Instructions about how to do this for any domain manager are given in “Changing a domain manager” on page 234, and for a master domain manager, in “Changing a master domain manager” on page 236.

Monitoring the disk space used by Tivoli Workload Scheduler

You can use event-driven workload automation (EDWA) to monitor the disk space used by Tivoli Workload Scheduler and to start a predefined set of actions when one or more specific events take place. You can use EDWA to monitor the used disk space, to verify that there is enough space to generate the Symphony and log files, and to allow the product to work correctly. For more information about event-driven workload automation, see *Tivoli Workload Scheduler: User’s Guide and Reference*.

The following .XML file contains the definition of a sample event rule to monitor the disk filling percentage. This event rule calls the MessageLogger action provider to write a message in a log file in an internal auditing database. If the condition described in the rule already exists when you deploy the rule, the related event is not generated. For more information about the MessageLogger action provider, see *Tivoli Workload Scheduler: User’s Guide and Reference*

```
<?xml version="1.0"?>
<eventRuleSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules"
  xsi:schemaLocation="http://www.ibm.com/xmlns/prod/tws/1.0/event-management
/rules http://www.ibm.com/xmlns/prod/tws/1.0/event-management/rules/EventRules.xsd">
  <eventRule name="FILESYSTEMFULL" ruleType="filter" isDraft="yes">
    <eventCondition name="twsDiskMonEvt1" eventProvider="TWSApplicationMonitor"
eventType="TWSDiskMonitor">
      <scope>
        * Disk is filling up
      </scope>
      <filteringPredicate>
        <attributeFilter name="FillingPercentage" operator="ge">
          <value>filling_percentage</value>
        </attributeFilter>
        <attributeFilter name="Workstation" operator="eq">
          <value>workstation_name</value>
        </attributeFilter>
        <attributeFilter name="SampleInterval" operator="eq">
          <value>sample_interval</value>
        </attributeFilter>
        <attributeFilter name="MountPoint" operator="eq">
          <value>mount_point</value>
        </attributeFilter>
      </filteringPredicate>
    </eventCondition>
  </eventRule>
</eventRuleSet>
```

Maintaining file system

```

1         </filteringPredicate>
1     </eventCondition>
1     <action actionProvider="MessageLogger" actionType="MSGLOG" responseType=
1 "onDetection">
1         <scope>
1             OBJECT=ADWDAD MESSAGE=Disk is filling up
1         </scope>
1         <parameter name="ObjectKey">
1             <value>object_key</value>
1         </parameter>
1         <parameter name="Severity">
1             <value>message_severity</value>
1         </parameter>
1         <parameter name="Message">
1             <value>log_message</value>
1         </parameter>
1     </action>
1 </eventRule>

```

where:

filling_percentage

Is the filling percentage. The following operators are supported:

ge Causes the event generation when the disk filling percentage increases above the threshold value. The event is generated only the first time that the specified disk filling percentage is reached. If you restart the SSM agent and the filling percentage is higher than the threshold value, the event is generated again. Table 56 provides an example in which the **ge** operator is set to 70%.

Table 56. Example for the *ge* operator

Mailbox name	Filling percentage	Action
Sample (0)	>= 70%	event not generated
Sample (0)	< 70%	event not generated
Sample (n-1)	< 70%	event not generated
Sample (n)	>= 70%	event generated
Sample (n+1)	>= 70%	event not generated

le Causes the event generation when the disk filling percentage decreases below the threshold value. The event is generated only the first time that the specified disk filling percentage is reached. If you restart the SSM agent and the filling percentage is lower than the threshold value, the event is not generated until the filling percentage increases above the threshold value and then decreases below it again. Table 57 provides an example in which the **le** operator is set to 50%:

Table 57. Example for the *le* operator

Mailbox name	Filling percentage	Action
Sample (0)	<= 50%	event not generated
Sample (0)	> 50%	event not generated
Sample (n-1)	> 50%	event not generated
Sample (n)	<= 50%	event generated
Sample (n+1)	<= 50%	event not generated

workstation_name

Is the workstation on which the event is generated.

- 1 *sample_interval*
Is the interval, expressed in seconds, for monitoring the disk filling percentage.
- 1 *mount_point*
Is the mount point of the file system where Tivoli Workload Scheduler is installed.
- 1 *object_key*
Is a key identifying the object to which the message belongs.
- 1 *message_severity*
Is the severity of the message.
- 1 *log_message*
Is the message to be logged.

Log files and archived files

Log files are produced from a variety of Tivoli Workload Scheduler activities. Other activities produce files which are archived after they have been used. The details are given in Table 58:

Table 58. Log and trace file maintenance

Activity	Description	Location	Maintenance method
Process	Each Tivoli Workload Scheduler process logs its activities, writing them in log and trace message files:		
	<p>Log messages</p> <p>These are messages intended for use directly by you, and provide information, errors and warnings about the processes.</p>	<p>netman</p> <p><TWA_home>/TWS/stdlist/logs/ <yyyymmdd>_NETMAN.log</p> <p>Other processes</p> <p><TWA_home>/TWS/stdlist/logs/ <yyyymmdd>_TWSMERGE.log</p> <p>This is the default situation. You can set an option in the localopts file to create separate log files for the major processes.</p>	rmstdlist
	<p>Trace messages</p> <p>These are messages written when a problem occurs that you can probably not solve without the assistance of IBM Software Support.</p>	<p>netman</p> <p><TWA_home>/TWS/stdlist/traces/ <yyyymmdd>_NETMAN.log</p> <p>Other processes</p> <p><TWA_home>/TWS/stdlist/traces/ <yyyymmdd>_TWSMERGE.log</p> <p>This is the default situation. You can set an option in the localopts file to create separate trace files for the major processes.</p>	
Master domain manager job management	The job manager process on the master domain manager archives the previous period's Symphony file.	<TWA_home>/TWS/schedlog	Manual
Job	Each job that runs under Tivoli Workload Scheduler control creates an output file. These files are archived.	<TWA_home>/TWS/stdlist/<date> where <date> is in the format yyyy.mm.dd	rmstdlist

Maintaining file system

Table 58. Log and trace file maintenance (continued)

Activity	Description	Location	Maintenance method
Forecast and trial plan creation	The creation of forecast and trial plans writes to a log file.	Trial <TWA_home>/TWS/schedTrial/*.log Other processes <TWA_home>/TWS/schedForecast/*.log	Manual
Audit	The audit facility writes log files.	<TWA_home>/TWS/audit	Manual
DB2 UDB	DB2 logs its activities.	Information about the location and viewing method for DB2 log files is supplied in the DB2 documentation. Go to the Information Center for DB2 (see <i>Tivoli Workload Automation: Publications</i> for the link). The main file to control is the db2diag.log file, which is the most important DB2 diagnostic file, which, without intervention, grows endlessly with no reuse of wasted space. This does not apply, however, to the database log files used by Tivoli Workload Scheduler, which are set up for circular reuse of disk space, so they don't grow in size over a maximum value.	See the DB2 documentation.
Oracle database	Oracle logs its activities.	See the Oracle documentation.	See the Oracle documentation.
The embedded WebSphere Application Server	The application server writes log files	<TWA_home>/eWAS/ profiles/twaprofile/logs/ ffdc server1	Manual
Netcool SSM monitoring agent	The agent writes log files	<TWA_home>/ssm/Log/ ssmagent.log traps.log	Manual
Other	Other activities also write trace and log files.	<TWA_home>/TWS/methods	Manual

The easiest method of controlling the growth of these directories is to decide how long the log files are needed, then schedule a Tivoli Workload Scheduler job to remove any files older than the given number of days. Use the **rmstdlist** command for the process and job log files, and use a manual date check and deletion routine for the others. Make sure that no processes are using these files when you perform these activities.

See the *Tivoli Workload Scheduler: User's Guide and Reference* for full details of the **rmstdlist** command.

Note: The **rmstdlist** command might give different results on different platforms for the same scenario. This is because on UNIX platforms the command uses the **-mtime** option of the **find** command, which is interpreted differently on different UNIX platforms.

Temporary files

The Tivoli Workload Scheduler master domain manager uses temporary files, located in `<TWA_home>/TWS/tmp` or `/tmp` and named `TWS<XXXX>`, when compiling new production control databases. These files are deleted when compiling is complete.

This directory also contains the Tivoli Workload Scheduler installation files and log files.

Managing event message queue file sizes

This publication contains the following information with respect to managing event message queue file sizes:

- See “Planning space for queues” on page 153 to learn about planning space for message event queues (and also how to use `evtsize` to resize the queues)
- See “Managing the event processor” on page 263 to learn about managing the EIF event queue
- See “Disk Space” on page 287 to learn about the impacts that increased fault tolerance can have on message queues
- See “Workload spreading” on page 284 to learn about how to avoid bottlenecks in the `Mailbox.msg` queue.

Administrative tasks - DB2

This section describes how to perform some specific administrative tasks on DB2, as follows:

- “Changing DB2 passwords”
- “Locating the DB2 tools”
- “User permissions for running the DB2 tools” on page 210
- “Administering the DB2 maintenance feature” on page 210
- “Reorganizing the DB2 database” on page 212
- “Monitoring the lock list memory” on page 213

Changing DB2 passwords

To change passwords used by DB2 other than the `<TWS_user>` password or the passwords of the user IDs used by Tivoli Workload Scheduler to access the database (see “Changing key Tivoli Workload Scheduler passwords” on page 241) follow the instructions in the DB2 documentation; they do not directly impact Tivoli Workload Scheduler.

Locating the DB2 tools

Tivoli Workload Scheduler is supplied with a small set of tools that you use to perform the following administrative tasks for DB2:

- Run the DB2 statistics program, to maximize the performance of DB2 (`dbrunstats`). See “Running DB2 maintenance manually” on page 211 for a full description of how to use the tool.
- Reorganize the database (`dbreorg`). See “Reorganizing the DB2 database” on page 212 for a full description of how to use the tool.

Find these tools in the following directory:

`<TWA_home>/TWS/dbtools/db2/scripts`

Administrative tasks: DB2

Note: The tools in this directory include some that are for the use of IBM Software Support:

dbcatalog
dbsetup

Do not run these scripts. To do so might damage or overwrite the data in your database.

User permissions for running the DB2 tools

The DB2 tools must be run by a user who has the following permissions:

- DB2 administrator permissions – the user must be defined to DB2 as a DB2 Administrator
- Full access (777) to the Tivoli Workload Scheduler installation directory

Administering the DB2 maintenance feature

At installation, DB2 automatic maintenance is switched on, which means that DB2 periodically checks to see if it needs to collect new database statistics, so that it can perform the maintenance, adjusting the performance parameters to maximize performance.

This section describes how to administer the automatic maintenance, by changing how and when it is run, switching it off and on again, and running it manually. See the following:

- “Modifying the DB2 automatic maintenance policy”
- “Switching off automatic maintenance”
- “Switching on automatic maintenance” on page 211
- “Running DB2 maintenance manually” on page 211

Modifying the DB2 automatic maintenance policy

To know when and how the statistics used by the automatic maintenance must be collected, DB2 uses a default policy, which can be customized. The procedure is as follows:

1. Right-click the database in the DB2 Control Center and select **Configure Automatic Maintenance** from the context menu.
2. Follow the instructions in the wizard, modifying any of the default policy parameters that you think might improve the way DB2 chooses when to run the automatic maintenance.

Switching off automatic maintenance

If you want to take full manual control of the database, switch off the automatic maintenance as follows:

1. Check that the user who is going to run the procedure has the appropriate rights (see “User permissions for running the DB2 tools”)
2. On the DB2 server computer, open a DB2 shell, as follows:

UNIX

Follow these steps:

- a. Issue the command **su - db2inst1**, or change to the subdirectory `sql1ib` of the home directory of the owner of the DB2 instance (by default `db2inst1`)

- b. Launch the command `./db2profile`

Windows Select from the **Start** menu, **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**

3. Check that the command shell is correctly initialized by issuing the command `db2`, and checking that the command is recognized.
4. Issue the command `quit` to leave the DB2 Processor mode.
5. Issue the following command:
`db2 UPDATE DB CFG FOR <database_name> USING AUTO_MAINT OFF`
where `<database_name>` is the name of the Tivoli Workload Scheduler database (the installed default name is `TWS`; supply this value unless you have changed it).
6. To make the changes effective, either disconnect and reconnect all the DB2 clients, or restart the DB2 instance (using `db2stop` and `db2start`).

Switching on automatic maintenance

To switch the automatic maintenance back on again, do as follows:

1. Check that the user who is going to run the procedure has the appropriate rights (see “User permissions for running the DB2 tools” on page 210)
2. On the DB2 server computer, open a DB2 shell, as follows:

UNIX Follow these steps:

- a. Issue the command `su - db2inst1`, or change to the subdirectory `sql1lib` of the home directory of the owner of the DB2 instance (by default `db2inst1`)
- b. Launch the command `./db2profile`

Windows Select from the **Start** menu, **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**

3. Check that the command shell is correctly initialized by issuing the command `db2`, and checking that the command is recognized.
4. Issue the command `quit` to leave the DB2 Processor mode.
5. Issue the following command:
`db2 UPDATE DB CFG FOR <database_name> USING AUTO_MAINT ON`
where `<database_name>` is the name of the Tivoli Workload Scheduler database (the installed default name is `TWS`; supply this value unless you have changed it).
6. To make the changes effective, either disconnect and reconnect all the DB2 clients, or restart the DB2 instance (using `db2stop` and `db2start`).

Running DB2 maintenance manually

This section describes how to perform the DB2 maintenance process on demand, instead of waiting for DB2 to do it according to its automatic maintenance policy. The process is run by the tool `dbrunstats` which you can run whenever you need to, without stopping DB2 or interrupting its processing.

To run this tool, follow this procedure:

1. Locate the DB2 tools: see “Locating the DB2 tools” on page 209.
2. Check that the user who is going to run the procedure has the appropriate rights (see “User permissions for running the DB2 tools” on page 210)

Administering the DB2 maintenance feature

3. Open a DB2 shell, as follows:

- | | |
|----------------|---|
| UNIX | Follow these steps: <ol style="list-style-type: none">Issue the command su - db2inst1, or change to the subdirectory <code>sql1lib</code> of the home directory of the owner of the DB2 instance (by default <code>db2inst1</code>)Launch the command ./db2profile |
| Windows | Select from the Start menu, Programs → IBM DB2 → Command Line Tools → Command Window |

4. Check that the command shell is correctly initialized by issuing the command **db2**, and checking that the command is recognized.

5. Issue the command **quit** to leave the DB2 Processor mode.

6. From within the shell, change to the directory `<TWA_home>/TWS/dbtools/db2/scripts`

7. Run the script:

UNIX **dbrunstats.sh** *database* [*user* [*password*]]

Windows **dbrunstats** *database* [*user* [*password*]]

where:

database

The name of the database:

- If you are running this from the computer where the DB2 server is installed, the installed default name is `TWS`. Supply this value unless you have changed it.
- If you are running this from the computer where the DB2 client is installed, the installed default name is `TWS_DB`. Supply this value unless you have changed it.

user The DB2 administration user. If this is omitted the ID of the user running the command will be used.

password

The password of the DB2 administration user. If this is omitted it will be requested interactively.

The script runs, giving you various messages denoting its progress and successful conclusion. At the end (it is not particularly time-consuming) the database performance parameters have been reset to maximize performance.

Reorganizing the DB2 database

Using this tool, the database physically reorganizes the data tables and indexes, optimizing disk space usage and ease of data access. The process is time-consuming, requires that the database is backed up, and that Tivoli Workload Scheduler is stopped. However, at the end you have a database that is completely reorganized.

To reorganize the database follow this procedure:

1. Back up the Tivoli Workload Scheduler database. Use the method described in “Backing up the database to offline storage” on page 201.
2. Stop all Tivoli Workload Scheduler processes. See “Unlinking and stopping Tivoli Workload Scheduler” on page 250 for full details.

3. Check that the user who is going to run the procedure has the appropriate rights (see "User permissions for running the DB2 tools" on page 210)
4. Open a DB2 shell, as follows:

UNIX Follow these steps:

- a. Issue the command **su - db2inst1**, or change to the subdirectory `sql1ib` of the home directory of the owner of the DB2 instance (by default `db2inst1`)
- b. Launch the command **./db2profile**

Windows Select from the **Start** menu, **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**

5. Check that the command shell is correctly initialized by issuing the command **db2**, and checking that the command is recognized.
6. Issue the command **quit** to leave the DB2 Processor mode.
7. From within the shell, change to the directory `<TWA_home>/TWS/dbtools/db2/scripts`
8. Run the script:

UNIX **dbreorg.sh** *database* [*user* [*password*]]

Windows **dbreorg** *database* [*user* [*password*]]

where:

database

The name of the database:

- If you are running this from the computer where the DB2 server is installed, the installed default name is *TWS*. Supply this value unless you have changed it.
- If you are running this from the computer where the DB2 client is installed, the installed default name is *TWS_DB*. Supply this value unless you have changed it.

user The DB2 administration user. If this is omitted the ID of the user running the command will be used.

password

The password of the DB2 administration user. If this is omitted it will be requested interactively.

The script runs, giving you various messages denoting its progress and successful conclusion.

9. Restart Tivoli Workload Scheduler.

Monitoring the lock list memory

If the memory that DB2 allocates for its lock list begins to be fully used, DB2 can be forced into a "lock escalation", where it starts to lock whole tables instead of just individual table rows, and increasing the risk of getting into a deadlock.

This happens especially when there are long transactions, such as the creation or extension of a plan (production, trial, or forecast).

To avoid this problem occurring, set the automatic notification in the DB2 Health Center, so that you can be advised of any lock list problems building up.

However, if you think that deadlock situations have been occurring, follow this procedure to verify:

Monitoring the lock list memory

1. With the WebSphere Application Server active, log on as DB2 administrator to the DB2 server, for example,

```
su - db2inst1
```

2. Run the following command to determine where the Tivoli Workload Scheduler database is located:

```
db2 list active databases
```

The output might be as follows:

```
Database name           = TWS
Applications connected currently = 2
Database path           = /home/db2inst1/db2inst1/NODE0000/SQL00002/
```

3. Run:

```
cd <Database path>/db2event/db2detaildeadlock
```

4. Connect to the Tivoli Workload Scheduler database, for example:

```
db2 connect to TWS
```

5. Flush the event monitor that watches over deadlocks (active by default) with the following:

```
db2 flush event monitor db2detaildeadlock
```

6. Disconnect from the database with:

```
db2 terminate
```

7. Obtain the event monitor output with:

```
db2evmon -path . > deadlock.out
```

The file `deadlock.out` now contains the complete deadlock history since the previous flush operation.

8. To find out if there have been deadlocks and when they occurred, run:

```
grep "Deadlock detection time" deadlock.out
```

The output might be as follows:

```
Deadlock detection time: 11/07/2008 13:02:10.494600
Deadlock detection time: 11/07/2008 14:55:52.369623
```

9. But the fact that a deadlock occurred does not necessarily mean that the lock list memory is inadequate. For that you need to establish a relationship with lock escalation. To find out if there have been lock escalation incidents prior to deadlocks, run:

```
grep "Requesting lock as part of escalation: TRUE" deadlock.out
```

The output might be as follows:

```
Requesting lock as part of escalation: TRUE
Requesting lock as part of escalation: TRUE
```

If there has been lock escalation related to deadlocks, it is a good idea to modify the values of the following parameters.

LOCKLIST

This configures, in 4KB pages, the amount of memory allocated to locking management

MAXLOCKS

This configures the percentage of the memory that a single transaction can use, above which DB2 escalates, even though the memory might not be full

10. To determine the values currently being applied to the Tivoli Workload Scheduler database, do the following:

```
db2 get db cfg for TWS | grep LOCK
```

The output might be as follows:

Max storage for lock list (4KB)	(LOCKLIST) = 8192
Percent. of lock lists per application	(MAXLOCKS) = 60
Lock timeout (sec)	(LOCKTIMEOUT) = 180

The example shows the typical output for the Tivoli Workload Scheduler database if no modification has taken place to these values:

- "8192" = 4KB x 8192 pages = 32 MB of memory
 - "60" = 60% – the percentage of memory that a single transaction can occupy before triggering an escalation
 - "180" = 3 minutes of timeout for the period a transaction can wait to obtain a lock
11. The most straightforward action to take is to double the amount of memory to 64MB, which you do with the command:

db2 update db cfg for TWS using LOCKLIST 16384 immediate

12. Alternatively, you can set DB2 to automatically modify the LOCKLIST and MAXLOCKS parameters according to the amount of escalation being experienced and the available system memory. This self-tuning is a slow process, but adapts the database to the needs of the data and the available system configuration. It is done by setting the values of these parameters to AUTOMATIC, as follows:

db2 update db cfg for TWS using LOCKLIST AUTOMATIC immediate

DB2 responds with messages telling you that MAXLOCKS has also been set to AUTOMATIC:

```
SQL5146W "MAXLOCKS" must be set to "AUTOMATIC" when "LOCKLIST" is "AUTOMATIC".
```

```
"MAXLOCKS" has been set to "AUTOMATIC"
```

Note: The self-tuning facility is only available from V9.1 of DB2.

Administrative tasks - Oracle

This section describes how to perform some specific administrative tasks for the Oracle database.

- "Changing the Oracle access password"
- "Locating the Oracle tools"
- "Maintaining the Oracle database" on page 216
- "Obtaining information about the Tivoli Workload Scheduler databases installed on an Oracle instance" on page 216
- "User permissions for running the Oracle tools" on page 216
- "Changing the Oracle host name, port, or database name" on page 257

Changing the Oracle access password

This is described as part of the process of changing the password for a master domain manager or backup master domain manager. See "Changing key Tivoli Workload Scheduler passwords" on page 241.

Locating the Oracle tools

Tivoli Workload Scheduler is supplied with a small set of tools that you use to perform the following administrative tasks for Oracle:

Administrative tasks - Oracle

- Grant the user permissions for the Tivoli Dynamic Workload Console views (dbgrant). See the Tivoli Dynamic Workload Console online help for full details.
- Migrating from DB2 to Oracle or vice versa (prepareSQLScripts, createdb_root_ora, updateSetupCmdLine). See "Migrating data from DB2 to Oracle and *vice versa*" on page 217 for full details.

Locate these tools in the following directory:

```
<TWA_home>/TWS/dbtools/oracle/scripts
```

Note: The directory also includes some scripts that are only for the use of IBM Software Support:

```
dbmigrate
dbpartition
dbsetup
dbupgrade
launchdb_root_ora
_migratedb_root_ora
```

Do not run these scripts. To do so might damage or overwrite the data in your database.

Maintaining the Oracle database

Like DB2, Oracle has a routine that regularly maintains the database. Similarly, this too can be run manually. The tool is invoked as follows:

```
dbms_stats.gather_schema_stats<schema_owner>
```

See the Oracle documentation for full details of how and when to run it.

Obtaining information about the Tivoli Workload Scheduler databases installed on an Oracle instance

To determine which Tivoli Workload Scheduler databases are installed on an Oracle instance, do the following:

```
su - oracle (UNIX only)
sqlplus system/<system_password>@<service_name>
SQL> select * from all_tws_schemas;
```

The output should look like the following:

```
SCHEMA_NAME
-----
MDL
mdm85
TWS_user
```

Notes:

1. More than one instance of Tivoli Workload Scheduler can be shared in one instance of Oracle, using different schemas.
2. In Oracle, the concept of "schema" and "user" are the same, so dropping an Oracle schema means dropping an Oracle user, which you do as follows:
SQL> drop user MDL cascade;

User permissions for running the Oracle tools

The Oracle tools must be run by a user who has the following permissions:

- Oracle administrator permissions – the user must be defined to Oracle as an administrator
- Full access (777) to the Tivoli Workload Scheduler installation directory

Migrating data from DB2 to Oracle and *vice versa*

This section applies to Tivoli Workload Scheduler master domain managers and backup masters. It documents how to migrate the Tivoli Workload Scheduler data from one RDBMS to another.

There are two ways to accomplish the migration. You can use either way to migrate your data from DB2 to Oracle or vice versa.

Parallel data migration

The migration is between two instances of Tivoli Workload Scheduler, one that uses DB2 while the other uses Oracle

Reconfiguration

The database is migrated from one RDBMS support mechanism to another, and Tivoli Workload Scheduler instance is re-configured to point to a different database without installing another instance.

Note: Neither of these procedures migrate the following information from the source database:

- The pre-production plan
- The history of job runs and job statistics
- The state of running event rule instances. This means that any complex event rules, where part of the rule has been satisfied prior to the database migration, are generated after the migration as new rules. Even if the subsequent conditions of the event rule are satisfied, the record that the first part of the rule was satisfied is no longer available, so the rule will never be completely satisfied.

The following sections describe the migration procedures.

- “Parallel data migration from DB2 to Oracle”
- “Parallel data migration from Oracle to DB2” on page 219
- “Reconfiguration from DB2 to Oracle” on page 220
- “Reconfiguration from Oracle to DB2” on page 225

Parallel data migration from DB2 to Oracle

With the following steps all scheduling object definitions and global options can be migrated from the DB2 database of a Tivoli Workload Scheduler version 8.5 instance to the Oracle database of another instance.

1. Fresh-install another instance of a Tivoli Workload Scheduler version 8.5 master domain manager and make it point to an Oracle database.
2. Use **composer**, the Tivoli Dynamic Workload Console, or the Job Scheduling Console to define this instance as a fault-tolerant agent in the database of the current master domain manager that points to DB2.
3. On the master domain manager that points to DB2 run the `dataexport` command or script to export all scheduling object definitions and global options from DB2. Find this file in the `bin` subdirectory of the Tivoli Workload Scheduler home directory.

Run `dataexport` from a Windows or UNIX command prompt as follows:

Parallel data migration from DB2 to Oracle

```
dataexport <source_dir> <export_dir>
```

where:

source_dir

The installation directory of the instance of Tivoli Workload Scheduler version 8.5 that points to the DB2 database.

export_dir

The directory where the export files are to be created.

For example:

```
dataexport.cmd F:\TWS85\twsDB2user F:\TWS85\export
```

The object definitions and the global options are retrieved from the DB2 database and placed in the F:\TWS85\export directory.

4. Verify the following files were created in export_dir:

- calendars.def
- jobs.def

Note: The record length supported by DB2 is 4095 characters, but it decreases to 4000 characters with Oracle. When you migrate your job definitions to Oracle, any job scripts or commands exceeding 4000 characters in length are not migrated. In such case, the data import utility replaces the job definition with a dummy job definition and sets the job priority to 0, guaranteeing that successors are not run.

- globalOpts.def
- erules.def
- parms.def
- prompts.def
- resources.def
- scheds.def
- topology.def
- users.def (includes encrypted user passwords)

5. On the master domain manager that points to DB2 do the following:

- a. Ensure that the carry forward option is set to ALL. Run:

```
optman chg cf=ALL
```
- b. Add the new instance (that you installed in step 1 and that you momentarily defined as a fault-tolerant agent) in the current plan. To do this, run:

```
JnextPlan -for 0000
```
- c. Check that the new instance was linked by running:

```
conman sc
```

6. On the new instance run the dataimport command or script to import all scheduling object definitions and global options to the Oracle database. Find this file in the bin subdirectory of the Tivoli Workload Scheduler home directory.

Run dataimport from a Windows or UNIX command prompt as follows:

```
dataimport <source_dir> <export_dir>
```

where:

source_dir

The installation directory of the new instance of Tivoli Workload Scheduler version 8.5 pointing to the Oracle database.

export_dir

The directory from where the export files are to be read from. This directory is the same export_dir directory specified for dataexport.

For example:

```
dataimport.cmd F:\TWS85\twsORACLEuser F:\TWS85\export
```

The object definitions and the global options are retrieved from the F:\TWS85\export directory and stored in the Oracle database.

7. On the master domain manager that points to DB2 run the conman switchmgr command to make the Tivoli Workload Scheduler instance pointing to Oracle as the acting master domain manager. For information on this command see *User's Guide and Reference*.

You have now completed the data migration steps.

Parallel data migration from Oracle to DB2

With the following steps all scheduling object definitions and global options can be migrated from the Oracle database of a Tivoli Workload Scheduler version 8.5 instance to the DB2 database of another instance (freshly installed or upgraded to version 8.5).

1. On the current master domain manager pointing to the Oracle database run the dataexport command or script to export all scheduling object definitions and global options. Find this file in the bin subdirectory of the Tivoli Workload Scheduler home directory.

Run dataexport from a Windows or UNIX command prompt as follows:

```
dataexport <source_dir> <export_dir>
```

where:

source_dir

The installation directory of the instance of Tivoli Workload Scheduler that points to the Oracle database.

export_dir

The directory where the export files are to be created.

For example:

```
dataexport.cmd F:\TWS85\twsORACLEuser F:\TWS85\export
```

The object definitions and the global options are retrieved from the Oracle database and placed in the F:\TWS85\export directory.

2. Verify the following files were created in export_dir:
 - calendars.def
 - erules.def
 - jobs.def
 - globalOpts.def
 - parms.def
 - prompts.def
 - resources.def
 - scheds.def
 - topology.def
 - users.def (includes encrypted user passwords)
3. Fresh-install another instance of Tivoli Workload Scheduler version 8.5 or upgrade an existing instance to version 8.5 making it point to a DB2 database.

Parallel data migration from Oracle to DB2

4. Use composer, the Tivoli Dynamic Workload Console, or the Job Scheduling Console to define this instance as a fault-tolerant agent in the database of the master domain manager that points to Oracle.
5. On the current master domain manager that points to Oracle do the following:
 - a. Ensure that the carry forward option is set to ALL. Run:

```
optman chg cf=ALL
```
 - b. Add the new instance (that you installed in step 3 and that you momentarily defined as a fault-tolerant agent) in the current plan. To do this, run:

```
JnextPlan -for 0000
```
 - c. Check that the new instance was linked by running:

```
conman sc
```
6. On the new instance run the `dataimport` command or script to import all scheduling object definitions and global options to DB2. Find this file in the `bin` subdirectory of the Tivoli Workload Scheduler home directory.
Run `dataimport` from a Windows or UNIX command prompt as follows:

```
dataimport <source_dir> <export_dir>
```

where:

source_dir

The installation directory of the instance of Tivoli Workload Scheduler that points to the DB2 database.

export_dir

The directory from where the export files are to be read from. This directory is the same `export_dir` directory specified for `dataexport`.

For example:

```
dataimport.cmd F:\TWS85\twsDB2user F:\TWS85\export
```

The object definitions and the global options are retrieved from the `F:\TWS85\export` directory and stored in the DB2 database.

7. On the master domain manager that points to Oracle run the `conman switchmgr` command to make the Tivoli Workload Scheduler instance pointing to DB2 as the acting master domain manager. For information on the `switchmgr` command see *User's Guide and Reference*.

You have now completed the data migration steps.

Reconfiguration from DB2 to Oracle

With the following steps all scheduling object definitions and global options can be migrated from the DB2 database of a Tivoli Workload Scheduler version 8.5 master domain manager and made to point to an Oracle database.

1. Run the `dataexport` command or script to export all scheduling object definitions and global options from DB2. Find this file in the `bin` subdirectory of the Tivoli Workload Scheduler version 8.5 home directory.
Run `dataexport` from a Windows or UNIX command prompt as follows:

```
dataexport <source_dir> <export_dir>
```

where:

source_dir

The Tivoli Workload Scheduler version 8.5 installation directory.

export_dir

The directory where the export files are to be created.

For example:

```
dataexport.cmd F:\TWS85\tws85user F:\TWS85\tws85user\export
```

The object definitions and the global options are retrieved from the DB2 database and placed in the F:\TWS85\tws85user\export directory.

2. Verify the following files were created in export_dir:
 - calendars.def
 - erules.def
 - jobs.def
 - globalOpts.def
 - parms.def
 - prompts.def
 - resources.def
 - scheds.def
 - topology.def
 - users.def (includes encrypted user passwords)
3. Stop the WebSphere Application Server using the **conman stopappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
4. Run prepareSQLScripts.bat (.sh) to customize the SQL scripts with the parameters needed to create the Tivoli Workload Scheduler schema in the Oracle database. Find this file in the *TWA_home/TWS/dbtools/oracle/scripts* directory.

Run prepareSQLScripts from a Windows or UNIX command prompt as follows:

- From a UNIX shell run:

```
prepareSQLScripts
  -dbRoot <dbRoot>
  -dbName <dbName>
  -twsDbUser <twsDbUser>
  -twsDbPassword <twsDbUser_password>
  [-tempDir <tempDir>]
  [-dataTablespace <dataTablespace_name>]
  [-logTablespace <logTablespace_name>]
  [-tempTablespace <tempTablespace_name>]
  [-companyName <companyName>]
  [-masterDmName <masterDmName>]
  [-eifPort <eifPort>]
```

- From a Windows command prompt run:

```
cmd /K prepareSQLScripts
  -dbRoot <dbRoot>
  -dbName <dbName>
  -twsDbUser <twsDbUser>
  -twsDbPassword <twsDbUser_password>
  [-tempDir <tempDir>]
  [-dataTablespace <dataTablespace_name>]
  [-logTablespace <logTablespace_name>]
  [-tempTablespace <tempTablespace_name>]
  [-companyName <companyName>]
  [-masterDmName <masterDmName>]
  [-eifPort <eifPort>]
```

where:

Reconfiguration from DB2 to Oracle

dbRoot

The path where the RDBMS software is installed - the *Oracle home* directory.

dbName

The name of the database.

twSdbUser

The database user for Tivoli Workload Scheduler.

twSdbPassword

The password of this user.

tempDir

The directory where the temporary files created by this process are placed. On Windows the default is *<drive>\Documents and Settings\<current_user>\Local Settings\Temp*.

dataTablespace

The name of the table space for the Tivoli Workload Scheduler data. The default is USERS. If you provided a different value at installation time, enter that value again.

logTablespace

The name of the table space for the Tivoli Workload Scheduler log. The default is USERS. If you provided a different value at installation time, enter that value again.

tempTablespace

The name of the table space for temporary data. The default is TEMP. If you provided a different value at installation time, enter that value again.

companyName

The name of your company. The default is MYCOMPANY. If you provided a different value at installation time, enter that value again.

masterDmName

The name of the master domain. The default is MASTERDM. If you provided a different value at installation time, enter that value again.

EIFPort

The EIF port. The default is 31123. If you provided a different value at installation time, enter that value again.

For example:

```
cmd /K prepareSQLScripts.bat -dbRoot D:\Oracle
                                -dbName TWS
                                -twSdbUser tws85User
                                -twSdbPassword mypassw0rd
```

The SQL scripts are customized to create an Oracle schema named tws85user on the TWS database. The names of the table spaces are the defaults: USERS and TEMP.

5. Run `createdb_root.bat (.sh)` to create the database and schema following the specifications of the previous step. Find this file in the *tempDir/TWA/tws85/scripts* directory, where *tempDir* is the parameter you specified in step 4.

Run `createdb_root` as follows:

- From a UNIX shell, run:

```
createdb_root
<netService>
<oracleAdmin>
<oracleAdminPassword>
<twSdbUser>
<twSdbPassword>
<isBackupManager>
<isPartitioned>
```

- From a Windows command prompt, run:

```
cmd /K createdb_root
<netService>
<oracleAdmin>
<oracleAdminPassword>
<twSdbUser>
<twSdbPassword>
<isBackupManager>
<isPartitioned>
```

where:

netService

The net service name required to connect to the Oracle database.

oracleAdmin

The user ID of the Oracle database administrator.

oracleAdminPassword

The password for oracleAdmin.

twSdbUser

The owner of the Tivoli Workload Scheduler schema that you specified also in step 4.

twSdbPassword

The password for twSdbUser that you specified also in step 4.

isBackupManager

Specify TRUE if you are migrating a backup master domain manager. Specify FALSE otherwise.

isPartitioned

Specify TRUE if the Oracle Partitioning feature is enabled for the database. Specify FALSE otherwise.

For example:

```
createdb_root TWS SYSTEM passw1rd tws85user passw0rd FALSE TRUE
```

creates a database schema named tws85user on the TWS database.

If something goes wrong and you have to rerun this step after finding (and fixing) the error, you must first log in to the database as the administrator and drop the *twSdbUser* (tws85user in the example).

6. Change the data source properties from DB2 to Oracle using the `changeDataSource.bat` (.sh) command or script to switch the data source in WebSphere Application Server from DB2 to Oracle. See “Changing data source properties” on page 252 for details on how to use the command.

- a. Clear the values of the following properties:

```
DB2Type4JndiName
DB2Type4DatabaseName
DB2Type4ServerName
DB2Type4PortNumber
```

Reconfiguration from DB2 to Oracle

Note: For the property DB2Type4JndiName replace the value with any character or string to nullify it.

- b. Set these properties to the following values:

```
OracleType2JndiName=jdbc/twsdb
OracleType2DatabaseName=the Oracle instance name
OracleType2PortNumber=the Oracle listener port number
```

- c. Set the JDBC driver path for the Oracle database to
ORACLE_JDBC_DRIVER_PATH=*Oracle_home*/jdbc/lib and the Oracle instance
type and name to ORACLETYPE2URL=JDBC:ORACLE:OCI:@*instance_name*

7. Use the changeSecurityProperties.bat (.sh) command or script to change the following security settings (see “Changing security properties” on page 259 for details on using the command):

j2cUserid

Write the value you used for twsDbUser in prepareSQLScripts.

j2cPassword

Write the password for the twsDbUser.

Note: After you updated these two values, make sure that you also erase all the other lines in the security properties file before you export it again with the changeSecurityProperties command. Failure to do so will result in all the passwords contained in the file being saved as strings of asterisks (*).

8. Modify the TWSConfig.properties file located in the *TWA_home*/eWAS/profiles/twaprofile/properties directory. Take the comment marks off the following lines and edit them as shown:

```
com.ibm.tws.dao.rdbms.rdbmsName = Oracle
com.ibm.tws.dao.rdbms.modelSchema = <twsDbUser>
com.ibm.tws.dao.rdbms.eventRuleSchema=<twsDbUser>
com.ibm.tws.dao.rdbms.logSchema=<twsDbUser>
```

where twsDbUser is the owner of the Tivoli Workload Scheduler schema that you specified also in the previous steps.

9. For UNIX only: run the updateSetupCmdLine.sh command or script to set the paths for the new database in the WebSphere Application Server profile. Locate this script in the *<TWA_home>*/TWS/dbtools/oracle/scripts directory. The syntax is as follows:

```
updateSetupCmdLine.sh -installRoot <TWA_home> -dbRoot <DB_home>
```

where:

-installRoot *<TWA_home>*

The installation directory of Tivoli Workload Scheduler.

-dbRoot *<DB_home>*

The installation directory of the database.

10. Start the WebSphere Application Server using the **conman startappserver** command (see “Starting and stopping the application server and appservman” on page 268)
11. Manually copy the master domain manager definition from the topology.def file you exported in step 1 (you can find it in export_dir) and use composer new to add it in the Oracle database.
12. Run dataimport to import all scheduling object definitions and global options to the Oracle database. Find this file in the bin subdirectory of the Tivoli Workload Scheduler home directory.

Reconfiguration from DB2 to Oracle

Run `dataimport` from a Windows or UNIX command prompt as follows:

```
dataimport source_dir export_dir
```

where:

source_dir

The Tivoli Workload Scheduler installation directory.

export_dir

The directory from where the export files are to be read from. This directory is the same `export_dir` directory specified for `dataexport`.

For example:

```
dataimport.cmd F:\TWS85\tws85user F:\TWS85\tws85user\export
```

The object definitions and the global options are retrieved from the `F:\TWS85\tws85user\export` directory and stored in the new Oracle database.

13. Run the following command to set the carry forward option to ALL:

```
optman chg cf=ALL
```

14. Update the Symphony file by creating a plan with 0 extension period that begins at the end of the current plan:

```
JnextPlan -from start_time -for 0000
```

where *start_time* is the date and time when the current plan ends.

You have now completed the reconfiguration steps.

To migrate a backup master domain manager, perform the following steps:

- Stop the WebSphere Application Server using the **conman stopappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
- Run steps 6, 7, 9
- Start the WebSphere Application Server using the **conman startappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
- Set the `isBackupManager` parameter of the `createdb_root` command (script) to TRUE.

Reconfiguration from Oracle to DB2

With the following steps all scheduling object definitions and global options can be migrated from the Oracle database of a Tivoli Workload Scheduler version 8.5 master domain manager and made to point to a DB2 database.

1. Run the `dataexport` command or script to export all scheduling object definitions and global options from Oracle. Find this file in the `bin` subdirectory of the Tivoli Workload Scheduler version 8.5 home directory.

Run `dataexport` from a Windows or UNIX command prompt as follows:

```
dataexport <source_dir> <export_dir>
```

where:

source_dir

The Tivoli Workload Scheduler version 8.5 installation directory.

Reconfiguration from Oracle to DB2

export_dir

The directory where the export files are to be created.

For example:

```
dataexport.cmd F:\TWS85\tws85user F:\TWS85\tws85user\export
```

The object definitions and the global options are retrieved from the Oracle database and placed in the F:\TWS85\tws85user\export directory.

2. Verify the following files were created in export_dir:
 - calendars.def
 - erules.def
 - jobs.def
 - globalOpts.def
 - parms.def
 - prompts.def
 - resources.def
 - scheds.def
 - topology.def
 - users.def (includes encrypted user passwords)
3. Stop WebSphere Application Server as described in “Application server - starting and stopping” on page 264.
4. Run prepareSQLScripts.bat (.sh) to customize the SQL scripts with the parameters needed to create the Tivoli Workload Scheduler database in DB2. Find this file in the *TWA_home*/TWS/dbtools/db2/scripts directory.

Run prepareSQLScripts from a Windows or UNIX command prompt as follows:

- From a UNIX shell run:

```
prepareSQLScripts
  -dbRoot <dbRoot>
  -dbName <dbName>
  -dbLocalAdmin <dbLocalAdmin>
  -twsDbUser <twsDbUser>
  [-tempDir <tempDir>]
  [-dataTablespace <dataTablespace_name>]
  [-dataTablespacePath <dataTablespacePath>]
  [-logTablespace <logTablespace_name>]
  [-logTablespacePath <logTablespacePath>]
  [-tempTablespace <tempTablespace_name>]
  [-userTempTablespace <userTempTablespace_name>]
  [-companyName <companyName>]
  [-masterDmName <masterDmName>]
  [-eifPort <eifPort>]
```

- From a Windows command prompt run:

```
cmd /K prepareSQLScripts
  -dbRoot <dbRoot>
  -dbName <dbName>
  -dbLocalAdmin <dbLocalAdmin>
  -twsDbUser <twsDbUser>
  [-tempDir <tempDir>]
  [-dataTablespace <dataTablespace_name>]
  [-dataTablespacePath <dataTablespacePath>]
  [-logTablespace <logTablespace_name>]
  [-logTablespacePath <logTablespacePath>]
  [-tempTablespace <tempTablespace_name>]
  [-userTempTablespace <userTempTablespace_name>]
  [-companyName <companyName>]
  [-masterDmName <masterDmName>]
  [-eifPort <eifPort>]
```

where:

dbRoot

The path where the RDBMS software is installed.

dbName

The name of the database.

dbLocalAdmin

The user ID of the local database administrator.

twSdbUser

The database user for Tivoli Workload Scheduler.

tempDir

The directory where the temporary files created by this process are placed. On Windows the default is <drive>\Documents and Settings\<current_user>\Local Settings\Temp.

dataTablespace

The name of the table space for the Tivoli Workload Scheduler data. The default is TWSDATA. If you provided a different value at installation time, enter that value again.

dataTablespacePath

The path of the table space for the Tivoli Workload Scheduler data.

logTablespace

The name of the table space for the Tivoli Workload Scheduler log. The default is TWSLOG. If you provided a different value at installation time, enter that value again.

logTablespacePath

The path of the table space for the Tivoli Workload Scheduler log files.

tempTablespace

The name of the table space for temporary data. The default is TEMP. If you provided a different value at installation time, enter that value again.

userTempTablespace

The name of the table space for temporary user data. The default is USERTEMP. If you provided a different value at installation time, enter that value again.

companyName

The name of your company. The default is MYCOMPANY. If you provided a different value at installation time, enter that value again.

masterDmName

The name of the master domain. The default is MASTERDM. If you provided a different value at installation time, enter that value again.

EIFPort

The EIF port. The default is 31123. If you provided a different value at installation time, enter that value again.

For example:

```
cmd /K prepareSQLScripts.bat -dbRoot D:\DB2
                                -dbName TWS
                                -dbLocalAdmin db2admin
                                -twSdbUser tws85User
```

Reconfiguration from Oracle to DB2

The SQL scripts are customized to create a database named TWS in DB2 for user `tw85user`. The names of the table spaces are the defaults: `TWSDATA`, `TWSLOG`, `TEMP` and `USERTEMP`.

5. Run `createdb_root.bat (.sh)` to create the database following the specifications of the previous step. Find this file in the `tempDir/TWA/tws85/scripts` directory, where `tempDir` is the parameter you specified in step 4.

Run `createdb_root` as follows:

- From a UNIX shell, run:

```
createdb_root
<dbName>
<isClientInstallation>
<dbNameNodeName>
<hostName>
<srvPortNumber>
<db2Admin>
<db2AdminPwd>
<instanceName>
<isBackupManager>
```

- From a Windows command prompt, run:

```
cmd /K createdb_root
<dbName>
<isClientInstallation>
<dbNameNodeName>
<hostName>
<srvPortNumber>
<db2Admin>
<db2AdminPwd>
<instanceName>
<isBackupManager>
```

where:

dbName

The name of the DB2 database. The maximum length is 5 characters.

isClientInstallation

The value is:

- TRUE if the database is a DB2 client.
- FALSE if the database is a DB2 server.

dbNameNodeName

The name of the DB2 node.

hostName

The host name of the computer where DB2 is to be installed.

srvPortNumber

The TCP/IP port number used to communicate with the DB2 server.
The default is 50000.

db2Admin

The user ID of the DB2 administrator.

db2AdminPwd

The password for `db2Admin`.

instanceName

The name of the DB2 server instance.

isBackupManager

Specify TRUE if you are migrating a backup master domain manager.
Specify FALSE otherwise.

For example:

```
createdb_root TWS FALSE TWS_ND myhost 50000 db2admin passw1rd DB2 FALSE
```

creates a database named TWS on a DB2 server instance named DB2.

6. Use the `changeDataSource.bat (.sh)` command or script to switch the data source in WebSphere Application Server from Oracle to DB2.
See “Changing data source properties” on page 252 for details on how to use the command.
 - a. Clear the following properties:
 - OracleType2JndiName
 - OracleType2DatabaseName
 - OracleType2ServerName
 - OracleType2PortNumber
 - b. Set the following properties:
 - DB2Type4JndiName
 - DB2Type4DatabaseName
 - DB2Type4ServerName
 - DB2Type4PortNumber
 - c. Set the JDBC driver path for the DB2 in both `DB2_JDBC_DRIVER_PATH` and `DB2UNIVERSAL_JDBC_DRIVER_PATH` (the path is the same for both properties).
7. Reset to a name of your choice the `...JndiName` property of the RDBMS from which you are changing.
8. Set to `jdbctwsdb` the `...JndiName` property of the new RDBMS
 - See that the following properties are set:
 - For DB2:
 - DB2Type4JndiName
 - DB2Type4DatabaseName
 - DB2Type4ServerName
 - DB2Type4PortNumber
9. Run the `changeSecurityProperties.bat (.sh)` command or script to change the following security settings:

j2cUserId

Write the value you used for `twsDbUser` in `prepareSQLScripts`.

j2cPassword

Write the password for `twsDbUser`.

Note: After you updated these two values, make sure that you also erase all the other lines in the security properties file before you export it again with the `changeSecurityProperties` command. Failure to do so will result in all the passwords contained in the file being saved as strings of asterisks (*).

See “Changing security properties” on page 259 for details.

10. Modify the `TWSConfig.properties` file located in the `TWA_home/eWAS/profiles/twaprofile/properties` directory. Comment the following four lines:


```
com.ibm.tws.dao.rdbms.rdbmsName = Oracle
com.ibm.tws.dao.rdbms.modelSchema = <twsDbUser>
com.ibm.tws.dao.rdbms.eventRuleSchema
com.ibm.tws.dao.rdbms.logSchema
```

Reconfiguration from Oracle to DB2

where `twSdbUser` is the owner of the Tivoli Workload Scheduler Oracle schema.

11. For UNIX only: run the `updateSetupCmdLine.sh` command or script to set the paths for the new database. Locate this script in the `<TWA_home>/TWS/dbtools/db2/scripts` directory. The syntax is as follows:

```
updateSetupCmdLine.sh -installRoot <TWA_home> -dbRoot <DB_home>
```

where:

-installRoot `<TWA_home>`

The installation directory of Tivoli Workload Scheduler.

-dbRoot `<DB_home>`

The installation directory of the database.

12. Start the WebSphere Application Server using the **conman startappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
13. Manually copy the master domain manager definition from the `topology.def` file you exported in step 1 (you can find it in `export_dir`) and use `composer new` to add it in the DB2 database.
14. Run `dataimport` to import all scheduling object definitions and global options to DB2. Find this file in the `bin` subdirectory of the Tivoli Workload Scheduler home directory.

Run `dataimport` from a Windows or UNIX command prompt as follows:

```
dataimport source_dir export_dir
```

where:

source_dir

The Tivoli Workload Scheduler installation directory.

export_dir

The directory from where the export files are to be read from. This directory is the same `export_dir` directory specified for `dataexport`.

For example:

```
dataimport.cmd F:\TWS85\tws85user F:\TWS85\tws85user\export
```

The object definitions and the global options are retrieved from the `F:\TWS85\tws85user\export` directory and stored in the new DB2 database.

15. Run the following command to set the carry forward option to ALL:

```
optman chg cf=ALL
```
16. Update the Symphony file by creating a plan with 0 extension period that begins at the end of the current plan:

```
JnextPlan -from start_time -for 0000
```

where `start_time` is the date and time when the current plan ends.

You have now completed the reconfiguration steps.

To migrate a backup master domain manager, perform the following steps :

- Stop the WebSphere Application Server using the **conman stopappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
- Perform steps 6 to 11.

- Start the WebSphere Application Server using the **conman startappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
- Set the `isBackupManager` parameter of the `createdb_root` command (script) to `TRUE`.

Upgrading your database

If you want to upgrade your database, change the instance owner, or relocate it to a different host, the procedure for upgrading your database, changing the instance owner, or relocating it, is as follows:

1. If you are changing DB2, check the *node directory* and *database directory* and make a note of the current configuration. To do this, issue the following commands at the DB2 command-line:

```
db2 list node directory show detail
```

```
db2 list database directory
```

where the `show detail` attribute is specified to give the full information in the directory.

Make a note of the displayed details.

2. Stop the application server, using the command


```
stopWas -direct -user <user> -password <password>
```
3. Make the upgrade, instance owner change, or relocation, of the database following the instructions from your database supplier.
4. If you have changed the database host, port, or database name, you will need to update the application server's data source properties, as described in “Changing the database host name, port, or database name” on page 251.
5. If you have changed the database access credentials, you will need to update the application server's security properties, as described in “Changing the security settings” on page 258.
6. Reconfigure the database for Tivoli Workload Scheduler, as follows:

DB2

- a. Check the *node directory* and *database directory*, as you did in step 1
- b. If necessary, modify the data displayed by these commands to match the data you noted in step 1. If you are not certain of how to do this, contact IBM Software Support for assistance.

Oracle Check the Oracle Listener and make sure that the service name is correctly specified.

7. Restart the database.
8. Restart the application server, using the command:


```
startWas -direct -user <user> -password <password>
```

Chapter 9. Administrative tasks

This chapter describes how to perform some specific administrative tasks on Tivoli Workload Scheduler, as follows:

The tasks

“Changing a domain manager” on page 234

Change a domain manager, either in the event of the failure of the computer where it is installed, or as part of a planned replacement activity.

“Changing a master domain manager” on page 236

Change a master domain manager, either in the event of the failure of the computer where it is installed, or as part of a planned replacement activity.

“Changing key Tivoli Workload Scheduler passwords” on page 241

Change the password of the TWS_user, or any other of the users that have an infrastructure role in Tivoli Workload Scheduler.

“Unlinking and stopping Tivoli Workload Scheduler” on page 250

The correct procedure to unlink the master domain manager from its agents and stop the master processing.

“Changing the database host name, port, or database name” on page 251

If you need to change the host, port or name of the database, effect the change in the application server, where the data source configuration is maintained.

“Changing the security settings” on page 258

“Managing the event processor” on page 263

If you are using event-driven workload automation, you will need to perform periodic maintenance on the event processor.

“Starting, stopping and displaying dynamic workload broker status” on page 264

The procedure to start or stop dynamic workload broker.

Application server tasks

The following tasks might need to be performed on the application server:

“Application server - starting and stopping” on page 264

How to stop and start the application server when you need to.

“Application server - automatic restart after failure” on page 266

The application server is managed by a utility that restarts it if it stops for any reason (subject to a configurable policy). This section describes how to modify the policy and deal with any situations that the policy cannot handle.

“Application server - encrypting the profile properties files” on page 269

Several of the application server configuration files contain passwords. To avoid that these remain in the files in plain text, run a utility to encrypt them.

“Application server - updating the Windows services after modifications” on page 270

On Windows, after changing certain data you must also update the Windows service that runs the embedded WebSphere Application Server.

“Application server - configuration files backup and restore” on page 272

The application server configuration manages the data source and security aspects of your Tivoli Workload Scheduler environment. The files should be regularly backed up and when necessary can be restored.

“Application server - changing the host name or TCP/IP ports” on page 273

If you need to change the host or ports used by the application server, follow the correct procedure.

“Application server - changing the trace properties” on page 276

The application server has a trace facility. This section describes how to increase the trace level to obtain more information for troubleshooting, and how to reduce the level to improve performance.

Changing the application server properties

Several of the above tasks require you to run a common procedure whereby you:

1. Run a utility that displays a set of current application server properties and saves them to a file
2. Edit the file to change the properties
3. Run another procedure to update the application server with the changed properties

This procedure is fully described in “Application server - using the utilities that change the properties - reference” on page 277

Application server utilities background information

Some background information on the application server utilities is also provided in “Application server - background information about the utilities” on page 278. For further reading, see *IBM Redbooks®: WebSphere Application Server V6 System Management & Configuration Handbook*.

Changing a domain manager

A domain manager might need to be changed because you want it to run on a different workstation, or it might be forced on you as the result of network linking problems or the failure of the domain manager workstation itself. This section, and its subsections, describes how to prepare for and use a backup domain manager, and applies to all domain managers. However, if the domain manager to be changed is a master domain manager, there are some specific additional steps to perform; see “Changing a master domain manager” on page 236.

Running without a domain manager has the following effects:

- Agents and subordinate domain managers cannot resolve inter-workstation dependencies, because activity records broadcast by the master domain manager are not being received.

- The upward flow of events is interrupted. This impacts events that report the status of jobs, job streams and dependencies defined on workstations in the Tivoli Workload Scheduler network hierarchy under the failed domain manager.
- Standard agents that are hosted by the failed domain manager cannot perform any processing, since they depend on the domain manager for all scheduling and job launching.

If the problem is expected to be of short duration, you can wait for the problem to be resolved and Tivoli Workload Scheduler will recover on its own, as described in the *Tivoli Workload Scheduler: Troubleshooting Guide* in the section about network linking problems. If you are uncertain about the duration, or if you want to restore normal agent operation, you must switch to a backup, as described in the following sections.

Choosing a backup domain manager

Being prepared for network problems makes recovery easier. Set up a backup domain manager for each domain manager in your network to more easily ensure that Tivoli Workload Scheduler peak job scheduling loads are met. Choose any fault-tolerant agent in the domain to be a backup domain manager.

Setting up a backup domain manager

Make certain that the *FullStatus* mode is selected in the backup domain manager's workstation definition.

Also ensure that the backup domain manager is synchronized with respect to time with the domain manager. The securest way is to use a Network Time Protocol Server to control the time on both systems, with the same repeat interval.

Network security

Network security is enforced using IP address validation. As a consequence, workstation linking (*autolink* option or **link** command) might fail if an agent has an old Symphony file that does not contain the new domain manager. If a connection fails, remove the old Symphony file on the agent and retry the connection.

Switching a domain manager

Use one of these procedures when you have a short-term loss of a domain manager.

Using the command line

See the procedure described under the **switchmgr** command in the *Tivoli Workload Scheduler: User's Guide and Reference*.

Using the Tivoli Dynamic Workload Console

1. Launch the Tivoli Dynamic Workload Console
2. Connect to the engine of the current domain manager
3. Select **All workstations in Plan**
4. Select the workstation you want to become the domain manager
5. Click the **More** action
6. Select **Become Domain Manager**.
7. Click **OK**.

Changing domain manager

Using the Job Scheduling Console

1. Launch the Job Scheduling Console.
2. Connect to the engine of the current domain manager
3. Select **Default Plan List**.
4. Click Status of All Domains.
5. Click **refresh**.
6. Select **Switch Manager**.
7. Specify the name of the backup domain manager you want to use.
8. Click **OK**.

Domain managers remain switched until you perform another switch manager operation, or run **JnextPlan**. To return to the original domain manager without running **JnextPlan**, repeat this procedure.

Changing a master domain manager

If you lose or want to plan to change a master domain manager, all the comments in the section “Changing a domain manager” on page 234 apply, but in addition consider the following:

Choosing a workstation for backup master domain manager

Since you must transfer files between the master domain manager and its backup, the workstations must have compatible operating systems. Do not combine UNIX with Windows workstations, and in UNIX, do not combine big-endian workstations (HP-UX, Solaris, and AIX[®]) with little-endian workstations (most Intel-based operating systems, including Windows and Linux).

See the *Tivoli Workload Scheduler: Planning and Installation Guide* for details of the prerequisite requirements of a backup master domain manager.

Promoting an agent to backup master domain manager

It is the normal process to install a backup master domain manager when you set up your scheduling network. However, if you have not done so, and decide later that you need a backup master domain manager, you have two options:

- Install a backup master domain manager on a system that is not currently in the workload scheduling network. Follow the instructions in the *Tivoli Workload Scheduler: Planning and Installation Guide*
- Promote an agent to backup master domain manager. This option is time-consuming and requires you to interrupt your workload scheduling activities, but if you want to do it, follow the procedure described in this section

You *cannot* promote an agent to backup master domain manager, using a command or procedure that allows continuity of workload scheduling activities.

Instead, if you need to change an agent workstation to become the backup master domain manager, you must interrupt the workload scheduling activities. The procedure is as follows:

1. Check that the workstation satisfies the prerequisites for a backup master domain manager
2. If it does, stop and disable all workload scheduling operations on the workstation

3. Uninstall the agent, following the instructions in the *Tivoli Workload Scheduler: Planning and Installation Guide*
4. Install the backup master domain manager on the system where the agent was installed, following the instructions in the *Tivoli Workload Scheduler: Planning and Installation Guide*
5. Ensure that the database entry for the workstation is correct for a backup master domain manager (see the *Tivoli Workload Scheduler: User's Guide and Reference* for information about the workstation definition)
6. Define and start any workload scheduling operations you require on the workstation in its new role.

Setting up a backup master domain manager

Ensure that the master domain manager and the backup master domain manager have *FullStatus* turned on in the workstation definition. This is important if you need to resort to long-term recovery, where the backup master domain manager generates a Symphony file (runs JnextPlan). If *FullStatus* is not turned on, the former master domain manager shows up as a regular fault-tolerant agent after the first occurrence of JnextPlan. During normal operations, the JnextPlan job automatically turns on the *FullStatus* flag for the master domain manager, if it is not already turned on. When the new master domain manager runs JnextPlan, it does not recognize the former master domain manager as a backup master domain manager unless the flag is enabled. The former master domain manager does not have an accurate Symphony file when the time comes to switch back.

Also ensure that the backup master domain manager is synchronized with respect to time with the master domain manager. The surest way is to use a Network Time Protocol Server to control the time on both systems, with the same repeat interval.

Copying files to use on the backup master domain manager

To back up the important master domain manager files to the backup master domain manager, use the following procedure:

1. Copy the Security file on the master domain manager to the `<TWA_home>/TWS` directory on the backup master domain manager. Add a suffix to the file so that it does not overwrite the backup master domain manager's own Security file, for example, `Security_from_MDM`.
2. Copy all files in the `<TWA_home>/TWS/mozart` directory.
3. Copy the `localopts` file (see "Setting local options" on page 20 for the location). Add a suffix to the file so that it does not overwrite the backup master domain manager's own `localopts` file; for example, `localopts_from_MDM`.

This procedure must be performed each production period, or whenever there are significant changes to any objects. It can be incorporated into a script.

In addition to these required files, you might also want to copy the following:

- Any scripts you might have written.
- Archived Symphony files, for reference.
- Log files, for reference.

Note: Another approach could be to place all of the above files on a separately mountable file system, that could easily be unmounted from the master domain manager and mounted on the backup master domain manager in

Changing master domain manager

the event of need. You would almost certainly want to backup these files in addition, to protect against loss of the separately mountable file system.

To prevent the loss of messages caused by a master domain manager error, you can use the fault-tolerant switch-manager facility.

Switching a master domain manager

Use the procedure described in “Switching a domain manager” on page 235 when you have a short-term loss of a master domain manager.

Master domain managers remain switched until you perform another switch manager operation. To return to the original master domain manager, repeat this procedure before the next production period turnover, unless you do not expect the master domain manager to be available for the next production period turnover (final job stream and JnextPlan job). In this case, use the procedure in the following section.

Extended loss or permanent change of master domain manager

Use the following procedure to switch to the backup if the original master domain manager is not expected to return to service before the next new production period turnover (final job stream and JnextPlan job). For UNIX, use forward slashes in path names.

1. Use the conman **stop** function to stop Tivoli Workload Scheduler on the master domain manager and its backup.
2. If you copied the Security file from the master domain manager to the backup master domain manager *with a suffix*, now delete the backup master domain manager's own Security file and rename the Security file with the suffix as just Security.
3. If you copied the localopts file from the master domain manager to the backup master domain manager *with a suffix*, now merge the backup master domain manager's own localopts file with the localopts file from the master domain manager. Look at each property in turn and determine which version you want to keep on what is going to be your new master domain manager. For example, the property *thiscpu* needs to be the one from the backup master domain manager, but the options for controlling how the processes run can be taken from the master domain manager.
4. On the backup master domain manager cancel the *final* job stream in the Symphony file (it refers to the next production period's JnextPlan on the old master domain manager).
5. On the backup master domain manager, use composer to modify any important job streams that run on the master domain manager, in particular the *final* job stream. For each of these, change the workstation name to the name of the backup.
6. Change the workstation definition of the master domain manager from *manager* to *fault-tolerant agent*.
7. Change the workstation definition of the backup master domain manager from *fault-tolerant agent* to *manager*.

Note: These two steps must be done in the order given, as the system will not allow you to have two *managers* at the same time.

8. On the backup master domain manager, edit the `<TWA_home>/TWS/mozart/globalopts` file and change the `master` option to the name of the backup master domain manager workstation (this is used mainly for reports production)
9. Use the conman `switchmgr` function to switch to the backup master domain manager. See “Switching a domain manager” on page 235.
10. Submit a new `final` job stream to the new master domain manager (old backup master domain manager).
11. Run `JnextPlan` on the new master domain manager to generate the new Symphony file.
12. Remember to log on to the backup master domain manager when opening the Tivoli Dynamic Workload Console or Job Scheduling Console, first defining a new engine to access it.
13. If the old master domain manager has failed or is being replaced, you can now delete its workstation definition and remove it from the network.

Implications on dynamic workload broker

Switching a master domain manager has implications on the running dynamic workload broker instance. The following background information explains why.

The installation of a master domain manager and of every backup master includes also an instance of dynamic workload broker. If at installation time you chose to run dynamic workload scheduling, once your Tivoli Workload Scheduler network is up and running, it has one active instance of dynamic workload broker (the one hosted by the master domain manager) and a number of idle dynamic workload broker instances (each hosted by a backup master). What is important is that your network can run no more than one active dynamic workload broker instance at a time. This is because only the dynamic workload broker instance running on the active master can record the resource allocation data in the Tivoli Workload Scheduler database

Before you switch your master domain manager to a backup master, you must stop dynamic workload broker. After the masters have switched, you must start it on the new master. The process is not automatic and you must take all possible care to avoid ending up with two concurrently active instances.

If you are impelled to switch masters because the system running the current master broke down, make sure also the dynamic workload broker instance is positively down. If it is, you only need to start the new dynamic workload broker instance after switching the master.

If you switch masters for any other reason than a system failure, and you are switching to a backup master while the system running the current master is up, you risk having two concurrently active instances. This is why it is important that you stop the current dynamic workload broker instance before you switch, and that you start the new instance after the backup master takes over.

Here is the procedure you must follow every time you switch masters if you run dynamic scheduling in your network:

1. If the dynamic workload broker instance on the current master is still running, stop it. To do so, use wastool `stopBrokerApplication.sh` on UNIX and Linux or `stopBrokerApplication.bat` on Windows as follows:
`stopBrokerApplication -user username -password password [-port portnumber]`

Changing master domain manager

where *username* and *password* are the credentials used at installation. The parameter *portnumber* is optional. If it is not defined, the default is used.

2. Switch the master domain manager to a backup master. Use either the `switchmgr` command or the Tivoli Dynamic Workload Console.
3. Start the dynamic workload broker instance running with the new master. To do so, use `wastool startBrokerApplication.sh` on UNIX and Linux or `startBrokerApplication.bat` on Windows as follows:

```
startBrokerApplication -user username -password password [-port portnumber]
```

where *username* and *password* are the credentials used at installation. The parameter *portnumber* is optional. If it is not defined, the default is used.

The active and idle dynamic workload broker instances are recorded in the Tivoli Workload Scheduler database and their list is cached in the master domain manager. The Resource Advisor agents, which run as part of the Tivoli Workload Scheduler agents, cyclically run a scan of the resources available on each computer and send these data to the running dynamic workload broker. The Resource Advisor agents are able at all times to find the active dynamic workload broker instance by polling the dynamic workload broker instances list until they get a reply from the active one.

The main idea for ensuring that the switch of master domain managers is smooth on dynamic workload broker is to set exact replicas of all the dynamic workload broker instances configured with your master domain manager and your backup masters. This implies that when you change any configuration settings on the instance running on the master domain manager, you make the changes also on the instances idling on the backup masters (this applies especially to the `ResourceAdvisorConfig.properties` and `JobDispatcherConfig.properties` files).

It is important that you also keep the data pertinent to every dynamic workload broker up-to-date. If you do change the host name or port number of any of your dynamic workload broker instances, use the `exportserverdata` and `importserverdata` commands from the dynamic workload broker command line to record these changes in the Tivoli Workload Scheduler database. See the *Scheduling Workload Dynamically* guide for details on these commands.

Two final recommendations apply to the workload broker workstation:

- Set the same port number (the default is 41111) for all your active and idle workload broker workstation instances (one is setup with the master and one with every backup master you install). If after installation you change the port number on your active workload broker workstation, set the same port number for the idle workload broker workstations on your backup masters.
- The database records for your workload broker workstations all have `LOCALHOST` as the host name of the workstation. Leave the record as it is. Do not replace `LOCALHOST` with the actual host name or IP address of the workstation. `LOCALHOST` is used intentionally to ensure the jobs submitted from Tivoli Workload Scheduler are successfully sent to the (new) local dynamic workload broker when you switch masters.

Note: This setup requires that all your network be at the 8.5.1 version. In fact, if you have older fault-tolerant agents and you run the `showjobs` command to display data on workload run dynamically on any of them, the command fails and you get an error message instead. If this happens, you have to upgrade the agent to version 8.5.1.

Changing key Tivoli Workload Scheduler passwords

When you change passwords for key users in your Tivoli Workload Scheduler environment, there are various operations to perform, depending on which user's password is being changed, the type of operating system on which it is deployed, and the type of Tivoli Workload Scheduler node where the password is being changed. You can perform these operations manually, or you can use the **changePassword** script described in "Using the changePassword script" on page 248 to accomplish the necessary operations automatically.

If you decide to proceed manually, the following pages describe what you have to do if the passwords of any of the following users change:

Tivoli Workload Scheduler instance owner

The `<TWS_user>` (the instance owner) of a Tivoli Workload Scheduler component (on Windows only).

WebSphere Application Server user

The WebSphere Application Server user (as identified by the WebSphere Application Server tools) which authenticates the `<TWS_user>` being used by Tivoli Workload Scheduler components.

The database user (J2C) of a Tivoli Workload Scheduler component:

DB2 If you are using a DB2 database, this is the user ID used to access DB2.

Note: This is different according to whether you have the server or the client installed:

DB2 Server installed

The DB2 administration user (local) is used.

DB2 Client installed

The Tivoli Workload Scheduler DB2 user on the remote server is used.

Oracle If you are using an Oracle database, the Oracle schema owner user.

Note: The Oracle schema owner is not an operating system ID. Even if it has the same value as an operating system ID on the same computer, it is completely separate, and the passwords are changed separately.

Streamlogon user

The streamlogon user of any job run in the Tivoli Workload Scheduler environment (jobs running on Windows only)

For all other users of Tivoli Workload Scheduler, no action is required if their passwords change.

If you use the **changePassword** script, the password changes and corresponding operations are performed automatically. For detailed information about the script, refer to "Using the changePassword script" on page 248. If you decide to proceed manually, consult Table 59 on page 242 to determine if a change of password requires actions to be taken for a role on the different Tivoli Workload Scheduler components. Look up the role and the component and determine from the corresponding table cell where the changes must be made:

- If the cell contains a "✓", make the change on the system where the indicated component is running

Changing Tivoli Workload Scheduler passwords

- If the cell contains "MDM", make the change on the master domain manager to which the component belongs

Table 59. If and where password changes are required

Role	MDM	BKM	FTA	FTA + CONN
Tivoli Workload Scheduler instance owner (Windows)	✓	✓	✓	✓
WebSphere Application Server user	✓	✓		✓
Database user	✓	✓		
Streamlogon user (Windows)	✓	✓	MDM	MDM

For example, if you are the TWS_user (the instance owner) of a fault-tolerant agent, you need to implement the password change on the system where the fault-tolerant agent is installed, but if you are also the streamlogon user of jobs running on that system, the changes required for the new password must be applied at the master domain manager to which the fault-tolerant agent belongs.

If you are not certain which user role you are playing, consult "Determining the role of the user whose password has changed."

When you have determined what role you are playing, determine if you need to take any actions, and if so, where, by consulting "Determining the actions to take" on page 244.

Determining the role of the user whose password has changed

Use the following procedure to determine which role or roles the user whose password has changed is playing.

Attention: A user might have more than one role, in which case you must follow more than one procedure to change the password

1. Check if the user is the Tivoli Workload Scheduler instance owner:

Windows Check if the user whose password is to be changed is the user that owns the *Tivoli Workload Scheduler* for <TWS_user> service.

UNIX Run the following command:
ps -ef | grep netman

If the user whose password has changed matches the user ID revealed by the above, the user is the *Tivoli Workload Scheduler instance owner*.

2. Check if the user is the WebSphere Application Server user or the database user, or both:

1. Log on to the computer where Tivoli Workload Scheduler is installed as the following user:

UNIX root

Changing Tivoli Workload Scheduler passwords

Windows Any user in the *Administrators* group.

2. Access the directory: `<TWA_home>/wastools`
3. From that same directory run the following script:

UNIX `showSecurityProperties.sh > <output_file.txt>`

Windows `showSecurityProperties.bat > <output_file.txt>`

Note: This command might display a message from the application server (WASX7357I:) in the output file. You can ignore this message.

4. Open `<output_file.txt>` in a text editor.
5. Check the value of the key: `activeUserRegistry`. Depending on the value, check if the user whose password has changed is one of the ID keys:

Table 60. Values of `activeUserRegistry` to check

Value of <code>activeUserRegistry</code> key:	User ID key to check:	Password to change
LocalOS	LocalOSServerID	LocalOSServerpassword
LDAP	LDAPServerid	LDAPPassword

If the user whose password has changed matches the appropriate ID key, the user is the *WebSphere Application Server user*

6. Check the value of the key `j2cUserId`. If the user whose password is to be changed matches this key, the user is the *database user*.

Note: If the user is the Oracle schema owner, the password must also be changed within Oracle (see the Oracle documentation).

3. Check if the user is a streamlogon user

Using **composer**, the Tivoli Dynamic Workload Console or the Job Scheduling Console, check if the user is identified as a Windows user. If so, the user is a *streamlogon user*.

When you have determined which roles the user plays, see Table 59 on page 242 to determine if and where the password change must be implemented, and then “Determining the actions to take” on page 244.

Changing Tivoli Workload Scheduler passwords

Determining the actions to take

Consult Table 61 to determine which actions you need to perform for a change of password:

Table 61. Password change actions

	TWS instance owner (Windows only)	WebSphere Application Server user	Database user	Streamlogon user (Windows only)
"Action 1 - change the WebSphere Application Server user ID password"		✓ (1)		
"Action 2 - change password used by command-line clients to access the master domain manager" on page 245		✓		
"Action 3 - change password used by fault-tolerant agent systems to access the master domain manager (for conman)" on page 246		✓		
"Action 4 - update the engine connection parameters in the GUIs" on page 246		✓		
"Action 5 - change the j2c user ID password" on page 246			✓ (1)	
The following step only applies to passwords of users on UNIX computers				
"Action 6 - UNIX - update SOAP properties" on page 247		✓		
The following step only applies to passwords of users on Windows computers				
"Action 7 - Windows - update Windows services" on page 247	✓	✓		
"Action 8 - change the Tivoli Workload Scheduler Windows user definition" on page 248				✓

Notes:

1. If the user is both the WebSphere Application Server user *and* the database user, the changes made by running **changeSecurityProperties** can be performed as one action, modifying both passwords with the same value.

Action 1 - change the WebSphere Application Server user ID password

Use the **changeSecurityProperties** utility to change the WebSphere Application Server user ID password.

The procedure requires you to create a text file of the current security properties, edit the file, stop the application server, run the utility and restart the application server.

Note: You might have already created the text file while determining your role (see "Determining the role of the user whose password has changed" on page 242).

Find information about how to do this as follows:

Changing Tivoli Workload Scheduler passwords

- “Application server - using the utilities that change the properties - reference” on page 277 gives a generic description of the procedure for making any change to the WebSphere Application Server properties
- “Changing security properties” on page 259 lists all the security properties and gives other reference information about the utility
- When editing the text file of the current security properties, Locate either LocalOSServerpassword or LDAPPassword, depending on the type of authentication you are using (see Table 60 on page 243), and change the password to the new value, in plain text.

Notes:

1. If the user is both the WebSphere Application Server user *and* the database user, you can change the properties for both in the same action. See “Action 5 - change the j2c user ID password” on page 246. for details of the property to change.
2. The **changeSecurityProperties** utility might display a message from the application server (WASX7357I:). You can ignore this message.
3. When you supply a password in a text file for **changeSecurityProperties**, there is a small security exposure. When you enter a password in the file, the password is entered in clear (unencrypted). After you have run **changeSecurityProperties**, the password remains in clear in the text file you have edited, but if you run **showSecurityProperties** the password is output encrypted. Thus, your potential security exposure is limited to the time from when you entered the password in the text file until when you manually deleted the text file after using **changeSecurityProperties**.

Attention: if you subsequently want to change *other* parameters and do *not* want to change any passwords, you must do one of the following before running **changeSecurityProperties**:

- Resupply the passwords in clear
- Comment the password properties
- Delete the password properties

This is to avoid that the row of asterisks is applied as the password.

Note that if you run **showSecurityProperties** and see that any password in the encrypted form is shown as a sequence of 5 asterisks (*), this is wrong. A maximum of 3 asterisks is supported. Sequences of 5 asterisks are not supported and your passwords, encrypted as such, are ignored. If this happens, reset the password to a shorter one.

Action 2 - change password used by command-line clients to access the master domain manager

If you have changed the password of the WebSphere Application Server user that command-line clients use to connect to the master domain manager, the connection parameters must be updated.

Follow this procedure:

Changing Tivoli Workload Scheduler passwords

1. Identify all systems that have a command line client remote connection defined with the master domain manager
2. On these workstations, open the user options files (one for each user). The default file name is `<User_home>/.TWS/useropts`, but if you have more than one instance of Tivoli Workload Scheduler on a system, you might have implemented separate user options files to make separate connections, in which case consult the `useropts` key in the `localopts` file on each instance to determine the name of the specific `useropts` file for that instance.
3. For each file, locate the password key (encrypted) and change its value to that of the new password in plain text, enclosed in double quotes. The password is saved in clear, but will be encrypted at first logon of the User ID.
4. Save the files.
5. Check if the following file exists: `<Root_home>/.TWS/useropts`. If it does, change the password in the same way.

Action 3 - change password used by fault-tolerant agent systems to access the master domain manager (for conman)

If you have changed the password of the WebSphere Application Server user that is used by fault-tolerant agents with an HTTP or HTTPS connection defined in the local options that points to the master domain manager, the connection parameters must be updated.

Follow this procedure:

1. Identify all fault-tolerant agents with an HTTP or HTTPS connection defined in the local options that points to the master domain manager.
2. On these workstations, open the user options file `<Root_home>/.TWS/useropts`
3. Locate the password key (encrypted) and change its value to that of the new password in plain text, enclosed in double quotes. The password is saved in clear, but will be encrypted at first logon of the User ID.
4. Save the file.

Action 4 - update the engine connection parameters in the GUIs

If you have changed the password of the WebSphere Application Server user that is used by the Tivoli Dynamic Workload Console or the Job Scheduling Console to connect to the distributed engine, the engine connection parameters must be updated, as follows:

1. On each instance of the Tivoli Dynamic Workload Console or the Job Scheduling Console, locate the page where you modify the distributed engine connection parameters
2. Change the password and submit the page.

Action 5 - change the j2c user ID password

Use the `changeSecurityProperties` utility to change the j2c database user ID password.

The procedure requires you to create a text file of the current security properties, edit the file, stop the application server, run the utility and restart the application server.

Changing Tivoli Workload Scheduler passwords

Note: You might have already created the text file while determining your role (see “Determining the role of the user whose password has changed” on page 242).

Find information about how to do this as follows:

- “Application server - using the utilities that change the properties - reference” on page 277 gives a generic description of the procedure for making any change to the WebSphere Application Server properties
- “Changing security properties” on page 259 lists all the security properties and gives other reference information about the utility
- When editing the text file of the current security properties, Locate the `j2cPassword` and change the password to the new value, in plain text.

Notes:

1. If the user is both the WebSphere Application Server user *and* the database user, you can change the properties for both in the same action. See “Action 1 - change the WebSphere Application Server user ID password” on page 244. for details of the property to change.
2. The **changeSecurityProperties** utility might display a message from the application server (WASX7357I:). You can ignore this message.
3. When you supply a password in a text file for **changeSecurityProperties**, there is a small security exposure. When you enter a password in the file, the password is entered in clear (unencrypted). After you have run **changeSecurityProperties**, the password remains in clear in the text file you have edited, but if you run **showSecurityProperties** the password is output encrypted. Thus, your potential security exposure is limited to the time from when you entered the password in the text file until when you manually deleted the text file after using **changeSecurityProperties**.

Attention: if you subsequently want to change *other* parameters and do *not* want to change any passwords, you must do one of the following before running **changeSecurityProperties**:

- Resupply the passwords in clear
- Comment the password properties
- Delete the password properties

This is to avoid that the row of asterisks is applied as the password.

Action 6 - UNIX - update SOAP properties

On UNIX, after the password of the WebSphere Application Server administration user has been modified, it is important to change the SOAP client properties using the **updateWas.sh** script (see “Application server - updating the SOAP properties after changing the WebSphere Application Server user or its password” on page 271 for full details). For example:

```
updateWas.sh -user john.smith@domain.com -password zzzz
```

where the **user** and **password** options are the user that is authorized to stop the WebSphere Application Server.

Stop and restart WebSphere Application Server using the **stopappserver** and **startappserver** commands to make the change effective.

Action 7 - Windows - update Windows services

On Windows, the `<TWS_user>` account is used to start the following services:

Changing Tivoli Workload Scheduler passwords

- Tivoli Token Service for `<TWS_user>`
- Tivoli Workload Scheduler for `<TWS_user>`
- IBM WebSphere Application Server V6 - `<TWS_user>`.

The password must be updated in the properties of these services, or they are not able to start at next reboot. This is done as follows:

1. Stop all Tivoli Workload Scheduler processes. See “Unlinking and stopping Tivoli Workload Scheduler” on page 250 for details.
2. Locate the script **updateWasService.bat** in the `<TWA_home>/wastools` directory.
3. Run **updateWasService.bat**, as described in “Application server - updating the Windows services after modifications” on page 270, giving the new password as the `<WAS_user_password>`.
4. Restart all Tivoli Workload Scheduler processes using the **StartUp** command.

Action 8 - change the Tivoli Workload Scheduler Windows user definition

If the user ID is used within Tivoli Workload Scheduler to run jobs, follow this procedure:

1. Run the **composer modify user** command. The user details of the selected user are written to a temporary file, which is opened.
2. Edit the password field so that it contains the new password value delimited by double quote characters (“”).
3. Save the file, and the contents are added to the database.
4. To make the change immediately effective in the current plan, issue the **conman altpass** command.

For the full syntax of these commands see the *Tivoli Workload Scheduler: User's Guide and Reference*.

Using the changePassword script

Use the **changePassword** script from the `<TWA_home>/wastools` directory to change the passwords of any of the following users:

- Tivoli Workload Scheduler instance owner (`<TWS_user>`)
- WebSphere Application Server user
- Database (J2C) user for either Oracle or DB2
- Streamlogon user (Windows only)

If required, the script performs the necessary changes to the `useropts` file and stops and restarts the WebSphere Application Server. You can run this script from your master domain manager or Tivoli Workload Scheduler agent. The script determines the role of the users for which the password must be changed and performs the checks and actions of the manual procedure described in actions 1 through 8. Run the script as follows:

UNIX

```
changePassword.sh -user <USERID>
                  -password <PASSWORD>
                  [-wasuser <WASUSER>]
                  [-waspassword <WASPASSWORD>]
                  [-usroptshome <HOMEDIR>]
```

Where the arguments are as follows:

Changing Tivoli Workload Scheduler passwords

- | **-user** <USERID>
| This argument is mandatory. Specify the user whose password you
| are changing.
- | **-password** <PASSWORD>
| This argument is mandatory. Specify the new password for the
| user.
- | **-wasuser** <WASUSER>
| This argument is optional. Specify the WebSphere Application
| Server user. By default, the <USERID> value is used.
- | **-waspassword** <WASPASSWORD>
| This argument is optional. Specify the WebSphere Application
| Server user password. By default, the <PASSWORD> value is used.
| The <WASUSER> and <WASPASSWORD> values are ignored if the
| WebSphere Application Server is not present on this instance or if
| the script is running for a Tivoli Workload Scheduler instance.
- | **-usroptshome**<HOMEDIR>
| This argument is optional. The script searches for the USEROPTS file
| in the *TWA_home/.TWS*WebSphere Application Server directory. This
| argument is ignored if the script is not running for a Tivoli
| Workload Scheduler instance.

Windows

```
changePassword.bat -user <USERID>  
                   -password <PASSWORD>  
                   [-srvuser <SRVUSERID>]  
                   [-srvpasword <SRVPASSWORD>]  
                   [-wasuser <WASUSERID>]  
                   [-waspassword <WASPASSWORD>]  
                   [-useroptshome <USEROPTSHOME>]  
                   [-streamlogonws <WS>]  
                   [-streamlogondm <DOMAIN>]
```

Where the arguments are as follows:

- | **-user** <USERID>
| This argument is mandatory. Specify the user whose password you are
| changing.
- | **-password** <PASSWORD>
| This argument is mandatory. Specify the new password for the user.
- | **-srvuser** <SRVUSERID>
| This argument is optional. Specify the Windows service user. If you are
| running the script for a Tivoli Workload Scheduler instance, the value
| specified here is the same as the Tivoli Workload Scheduler user. By
| default, the <USERID> value is used.
- | **-srvpasword** <SRVPASSWORD>
| This argument is optional. The password of the Windows service user. By
| default, the <PASSWORD> value is used.
- | **-wasuser** <WASUSER>
| This argument is optional. Specify the WebSphere Application Server user.
| By default, the <USERID> value is used.

Changing Tivoli Workload Scheduler passwords

| **-waspassword** <WASPASSWORD>
| This argument is optional. Specify the WebSphere Application Server user
| password. By default, the <PASSWORD> value is used. The <WASUSER>
| and <WASPASSWORD> values are ignored if the WebSphere Application
| Server is not present on this instance or if the script is running for a Tivoli
| Workload Scheduler instance.

| **-usroptshome**<HOMEDIR>
| This argument is optional. The script searches for the USEROPTS file in the
| *TWA_home/.TWS* WebSphere Application Server directory. By default, the
| home directory of the user running the script is used.

| **-streamlogonws**<WS>
| This argument is optional. The script updates the Windows user definition
| for the <WS>#<DOMAIN>/<USER> user in the Tivoli Workload Scheduler
| database. By default, the Windows user definition for the <USER> is
| updated. The update is performed only if the tool is running on the master
| domain manager in a Windows environment.

| **-streamlogondm**<DOMAIN>
| This argument is optional. Specify the domain of the user specified for the
| <USER>.

Unlinking and stopping Tivoli Workload Scheduler

Before you perform an upgrade or uninstall, install a fix pack, or perform maintenance activities, ensure that all Tivoli Workload Scheduler processes and services are stopped. Follow these steps:

1. If you have jobs that are currently running on the workstation, wait for them to finish. To determine which are not finished, check for jobs that are in the *exec* state. When there are no jobs in this state, and you have allowed sufficient time for all events to be distributed in your network, you can continue with the rest of the procedure.
2. If the workstation that you want to stop is not the master domain manager, unlink the workstation by issuing the following command from the command line of the master domain manager:

```
conman "unlink workstationname;noask"
```
3. All Tivoli Workload Scheduler processes on the workstation must then be stopped manually. From the command line, while logged on as the <TWS_user>, use the following command:

```
conman "stop;wait"
```
4. From the command line, stop the netman process as follows:

UNIX Run:
 conman "shut"

Note: Do not use the UNIX **kill** command to stop Tivoli Workload Scheduler processes.

Windows Run the shutdown.cmd command from the Tivoli Workload Scheduler home directory.

5. If the workstation is at V8.4 or higher, stop the SSM agent, as follows:
 - On Windows, stop the Windows service: Tivoli Workload Scheduler SSM Agent (for <TWS_user>).
 - On UNIX, run **stopmon**.
6. If you are updating an agent, remove (unmount) any NFS mounted directories from the master domain manager.
7. If you are upgrading an installation that includes the connector, stop the connector.
8. Stop the WebSphere Application Server using the **conman stopappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)

To verify if there are services and processes still running, complete the following steps:

UNIX Type the command:

```
ps -u <TWS_user>
```

Verify that the following processes are not running: netman, mailman, batchman, writer, jobman, JOBMAN, stageman, logman, planman, monman, ssmagent.bin, and appservman.

Windows Run **Task Manager**, and verify that the following processes are not running: netman, mailman, batchman, writer, jobman, stageman, JOBMON, tokensrv, batchup, logman, planman, monman, ssmagent, and appservman.

Also, ensure that no system programs are accessing the directory or subdirectories, including the command prompt and Windows Explorer.

Changing the database host name, port, or database name

To change the database host name, port, or database name, the procedure differs, depending on whether the database is on DB2 or Oracle:

- “Change the DB2 host name, port, or database name”
- “Changing the Oracle host name, port, or database name” on page 257

Change the DB2 host name, port, or database name

If you need to change the DB2 host name, port, or database name, use the **changeDataSourceProperties** utility to reflect these changes in the application server on the master domain manager.

When you installed Tivoli Workload Scheduler, the default database name that was used for the creation of the database was *TWS* (which you might have changed). You also supplied the port and the host name of the DB2 server. If you want to change any of these details, you must do the following:

1. Stop DB2 and Tivoli Workload Scheduler
2. Use the facilities of DB2 (see DB2 documentation for details) or the operating system to change the database name, port or host name.

Changing database host name, port, or database name

3. Change the configuration of the Tivoli Workload Scheduler application server so that it points correctly to the changed DB2 configuration.

The procedure requires you to stop the application server, create a text file of the current data source properties, edit the file, run the utility and restart the application server. Find information about how to do this as follows:

- “Application server - using the utilities that change the properties - reference” on page 277 gives a generic description of the procedure for making any change to the WebSphere Application Server properties
- “Changing data source properties” lists all the data source properties and gives other reference information about the utility
- When editing the text file of the current data source properties, do as follows:
 - a. Edit the text file and locate the following properties:

```
#####  
# DB2 Type4 Resource Properties  
#####  
DB2Type4DatabaseName=TWS  
DB2Type4ServerName=localhost  
DB2Type4PortNumber=50083
```

- b. Set the following entries:

DB2Type4DatabaseName

The new name of the Tivoli Workload Scheduler database.

DB2Type4ServerName

The new DB2 server host name.

DB2Type4PortNumber

The new DB2 server port.

Do not change any other properties.

Note: The utility might display a message from the application server (WASX7357I:). You can ignore this message.

4. Start DB2 and Tivoli Workload Scheduler.

This script can also be used to change other data source properties. However, if you do so, Tivoli Workload Scheduler might not work correctly. You are advised to make any other changes only under the instructions of IBM Software Support, to correct specific problems. One of those specific problems could be the need to resolve problems with the JDBC driver, see “Resolving problems with the JDBC driver” on page 256.

Changing data source properties

You run the **changeDataSourceProperties** script on the master domain manager to change the data source properties of the RDBMS in use with the master domain manager. You are required to update the data source properties in the following cases:

- You migrate your data from an Oracle database to DB2 using the reconfiguration method.
- You change the database name, server, host, or port.
- You change the path to the JDBC driver of the RDBMS.

The procedure for running the script is described in detail in “Application server - using the utilities that change the properties - reference” on page 277, but in summary you do the following:

Changing database host name, port, or database name

- Run **showDataSourceProperties.sh (.bat)** > *my_file_name* to obtain the current properties
- Edit *my_file_name*
- Run **changeDataSourceProperties.sh (.bat)** *my_file_name*

Note: *my_file_name* must be the fully qualified path of the file.

The change utility calls the **wsadmin** utility by running **ChangeDataSourceProperties.jacl** with the properties file as input.

Only the WebSphere Application Server resources.xml are affected by this script. The full path of the file is:

```
<TWA_home>/eWAS/profiles/twaprofile/config/cells/DefaultNode/nodes/  
DefaultNode/servers/server<n>/resources.xml
```

The following is an example of the properties that can be changed with this utility. Only some of the options listed are currently in use by Tivoli Workload Scheduler.

```
#####  
JDBC Path Variables  
#####  
ORACLE_JDBC_DRIVER_PATH=  
DB2_JDBC_DRIVER_PATH=c:/ibm/sql11ib/java  
DB2UNIVERSAL_JDBC_DRIVER_PATH=c:/ibm/sql11ib/java  
  
#####  
DB2 Type2 Resource Properties  
#####  
DB2Type2JndiName=  
DB2Type2Description=  
DB2Type2ConnectionAttribute=cursorhold=0  
DB2Type2EnableMultithreadedAccessDetection=false  
DB2Type2Reauthentication=false  
DB2Type2JmsOnePhaseOptimization=false  
DB2Type2DatabaseName=TWSZ_DB  
DB2Type2PreTestSQLString=SELECT 1 FROM SYSIBM.SYSDUMMY1  
  
#####  
DB2 Type4 Resource Properties  
#####  
DB2Type4JndiName=jdbc/twsdb  
DB2Type4DatabaseName=TWSZ  
DB2Type4DriverType=4  
DB2Type4ServerName=myhost.mydomain.com  
DB2Type4PortNumber=50000  
DB2Type4SslConnection=false  
DB2Type4Description=  
DB2Type4TraceLevel=  
DB2Type4TraceFile=  
DB2Type4FullyMaterializeLobData=true  
DB2Type4ResultSetHoldability=2  
DB2Type4CurrentPackageSet=  
DB2Type4ReadOnly=false  
DB2Type4DeferPrepares=true  
DB2Type4CurrentSchema=  
DB2Type4CliSchema=  
DB2Type4RetrieveMessagesFromServerOnGetMessage=true  
DB2Type4ClientAccountingInformation=  
DB2Type4ClientApplicationInformation=  
DB2Type4ClientUser=  
DB2Type4ClientWorkstation=  
DB2Type4CurrentPackagePath=
```

Changing database host name, port, or database name

```
DB2Type4CurrentSQLID=
DB2Type4KerberosServerPrincipal=
DB2Type4LoginTimeout=0
DB2Type4SecurityMechanism=
DB2Type4TraceFileAppend=false
DB2Type4CurrentFunctionPath=
DB2Type4CursorSensitivity=
DB2Type4KeepDynamic=
DB2Type4CurrentLockTimeout=
DB2Type4EnableMultithreadedAccessDetection=false
DB2Type4Reauthentication=false
DB2Type4JmsOnePhaseOptimization=false
DB2Type4PreTestSQLString=SELECT 1 FROM SYSIBM.SYSDUMMY1
DB2Type4DbFailOverEnabled=false
DB2Type4ConnRetriesDuringDBFailover=100
DB2Type4ConnRetryIntervalDuringDBFailover=3000
# DB2Type4IsolationLevel can be one of the following:
#         CURSOR_STABILITY or READ_STABILITY
DB2Type4IsolationLevel=CURSOR_STABILITY
```

```
#####
Oracle Type2 Resource Properties
#####
OracleType2JndiName=
OracleType2DriverType=
OracleType2URL=jdbc:oracle:oci:@ORCL
OracleType2DatabaseName=
OracleType2ServerName=
OracleType2PortNumber=1521
OracleType2OracleLogFileSizeMode=0
OracleType2OracleLogFileCount=1
OracleType2OracleLogFileName=
OracleType2OracleLogTraceLevel=INFO
OracleType2OracleLogFormat=SimpleFormat
OracleType2OracleLogPackageName=oracle.jdbc.driver
OracleType2TNSEntryName=
OracleType2NetworkProtocol=
OracleType2DataSourceName=
OracleType2LoginTimeout=
OracleType2Description=
OracleType2EnableMultithreadedAccessDetection=false
OracleType2Reauthentication=false
OracleType2JmsOnePhaseOptimization=false
OracleType2PreTestSQLString=SELECT 1 FROM DUAL
OracleType2DbFailOverEnabled=false
OracleType2ConnRetriesDuringDBFailover=100
OracleType2ConnRetryIntervalDuringDBFailover=3000
```

```
#####
Oracle Type4 Resource Properties
#####
OracleType4JndiName=
OracleType4DriverType=
OracleType4URL=jdbc:oracle:thin:@//localhost:1521/ORCL
OracleType4DatabaseName=
OracleType4ServerName=
OracleType4PortNumber=1521
OracleType4OracleLogFileSizeMode=0
OracleType4OracleLogFileCount=1
OracleType4OracleLogFileName=
OracleType4OracleLogTraceLevel=INFO
OracleType4OracleLogFormat=SimpleFormat
OracleType4OracleLogPackageName=oracle.jdbc.driver
OracleType4TNSEntryName=
OracleType4NetworkProtocol=
```

Changing database host name, port, or database name

```
OracleType4DataSourceName=  
OracleType4LoginTimeout=  
OracleType4Description=  
OracleType4EnableMultithreadedAccessDetection=false  
OracleType4Reauthentication=false  
OracleType4JmsOnePhaseOptimization=false  
OracleType4PreTestSQLString=SELECT 1 FROM DUAL  
OracleType4DbFailOverEnabled=false  
OracleType4ConnRetriesDuringDBFailover=100  
OracleType4ConnRetryIntervalDuringDBFailover=3000
```

When you change data source properties, observe the following rules:

- If a property is not provided in the properties file, the current value is not changed
- If a property is provided with a non-blank value, the current value is updated.
- If a property is provided with a blank value, the setting is set to blank if the property is classified as erasable or left unchanged if not.
- Always use type 4 data sources for DB2 and type 2 data sources for Oracle.
- Set the appropriate JDBC driver path variable for the RDBMS of your choice.
 - For DB2, the JDBC driver is located in the java subfolder of the sqllib directory. For example:
DB2_JDBC_DRIVER_PATH=c:/program files/ibm/sqllib/java
 - or
DB2UNIVERSAL_JDBC_DRIVER_PATH=c:/program files/ibm/sqllib/java
 - For Oracle, it is located in the jdbc/lib subfolder of the Oracle home directory. For example:
ORACLE_JDBC_DRIVER_PATH=C:/Oracle/product/10.2.0/db_1/jdbc/lib
- Make sure that the data source JNDI name is always set to jdbc/twsdb in the ...JndiName property of the RDBMS you use. If you change the RDBMS, proceed as follows:
 1. Reset to a name of your choice the ...JndiName property of the RDBMS from which you are changing.
 2. Set to jdbc/twsdb the ...JndiName property of the new RDBMS.
- See that the following properties are set:
 - For DB2:
DB2Type4JndiName
DB2Type4DatabaseName
DB2Type4ServerName
DB2Type4PortNumber
 - For Oracle:
OracleType2JndiName
OracleType2DatabaseName
OracleType2ServerName
OracleType2PortNumber

Displaying the current data source properties: To display the current properties, use the following utility:

UNIX **showDataSourceProperties.sh**

Windows **showDataSourceProperties.bat**

Changing database host name, port, or database name

Resolving problems with the JDBC driver

Tivoli Workload Scheduler is supplied using the JDBC driver type 4 for DB2 and type 2 for Oracle. However, each can use the other driver type, if necessary. IBM Software Support might ask you to change to this driver. This section tells you how.

Attention: This procedure must only be performed under the control of IBM Software Support.

To change the driver you need to change the data source properties following the procedure described in “Changing the database host name, port, or database name” on page 251. However, the parameters that you change are different. This is an example of the type 4 and type 2 parameters for DB2:

JDBC driver type 4 parameters

```
#####  
# DB2 Type4 Resource Properties  
#####  
DB2Type4JndiName=jdbc/twsdb  
DB2Type4DatabaseName=TWSZ  
DB2Type4DriverType=4  
DB2Type4ServerName=IT041866-9088  
DB2Type4PortNumber=50000  
DB2Type4SslConnection=false  
DB2Type4Description=  
DB2Type4TraceLevel=  
DB2Type4TraceFile=  
DB2Type4FullyMaterializeLobData=true  
DB2Type4ResultSetHoldability=2  
DB2Type4CurrentPackageSet=  
DB2Type4ReadOnly=false  
DB2Type4DeferPrepares=true  
DB2Type4CurrentSchema=  
DB2Type4CliSchema=  
DB2Type4RetrieveMessagesFromServerOnGetMessage=true  
DB2Type4ClientAccountingInformation=  
DB2Type4ClientApplicationInformation=  
DB2Type4ClientUser=  
DB2Type4ClientWorkstation=  
DB2Type4CurrentPackagePath=  
DB2Type4CurrentSQLID=  
DB2Type4KerberosServerPrincipal=  
DB2Type4LoginTimeout=0  
DB2Type4SecurityMechanism=  
DB2Type4TraceFileAppend=false  
DB2Type4CurrentFunctionPath=  
DB2Type4CursorSensitivity=  
DB2Type4KeepDynamic=  
DB2Type4CurrentLockTimeout=  
DB2Type4EnableMultithreadedAccessDetection=false  
DB2Type4Reauthentication=false  
DB2Type4JmsOnePhaseOptimization=false  
DB2Type4PreTestSQLString=SELECT 1 FROM SYSIBM.SYSDUMMY1  
DB2Type4DbFailOverEnabled=false  
DB2Type4ConnRetriesDuringDBFailover=100  
DB2Type4ConnRetryIntervalDuringDBFailover=3000  
# DB2Type4IsolationLevel can be one of the following:  
#         CURSOR_STABILITY or READ_STABILITY  
DB2Type4IsolationLevel=CURSOR_STABILITY
```

Changing database host name, port, or database name

JDBC Driver type 2 parameters

```
#####  
# DB2 Type2 Resource Properties  
#####  
DB2Type2JndiName=  
DB2Type2Description=  
DB2Type2ConnectionAttribute=cursorhold=0  
DB2Type2EnableMultithreadedAccessDetection=false  
DB2Type2Reauthentication=false  
DB2Type2JmsOnePhaseOptimization=false  
DB2Type2DatabaseName=TWSZ_DB  
DB2Type2PreTestSQLString=SELECT 1 FROM SYSIBM.SYSDUMMY1
```

Switching drivers or changing the JNDI name: The data source JNDI name must be unique. In the above examples the JNDI name for driver 4 is set to the correct value. To switch drivers, modify the parameters so that the values are reversed, as follows:

Example 1: default values for the JNDI name:

```
#DB2Type4JndiName=jdbc/twsdb
```

...

```
#DB2Type2JndiName=jdbc/twsdb2
```

Example 2: switched values for the JNDI name:

```
#DB2Type4JndiName=jdbc/twsdb2
```

...

```
#DB2Type2JndiName=jdbc/twsdb
```

To change the driver names to a different value, see the following:

Example 3: different values for the JNDI name:

```
#DB2Type4JndiName=jdbc/twsdb_test4
```

...

```
#DB2Type2JndiName=jdbc/twsdb_test2
```

Changing the Oracle host name, port, or database name

If you need to change the Oracle host name, port, or database name, you can normally manage the change within Oracle. This is because WebSphere Application Server points at the Oracle service where these items are defined. See the Oracle documentation for information on how to change them.

However, the properties that you are changing might be defined in `<TWA_home>/eWAS/profiles/twaprofile/properties/TWSConfig.properties`. In this case you must ensure that they are changed here, as well. The properties in question are:

```
com.ibm.tws.dao.rdbms.rdbmsName = ORACLE  
com.ibm.tws.dao.rdbms.modelSchema = <TWS_Oracle_User>  
com.ibm.tws.dao.rdbms.eventRuleSchema = <TWS_Oracle_User>  
com.ibm.tws.dao.rdbms.logSchema = <TWS_Oracle_User>
```

Changing the security settings

This section describes how to modify the security settings of Tivoli Workload Scheduler. It has the following topics:

- “Changing security settings - general”
- “Modify or display security settings - custom user registry” on page 263

Changing security settings - general

Use the **changeSecurityProperties** script to change various security settings on the application server. Those relating to SSL are described in Chapter 7, “Setting connection security,” on page 169. Those relating to the passwords of the database access users are described in “Changing key Tivoli Workload Scheduler passwords” on page 241. Other items, such as the active user registry and the local operating system ID and password, can also be changed.

The procedure requires you to stop the application server, create a text file of the current security properties, edit the file, run the utility and restart the application server. Find information about how to do this as follows:

- “Application server - using the utilities that change the properties - reference” on page 277 gives a generic description of the procedure for making any change to the WebSphere Application Server properties
- “Changing security properties” on page 259 lists all the security properties and gives other reference information about the utility
- When editing the text file of the current security properties, do as follows:
 1. Edit the text file and locate the following properties:

```
#####  
Global Security Panel  
#####  
enabled=true  
enforceJava2Security=false  
useDomainQualifiedUserNames=false  
cacheTimeout=600  
ltpaTimeOut=720  
issuePermissionWarning=true  
activeProtocol=CSI  
useFIPS=false  
activeAuthMechanism=LTPA  
activeUserRegistry=LocalOS  
  
#####  
Local OS Registry  
#####  
LocalOSServerID=tw8511  
LocalOSServerpassword=*****  
LocalOSServerREALM=myhost.mydomain.com
```

2. Make any changes to these properties that are necessary.
Do not change any other properties.

Notes:

1. The utility might display a message from the application server (WASX7357I:). You can ignore this message.
2. When you supply a password in a text file for **changeSecurityProperties**, there is a small security exposure. When you enter a password in the file, the password is entered in clear (unencrypted). After you have run **changeSecurityProperties**, the password remains in clear in the text file you have edited, but if you run **showSecurityProperties** the password is output

encrypted, as a row of asterisks. Thus, your potential security exposure is limited to the time from when you entered the password in the text file until when you manually deleted the text file after using **changeSecurityProperties**.

Attention: if you want to change parameters *other* than passwords, and do *not* want to change a password, you must do one of the following before running **changeSecurityProperties**:

- Resupply the passwords in clear
- Comment the password properties
- Delete the password properties

This is to avoid that the row of asterisks is applied as the password.

Changing security properties

You can use the **changeSecurityProperties** script to change the embedded WebSphere Application Server security settings related to the usage of Tivoli Workload Scheduler. In particular they include:

- General security settings, such as the user registry to use (local operating system or LDAP)
- Local operating system user registry settings (user and password)
- LDAP configuration settings
- SSL key-stores
- DB2 user settings for Tivoli Workload Scheduler data source

The procedure for running the script is described in detail in “Application server - using the utilities that change the properties - reference” on page 277, but in summary you do the following:

- Run **showSecurityProperties.sh (.bat)** > *my_file_name* to obtain the current properties
- Edit *my_file_name*
- Run **changeSecurityProperties.sh (.bat)** *my_file_name*

Note: *my_file_name* must be the fully qualified path of the file.

The utility calls **wsadmin** utility by running the **ChangeSecurityProperties.jacl** with the properties file as input.

Only the WebSphere Application Server `security.xml` configuration files is affected by this script. The path of the file is:

```
TWA_home/eWAS/profiles/twaprofile/config/cells/DefaultNode/security.xml
```

where *<profile>* is one of **twspprofile**, **twconnprofile** or **twszconnprofile** for the master domain manager, the distributed connector and the z/OS connector respectively.

The security properties vary according to how your instance of Tivoli Workload Scheduler is configured. The following is an example of some of the properties you can change with this utility. The example specifies Local OS as the active user registry:

```
#####
Global Security Panel
#####
enabled=true
enforceJava2Security=false
```

Changing security properties

```
useDomainQualifiedUserNames=false
cacheTimeout=600
ltpaTimeOut=720
issuePermissionWarning=true
activeProtocol=CSI
useFIPS=false
activeAuthMechanism=LTPA
activeUserRegistry=LocalOS

#####
Local OS Registry
#####
LocalOSServerID=tw8511
LocalOSServerpassword=*****
LocalOSServerREALM=myhost.mydomain.com

#####
Custom User Registry
#####
CustomServerID=
CustomServerpassword=*****
CustomServerREALM=
#####
WIM Registry
#####
#WIMServerID=<Unknown>
#WIMServerpassword=*****
#WIMServerREALM=defaultWIMFileBasedRealm
#####
LDAP Panel
#####
LDAPServerId=tw8511
LDAPPassword=*****
LDAPPrimaryAdminId=
LDAPUseRegistryServerId=true
LDAPServerType=IBM_DIRECTORY_SERVER
LDAPHostName=
LDAPPort=389
LDAPBaseDN=
LDAPBindDN=
LDAPBindPassword=*****
LDAPsearchTimeout=120
LDAPreuseConnection=true
LDAPIgnoreCase=true
LDAPsslEnabled=false
LDAPsslConfig=DefaultNode/DefaultSSLSettings
#####
Advanced LDAP Panel
#####
LDAPUserFilter=(amp;(uid=%v)(objectclass=person))
LDAPGroupFilter=(amp;(cn=%v)(|(objectclass=groupofnames)
(objectclass=groupofuniqueNames)(objectclass=groupofURLs)))
LDAPUserIdMap=*:uid
LDAPGroupIdMap=*:cn
LDAPGroupMemberIdMap=ibm-allGroups:member;ibm-allGroups:uniqueMember
LDAPCertificateFilter=
LDAPCertificateMapMode=EXACT_DN
#####
SSL Panel
#####
alias=DefaultSSLSettings
keyFileName=${USER_INSTALL_ROOT}/etc/TWSServerKeyFile.jks
keyFilePassword=*****
keyFileFormat=JKS
trustFileName=${USER_INSTALL_ROOT}/etc/TWSServerTrustFile.jks
trustFilePassword=*****
```

```
trustFileFormat=JKS
clientAuthentication=false
securityLevel=HIGH
enableCryptoHardwareSupport=false
```

```
#####
J2C Authentication Data Panel
#####
j2cAlias=twsj2c
j2cUserId=db2admin
j2cPassword=*****
j2cDescription=TWS authentication data entry for data source
```

All of the properties in the properties file are optional. When you are modifying the properties file you should be aware of the following:

- If a property is not provided in the properties file, the current value is not changed
- If a property is provided with a non-blank value, the current value is updated.
- If a property is provided with a blank value, the setting is set to blank if the property is classified as erasable or left unchanged if not.

Non-erasable properties are the following:

- enabled
- enforceJava2Security
- useDomainQualifiedUserNames
- cacheTimeout
- issuePermissionWarning
- activeAuthMechanism
- activeUserRegistry
- activeProtocol
- LocalOSServerID
- LocalOSServerpassword
- LocalOSServerREALM
- LDAPHostName
- LDAPPort
- LDAPsearchTimeout
- LDAPPreuseConnection
- LDAPIgnoreCase
- LDAPsslEnabled
- LDAPSearchTimeout
- LDAPCertificateMapMode
- keyFileFormat
- trustFileFormat
- clientAuthentication
- securityLevel
- timeout

“Example SecurityProps_TEMPLATE.properties file” on page 262 provides a complete list of options supported by the script. Not all of the properties are used for this release, but have been reserved for future releases or for special needs.

Changing security properties

Example SecurityProps_TEMPLATE.properties file: The following is a sample of the SecurityProps_TEMPLATE.properties file:

```
#####  
Global Security Panel  
#####  
enabled=true  
enforceJava2Security=false  
useDomainQualifiedUserNames=false  
cacheTimeout=600  
ltpaTimeOut=720  
issuePermissionWarning=true  
activeProtocol=CSI  
useFIPS=false  
activeAuthMechanism=LTPA  
activeUserRegistry=LDAP  
  
#####  
Local OS Registry  
#####  
LocalOSServerID=tw8511  
LocalOSServerpassword=*****  
LocalOSServerREALM=myhost.mydomain.com  
  
#####  
Custom User Registry  
#####  
CustomServerID=  
CustomServerpassword=*****  
CustomServerREALM=  
#####  
WIM Registry  
#####  
#WIMServerID=<Unknown>  
#WIMServerpassword=*****  
#WIMServerREALM=defaultWIMFileBasedRealm  
#####  
LDAP Panel  
#####  
LDAPServerId=tw8511  
LDAPPassword=*****  
LDAPPrimaryAdminId=  
LDAPUseRegistryServerId=true  
LDAPServerType=IBM_DIRECTORY_SERVER  
LDAPHostName=  
LDAPPort=389  
LDAPBaseDN=  
LDAPBindDN=  
LDAPBindPassword=*****  
LDAPsearchTimeout=120  
LDAPreuseConnection=true  
LDAPIgnoreCase=true  
LDAPsslEnabled=false  
LDAPsslConfig=DefaultNode/DefaultSSLSettings  
  
#####  
Advanced LDAP Panel  
#####  
LDAPUserFilter=(&(uid=%v)(objectclass=ePerson))  
LDAPGroupFilter=(&(cn=%v)(|(objectclass=groupName)  
                  (objectclass=groupOfUniqueNames)(objectclass=groupOfURLs)))  
LDAPUserIdMap=*:uid  
LDAPGroupIdMap=*:cn  
LDAPGroupMemberIdMap=ibm-allGroups:member;ibm-allGroups:uniqueMember  
LDAPCertificateFilter=
```

```

LDAPCertificateMapMode=EXACT_DN

#####
SSL Panel
#####
alias=DefaultSSLSettings
keyFileName=${USER_INSTALL_ROOT}/etc/TWSServerKeyFile.jks
keyFilePassword=*****
keyFileFormat=JKS
trustFileName=${USER_INSTALL_ROOT}/etc/TWSServerTrustFile.jks
trustFilePassword=*****
trustFileFormat=JKS
clientAuthentication=false
securityLevel=HIGH
enableCryptoHardwareSupport=false

#####
J2C Authentication Data Panel
#####
j2cAlias=twsj2c
j2cUserid=db2admin
j2cPassword=*****
j2cDescription=TWS authentication data entry for data source

```

Displaying the current security properties: To display the current properties, use the following utility:

```

UNIX          showSecurityProperties.sh
Windows      showSecurityProperties.bat

```

Modify or display security settings - custom user registry

This section describes how to display and modify the security properties when the active user registry is *Custom*.

The *Custom* active user registry provides a single authentication mechanism that is capable of authenticating users no matter how their user registry is implemented: local OS or LDAP.

If you want to modify the security properties of the LocalOS section, when the active user registry value is Custom, you can modify the LocalOS section of the showSecurityProperties.sh/bat output file, and then pass it as an argument to the changeSecurityProperties.sh/bat, as described in “Changing security settings - general” on page 258.

However, if you want to *display* the current LocalOS security properties when the active user registry is "Custom", open the file: TWA_home/eWAS/profiles/twaprofile/config/cells/DefaultNode/security.xml and check the XML section

```

<userRegistries
xmi:type="security:CustomUserRegistry"...

```

Managing the event processor

The only maintenance issue for the event processor is the management of the EIF event queue, cache.dat. The event queue is circular, with events being added at the end and removed from the beginning. However, if there is no room to write an event at the end of the queue it is written at the beginning, overwriting the event at the beginning of the queue.

To increase the size of the event processor queue, follow this procedure:

Managing event processor

1. At the workstation running the event processor, locate the file:
`<TWA_home>/eWAS/profiles/twaprofile/temp/TWS/EIFListener/eif.temp1`
2. Edit the file and locate the keyword:
`BufEvtMaxSize`
3. Increase the value of this keyword, according to your requirements.
4. Stop and restart the WebSphere Application Server using the **conman stopappserver** and **conman startappserver** commands (see “Starting and stopping the application server and **appservman**” on page 268).

Starting, stopping and displaying dynamic workload broker status

To start or stop dynamic workload broker, use the **startBrokerApplication** or **stopBrokerApplication** commands on the active master domain manager. Since these commands are processed asynchronously, the **brokerApplicationStatus** command allows you to check the status of dynamic workload broker following a start or a stop. Ensure that WebSphere Application Server is running and proceed as follows:

Starting dynamic workload broker

Use **startBrokerApplication.sh** on UNIX and Linux or **startBrokerApplication.bat** on Windows as follows:

```
startBrokerApplication -user username -password password [-port  
portnumber]
```

where *username* and *password* are the credentials used during the installation. The parameter *portnumber* is optional. If it is not defined, the default is used.

Stopping dynamic workload broker

Use **stopBrokerApplication.sh** on UNIX and Linux or **stopBrokerApplication.bat** on Windows as follows:

```
stopBrokerApplication -user username -password password [-port  
portnumber]
```

where *username* and *password* are the credentials used during the installation. The parameter *portnumber* is optional. If it is not defined, the default is used.

Displaying dynamic workload broker status

Use **brokerApplicationStatus.sh** on UNIX and Linux or **brokerApplicationStatus.bat** on Windows as follows:

```
brokerApplicationStatus -user username -password password [-port  
portnumber]
```

where *username* and *password* are the credentials used during the installation. The parameter *portnumber* is optional. If it is not defined, the default is used.

Application server - starting and stopping

Use the **startappserver** and **stopappserver** commands or the equivalent from the Tivoli Dynamic Workload Console or Job Scheduling Console to start or stop the embedded WebSphere Application Server. (see *Tivoli Workload Scheduler: User's Guide and Reference* for a description of these commands.)

Starting and stopping the embedded WebSphere Application Server

These commands also stop **appservman**, the service that monitors and optionally restarts the application server.

If you do not want to stop **appservman**, you can issue **startWas** or **stopWas**, supplying the **-direct** argument.

The full syntax of **startWas** and **stopWas** is as follows:

UNIX

Start the application server
startWas.sh [-direct]

Stop the application server
stopWas.sh [-direct]
-user <user_ID>
-password <password>

Note: The above syntax for stopping the embedded WebSphere Application Server is applicable only if all components are integrated with your Tivoli Workload Scheduler environment. If your Tivoli Dynamic Workload Console or z/OS Connector are not integrated (they do not share the same WebSphere Application Server with your Tivoli Workload Scheduler installation), you must use the following syntax:

```
stopWas.sh -direct  
-user <user_ID>  
-password <password>
```

where the **-direct** argument is mandatory.

Windows

Start the application server
startWas.bat [-direct]
[-service <service_name>]
[-options <parameters>]

Stop the application server
stopWas.bat [-direct]
[-service <service_name>]
[-washome <installation_directory>]
[-options <parameters>]

where the arguments are as follows:

-direct

Optionally starts or stops the application server without starting or stopping the application server monitor **appservman**.

For example, you might use this after changing some configuration parameters. By stopping WebSphere Application Server without stopping **appservman**, the latter will immediately restart WebSphere Application Server, using the new configuration properties. This argument is mandatory on UNIX when the product components are not integrated.

-options <parameters>

Optionally supplies parameters to the WebSphere Application Server **startServer** or **stopServer** commands. See the WebSphere Application Server documentation for details.

Starting and stopping the embedded WebSphere Application Server

- password** <password>
Defines the password to be used when stopping the application server on UNIX.
- service** <service_name>
Defines the WebSphere Application Server service name, if it is not the default value of IBM WebSphere Application Server V6 - <TWS_user>
- user** <user_ID>
Defines the user ID to be used when stopping the application server on UNIX.
- wasHome** <installation_directory>
Defines the WebSphere Application Server installation directory, if it is not the default value.

Application server - automatic restart after failure

If you experience any problems with the application server failing, a service is available that not only monitors its status, but can also restart it automatically in the event of failure. The service is called **appservman**, and it is enabled and controlled by the local options on the computer where the application server is running.

The following sections describe the service, how it works, and how it is controlled:

- “**Appservman** - how it works”
- “Controlling **appservman**” on page 267
- “Starting and stopping the application server and **appservman**” on page 268
- “Monitoring the application server status” on page 269
- “Obtaining information about application server failures” on page 269
- “Events created by **appservman**” on page 269

Appservman - how it works

Appservman is a service that starts, stops and monitors the application server. It also optionally restarts the application server in the event that the latter fails. **Appservman** can be controlled not just from nodes running the application server, but also from any other node running **conman**.

It is launched as a service by **netman** when starting Tivoli Workload Scheduler, and it itself then launches the application server. **Netman** also launches it when the **conman startappserver** command is run.

Appservman is stopped when Tivoli Workload Scheduler is shutdown. In addition, **Netman** stops both the application server and **appservman** when you use the **conman stopappserver** command, or, on Windows only, when you issue the **Shutdown -appsrv** command.

While it is running **appservman** monitors the availability of the application server, sending events that report the status of the application server. If the automatic restart facility is enabled, and the application server fails, the service determines from the restart policy indicated in the `localopts` options if it is to restart the application server. If the policy permits, it will restart the application server, and send events to report its actions.

Application server: automatic restart after failure

The WebSphere Application Server utilities **startWas** and **stopWas** by default start and stop the application server and **appservman** using the **startappserver** and **stopappserver** commands. However, these utilities can be instructed to start and stop the application server without stopping **appservman** by using the **startWas** and **stopWas** utilities with the **-direct** option.

Controlling appservman

Appservman is controlled by the following local options (in the `localopts` file):

Appserver auto restart

Determines if the automatic restart facility is enabled.

The default is *yes*. To disable the option set it to *no*.

Appserver check interval

Determines how frequently the service checks on the status of the application server. You should not set this value to less than the typical time it takes to start the application server on the computer.

The default is every 5 minutes.

Appserver min restart time

Determines the minimum time that must elapse between failures of the application server for the automatic restart to work. This option stops **appservman** from immediately restarting the application server if it fails on initial startup or when being restarted.

The default is 10 minutes.

Appserver max restarts

Determines the maximum number of times that **appservman** will automatically restart the application server within a time frame determined by you (Appserver count reset interval).

The default is 5 restarts.

Appserver count reset interval

Determines the time frame for the maximum number of restarts (Appserver max restarts).

The default is 24 hours.

Appserver service name

This is used in Windows only. It is generated as follows:

```
IBMVAS61Service - <TWS_user>
```

How to use the options

The default settings are a good starting point. Follow the indications below if you are not satisfied that the settings are maintaining the correct availability of the application server:

- If the application server is not restarting after failure, check the following:
 - That the *Appserver auto restart* is set to *yes*.
 - That the *Appserver check interval* is not set to too high a value. For example, if this value is set to 50 minutes, instead of the default 5, an early failure of the application server might wait 45 minutes before being restarted.
 - That *Appserver min restart time* is sufficient for the application server to fully restart. If, when the server checks the status of the application server, it finds that the application server is still starting up, in some circumstances it is not able to distinguish the starting-up state from the failed state, will report it as

Application server: automatic restart after failure

failed, and try and restart it again. With the same result. This will continue until *Appserver max restarts* is exceeded. If this is the case, make *Appserver min restart time* larger.

- If the application server is failing infrequently, but after several failures is not restarting, set the *Appserver max restarts* option to a higher value or the *Appserver count reset interval* to a lower value, or both. In this case it might be advantageous to study the pattern of failures and tailor these options to give you the required availability

Starting and stopping the application server and appserverman

If you need to stop and restart the application server, for example to implement a change in the application server configuration, use the following commands:

stopappserver[*domain!*]*workstation* [**;***wait*]

This command stops the application server and **appserverman**. You can optionally stop the application server on a remote workstation. The optional **;***wait* parameter tells **conman** to suspend processing until the command reports that both the application server and the service have stopped.

startappserver[*domain!*]*workstation* [**;***wait*]

This command starts the application server and **appserverman**. You can optionally start the application server on a remote workstation. The optional **;***wait* parameter tells **conman** to suspend processing until the command reports that both the application server and the service are up and running.

To stop and start the application server without stopping **appserverman**, see “Application server - starting and stopping” on page 264.

Configuring user and password for running conman stopappserver

When you run **conman stopappserver**, the appserverman process first checks if WebSphere Application Server can retrieve the user's credentials (username and password) from the `soap.client.props` file located in the WebSphere Application Server profile. If the check is negative, appserverman reads them from the `useropts` file of the user and runs the `stopServer.sh` (bat) script to pass them to WebSphere Application Server.

To be able to run **conman stopappserver**, you must therefore complete one of the following two customization procedures to provide the user credentials to WebSphere Application Server:

- Customize the user name (`com.ibm.SOAP.loginUserId`) and password (`com.ibm.SOAP.loginPassword`) properties in the `soap.client.props` file located in:

<code>TWS_home/appserver/profiles/twsprofile/properties/</code>	(Version 8.4 and earlier master)
<code>TWS_home/appserver/profiles/twsconnprofile/properties/</code>	(Version 8.4 and earlier agents)
<code>TWA_home/eWAS/profiles/twaprofile/properties/</code>	(Version 8.5 and later master and agents)

You must also:

1. Set the property `com.ibm.SOAP.securityEnabled` to `true` in the same file to enable the SOAP client security
2. Run the `encryptProfileProperties.sh` script to encrypt the password. See the Tivoli Workload Scheduler *Administration Guide* for more information on this application server tool.

Application server: automatic restart after failure

- 1 • Customize the Attributes for conman (CLI in version 8.4) connections
1 section in the localopts file by specifying the details of the connector or of the
1 master domain manager.
1 You must also:
1 1. Create (or customize if already present) the useropts file manually, adding
1 the USERNAME and PASSWORD attributes for the user who will run
1 **stopappserver**. Make sure the useropts file name is entered in the USEROPTS
1 key in the Attributes for conman (CLI) connections section. See the
1 *Administration Guide* for further details.
1 2. Encrypt the password in the useropts file simply by running **conman**.

Monitoring the application server status

To see the current status of the application server at any time view the STATE field in the workstation details.

This field contains a string of characters that provide information about the statuses of objects and processes on the workstation. The state of the application server is a one-character flag in this string, which has one of the following values if the application server is installed:

[A|R]

where:

- A The WebSphere Application Server is running.
R The WebSphere Application Server is restarting.

If the application server is down or if it was not installed, neither value of the flag is present in the STATE entry.

Obtaining information about application server failures

Appservman does not provide information about why the application server has failed. To obtain this information, look in the application server log files (see the *Tivoli Workload Scheduler: Troubleshooting Guide*).

Events created by appservman

Appservman sends an event called *ApplicationServerStatusChanged* from the TWSObjectsMonitor provider to the configured event monitoring process every time the status of the application server changes.

Application server - encrypting the profile properties files

You use the **encryptProfileProperties** script to encrypt passwords in the following embedded WebSphere Application Server property files:

- *TWA_home/eWAS/profiles/twaprofile/properties/soap.client.props*
- *TWA_home/eWAS/profiles/twaprofile/properties/sas.client.props*
- *TWA_home/eWAS/profiles/twaprofile/properties/sas.stdclient.properties*
- *TWA_home/eWAS/profiles/twaprofile/properties/sas.tools.properties*

where *PROFILE* can be one of the following:

Component	PROFILE
Master domain manager	twprofile

Application server: encrypting profile properties

Component	PROFILE
Distributed connector	twconnprofile
z/OS connector	twoszconnprofile

An example of when you might use the encryption function is when creating SSL key-stores. You would type the passwords in the key-stores and then encrypt them using the **encryptProfileProperties** script. See *Tivoli Workload Scheduler: Planning and Installation Guide*.

The script uses **PropFilePasswordEncoder.bat**. Restart the server for the changes to take effect.

Encrypting profile properties usage

Encrypt properties uses the following syntax:

```
encryptProfileProperties.bat (.sh)
```

Application server - updating the Windows services after modifications

If you have modified any of the following, you must update the Windows service that runs the application server:

- The user ID and password of the local operating system user that runs the application server process
- The installation directory of the embedded WebSphere Application Server
- The directory where the Tivoli Workload Scheduler application server profile is stored

To update the service, run the **updateWasService** command from the `<TWA_home>/wastools` directory.

Note: You can also use this command to change the way that the application server is started.

At runtime, the script calls **WASService.exe**.

Updating the embedded WebSphere Application Server services usage

To update the embedded WebSphere Application Server services use the following syntax:

updateWasService

Format:

```
updateWasService -userid <TWS_user> -password <TWS_user_password>  
[-wasuser <WAS_user> -waspassword <WAS_user_password>]  
[-startType {automatic | manual | disabled}]  
[-wasHome <WebSphere_install_directory>]  
[-profilePath <server_profile_directory>]
```

Application server: updating the Windows services

Parameters:

-userid <TWS_user> **-password** <TWS_user_password>
Supply the <TWS_user> and its password.

[-wasuser <WAS_user> **-waspassword** <WAS_user_password>]
The user that the local operating system uses to run the application process is set by default to the <TWS_user>. If you want to change it to a different user and password, supply this parameter.

Note: Due to a known problem with this utility, when you change the password you must first use the Windows facility for changing the password, as described in "Action 7 - Windows - update Windows services" on page 247, and then run this utility, giving the new password you have just set as the <WAS_user_password>.

[-startType {automatic | manual | disabled}]
The application server starts automatically by default, when the computer is started. If you want to have a different behavior supply this parameter.

[-wasHome <WebSphere_install_directory>]
If you have changed the name of the installation directory of the embedded WebSphere Application Server, supply this parameter indicating the new name.

[-profilePath <server_profile_directory>]
If you have changed the name of the directory where the Tivoli Workload Scheduler application server profile is stored, supply this parameter indicating the new name.

Application server - updating the SOAP properties after changing the WebSphere Application Server user or its password

In UNIX, if you have changed the user ID or the password of the WebSphere Application Server administration user either for Tivoli Workload Scheduler or the Tivoli Dynamic Workload Console, you must also update the SOAP client properties.

To update the properties, run the **updateWas.sh** command from the <TWA_home>/wastools directory.

Updating the SOAP properties usage

To update the SOAP properties use the following syntax:

updateWas.sh

Format:

updateWas.sh **-user** <new_WAS_admin_user> **-password** <password>

Parameters:

-user <new_WAS_admin_user> **-password** <password>
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.

Application server - configuration files backup and restore

The application server has configuration files, which should be backed up whenever you change them. Use the **backupConfig** script in the `<TWA_home>/wastools` directory.

The files are restored, if necessary, using the **restoreConfig** script in the same directory.

There is no need to stop the application server to perform the backup, but you must stop and restart the server if you have restored the files from a previous backup.

For further information, see the *IBM Redbooks: WebSphere Application Server V6 System Management & Configuration Handbook*.

Backup usage

Backup uses the following syntax:

```
backupConfig.bat [backup_file]
                  [-nostop]
                  [-quiet]
                  [-logfile file_name]
                  [-replaceLog]
                  [-trace]
                  [-username user_ID]
                  [-password password]
                  [-profileName profile]
                  [-help]
```

The following is an example of **backupconfig.bat**:

```
C:\Program Files\ibm\TWA0\wastools>backupConfig.bat
ADMU0116I: Tool information is being logged in file
          C:\Program Files\ibm\TWA0\ewAS\profiles\twaprofile\logs\
          backupConfig.log
ADMU0128I: Starting tool with the twsprofile profile
ADMU5001I: Backing up config directory
          C:\Program Files\ibm\TWA0\ewAS\profiles\twaprofile\config to file
          C:\Program Files\ibm\TWA0\wastools\WebSphereConfig_2005-12-12.zip
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1
ADMU2010I: Stopping all server processes for node DefaultNode
ADMU0512I: Server server1 cannot be reached. It appears to be stopped.
.....
ADMU5002I: 137 files successfully backed up
```

Restore usage

Restore uses the following syntax:

```
restoreConfig.bat backup_file
                  [-location restore_location]
                  [-quiet]
                  [-nowait]
                  [-logfile file_name]
                  [-replaceLog]
                  [-trace]
                  [-username user_ID]
                  [-password password]
                  [-profileName profile]
                  [-help]
```

The following is an example of **restoreConfig.bat**:

Application server: configuration files: backup and restore

```
C:\Program Files\ibm\TWA0\wastools>restoreConfig.bat WebSphereConfig_2005-12-11.zip
ADMU0116I: Tool information is being logged in file
           C:\Program Files\ibm\TWA0\ewAS\profiles\twaprofile\logs\
           restoreConfig.log
ADMU0128I: Starting tool with the twsprofile profile
ADMU0505I: Servers found in configuration:
ADMU0506I: Server name: server1
ADMU2010I: Stopping all server processes for node DefaultNode
ADMU0512I: Server server1 cannot be reached. It appears to be stopped.
ADMU5502I: The directory C:\Program Files\ibm\TWA0\ewAS\profiles\twaprofile\config
           already exists; renaming to
           C:\Program Files\ibm\TWA0\ewAS\profiles\twaprofile\config.old
ADMU5504I: Restore location successfully renamed
ADMU5505I: Restoring file WebSphereConfig_2005-12-11.zip to location
           C:\Program Files\ibm\TWA0\ewAS\profiles\twaprofile\config
.....
ADMU5506I: 127 files successfully restored
ADMU6001I: Begin App Preparation -
ADMU6009I: Processing complete.
```

Application server - changing the host name or TCP/IP ports

To modify the host name of the computer where the application server is installed, or the TCP/IP ports it uses, run the **changeHostProperties** script.

The procedure requires you to stop the application server, create a text file of the current host properties, edit the file, run the utility and restart the application server. Find information about how to do this as follows:

- “Application server - using the utilities that change the properties - reference” on page 277 gives a generic description of the procedure for making any change to the WebSphere Application Server properties
- “Changing host properties” on page 274 lists all the host properties and gives other reference information about the utility
- When editing the text file of the current host properties, do as follows:
 1. Edit the text file and locate the following properties:

```
#####
# Host Configuration Panel
#####

# Old Hostname
oldHostname=myoldhost.mydomain.com

# New Hostname
newHostname=myhost.mydomain.com

#####
Ports Configuration Panel
#####
bootPort=41117
bootHost=myhost.mydomain.com
soapPort=41118
soapHost=myhost.mydomain.com
httpPort=41115
httpHost=*
httpsPort=41116
httpsHost=*
adminPort=41123
adminHost=*
adminSecurePort=41124
adminSecureHost=*
sasPort=41119
sasHost=myhost.mydomain.com
```

Application server: modifying host name or TCP/IP ports

```
csiServerAuthPort=41120
csiServerAuthHost=myhost.mydomain.com
csiMuthualAuthPort=41121
csiMuthualAuthHost=myhost.mydomain.com
orbPort=41122
orbHost=myhost.mydomain.com
```

The rules for changing these values are as follows:

- To change the host name, supply both `oldHostname` and `newHostname`. Also check that the values of `bootHost` and `csiServerAuthHost` are set correctly (normally to the new host name).
- If you change the host name, the old host port settings are used, unless you specifically change them.
- If you do not supply a port number it is not changed.

Do not change any other properties.

Note: The utility might display a message from the application server (WASX7357I:). You can ignore this message.

Changing host properties

You use the `changeHostProperties` script to change the workstation host name in the WebSphere Application Server configuration files, or the TCP/IP ports used by WebSphere Application Server. To change or disable TCP/IP Ports see “Disabling TCP/IP ports” on page 275.

The procedure for running the script is described in detail in “Application server - using the utilities that change the properties - reference” on page 277, but in summary you do the following:

- Run `showHostProperties.sh (.bat) > my_file_name` to obtain the current properties
- Edit `my_file_name`
- Run `changeHostProperties.sh (.bat) my_file_name`

Note: `my_file_name` must be the fully qualified path of the file.

The change utility calls the `wsadmin` utility by running `ChangeHostProperties.jacl` with the properties file as input.

Only the WebSphere Application Server `serverindex.xml` configuration file is affected by this script. The path of the file is:

```
TWA_home/eWAS/profiles/twaprofile/config/cells/DefaultNode/nodes/
DefaultNode/serverindex.xml
```

where `<profile>` is one of `twspprofile`, `twconnprofile`, or `twszconnprofile` for the master domain manager, the distributed connector and the z/OS connector respectively.

The following is a list of the properties that can be changed with this utility:

```
#####
# Host Configuration Panel
#####

# Old Hostname
oldHostname=myoldhost.rome1ab.ibm.it.com

# New Hostname
```

```

newHostname=mynewhost.romelab.ibm.it.com

#####
Ports Configuration Panel
#####
bootPort=41117
bootHost=myhost.mydomain.com
soapPort=41118
soapHost=myhost.mydomain.com
httpPort=41115
httpHost=*
httpsPort=41116
httpsHost=*
adminPort=41123
adminHost=*
adminSecurePort=41124
adminSecureHost=*
sasPort=41119
sasHost=myhost.mydomain.com
csiServerAuthPort=41120
csiServerAuthHost=myhost.mydomain.com
csiMuthualAuthPort=41121
csiMuthualAuthHost=myhost.mydomain.com
orbPort=41122
orbHost=myhost.mydomain.com

```

All of the properties in the properties file are optional. When you are modifying the properties file you should be aware of the following:

- When **oldHostname** and **newHostname** are provided (these settings must be provided together), the host property related to each port is updated when it has not been provided in the properties file and the current value matches **oldHostname**
- Port settings are updated only if they are provided in the properties file
- A port-specific host setting, such as **httpHost** is not updated if not specified except when its current setting matches **oldHostname**
- An empty setting is considered as not provided

Disabling TCP/IP ports

Using the **changeHostProperties** script you can also disable some TCP/IP ports by setting the value of the following corresponding properties to **false**. If you are using Secure Sockets Layer Communication (SSL) in your network, disabling the non-secure HTTP and Administrative Console ports will ensure that only encrypted communication occurs in your network. By default these ports are all enabled. To disable them use the following properties:

httpEnabled

To disable the httpPort.

httpsEnabled

To disable the httpsPort.

adminEnabled

To disable the adminPort.

adminSecureEnabled

To disable the adminSecurePort.

Displaying the current host properties

To display the current properties, use the following utility:

Changing host properties

UNIX	<code>showHostProperties.sh</code>
Windows	<code>showHostProperties.bat</code>

Application server - changing the trace properties

You can use the `changeTraceProperties` script to change the embedded WebSphere Application Server trace properties.

The script calls the `wsadmin` utility by running the `ChangeServerTracing.jacl` with a properties file whose template is `TracingProps.properties`.

See the *Tivoli Workload Scheduler: Troubleshooting Guide* for full details on how to change the trace settings.

The properties file defines the following trace modes:

```
wsmm_odr=com.ibm.ws.xd.comm.*=all:com.ibm.wsmm.grm.Controller=all
:com.ibm.ws.xd.work*=all
:com.ibm.ws.xd.arfm.*=all:com.ibm.wsmm.policing.*=all
:com.ibm.wsmm.xdglue.*=all:com.ibm.ws.odc.ODCTreeImpl$Save=all
wsmm_node=com.ibm.ws.xd.comm.*=all:com.ibm.ws.xd.placement*=all
:com.ibm.ws.xd.arfm.*=all
reset*=info
wsmm007=com.ibm.wsmm.policing.*=all
dcs=DCS=finest:RMM=finest
ham=hamanageditem=all
tcpdcs=DCS=finest:RMM=finest:com.ibm.ws.tcp.channel.*=finest
tcp=com.ibm.ws.tcp.channel.*=finest
vizcache=com.ibm.ws.xd.visualizationengine.cacheservice.cacheimpl.*=all
runtime=com.ibm.ws.console.xdruntime.*=all
proxy=com.ibm.ws.console.proxy.*=all
placement=com.ibm.ws.xd.placement*=all=enabled
charting=com.ibm.ws.console.chart.*=all
dwlml=com.ibm.ws.dwlml.*=all
operationalpolicy=com.ibm.ws.xd.operationalpolicymonitor.*=all
wsmm_na*=info:com.ibm.ws.xd.comm.*=all:com.ibm.ws.xd.placement*=all
:com.ibm.ws.xd.workprofiler.
*=all:com.ibm.ws.xd.arfm.*=all:com.ibm.ws.dwlml.*=all:com.ibm.ws.xd.hmm.
*=all:com.ibm.ws.xd.admin.utils.*=all:com.ibm.ws.clustersensor.impl.*=all
:com.ibm.ws.xd.placement.memory.profiler.impl.*=off
wsmm_o*=info:com.ibm.ws.xd.comm.*=all:com.ibm.wsmm.grm.Controller=all
:com.ibm.ws.xd.workprofiler.
*=all:com.ibm.ws.xd.arfm.*=all:com.ibm.wsmm.policing.*=all
:com.ibm.wsmm.xdglue.*=all
:com.ibm.ws.dwlml.*=all:com.ibm.ws.dwlml.client.*=off
dmgr=com.ibm.ws.odc.*=all
:com.ibm.ws.xd.visualizationengine.cacheservice.cacheimpl.
DeploymentTargetCache*=all
grid=grid.capacityplacement=all
webcontainer=com.ibm.ws.webcontainer.*=all:com.ibm.ws.http.*=all
odc=com.ibm.ws.odc.*=all:com.ibm.ws.dwlml.client.*=all
:com.ibm.ws.xd.dwlml.client.*=all:com.ibm.ws.proxy.*=all
tws_all=com.ibm.tws.*=all
tws_alldefault=com.ibm.tws.*=error=enabled
tws_db=com.ibm.tws.dao.model.*=all:com.ibm.tws.dao.rdbms.*=all
tws_planner=com.ibm.tws.planner.*=all:com.tivoli.icalendar.*=all
:com.ibm.tws.runcycles.
*=all:com.ibm.tws.conn.planner.*=all:com.ibm.tws.cli.planner.*=all
tws_cli=com.ibm.tws.cli.*=all:com.ibm.tws.objects.*=all
tws_utils=com.ibm.tws.util.*=all
tws_conn=com.ibm.tws.conn.*=all:com.ibm.tws.objects.*=all
:com.ibm.tws.updatemanager.*=all
:com.ibm.tws.dao.plan.*=all
tws_secjni=com.ibm.tws.audit.*=all:com.ibm.tws.security.*=all
```

```
sec_all=com.ibm.ws.security.*=all
tw_s_with_sec=com.ibm.ws.security.*=all:com.ibm.tws.*=all
tw_s_broker_all=com.ibm.scheduling.*=all
tw_s_broker_rest=com.ibm.scheduling.jobmanager.rest.*=all
tw_s_bridge=com.ibm.scheduling.tws.*=all
```

You can define other trace modes and update `TracingProps.properties` or create a new properties file.

Two settings that must not be changed are the server name (default `server1`), and the node (default `DefaultNode`).

Application server - using the utilities that change the properties - reference

This section documents a common procedure that you are advised to follow when using the following utilities:

- `changeDataSourceProperties`
- `changeHostProperties`
- `changeSecurityProperties`

To avoid the risk of changing a configuration value inadvertently, you should follow a procedure that creates a file containing the current properties, edit it to the values you require, and apply the changes. The details are as follows:

1. Log on to the computer where Tivoli Workload Scheduler is installed as the following user:

UNIX `root`

Windows Any user in the *Administrators* group.

2. Access the directory: `<TWA_home>/wastools`
3. Stop the WebSphere Application Server using the **conman stopappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
4. From that same directory run the following script to create a file containing the current properties:

UNIX `show<property_type>Properties.sh > my_file_name`

Windows `show<property_type>Properties.bat > my_file_name`

where `<property_type>` is one of the following:

- `DataSource`
 - `Host`
 - `Security`
5. Edit `my_file_name` with a text editor. Check the start of the file. The command might have written a message from the application server (WASX7357I:) at the beginning of the file. Delete this message.
 6. Change the value of the configuration parameters, according to your requirements. You do not need to supply all of the parameters in the file.
 7. Save the file `my_file_name`.
 8. Run the script:

Application server: using the utilities that change its properties

Windows `change<property_type>Properties.bat my_file_name`
UNIX `change<property_type>Properties.sh my_file_name`

where *<property_type>* is the same as used in step 4, and *my_file_name* is the *fully qualified path* of the file containing the new parameters.

The properties are updated, according to the rules given in the descriptions of each property type.

9. Start the WebSphere Application Server using the **conman startappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
10. Check that the change has been implemented in Tivoli Workload Scheduler.

Understanding the templates

As indicated in the overview to these utilities, a template file of properties is supplied with the product for each of these utilities. However, this template file does not contain any of the configuration values that were created by the product installation process or any previous uses of the configuration utilities. If you decide to use the template instead of creating the properties document as described in step 4 on page 277, you must be careful to ensure that the values you enter in the file for every parameter are those that you require to be used by the application server.

Application server - background information about the utilities

This section provides background information about the utilities provided with the embedded WebSphere Application Server that supports Tivoli Workload Scheduler.

The utilities are a set of scripts based on Windows batch files, UNIX and Linux shell scripts, and WebSphere Jacl procedures. The scripts run the WebSphere Application Server embedded utilities that perform the reconfiguration. Many of them load configuration settings from a properties file. Templates for these files are also provided.

Tivoli Workload Scheduler installs the following Windows scripts:

- backupConfig.bat
- changeDataSourceProperties.bat
- changeHostProperties.bat
- changeSecurityProperties.bat
- changeTraceProperties.bat
- encryptProfileProperties.bat
- restoreConfig.bat
- showDataSourceProperties.bat
- showHostProperties.bat
- showSecurityProperties.bat
- startWas.bat
- stopWas.bat
- updateWasService.bat
- manage_ltpa.bat
- modifyThreadPool.bat
- InstallOracledataSource.bat

Application server utilities - background info

Tivoli Workload Scheduler installs the following UNIX and Linux scripts:

- backupConfig.sh
- changeDataSourceProperties.sh
- changeHostProperties.sh
- changeSecurityProperties.sh
- changeTraceProperties.sh
- encryptProfileProperties.sh
- restoreConfig.sh
- showDataSourceProperties.sh
- showHostProperties.sh
- showSecurityProperties.sh
- startWas.sh
- stopWas.sh
- manage_ltpa.sh
- modifyThreadPool.sh
- InstallOracledataSource.sh
- updateWas.sh
- wasstart.sh

The following templates are installed for both Windows and UNIX operating systems:

- DataSourceProps.properties
- HostProps.properties
- SecurityProps_FULL.properties
- SecurityProps_TEMPLATE.properties
- TracingProps.properties

See “Application server - using the utilities that change the properties - reference” on page 277 to understand how the templates should be used.

Chapter 10. Performance

This chapter provides information about issues that impact performance. Use this information both to prevent problems occurring and to help resolve problems that have occurred.

The topics discussed are as follows:

- “Network traffic”
- “Tracing”
- “Logging” on page 282
- “Maintaining the database” on page 282
- “Symphony file sizing” on page 282
- “Tuning a UNIX domain manager to handle large numbers of fault-tolerant agents” on page 282
- “Tuning job processing on a workstation” on page 282
- “Tuning the database” on page 283
- “Tuning the embedded WebSphere Application Server” on page 284
- “Too many manual job submissions” on page 284
- “Too many file dependency checks” on page 284
- “Workload spreading” on page 284
- “Improving job-processing performance” on page 285
- “Mailbox caching - advantages and disadvantages” on page 285
- “Setting the synch level parameter” on page 286
- “The fault-tolerant switch manager - impact on performance” on page 287
- “Scalability” on page 288
- “Multiple Tivoli Dynamic Workload Console production plan reports” on page 292
- “Tivoli Dynamic Workload Console - adjusting session timeout settings” on page 293

Network traffic

A full description of how a Tivoli Workload Scheduler network is structured, and how the different nodes communicate, is provided at the beginning of Chapter 6, “Network administration,” on page 143. In particular, see “Optimizing the network” on page 151, which explains how to design and operate your Tivoli Workload Scheduler network to maximize performance.

Tracing

The performance of any workstation can be impacted by the level of tracing it has to perform. The *Tivoli Workload Scheduler: Troubleshooting Guide* has a chapter which explains the diagnostic tools that are available, and within that chapter there is a section about the Autotrace utility, which, as well as discussing how the autotrace feature works, also describes how to customize it to enhance workstation performance.

Tracing

The performance might also be impacted by the tracing activities on the WebSphere Application Server.

Logging

The performance of any workstation can be impacted by the way the Tivoli Workload Scheduler logging mechanism uses memory. The default settings applied in this version are designed to ensure the maximum performance. However, because these defaults are different from the defaults in earlier versions, if you are experiencing performance problems, it is advisable to check that these settings have not been in some way overwritten by the previous values. In the diagnostic tools chapter of *Tivoli Workload Scheduler: Troubleshooting Guide*, there is a section about CCLog, which, apart from discussing how to customize CCLog, also describes how to check the CCLog processing defaults.

Maintaining the database

Maintaining the database in a good state of organization is important to optimize performance. See “Reorganizing the database” on page 203 for details.

Symphony file sizing

To calculate the size of the Symphony file and understand its impact on performance, see “Avoiding full file systems” on page 204.

Tuning a UNIX domain manager to handle large numbers of fault-tolerant agents

The performance of domain managers on UNIX is impacted if they serve large numbers of fault-tolerant agents. Improvements can be obtained by modifying the kernel parameters. The precise settings differ according to operating system, and you might need to test different settings to obtain optimum performance.

The following is an example of the kernel settings for HP-UX to handle approximately 200 fault-tolerant agents:

```
max_thread_proc=256
nprocess=1800
maxusers=120
maxuprc=1700
nfllocks=500
maxfiles=1024
```

Tuning job processing on a workstation

This section explains how to tune selected options in the Tivoli Workload Scheduler localopts file to improve Tivoli Workload Scheduler performance. These options control the period between successive instances of an activity. Table 62 on page 283 shows the activities to be tuned, the corresponding option that can be set in the localopts file, and how the changed value impacts performance.

Table 62. Options for tuning job processing on a workstation

Activity	Option	Impact on performance
batchman periodically scans the Symphony file for jobs ready to be processed.	<i>bm look</i>	<p>In all these cases, a shorter time means more frequent scans, using more cpu resources, and impacting other processes that are running. However, it also means that for all activities waiting time is kept to a minimum. If throughput is important and the workstation has plenty of memory, try shortening the times.</p> <p>A longer period between successive activities means jobs take longer to run, because there are longer waits for each activity. However, the reduced frequency of the scans means that more memory is available for jobs because less is being used by these monitoring activities.</p> <p>Consider the meaning of the various options. If your objective is to run the jobs as quickly as possible, but you are not concerned about how quickly the information about completed jobs is distributed, you could reduce the wait periods for <i>bm look</i> and <i>jm read</i>, but increase the periods for the others.</p> <p>Alternatively, to speed up the overall job processing time (from initial job launch to the update with the completion status), you can tune <i>bm look</i>, <i>jm look</i>, and <i>mm read</i>.</p>
jobman accesses the Courier.msg file to see if there are jobs that need to be launched.	<i>jm read</i>	
After having launched a job jobman checks periodically for job completion status.	<i>jm look</i>	
mailman looks periodically in the Mailbox.msg for completed jobs.	<i>mm read</i>	
batchman checks periodically in Intercom.msg for jobs that are complete so that it can update the Symphony file.	<i>bm read</i>	

If you decide to tune these setting do the following:

- Test the result in a test system before applying changes in your production environment. To get worthwhile results, the test environment must have the same characteristics as the production environment.
- Modify only the parameters that are necessary. It is better to modify one at a time and thoroughly test the change in performance, rather than changing all at once.
- Make a backup copy of the localopts file to ensure you can revert to the default options if necessary.

Stop and start the agent to activate changes applied to the localopts file.

Tuning the database

To learn about tuning the database, consult the relevant product documentation:

DB2 Go to <http://publib.boulder.ibm.com/infocenter/db2luw/v9r5/index.jsp>, and select **Best practices**.

Oracle See the *Performance Tuning Guide* in the Oracle documentation set.

Tuning the embedded WebSphere Application Server

To learn about tuning the embedded WebSphere Application Server, consult the appropriate documentation.

Go to <http://publib.boulder.ibm.com/infocenter/wasinfo/v6r1/index.jsp>, select **WebSphere Application Server (Distributed platforms and Windows), Version 6.1** and then **Tuning performance**.

Inadequate Java heap size

The default Java maximum heap size might be too small for your requirements. If you have any reason to suspect the performance of the embedded WebSphere Application Server, increase the heap size as described in “Increasing application server heap size” on page 289.

Too many manual job submissions

Tivoli Workload Scheduler is designed for maximum efficiency when handling jobs submitted using a scheduled plan. Consequently, it is less adapted to processing manually submitted jobs. Thus, performance can be improved by reducing the number of manually submitted jobs.

Too many file dependency checks

Each file dependency check has an impact on performance. If you design a plan that is constantly checking many file dependencies, you reduce the performance of the workstation where these jobs are being run.

If multiple “opens” files are being used as a dependency, use the “-a” (and) option. For example, to check if three home directories /tom, /dick, and /harry exist, before launching myjob issue the following:

```
job2 opens "/users" (-d %p/tom -a -d %p/dick -a -d %p/harry)
```

This checks for all three directories at the same time, instead of looking for each directory separately.

Other factors that impact performance when evaluating file dependencies are the **bm check** parameters in the localopts file. These are documented in the *Tivoli Workload Scheduler: Planning and Installation Guide* in the chapter on customization.

Workload spreading

Whatever jobs you have to schedule, try and spread them out through the production period so that there is no concentration in any one moment. Try also to avoid scheduling activities during times when normal user traffic in the network is very heavy, for example during the morning when users commence working and deal with accumulated e-mails.

Failure to do this might cause a bottleneck at the Mailbox.msg queue, which causes delays in updating the Symphony file, which in turn creates delays in the availability of job statuses to **conman**, the Tivoli Dynamic Workload Console, or the Job Scheduling Console.

Improving job-processing performance

The processing and monitoring of jobs on a workstation is controlled primarily by various parameters in the `localopts` file and the global options maintained by **optman**. These parameters are described in the *Tivoli Workload Scheduler: Planning and Installation Guide*.

If you are experiencing problems of performance when processing and monitoring jobs, contact IBM Software Support for advice about how to tune these parameters in your particular environment to improve performance.

Mailbox caching - advantages and disadvantages

Mailman uses a parameter in the `localopts` file to decide whether to cache mailbox messages: `mm cache mailbox`. This section explains the advantages and disadvantages of the on and off settings of this parameter.

Setting the `mm cache mailbox` parameter to `no`

This means that mailman has to make a separate read action for each message before processing it, and then a separate delete action after successfully processing the message. The I/O activity in performing these activities one message at a time is proportionally high for the amount of data being read. This has an impact on performance. On the other hand, the processing is simple, in that each message is read, processed, and then removed from the mailbox. Any failure of the system at any point means that at most one message is replayed and no data is lost.

Setting the `mm cache mailbox` parameter to `yes` (default)

This means that mailman reads a block of messages into cache memory, processes all of the messages, and then deletes all of them from the mailbox. The advantage in I/O time is clear; reading and deleting a sequential set of messages in one action is a much more efficient use of I/O time, than reading and deleting them one-by-one, meaning improved performance.

However, if there is a failure of mailman or the operating system, the cache is lost. On restarting, mailman rereads the set of messages that were previously in cache, some of which might already have been processed. For example, if mailman reads a block of 32 messages into cache and has processed 30 of them when a problem occurs, when mailman is restarted it rereads those 32 records and has to process 30 duplicates before being able to continue where it stopped.

Most events deal with job state changes, and these events can be repeated without creating any problems, and the critical events mechanism is able to deal with the others. However, there is an impact on performance while this recovery processing is going on, and if the in-built mechanisms cannot handle the message duplication, a more serious error might occur, ultimately involving the full or partial loss of the mailbox contents.

The number of messages being read in one action is configurable, using the parameter `mm cache size`. The default value for this parameter is 32 messages, and the maximum is 512. Setting this parameter to a value higher than the default increases performance during correct working, but decreases the performance in the event of a failure, for the reasons stated above. In addition, the additional cache means that the memory required by the Tivoli Workload Scheduler engine also increases. If you have a workstation with limited memory, or memory-heavy applications running,

Mailbox caching

it might be counterproductive to increase the mailbox cache because the operating system might have to start paging the cache memory.

In conclusion, the default setting maximizes performance; only if you start losing events should you set it to *no*.

Setting the synch level parameter

This section describes the impact of the different settings of the *synch level* parameter in the *localopts* file. The *synch level* parameter only impacts UNIX environments.

The I/O activity performed by the Tivoli Workload Scheduler engine in managing plans, job streams, and jobs, consists in reading from and writing to the Symphony file and the event files (*Mailbox.msg*, *Intercom.msg*, and *Courier.msg*). When Tivoli Workload Scheduler writes to these files it has more than a straightforward *write* operation to perform. For example, when it writes to the *Mailbox.msg* file it performs the actions described in the following pseudo code:

```
TWS_write_event_lock {
    Lock Mailbox to write
}

TWS_write_event_update {
    Check Available Space
    Write Header
    Write Record
    Update Write Pointer
    Unlock Mailbox
}
```

Each action requires one or more write accesses to the disk. The way these actions are performed with the different *synch level* options is as follows:

synch level = high

Each write operation on the event files is immediately physically written to disk. This has a heavy impact on performance caused by the high I/O dependency.

synch level = medium

Each write event is considered as a single operation. For example, while *TWS_write_event_lock* contains only one action, *TWS_write_event_update* comprises five actions. With *synch level* at *medium*, the five actions in this write event would be completed in one physical disk access, thus drastically reducing the I/O overhead.

synch level = low (default)

The operating system decides how and when to synchronize the data to disk. The impact of this option is more difficult to assess, because the rules are different for each operating system and file system.

The fault-tolerant switch manager - impact on performance

This section describes the impact that the enablement of the fault-tolerant switch manager feature has on the performance of the general architecture and the individual system. The fault-tolerant switch manager is enabled by setting the `enSwfaultTo1` global option to *yes*. When it is set, the master domain manager distributes messages to all fault-tolerant agents with *FullStatus* set to *yes*.

Enabling this option impacts the following:

- Network traffic
- Disk space

Note: The fault-tolerant switch manager facility is only available if all of the workstations in the domain are at version 8.2, fix pack level 4, or higher.

Network Traffic

Network traffic is unchanged under normal conditions, but is increased during the replay phase, according to your choice and only under special conditions.

The replay phase is an essential part of the processing performed by the **switchmgr** command. It occurs when the new domain manager processes its Symphony file against its copies of the messages received, as it attempts to update its copy of the Symphony file.

Under normal conditions, the outbound reliability does not create any additional network traffic, because the messages are only stored for an eventual replay operation. The multiple inbound connections do not generate additional traffic because the traffic that was previously copied by the domain manager to the *FullStatus* member is now copied to the *FullStatus* members directly by the fault-tolerant agents.

During the replay phase, the connection protocol initiated by mailman on the backup domain manager includes a new phase for the replay of messages not sent by the failed domain manager. The impact of the message replay might be important, depending on the number of messages "trapped" in the old domain manager.

Disk Space

There are two places within the network where disk space use increases following the activation of the additional fault tolerance.

These places are as follows:

- On the single fault-tolerant agent. Here, in addition to the `tomaster.msg` queue, new queues are created for the other *FullStatus* fault-tolerant agents. These queues need not be considered, because the impact on a single agent is small.
- On the *FullStatus* fault-tolerant agents acting as backup domain managers. Here new `ftbox` message files are created. Upward traffic to the upper domain manager is in `ftbox/ftup.msg` and downward traffic to the lower domain manager is in `ftbox/ftdown.msg`.

Scalability

In an environment with large numbers of scheduling objects, the following impacts are felt:

- “Impact on **JnextPlan**”
- “Impact on reporting”
- “Impact on event rule deployment” on page 289
-

The resolution for these problems often includes making the following changes:

- “Increasing application server heap size” on page 289
- “Increasing maximum DB2 log capacity” on page 289

Impact on **JnextPlan**

The main impact on performance caused by a large network of workstations running many jobs over a production period of many days, is on **JnextPlan**. The key factor is the number of job stream instances that **JnextPlan** needs to handle. **JnextPlan** has to process each of these instances, and the time it takes to do so is a factor that can only be reduced by ensuring that the master domain manager and the database are on the most powerful computers possible, and that the communication, whether in local or remote, between the master domain manager and the database is maximized.

However, there are some specific measures that need to be taken as the number of jobs or job stream instances increases:

Number of jobs in the plan exceeds 40 000

In this event you need to increase the Java heap size used by the application server. The default is 512 MB, and you should at least double the heap size when job numbers exceed this level. Follow the procedure in “Increasing application server heap size” on page 289.

You have a large number of job stream instances in the plan

DB2 The default DB2 transaction log files cannot handle more than the transactions generated by about 180 000 job stream instances. You need to change the parameters that control the log file sizes or the numbers of log files that can be created, or both. Follow the procedure in “Increasing maximum DB2 log capacity” on page 289.

Oracle The number of transactions that can be managed by the Oracle log files depends on the way the Oracle database is configured. See the Oracle documentation for more details.

Note: If circumstances change and the number of job stream instances handled by **JnextPlan** falls below about 180 000, consider resetting the log and application server heap size settings to their default values, to avoid performance problems.

Impact on reporting

When a report is being processed, extra memory is required to handle large numbers of scheduling objects. The critical point is approximately 70 000 objects. This problem can be handled by increasing the Java heap size used by the application server. Follow the procedure in “Increasing application server heap size” on page 289.

Impact on event rule deployment

When deploying large numbers of event rules, extra memory is required. The critical point is approximately 8 000 rules. This problem can be handled by increasing the Java heap size used by the application server. Follow the procedure in “Increasing application server heap size.”

Increasing application server heap size

Follow this procedure to increase the Java heap size:

1. Log on to the computer where Tivoli Workload Scheduler is installed as the following user:

UNIX	root
Windows	Any user in the <i>Administrators</i> group.
2. Access the directory: <TWA_home>/wastools
3. Stop the WebSphere Application Server using the **conman stopappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)
4. Open the following file:


```
<TWA_home>/eWAS/profiles/twaprofile/config/cells/
DefaultNode/nodes/DefaultNode/servers/server1/server.xml
```
5. Locate the following line:


```
<jvmEntries xmi:id="..."
verboseModeClass="false"
verboseModeGarbageCollection="false"
verboseModeJNI="false"
initialHeapSize="16"
maximumHeapSize="512"
runHProf="false"
hprofArguments=""
debugMode="false"
debugArgs="..."
genericJvmArguments=""/>
```
6. Edit `maximumHeapSize="512"` and change it at least `1024`. This would double the maximum heap size from 512 MB to 1 GB. Ensure that the RAM usage of the computer can manage whatever increased size you choose. Save the file.
7. Start the WebSphere Application Server using the **conman startappserver** command (see “Starting and stopping the application server and **appservman**” on page 268)

Increasing maximum DB2 log capacity

The Tivoli Workload Scheduler DB2 database uses a transaction log the maximum size of which is fundamentally important for the successful running of **JnextPlan** on very large databases.

The default log consists of 40 primary log files, which are always present, and 20 secondary log files, created on demand. Each file is about 4 MB in size, so the maximum log capacity using all of the “secondary” log files as well as the primary files is $(40 + 20) \times 4 \text{ MB} = 240 \text{ MB}$.

The log space used by **JnextPlan** is dependent on the size of the preproduction plan. Approximately every 1000 job stream instances generate transactions that occupy 1 MB of space in the log file. Thus, the log files by default have a maximum theoretical capacity of 240 000 job stream instances. However, in

practice, you should allow for at least 25% more space than this algorithm indicates, so the capacity of the default log files is around 180 000 job stream instances.

If **JnextPlan** has neared or exceeded that level, you must make more log space available to DB2.

In addition to performing the above calculation, you can also determine the log space actually used by a specific instance of **JnextPlan** and base your log size requirement on that figure.

Determining actual DB2 log file usage

The following is the procedure to verify how much space was used by a successful instance of the **JnextPlan** command:

1. After **JnextPlan** has run, log on to the computer where the Tivoli Workload Scheduler DB2 server is installed, as the DB2 instance owner (UNIX) or DB2 Administrator (Windows).
2. Open a DB2 command line window or shell, as follows:

UNIX	Follow these steps: <ol style="list-style-type: none">a. Issue the command su - db2inst1, or change to the subdirectory <code>sql1ib</code> of the home directory of the owner of the DB2 instance (by default <code>db2inst1</code>)b. Launch the command ./db2profile
Windows	Select from the Start menu, Programs → IBM DB2 → Command Line Tools → Command Window

3. Run the following command:
`db2 "get snapshot for database on TWS" > snapdb.txt`

where "TWS" must be changed to the actual database name if different

4. Open the `snapdb.txt` file and look for a section like this:

```
Log space available to the database (Bytes)= 244315359
Log space used by the database (Bytes)      = 484641
Maximum secondary log space used (Bytes)   = 0
Maximum total log space used (Bytes)      = 581636
Secondary logs allocated currently         = 0
```

The value shown in "Maximum total log space used" is the actual space used for the DB2 logs. This space should be allocated to DB2 using primary log files only: therefore, you should change the number of primary log files and their size as necessary to meet this requirement as a minimum.

In addition, you are recommended to allocate a secondary log space to DB2. A good choice for the secondary log files is half the number allocated for the primary files.

The snapshot command described in step 3 can be run at any time to keep track of the current usage of the DB2 log space, without a noticeable impact on performance. All metrics shown are useful to monitor the current allocation of DB2 primary and secondary logs at any time, and to determine any required changes.

Procedure for changing the maximum DB2 log capacity

Do this as follows:

1. Log on to the computer where the Tivoli Workload Scheduler DB2 server is installed, as the DB2 instance owner (UNIX) or DB2 Administrator (Windows).
2. Open a DB2 command line window or shell, as follows:

- UNIX** Follow these steps:
- Issue the command `su - db2inst1`, or change to the subdirectory `sql1lib` of the home directory of the owner of the DB2 instance (by default `db2inst1`)
 - Launch the command `./db2profile`

Windows Select from the **Start** menu, **Programs** → **IBM DB2** → **Command Line Tools** → **Command Window**

3. Run the following commands:

```
db2 update db cfg for <database_name> using LOGFILSIZ <log_file_size>
db2 update db cfg for <database_name> using LOGPRIMARY <primary_log_files>
db2 update db cfg for <database_name> using LOGSECOND <secondary_log_files>
```

where:

<database_name>

The name of the database:

- If you are running this from the computer where the DB2 server is installed, the installed default name is *TWS*. Supply this value unless you have changed it.
- You are not recommended to run this procedure from the computer where the DB2 client is installed, but if you do so, the installed default name is *TWS_DB*. Supply this value unless you have changed it.

<log_file_size>

The log file size in 4 KB pages. The default is 1000 (hence the default log file size of 4MB). Look in the DB2 documentation for details of the implications of choosing a larger or a smaller log file size. The maximum value is 262 144 (making the maximum log file size about 1 GB).

<primary_log_files>

The number of primary log files. The default is 40. The total maximum number of log files that DB2 can handle (primary and secondary) is 256. Thus, there is a maximum limit of 256 GB for the log, or approximately 256 million job stream instances! (maximum 256 files x 1 GB maximum file size)

<secondary_log_files>

The number of secondary log files. The default is 20. If there is enough free space on the file system, these additional log files are dynamically allocated by DB2 as needed (with a small impact on the performance of **JnextPlan**). Because these are only created if required, it is preferable to increase the number of secondary files, rather than the primary files. Typically, you allocate 50% of the primary log file value.

In making the calculation to allocate the log files, allow at least 25% more space than you think you require, to avoid that any slight miscalculation causes **JnextPlan** to fail.

Example: if you have determined from the procedure described in “Determining actual DB2 log file usage” on page 290 that **JnextPlan** has a current use of 320 MB, you could calculate as follows:

- Increase 320 MB by 25%, giving 400 MB
- Determine if you want more log files, or bigger log files, or both, by reference to the DB2 documentation. For example, you could choose to allocate 40 files with a size of 10 MB, 80 files with a size of 5 MB, or 100

Scalability

- files with a size of 4 MB. For the sake of this example, assume you have chosen 80 files with a size of 5 MB, so your LOGPRIMARY value will be 80.
- c. Determine the log file size in 4 KB pages to give a log file size of 5 MB - your LOGFILSIZ value will thus be 1250.
 - d. Determine how many secondary log files are required. If you follow the 50% guideline you will need a LOGSECOND value of 40.
4. Log on to the computer where Tivoli Workload Scheduler is installed as the following user:

UNIX	root
Windows	Any user in the <i>Administrators</i> group.
 5. Access the directory: <TWA_home>/wastools
 6. Stop the WebSphere Application Server using the **conman stopappserver** command (see "Starting and stopping the application server and **appservman**" on page 268)
 7. On the computer where the DB2 server is installed, stop and start DB2, as follows:
 - a. Ensure that no other applications are using this instance of DB2, or if they are that they can be stopped.
 - b. Issue the following command:
db2stop
 - c. Issue the following command:
db2start

Note: It is strongly recommended that you stop and start DB2. If this is a problem for you, you must at least disconnect all applications from the DB2 instance and reconnect them. DB2 will apply the new parameters when you reconnect. If necessary, use the following command to force the disconnection of all open connections:

```
db2 "force application all"
```
 8. Start the WebSphere Application Server using the **conman startappserver** command (see "Starting and stopping the application server and **appservman**" on page 268)

Multiple Tivoli Dynamic Workload Console production plan reports

From the Dynamic Workload Console you can launch production plan reports. These are heavy users of CPU time, and if they are requested for the entire plan, they can also take some considerable time to produce. If several are running at once, they can have a noticeable impact on the performance of the master domain manager.

If you notice a degradation of performance, you can determine if there are any reports running by checking for the report work files, as follows;

1. Navigate to the operating system's temporary directory
2. Look for files that have the following file name template:
TWS-<sequential_number>-extr

Each report currently in progress has one of these work files open. The files are removed when the report is completed.

3. Check the dates of these files, and consider only recent files (if a report fails during production at any time, its file remains in the temporary directory until

Multiple Tivoli Dynamic Workload Console production plan reports

the next reboot of the master domain manager or you run an operating system cleanup process that discards all files in the temporary directory).

There is no direct action to take, as you must wait until the report completes for the performance to recover.

However, if you note that large numbers of reports are being issued, it might indicate the following scenario:

1. A user issues a report request, expecting it to be available immediately
2. When the report does not appear immediately, the user thinks it has hung, closes and reopens the browser, and reissues the report. The closing of the browser does not stop the report production.
3. The user might repeat this action several times.

In this case, you can take action to remind the user that the production of large reports can be time-consuming, and that it always better to wait.

Tivoli Dynamic Workload Console - adjusting session timeout settings

The value assigned to the session timeout settings defines after how many minutes a user is automatically logged out from the WebSphere Application Server. If you plan to perform long running operations, or to have many users connected concurrently to the Tivoli Dynamic Workload Console, or expect to have low performance on the system where the Tivoli Dynamic Workload Console is installed, you might want to increase these values.

Perform these steps to change the values assigned to the timeout settings:

1. Open the configuration file:

```
<TWA_home>\eWAS\profiles\tdwc_profile  
\config\cells\tdwc_cell\nodes\tdwc_node\servers\  
tdwc_server\server.xml
```

2. In the file, search for `invalidationTimeout` in the following tag:

```
<tuningParams xmi:id="TuningParams_1188622510500"  
  usingMultiRowSchema="false"  
  maxInMemorySessionCount="1000"  
  allowOverflow="true"  
  scheduleInvalidation="false"  
  writeFrequency="TIME_BASED_WRITE"  
  writeInterval="10"  
  writeContents="ONLY_UPDATED_ATTRIBUTES"  
  invalidationTimeout="30">
```

This is the parameter that sets the HTTP session timeout. By default `invalidationTimeout` is set to 30, which means that a user is logged out automatically after 30 minutes of inactivity.

3. Set `invalidationTimeout` to an appropriate value for your environment and for the activities you plan to perform.
4. Save the file.
5. Open the configuration file:

```
<TWA_home>\eWAS\profiles\tdwc_profile  
\config\cells\tdwc_cell\applications\isclite.ear\  
  deployments\isclite\deployment.xml
```

6. In the file, search for `invalidationTimeout` in the following tag:

```
<tuningParams xmi:id="TuningParams_1188878529796"  
  usingMultiRowSchema="false"  
  maxInMemorySessionCount="1000"
```

Application server: adjusting session timeout settings

```
allowOverflow="true"  
scheduleInvalidation="false"  
writeFrequency="TIME_BASED_WRITE"  
writeInterval="10"  
writeContents="ONLY_UPDATED_ATTRIBUTES"  
invalidationTimeout="30">
```

By default, `invalidationTimeout` is set to 30, which means that a user is logged out automatically after 30 minutes of inactivity.

7. Set `invalidationTimeout` to an appropriate value for your environment and for the activities you plan to perform.
8. Save the file.
9. Open the configuration file:

```
<TWA_home>\eWAS\profiles\tdwc_profile  
\config\cells\tdwc_cell\security.xml
```

10. In the file, search for `timeout` in the following tag:

```
<authMechanisms xmi:type="security:LTPA"  
  xmi:id="LTPA_1" OID="oid:1.3.18.0.2.30.2"  
  authContextImplClass="com.ibm.ISecurityLocalObjectTokenBaseImpl  
.WSSecurityContextLTPAImpl"  
  authConfig="system.LTPA"  
  simpleAuthConfig="system.LTPA"  
  authValidationConfig="system.LTPA"  
  timeout="120"  
  keySetGroup="KeySetGroup_1ab237165Node01_1">
```

By default `timeout` is set to 120, which means that a user is logged out automatically after 120 minutes regardless of whether the user performed any actions on the WebSphere Application Server.

11. Set the `timeout` value in the following section of the file to an appropriate value for your environment and for the activities you plan to perform.
12. Save the file.
13. Restart the WebSphere Application Server.

Chapter 11. Availability

This chapter describes factors that might affect the availability of Tivoli Workload Scheduler on a workstation. It covers the following topics:

- “Resolving Windows user ID account”
- “Using a temporary directory on UNIX” on page 296

Resolving Windows user ID account

Tivoli Workload Scheduler needs to resolve the user ID account on Windows operating systems to verify the security information.

Windows users can be classified as domain users or local users. Domain users are defined in the domain controller, while local users are defined in the workstations of the network.

For a domain user, Tivoli Workload Scheduler requests the primary domain controller (or any domain controller for Windows 2000 or 2003 Active Directory), to identify an available domain controller. It then uses this domain controller identity to fill out the structure for the user.

For a local user, Tivoli Workload Scheduler makes a request to the local workstation. Generally, Tivoli Workload Scheduler specifies two cases: one for the Tivoli Workload Scheduler user and one for the streamlogon user.

The following is a list of steps that Tivoli Workload Scheduler performs to authenticate Windows users, and the APIs involved:

1. Tivoli Workload Scheduler looks up the user in the reference domain. For the domain user, the reference domain is the name of the Windows network. For the local user, it is the name of the local workstation.

API: LookupAccountName.

2. If the user is a domain user, Tivoli Workload Scheduler asks the primary domain controller for any domain controller that is available to resolve the account for the user in the reference domain.

API: NetGetAnyDCName for Windows or DsGetDcName for Windows 2000 or 2003.

3. Tivoli Workload Scheduler requests the domain controller (or the local workstation if the user is local) for information about the user.

API: NetUserGetInfo.

Note: On Windows 2000 and 2003, the permissions for this API are contained in the BUILTIN\“Pre-Windows 2000 compatible access” group.

Using a temporary directory on UNIX

When performing Tivoli Workload Scheduler operations on UNIX, temporary files are written to the temporary directory on the local workstation. Ensure that the `<TWS_user>` running operations has *read* and *write* access to this directory.

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.

Notices

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licenses 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](http://www.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 is a trademark of Intel Corporation in the United States, other countries, or both.



Java and all Java-based trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a 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 company, product, and service names may be trademarks or service marks of others.

Index

Special characters

- jobmanrc 163
- .rhost, file 163
- \$MASTER variable, resolve in mailman, local option 27
- FullStatus mode, setting 146

Numerics

- 2001, service in NetConf file 160
- 2002, service in NetConf file 160
- 2003, service in NetConf file 160
- 2004, service in NetConf file 160
- 2005, service in NetConf file 160
- 2006, service in NetConf file 160
- 2007, service in NetConf file 160
- 2008, service in NetConf file 160
- 2009, service in NetConf file 160
- 2010, service in NetConf file 160
- 2011, service in NetConf file 160
- 2012, service in NetConf file 160
- 2013, service in NetConf file 160
- 2014, service in NetConf file 160
- 2015, service in NetConf file 160
- 2016, service in NetConf file 160
- 2017, service in NetConf file 160
- 2018, service in NetConf file 160
- 2021, service in NetConf file 160
- 2022, service in NetConf file 160
- 2023, service in NetConf file 160
- 2501, service in NetConf file 160
- 2502, service in NetConf file 160
- 2503, service in NetConf file 160

A

- access method, UNIX
 - local 163
 - remote 163
- access, remote, for command-line, configuring 36
- accessibility xiii
- action, object type, defining access 116
- Active Directory, configuring LDAP for 136
- administrative tasks 233
 - DB2 209
 - Oracle 215
- agent 236
- agent process
 - monitoring 150
- agents
 - critical, positioning 151
 - extended
 - See extended agent
 - fault-tolerant
 - See fault-tolerant agent
 - standard
 - See standard agent
- al, global option 9
- altpass, command 241
- application server
 - administrative tasks 233

- application server (*continued*)
 - automatic restart 266
 - backup master domain manager setting, local option 26
 - changing user password 241
 - configuration backup and restore 272
 - configuration files: backup and restore 272
 - configuration utilities: background 278
 - encrypting profile properties 269
 - host name, modify 273
 - increase heap size 289
 - master domain manager setting, local option 26
 - monitor 266
 - multiple instances on same system 80
 - security settings, modify 258
 - starting and stopping 264
 - TCP/IP ports, modify 273
 - tuning 284
 - updating the SOAP properties after changing user or password 271
 - updating the Windows service 270
 - using utilities to change properties 277
 - utilities 278
- ApplicationServerStatusChanged event 269
- approachingLateOffset, global option 9
- appserver
 - auto restart, local option 23
 - check interval, local option 23
 - count reset interval, local option 23
 - max restarts, local option 23
 - min restart time, local option 23
 - service name, local option 23
- appserverbox
 - monitoring 155
- appservman 266
 - local options 267
 - not stopping while stopping the application server 264
 - run from conman, process 161
- appservman process
 - monitoring 150
- archive settings for job data
 - configuring 44
- archived files 207
- at dependency, job streams without, preventing from starting, global option 14
- audit
 - directory
 - as log file location 208
 - plan, enabling, global option 14
- auditing
 - database, enabling, global option 12
- authentication on SMTP connection, use, global option 19
- auto link flag 145
- auto restart, appservman 267
- automatic maintenance, DB2
 - administer 210
 - modify policy 210
 - running manually 211
 - switch off 210
 - switch on 211
- automatic restart of application server 266
- automatically grant logon as batch, global option 14

- autostart monman, local option 23
- availability 295
- average run time calculation, weighting factor, global option 17

B

- backing up
 - log files 203
- backup
 - application server configuration 272
 - backup master domain manager 202
 - log files 201
 - to offline storage 201
- backup domain manager
 - choosing 235
 - configuring 235
 - definition 144
 - network security 235
 - performance 287
 - setting up 235
 - switching 235
- backup master domain manager
 - backing up files to 202
 - choosing 236
 - configuring 237
 - copying files to 237
 - definition 144
 - extended loss of 238
 - permanent change 238
 - promoting from agent 236
 - setting up 237
 - switching 238
- backupConfig, used to backup application server configuration 272
- baseRecPrompt, global option 10
- baseRecPropmt, additional prompts global option 16
- batchman
 - deadline, minimum wait to check , local option 23
 - dependency status, wait to check, local option 24
 - file, minimum wait to check , local option 23
 - intercom.msg file, maximum wait to read, local option 24
 - processes 146
 - production control file, minimum wait to update, local option 24
 - starting 147
 - statistics reporting, enable, local option 24
 - status messages, send to standard list, local option 24
 - tuning 282
 - until time, maximum wait to report expiry of, local option 24
- batchman process
 - monitoring 150
- behind firewall
 - extended agent 187
- behindfirewall option 187
- bm check deadline, local option 23
- bm check file, local option 23
- bm check status, local option 24
- bm check until, local option 24
- bm look, local option 24
- bm read, local option 24
- bm status, local option 24
- bm verbose, local option 24
- books
 - See publications
- bp, global option 10

- BrokerWorkstation.properties
 - workload broker workstation configuration 55

C

- cache.dat, increasing size of 263
- caching mailbox 285
- calendar, object type, defining access 114, 117, 118
- calendars, enabling the copying of into Symphony, global option 16
- can be event processor, local option 24
- caonly, SSL auth mode, local option 29
- carry forward internetwork dependencies, global option 12
- carry forward resource quantities, global option 12
- carry forward, global option 11
- carryStates, global option 10
- CCLog
 - shared memory management 282
- centralized security 98
- centralized security, global option 12
- certificates 170
- certification authority 171
- cf, global option 11
- changeDataSourceProperties script 251
- changeDataSourceProperties, application server utility 252
- changeHostProperties script 273
- changeHostProperties, application server utility 274
- changeSecurityProperties script 258
- changeSecurityProperties, application server utility 259
- changeTraceProperties, application server utility 276
- check interval, appservman 267
- check status of remote job, process start 161
- checkvptproc, run from conman on a client, process 161
- checks, file dependency, impacting performance 284
- ci, global option 12
- clbox
 - monitoring 155
- CLI
 - certificate file when using SSL with server, local option 24
 - default workstation when using, local option 25
 - enabling SSL communications with server, local option 24
 - host name when connecting from, local option 25
 - is installed as, local option 25
 - port, local option 28
 - SSL authentication string, local option 24
 - trusted certificate directory when using SSL with server, local option 24
- clisslcipher, local option 24
- clisslserverauth, local option 24
- clisslservercertificate, local option 24
- clissltrustedddir, local option 24
- cn, global option 10
- command line
 - See CLI
- command line client
 - configuring remote access 36
- command-line prompt
 - composer 24
 - conman 25
- commands
 - console 38
 - dumpsec 96
 - evtsize 158
 - makesec 97
 - optman 5
 - StartUp 149

- commands and scripts
 - .jobmanrc 163
 - altpass 241
 - appservman 266
 - backupConfig 272
 - changeDataSourceProperties 251
 - changeHostProperties 273
 - changeSecurityProperties 258
 - dbreorg 212
 - dbrunstats 211
 - evtsize
 - to monitor queue sizes 154
 - jobmanrc 163
 - link 145
 - restoreConfig 272
 - rmstdlist 207
 - startappserver 266, 268
 - StartUp 147
 - startWas 264
 - stopappserver 266, 268
 - stopWas 264
 - unlink
 - usage 145
 - updateWas 271
 - updateWasService 270
 - communications in the network 145
 - companyName, global option 10
 - composer
 - defining access for working with objects 114
 - prompt, local option 24
 - configuration
 - federated user registry 76
 - for LDAP 79
 - for single sign-on 79
 - ldap user registry 73
 - Tivoli Dynamic Workload Console 73
 - user 77
 - user's portfolio 77
 - configuration file, netman 160
 - configuration files
 - application server, backup and restore 272
 - backing up 202
 - configuring
 - archive settings for job data 44
 - global resource matching 42
 - heartbeat signal 42
 - J2EE communication channel 48
 - J2EE jobs 48
 - LDAP, for TWS 135
 - maximum number of results for global resource matching 42
 - time interval for job allocation to resources 42
 - time interval for notifications on resources 42
 - time interval for retrying failed operations 44
 - WebSphere Application Server 48
 - configuring reports
 - DB2 database 89, 91
 - Oracle database 90
 - configuring Tivoli Dynamic Workload Console
 - global settings file 84
 - conman
 - defining access for working with objects 114
 - prompt, local option 25
 - running appservman 161
 - running checkevtptrc on a client 161
 - running deployconf 161
 - running startappserver 161
 - conman (*continued*)
 - running startevtptrc 161
 - running stopappserver 161
 - running stopevtptrc 161
 - running stopevtptrc on a client 161
 - running stopmon 161
 - running switchevtptrc 161
 - starting the find and return of a stdlist file 161
 - connection failed, wait to retry in netman, local option 28
 - connection parameters
 - configuring 36
 - connectivity, impact on network 151
 - console
 - users and groups 77
 - console command 38
 - console manager, start process 161
 - console messages and prompts 37
 - conventions used in publications xii
 - copying files to backup master domain manager 237
 - count reset interval, appservman 267
 - courier
 - monitoring 155
 - courier.msg file, wait to read in jobman, local option 26
 - cpu, object type, defining access 114
 - cpu, SSL auth mode, local option 29
 - critical agents, positioning 151
 - critical path processing, enabling, global option 15
 - cs, global option 10
 - custom user registry, modify security settings 263
 - customer support
 - See* Software Support
 - customization 5
- ## D
- da, global option 12
 - data flows, planning for 153
 - data maintenance 201
 - data volumes, impact on network 151
 - database
 - back up
 - to backup master domain manager 202
 - to offline storage 201
 - maintenance 201
 - migrating from DB2 to Oracle and *vice versa* 217
 - name, change 251
 - reorganization 203
 - database audit enabling, global option 12
 - date format, local option 25
 - DB2
 - administrative tasks 209
 - automatic maintenance
 - administer 210
 - modify policy 210
 - running manually 211
 - switch off 210
 - switch on 211
 - changing database user password 241
 - database name, change 251
 - host name, change 251
 - increase maximum log capacity 289
 - migrating to Oracle 217
 - monitoring the lock list memory 213
 - passwords not used by TWS, changing 209
 - port, change 251
 - reorganization 203
 - reorganize database 212

- DB2 (*continued*)
 - tools, locating 209
 - tuning 283
 - user permissions for running the tools 210
 - DB2 database
 - configuring reports 89, 91
 - dbreorg, DB2 tool 212
 - dbrunstats, DB2 tool 211
 - deadline, minimum wait to check , local batchman option 23
 - deadlineOffset, global option 11
 - defaulttws, local option 25
 - dependency checks, file, impacting performance 284
 - dependency status, batchman wait to check, local option 24
 - deploymentFrequency, global option 11
 - deplyconf, run from conman, process 161
 - df, global option 11
 - direct scheduling
 - J2EE jobs 48
 - directories
 - audit 208
 - logs 207
 - methods 208
 - schedForecast 208
 - schedlog 207
 - schedTrial 208
 - stdlist 207
 - tmp, as location for temporary files 209
 - traces 207
 - disk filling
 - monitoring 205
 - disk full
 - monitoring 205
 - disk space
 - maintaining enough available space 204
 - monitoring 205
 - threshold 205
 - used by backup domain manager impacting performance 287
 - dm.msg message queue 154
 - do, global option 11
 - domain
 - definition 144
 - master, definition 144
 - parent, definition 144
 - structure of, impact on critical agents 151
 - domain manager
 - backup
 - See* backup domain manager
 - backup master
 - See* backup master domain manager
 - data flows 153, 154
 - definition 144
 - failure 234
 - IP address validation 167
 - log file maintenance 207
 - loss of 234
 - master
 - See* master domain manager
 - mitigating loss of 234
 - network planning, role of 151
 - optimizing for critical activities 151
 - positioning for critical activities 151
 - role of in network planning 151
 - running without 234
 - switching 235
 - temporary files 209
 - without 234
 - domain user, resolving account in Windows 295
 - download scripts from z/OS master domain manager, starting process 161
 - DsGetDcName, API used to resolve Windows user ID account 295
 - dumpsec command 96
 - duration, job, long, threshold, global option 17
 - dynamic workload broker
 - changing URI data 240
 - exportserverdata 240
 - importserverdata 240
 - start 264
 - stop 264
 - dynamic workload broker security
 - workload broker security roles and users and groups 56
- ## E
- ed, global option 13
 - education
 - See* Tivoli technical training
 - ee, global option 15
 - eh, global option 13
 - EIF event queue, increasing size of 263
 - EJB
 - J2EE jobs 48
 - empty job streams, global option 13
 - enable list security check 14
 - enCarryForward, global option 11
 - enCentSec, global option 12
 - enCFinterNetworkDeps, global option 12
 - enCFResourceQuantity, global option 12
 - encrypting application server profile properties 269
 - encryption ciphers
 - shortcuts
 - EXP 30
 - EXPORT40 30
 - EXPORT56 30
 - HIGH 30
 - LOW 30
 - MEDIUM 30
 - NULL 30
 - SSLv3 30
 - TLSv 30
 - encryption, strong, enabling, global option 15
 - enDbAudit, global option 12
 - enEmptySchedsAreSucc, global option 13
 - enEventDrivenWorkloadAutomation, global option 13
 - enEventProcessorHttpsProtocol, global option 13
 - engine
 - administrative tasks 233
 - enLegacyId, global option 13
 - enLegacyStartOfDayEvaluation, global option 13
 - enListSecChk, global option 14
 - enLogonBatch, global option 14
 - enPlanAudit, global option 14
 - enPreventStart, global option 14
 - enRetainNameOnRerunFrom, global option 14
 - enSSLFullConnection, global option 15
 - enStrEncrypt, global option 15
 - enSwfaultTol, global option 15
 - Enterprise Workload Manager
 - resource assignment 44
 - resource optimization 44
 - resource weight 44
 - retry on failure 44
 - enTimeZone, global option 15

- enWorkloadServiceAssurance, global option 15
- error messages
 - IP address validation 166
- es, global option 13
- event processor
 - managing 263
- event processor HTTPS protocol, global option 13
- event processor, enabling workstation to be, local option 24
- event rule management
 - log history maintenance, global option 16
 - mail sender name, global option 17
 - SMTP
 - port, global option 18
 - server name, global option 18
 - use authentication, global option 19
 - use SSL, global option 19
 - use TLS, global option 19
 - user name, global option 19
 - user password, global option 19
 - specify cleanup frequency, global option 16
 - TEC server
 - name, global option 20
 - port, global option 20
- event-driven workload automation enablement, global option 13
- event, ApplicationServerStatusChanged 269
- event, object type, defining access 119
- eventProcessorEIFPort, global option 15
- eventrule, object type, defining access 114
- evtsize command 158
- evtsize, command
 - to monitor queue sizes 154
- EXP, encryption shortcut 30
- EXPORT40, encryption shortcut 30
- EXPORT56, encryption shortcut 30
- exportserverdata 240
- extended agent
 - behind firewall 187
 - definition 144
 - jobs cannot launch 165
 - overview 162
- extRecPrompt, global option 16

F

- failure of
 - domain manager 234
 - master domain manager 236
- fault tolerant switch manager, enabling, global option 15
- fault-tolerant agent
 - backing up to, when used as backup master domain manager 202
 - data flows 154
 - definition 144
 - promoting to backup master domain manager 236
- fault-tolerant domain manager
 - See* domain manager
- fault-tolerant switch manager
 - See* domain manager
- fault-tolerant switch manager, impact on performance 287
- federated user registry
 - configuration 76
- file dependency checks impacting performance 284
- file sets
 - See* files
- file system
 - See* files

- file, object type, defining access 120
- files
 - .rhost 163
 - archived 207
 - avoiding full file systems 204
 - batchman minimum wait to check , local option 23
 - configuration
 - backing up 202
 - forecast plan logs 208
 - host.equiv 163
 - job output, archived 207
 - localopts 20
 - log files
 - backing up 201
 - maintaining file system 204
 - NetConf 160
 - Security
 - backing up 202
 - Symphony
 - archived 207
 - IP address validation 165
 - maximum number of records 162
 - overview 145
 - scanning by batchman 146
 - temporary
 - See* temporary files
 - trial plan logs 208
 - useropts 34
- filling mailboxes
 - monitoring 155
- filling message queues
 - monitoring 155
- final job stream, launch time, global option 19
- FIPS compliance 188
 - database configuration 196
 - EIF Listener port 196
 - embedded WebSphere Application Server 194
 - FIPS certificates 190
 - localopts parameters 194
- firewall bypass, starting 161
- firewall support 186
- flows, data, planning for 153
- forecast plan logs 208
- fta.msg message queue 154
- ftdown message queue, in backup domain manager 287
- ftup message queue, in backup domain manager 287
- full file systems, avoiding 204

G

- getaddrinfo() system call 165
- global option descriptions
 - approachingLateOffset 9
 - baseRecPrompt 10
 - carryforward 11
 - carryStates 10
 - companyName 10
 - deadlineOffset 11
 - deploymentFrequency 11
 - enCentSec 12
 - enCFinterNetworkDeps 12
 - enCFResourceQuantity 12
 - enDbAudit 12
 - enEmptySchedsAreSucc 13
 - enEventDrivenWorkloadAutomation 13
 - enEventProcessorHttpsProtocol 13
 - enLegacyId 13

global option descriptions (*continued*)

- enLegacyStartOfDayEvaluation 13
- enListSecChk 14
- enLogonBatch 14
- enPlanAudit 14
- enPreventStart 14
- enRetainNameOnRerunFrom 14
- enSSLFullConnection 15
- enStrEncrypt 15
- enSwfaultTol 15
- enTimeZone 15
- enWorkloadServiceAssurance 15
- eventProcessorEIFPort 15
- extRecPrompt 16
- ignoreCals 16
- logCleanupFrequency 16
- logHistory 16
- logmanMinMaxPolicy 16
- logmanSmoothPolicy 17
- longDurationThreshold 17
- mailSenderName 17
- maxLen 17
- minLen 18
- promotionOffset 18
- smtpServerName 18
- smtpServerPort 18
- smtpUserName 19
- smtpUserPassword 19
- smtpUseSSL 19
- smtpUseTLS 19
- startOfDay 19
- statsHistory 19
- TECServerName 20
- TECServerPort 20
- useAuthentication 19
- global options
 - optman command line 5
 - time zone feature 38
- global resource matching
 - configuring 42
- global settings
 - configuration 84
- glossary xii
- groups
 - creating 77

H

- heap size, application server, increase 289
- heartbeat signal
 - configuring 42
- HIGH, encryption shortcut 30
- history, job statistics, global option 19
- host name
 - application server, modify 273
 - database, change 251
 - impact of changing 167
- host, for extended agents, definition 144
- host, when connecting from command line client, local option 25
- host.equiv, file 163
- HTTPS protocol for event processor, global option 13

I

- IBMWAS61Service 267
- ic, global option 16
- ignoreCals, global option 16
- importserverdata 240
- incoming message cache
 - enable in mailman, local option 26
 - resize in mailman, local option 27
- indirect scheduling
 - J2EE jobs 48
- installation
 - checking
 - See* installation, verifying
 - directory 1
 - steps
 - See* steps, installation
- intercom
 - monitoring 155
- intercom.msg file, batchman maximum wait to read, local option 24
- Intercom.msg message queue 154
- internal Symphony table, determining the size of 162
- IP address
 - impact of changing 167
 - support for V6 165
 - validation 165
- is remote cli, local option 25
- ISMP
 - See* InstallShield wizard

J

- J2EE communication channel
 - configuring 48
- J2EE jobs
 - configuration for 48
 - configuring 48
 - direct scheduling 48
 - EJB 48
 - enabling 48
 - indirect scheduling 48
 - JMS 48
 - security settings 48
 - supported operations 48
 - WebSphere Application Server settings 48
- J2EE jobs on agent
 - configuration 48, 51
- J2EEJobExecutorConfig.properties
 - configuring 48
- 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 heap size, application server, increase 289
- Java Virtual Machine
 - See* Java Runtime Environment
- JDBC driver, resolving problems 256
- JDK
 - See* Java Runtime Environment
- jm job table size, local option 25
- jm look, local option 25
- jm nice, local option 25

- jm no root, local option 25
- jm promoted nice, local option 25
- jm promoted priority, local option 26
- jm read, local option 26
- JMS
 - J2EE jobs 48
- JnextPlan
 - when setting up a domain manager 237
- JnextPlan, avoiding lock list memory problems 213
- job management tasks wait in jobman, local option 25
- job output files, archived 207
- Job Scheduling Console
 - accessibility xiii
- job streams
 - empty. behavior, global option 13
 - more than 180 000, impacting DB2 log files 288
 - naming in mixed environments, global option 13
 - without at dependency, preventing from starting, global option 14
 - without jobs. behavior, global option 13
- job submissions, manual, impacting performance 284
- job table, size of in jobman, local option 25
- job times, minimum and maximum, logging and reporting, global option 16
- job, object type, defining access 114, 121
- JobDispatcherConfig.properties
 - job age in archive database 44
 - job age in database 44
- jobman
 - tuning 282
- jobman and JOBMAN
 - courier.msg file, wait to read, local option 26
 - job management tasks wait, local option 25
 - launching by batchman 146
 - nice value to apply to critical UNIX or Linux jobs, local option 25
 - nice value to apply to UNIX or Linux jobs, local option 25
 - priority value to apply to critical Windows jobs, local option 26
 - root jobs, enabling the launch of, local option 25
 - size of job table, local option 25
 - starting 147
- jobman process
 - monitoring 150
- jobmanrc 163
- jobs
 - failing to launch on extended agent 165
 - improving processing performance 285
 - late, when becoming, global option 9
 - long duration threshold, global option 17
 - more than 40 000, impacting Java heap size 288
 - promotion of critical, eligibility for, global option 18
 - retaining name on rerun, global option 14
 - statistics history, global option 19
- JRE
 - See Java Runtime Environment
- JVM
 - See Java Runtime Environment

L

- late jobs, when becoming, global option 9
- launch in context
 - Tivoli Dynamic Workload Console 67
- lb, global option 14
- lc, global option 16
- ld, global option 17
- LDAP
 - configuration 79
- LDAP authentication 135
- ldap user registry
 - configuration 73
- LDAP, configuring for TWS 135
- le, global option 13
- lh, global option 16
- li, global option 13
- Lightweight Directory Access Protocol
 - See LDAP
- link command, using 145
- link flag, auto 145
- link to non-responding workstation, wait to retry in mailman, local option 27
- linking
 - concept 145
- Linux
 - jobs, nice value to apply when critical, local option 25
 - jobs, nice value to apply, local option 25
- list permission
 - enable option 14
- lm, global option 16
- local option descriptions
 - appserver auto restart 23
 - appserver check interval 23
 - appserver count reset interval 23
 - appserver max restarts 23
 - appserver min restart time 23
 - appserver service name 23
 - autostart monman 23
 - bm check file 23
 - bm check status 24
 - bm check until 24
 - bm look 24
 - bm read 24
 - bm stats 24
 - bm verbose 24
 - can be event processor 24
 - caonly 29
 - clisslcipher 24
 - clisslserverauth 24
 - clisslservercertificate 24
 - clissltrusteddir 24
 - composer prompt 24
 - conman prompt 25
 - cpu 29
 - date format 25
 - defaultws 25
 - host 25
 - is remote cli 25
 - jm job table size 25
 - jm look 25
 - jm nice 25
 - jm no root 25
 - jm promoted nice 25
 - jm promoted priority 26
 - jm read 26
 - local was 26
 - merge stdlists 26
 - mm cache mailbox 26
 - mm cache size 27
 - mm resolve master 27
 - mm response 27
 - mm retry link 27
 - mm sound off 27
 - mm unlink 27

local option descriptions (*continued*)

- mozart directory 27
- nm mortal 27
- nm port 27
- nm read 28
- nm retry 28
- nm SSL port 28
- parameters directory 28
- port 28
- protocol 28
- proxy 28
- proxyport 28
- restricted stdlists 28
- SSL auth mode 29
- SSL auth string 29
- SSL CA certificate 29
- SSL certificate 29
- SSL certificate chain 30
- SSL encryption cipher 30
- SSL key 30
- SSL key pwd 30
- SSL random seed 30
- stdlist width 30
- string 29
- switch sym prompt 31
- sync level 31
- syslog local 30
- tcp connect timeout 31
- tcp timeout 31
- thiscpu 31
- timeout 31
- unison network directory 31
- useropts 31
- wr enable compression 31
- wr read 32
- wr unlink 32

local options

- file example 20
- file template 20
- setting 20
- setting sysloglocal 37
- syntax 20

local security 93

local user, resolving account in Windows 295

local was, local option 26

localopts

- nm ipvalidate 165
- option for setting synch level 286
- options for caching mailbox messages 285
- options used for tuning 282
- parameters for tuning mailman servers 160
- tuning for job-processing performance 285
- used for appservman 267

localopts file 20

lock list memory, DB2, monitoring 213

LOCKLIST, DB2 configuration parameter 213

log capacity, DB2, increase 289

log configuration

- Tivoli Workload Scheduler agent 60

log files

- backing up 203
- backup 201
- maintenance 207

logCleanupFrequency, global option 16

LOGFILSIZ, DB2 parameter 290

logging, impact on performance 282

logging.properties

- configuring 51

logHistory, global option 16

logmanMinMaxPolicy, global option 16

logmanSmoothPolicy, global option 17

logon as batch, granting automatically, global option 14

LOGPRIMARY, DB2 parameter 290

logs directory, as log file location 207

LOGSECOND, DB2 parameter 290

longDurationThreshold, global option 17

LookupAccountName, API used to resolve Windows user ID account 295

loss of

- domain manager 234
- master domain manager 236

LOW, encryption shortcut 30

lt, global option 17

LTPA keys

- sharing 79

LTPA token_keys

- using the same 80

M

mailbox

- monitoring 155

mailbox caching 285

mailbox files

- setting size 158

mailbox full

- monitoring 155

Mailbox.msg

- message queue 154

mailboxes

- monitoring 155

mailman

- \$MASTER variable, resolve, local option 27
- caching 285
- determining the size of its internal Symphony table 162
- incoming message cache
 - enable, local option 26
 - resize, local option 27
- link to non-responding workstation, wait to retry, local option 27
- processes timing out 145
- servers
 - configuring to maximize critical activities 151
 - tuning 160
- starting 147, 160
- starting with demgr parameter 161
- tellop command, respond to, local option 27
- tuning 282
- unlink non-responding workstation, wait to, local option 27
- workstation not responding, wait to report, local option 27

mailman process

- monitoring 150

mailSenderName, global option 17

maintenance

- database 201
- DB2, automatic
 - administer 210
 - modify policy 210
 - running manually 211
 - switch off 210
 - switch on 211

- maintenance (*continued*)
 - Oracle database 216
- makesec command 97
- manage_ltpa
 - script 81
- manual job submissions impacting performance 284
- manuals
 - See* publications
- master domain manager
 - backing up to backup master domain manager 202
 - backup
 - See* backup master domain manager
 - definition 144
 - extended loss of 238
 - failure 236
 - loss of 236
 - mitigating loss of 236
 - permanent change 238
 - running without 236
 - switching 238
 - without 236
- master domain, definition 144
- max restarts, appservman 267
- maximum and minimum job times, logging and reporting,
 - global option 16
- maximum number of results for global resource matching
 - configuring 42
- maximum records in Symphony file 162
- maxLen, global option 17
- MAXLOCKS, DB2 configuration parameter 213
- MEDIUM, encryption shortcut 30
- memory
 - management by logging processes impacting
 - performance 282
- merge stdlists, local option 26
- message caching 285
- message level 38
- message queues
 - dm.msg 154
 - fta.msg 154
 - ftdown, in backup domain manager 287
 - ftup, in backup domain manager 287
 - in backup domain manager 287
 - Intercom.msg 154
 - Mailbox.msg 154
 - monitoring 155
 - size 155
 - threshold 155
 - tomaster.msg 154
- methods directory, as log file location 208
- migration
 - of database (DB2 to Oracle and *vice versa*) 217
- min restart time, appservman 267
- minimum and maximum job times, logging and reporting,
 - global option 16
- minLen, global option 18
- mitigating loss of
 - domain manager 234
 - master domain manager 236
- mixed environments, naming of job streams in, global
 - option 13
- ml, global option 18
- mm cache mailbox, local option 26
- mm cache size, local option 27
- mm resolve master, local option 27
- mm response, local option 27
- mm retry link, local option 27

- mm sound off, local option 27
- mm unlink, local option 27
- Mm_unlink, localopts parameter 160
- monbox
 - monitoring 155
- moncmd
 - monitoring 155
- monman
 - autostart, local option 23
 - starting process 161
- monman process
 - monitoring 150
- mozart directory, local option 27
- ms, global option 17

N

- name, TEC server, global option 20
- naming, of job streams in mixed environments, global
 - option 13
- NetConf, file 160
- NetGetAnyDCName, API used to resolve Windows user ID
 - account 295
- netman
 - configuration file 160
 - connection failed, wait to retry, local option 28
 - IP address validation 165
 - port, local option 27
 - quit when child processes stopped, local option 27
 - SSL port, local option 28
 - starting 147
 - stop and start commands, wait to check for, local
 - option 28
 - support for Internet Protocol version 6 165
- netman process
 - monitoring 150
- NetUserGetInfo, API used to resolve Windows user ID
 - account 295
- network
 - capacity 151
 - changes, impact of 167
 - communications 145
 - impact of changes 167
 - IP address validation 165
 - linking 145
 - message queues, planning for 153
 - monitoring for unlinked workstations 145
 - operation 146
 - optimizing 151
 - overview 143
 - processes 147
 - structure, impact on critical agents 151
 - support for Internet Protocol version 6 165
 - traffic caused by backup domain manager impacting
 - performance 287
 - unlinking 145
- network traffic 281
- nice value to apply to critical UNIX or Linux jobs in jobman,
 - local option 25
- nice value to apply to UNIX or Linux jobs in jobman, local
 - option 25
- nm ipvalidate, localopts parameter 165
- nm mortal, local option 27
- nm port, local option 27
- nm read, local option 28
- nm retry, local option 28
- nm SSL port, local option 28

- nodename validation 165
- normal run time calculation, weighting factor, global option 17
- NULL, encryption shortcut 30

O

- object types, defining access to composer actions 114
- offline storage, used for backup and restore 201
- opens, file dependency 163
- options, global 5
- optman
 - security settings 5
- Oracle
 - administrative tasks 215
 - changing user password 241
 - database name, change 251
 - host name, change 251
 - maintaining database 216
 - migrating to DB2 217
 - obtaining database information 216
 - passwords not used by TWS, changing 215
 - port, change 251
 - reorganization 203
 - running maintenance manually 215
 - tools, locating 215
 - tuning 283
 - user permissions for running the tools 216
- Oracle database
 - configuring reports 90

P

- pa, global option 14
- parallel database migration (DB2 to Oracle and *vice versa*) 217
- parameter, object type, defining access 124
- parameters directory, local option 28
- parent domain, definition 144
- password for SMTP connection, global option 19
- passwords
 - other DB2, changing 209
 - TWS_user, changing 241
- performance
 - fault-tolerant switch manager 287
 - file dependency checks, too many 284
 - impacted by multiple TDWC production plan reports 292
 - job submissions, manual, too many 284
 - job-processing, improving 285
 - network 151
 - too many file dependency checks 284
 - too many manual job submissions 284
 - tuning job processing on a workstation 282
 - tuning on UNIX 282
 - workload spreading 284
- permissions, user
 - for running DB2 tools 210
 - for running Oracle tools 216
- plan auditing, enabling, global option 14
- plan, preproduction, maximum length, global option 17
- plan, preproduction, minimum length, global option 18
- planned maintenance
 - See maintenance
- planning to minimize network message queues 153
- Pluggable Authentication Module, using in TWS 141
- po, global option 18

- pobox
 - monitoring 155
- port
 - SMTP server, global option 18
 - SSL, used by netman, local option 28
 - TEC server, global option 20
- port for command line client, local option 28
- port number, event processor, global option 15
- port, database, change 251
- port, for netman, local option 27
- preproduction plan, maximum length, global option 17
- preproduction plan, minimum length, global option 18
- preventing job streams without at dependency from starting, global option 14
- priority value to apply to critical Windows jobs in jobman, local option 26
- private keys 170
- problems
 - See troubleshooting
- process messages 37
- process prompts 37
- processes status
 - monitoring 150
- production control file, batchman minimum wait to update, local option 24
- production plan reports, TDWC, affecting performance 292
- production, managing on extended agents 165
- profile properties, application server, encrypting 269
- promoting an agent to backup master domain manager 236
- promoting to backup master domain manager 236
- promotion of critical jobs, eligibility for, global option 18
- promotionOffset, global option 18
- prompt, object type, defining access 114, 125
- prompts, additional, global option 16
- properties
 - of application server, utilities for changing 277
- protocol, local option 28
- proxy, local option 28
- proxyport, local option 28
- ps, global option 14
- publications xii

Q

- queues, message
 - See message queues
- quit netman when child processes stopped, local option 27

R

- r3batch extended agent, interaction process start 161
- RACF, configuring LDAP for 139
- reconfiguration of database (DB2 to Oracle and *vice versa*) 217
- registry, user, custom, modify security settings 263
- remote access for command-line, configuring 36
- remote job, check status of, process start 161
- remove
 - See uninstallation
- reorganize DB2 database 212
- reorganizing database 203
- report, object type, defining access 126
- reports configuring
 - DB2 database 89, 91
 - Oracle database 90
- reports, configuring Tivoli Dynamic Workload Console to view 89

- required maintenance
 - See maintenance
- rerun jobs, retaining name, global option 14
- resource advisor agent 239
- resource quantities carried forward, global option 12
- resource, object type, defining access 114, 127
- restart, automatic, of application server 266
- restore
 - application server configuration 272
- restore of application server configuration 272
- restore, from offline storage 201
- restoreConfig, used to restore application server configuration 272
- restricted stdlists, local option 28
- RHEL
 - See Red Hat Enterprise Linux
- rights, user
 - for running DB2 tools 210
 - for running Oracle tools 216
- rmstdlist command
 - used for archiving log files 207
- roles
 - for Tivoli Dynamic Workload Broker 79
 - for Tivoli Workload Scheduler 77
- root jobs, enabling the launch of in jobman, local option 25
- root user, changing password 241
- rq, global option 12
- rr, global option 14
- run time (average) calculation, weighting factor, global option 17
- running without
 - domain manager 234
 - master domain manager 236

S

- sc, global option 14
- scalability 288
- schedlog directory, as log file location 207
- schedule, object type, defining access 114, 128
- schedules
 - See job streams
- scheduling events, communications 145
- script
 - manage_ltpa 81
 - webui 92
- scripts
 - See commands and scripts
- sd, global option 19
- se, global option 15
- security
 - centralized 98
 - information, verifying in Windows 295
 - local 93
 - network, for backup domain manager 235
 - overview 93
 - specifying accesses 112
 - specifying object types 107
 - specifying objects 108
 - specifying user attributes 102
 - template file 99
- security check when listing, global option 14
- security file
 - enabling centralized security, global option 12
- security file, template 99
- security level
 - enabled 173
- security level (*continued*)
 - force 173
 - on 173
- security settings, application server, modify 258
- Security, file
 - backing up 202
- sender name, mail, event rule management, global option 17
- server
 - monitoring 155
- server configuration
 - ResourceAdvisorConfig.properties file 42
- servers
 - mailman
 - configuring to maximize critical activities 151
 - tuning 160
- service
 - Windows
 - for the application server, updating 270
- service name, appservman 267
- services
 - IBMWAS61Service 267
- services (Windows)
 - changing password 241
 - for application server, update 270
 - stopping 250
 - Tivoli Workload Scheduler, configuring in NetConf file 160
- setting the local options 20
- setting the user options 34
- sf, global option 15
- sh, global option 19
- showDataSourceProperties, application server utility 252
- showHostProperties, application server utility 274
- showSecurityProperties, application server utility 259
- single sign-on
 - configuration 79
 - LTPA token_keys 80
- sizing the internal Symphony table 162
- SMTP
 - authentication on connection, use, global option 19
 - port, global option 18
 - server name, global option 18
 - SSL on connection, use, global option 19
 - TLS on connection, use, global option 19
 - user name for connection, global option 19
 - user password for connection, global option 19
- smtpServerName, global option 18
- smtpServerPort, global option 18
- smtpUseAuthentication, global option 19
- smtpUserName, global option 19
- smtpUserPassword, global option 19
- smtpUseSSL, global option 19
- smtpUseTLS, global option 19
- sn, global option 18
- soap.client.props
 - configuring 51
- sp, global option 18
- space, disk
 - See disk space
- SSL
 - authentication string when using command-line client, local option 24
 - certificate file for communications with command-line client, local option 24
 - enabling command-line client to use, local option 24
 - full connection enablement, global option 15
 - port, used by netman, local option 28

- SSL (*continued*)
 - trusted certificate directory for communications with command-line client, local option 24
- SSL attributes
 - configuring 173
- SSL auth mode, local option 29
- SSL auth string, local option 29
- SSL CA certificate, local option 29
- SSL certificate chain, local option 30
- SSL certificate, local option 29
- SSL communication
 - enabled 173
 - force 173
 - on 173
- SSL encryption cipher, local option 30
- SSL key pwd, local option 30
- SSL key, local option 30
- SSL on SMTP connection, use, global option 19
- SSL random seed, local option 30
- SSL security
 - HTTP/HTTPS 169
 - keystore passwords 179
 - overview 169
 - RMI/IIOP 169
 - topology connection 169
- SSL support
 - configuring 174
- SSLv3, encryption shortcut 30
- standard agent, definition 144
- start dynamic workload broker 239, 264
- start-of-plan-period
 - initialization, communications 145
- startappserver 266, 268
 - run from conman, process 161
- startevtptroc, run from conman, process 161
- starting the application server 264
- startOfDay, global option 19
- startOfDay, how evaluated in time zones, global option 13
- StartUp
 - used for starting netman 147
- StartUp command 149
- startWas command 264
- statistics reporting by batchman, enable, local option 24
- statsHistory, global option 19
- status
 - application server 269
- status messages, batchman send to standard list, local option 24
- stdlist directory
 - information about extended agent jobs 163
 - maintaining 207
- stdlist width, local option 30
- stdlist, merge console messages into, local option 26
- stop and start commands, wait to check for in netman, local option 28
- stop dynamic workload broker 239, 264
- stopappserver 266, 268
 - configure user credentials 268
 - run from conman, process 161
- stopevtptroc, run from conman on a client, process 161
- stopevtptroc, run from conman, process 161
- stopmon, run from conman, process 161
- stopping
 - services 250
- stopping the application server 264
- stopping workstations hierarchically, starting process 161
- stopWas command 264

- streamlogon, user 295
- string, SSL auth mode, local option 29
- strong encryption, enabling, global option 15
- structure of network, impact on critical agents 151
- submitting too many jobs manually, impact on performance 284
- Sun
 - See Solaris
- sw, global option 15
- switch dynamic workload broker instances 239
- switch manager, fault tolerant, enabling, global option 15
- switch manager, fault-tolerant
 - See backup domain manager
- switch sym prompt, local option 31
- switchevtptroc, run from conman, process 161
- switching a domain manager, short-term 235
- switching a master domain manager
 - long-term 238
 - short-term 238
- switchmgr
 - starting normal process 161
 - starting process so that links are not started until event received 161
- Symphony file
 - archived 207
 - enabling the copying of calendars into, global option 16
 - IP address validation 165
 - maximum number of records 162
 - monitoring space used 204
 - overview 145
 - scanning by batchman 146
 - support for Internet Protocol version 6 165
- Symphony table, internal, determining the size of 162
- sync level, local option 31
- synch level option, setting 286
- syslog 37
- syslog local, local option 30
- sysloglocal options
 - LOG_ERR 37
 - LOG_INFO 37
 - LOG_NOTICE 37
 - LOG_WARNING 37

T

- tcp connect timeout, local option 31
- tcp timeout, local option 31
- TCP/IP ports, application server, modify 273
- technical training
 - See Tivoli technical training
- TECServerName, global option 20
- TECServerPort, global option 20
- tellop command, respond to in mailman, local option 27
- temporary files 209
- text files, used for backup and restore
 - See files
- th, global option 20
- thiscpu, local option 31
- threshold, long job duration, global option 17
- time interval for job allocation to resources
 - configuring 42
- time interval for notifications on resources
 - configuring 42
- time interval for retrying failed operations
 - configuring 44
- time zone feature, enabling, global option 15

- time zones
 - evaluating startOfDay, global option 13
- timeout, local option 31
- Tivoli Dynamic Workload Broker
 - roles 79
- Tivoli Dynamic Workload Console
 - accessibility xiii
 - configuration 67, 73
 - configure to view reports 89
 - launch in context 67
 - multiple production plan reports, affecting performance 292
- Tivoli technical training xiii
- Tivoli Workload Scheduler
 - preventing access 92
 - roles 77
 - security file 79
- Tivoli Workload Scheduler agent
 - log configuration 60
 - trace configuration 60
- Tivoli Workload Scheduler agent traces
 - modifying 61
 - viewing settings 61
- tl, global option 19
- TLS on SMTP connection, use, global option 19
- TLSv, encryption shortcut 30
- tmp directory, as location for temporary files 209
- Tokenstrv
 - See Tivoli Token Service
- tomaster
 - monitoring 155
- tomaster.msg message queue
 - data flow 154
 - in backup domain manager 287
- tp, global option 20
- trace configuration
 - Tivoli Workload Scheduler agent 60
- traces directory, as trace file location 207
- tracing 281
- traffic caused by backup domain manager impacting performance 287
- training
 - See also Tivoli technical training
 - technical xiii
- tree structure, impact on critical agents 151
- trial plan logs 208
- troubleshooting
 - data flows 153
 - message queues 153
- ts, global option 12
- tuning
 - database 283
 - job processing on a workstation 282
 - localopts file, for job-processing performance 285
 - mailman servers 160
 - the application server 284
 - UNIX operating systems 282
- TWA_home 1
- TWS disk space
 - monitoring 205
- TWS processes status
 - monitoring 150
- TWS_user
 - changing password 241
 - owning processes 147
 - required security access for workload service assurance 131

- TWSObjectsMonitor events,
 - ApplicationServerStatusChanged 269
- tz, global option 15

U

- ua, global option 19
- un, global option 19
- unison network directory, local option 31
- UNIX
 - changing passwords on 241
 - configuration for IP address validation 166
 - extended agents 163
 - jobs, nice value to apply when critical, local option 25
 - jobs, nice value to apply, local option 25
 - temporary directory, on UNIX, access rights 296
 - tuning 282
 - updating SOAP properties after changing application server user or password 271
- unixloc, access method 163
- unixrsh, access method 163
- unixssh, access method 163
- unlink non-responding workstation, wait to in mailman, local option 27
- unlink, command
 - usage 145
- unlinking
 - concept 145
 - workstations 250
- until time, batchman maximum wait to report expiry of, local option 24
- up, global option 19
- updateWas, using to update the SOAP properties after changing application server user or password 271
- updateWasService
 - using to update the application server Windows service 270
- us, global option 19
- used disk space
 - monitoring 205
- user
 - configuration 77
 - portfolio 77
- user name for SMTP connection, global option 19
- user options
 - setting 34
 - syntax 34
- user password for SMTP connection, global option 19
- user permissions
 - for running DB2 tools 210
 - for running Oracle tools 216
- user registry, custom, modify security settings 263
- user security
 - commands
 - dumpsec 96
 - makesec 97
 - local security 93
 - security file
 - access capabilities 112
 - modifying 94
 - sample 131
 - syntax 99
 - user qualification 104
 - variables 111
 - wildcards 100
 - security files 94
 - setting 93

- userobj, object type, defining access 114, 129
- useropts file 34
- useropts, local option 31
- users
 - console 77
 - creating 77
 - domain, resolving account in Windows 295
 - local, resolving account in Windows 295
 - streamlogon 295
- utilities
 - application server 278
 - changeDataSourceProperties 252
 - changeHostProperties 274
 - changeSecurityProperties 259
 - changeTraceProperties 276
 - defining access for working with objects 114
 - showDataSourceProperties 252
 - showHostProperties 274
 - showSecurityProperties 259
- utilities that change application server properties, using 277
- utility commands
 - setting mailbox file size 158
 - starting up netman 149

V

- validating IP address 165
- variables
 - \$MASTER, resolve, local option 27
- variable, object type, defining access 130
- volumes, data, impact on network 151

W

- wa, global option 15
- warning messages
 - IP address validation 166
- Web User Interface
 - See* Tivoli Dynamic Workload Console
- WebSphere Application Server
 - See also* application server
 - configuring 48
- webui
 - script 92
- weighting factor for calculating average run time, global option 17
- Windows
 - changing passwords on 241
 - jobs, nice value to apply when critical, local option 26
 - resolving user ID account 295
 - service
 - for the application server, updating 270
- Windows OS
 - special characters, handling 37
- without
 - domain manager 234
 - master domain manager 236
- workload
 - spreading to improve performance 284
- workload automation, event-driven, enablement, global option 13
- workload broker users and roles
 - mapping security roles in Websphere Application Server 56
 - modifying 56

- workload broker workstation configuration
 - modifying 55
- workload service assurance
 - approaching late offset, global option 9
 - deadline offset, global option 11
 - jobs eligible for promotion, global option 18
 - nice value to apply to critical UNIX or Linux jobs in jobman, local option 25
 - priority value to apply to critical Windows jobs in jobman, local option 26
 - required security access for TWS_user 131
- workload service assurance, enabling, global option 15
- workstation
 - application server status 269
 - Tuning job processing on 282
- workstation not responding, wait to report in mailman, local option 27
- workstations
 - default when using the command line client, local option 25
 - enabling to be event processor, local option 24
 - unlinking 250
- wr enable compression, local option 31
- wr read, local option 32
- wr unlink, local option 32
- Wr_unlink, localopts parameter 160
- writer
 - starting 147
 - starting, for incoming mailman messages 160
 - stopping, for incoming mailman messages 160
- wsadmin utility 251

X

- xl, global option 17
- xp, global option 16



Program Number: 5698-WSH

Printed in USA

SC23-9113-02



Spine information:

IBM Tivoli Workload Scheduler **Version 8.5.1 (Revised October 2010)**

Administration Guide

