IBM Workload Scheduler for z/OS

Scheduling End-to-end with z-centric Capabilities

Version 9.3 (Revised July 2018)

IBM Workload Scheduler for z/OS

Scheduling End-to-end with z-centric Capabilities

Version 9.3 (Revised July 2018)

Note
Before using this information and the product it supports, read the information in "Notices" on page 163.

This edition applies to version 9, release 3, modification level 0 of IBM Workload Scheduler for z/OS (program number 5698-T08) and to all subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this publication

Scheduling End-to-end with z-centric Capabilities describes how to set up IBM Workload Scheduler for z/OS to generate a lightweight end-to-end scheduling environment in which to schedule and control workload from the mainframe to distributed systems. While it exhaustively describes the required installation and customization tasks, this first edition provides only miscellaneous notes about the z-centric end-to-end scheduling solution. This publication is therefore to be used in conjunction with the rest of the IBM Workload Automation library. In particular, refer to the following publications:

- *Customization and Tuning* for information about customizing and tuning IBM Workload Scheduler for z/OS.
- *Managing the Workload* for details about organizing your workload and information about managing the workload when it becomes part of a plan and is run.
- *Messages and Codes* for message information.
- *Planning and Installation* for details about how to install the IBM Workload Scheduler for z/OS Agent.
- *Memo to Users* for interoperability information among component versions.

What is new in this release

Learn what is new in this release.

For information about the new or changed functions in this release, see *Overview*, section *Summary of enhancements*.

For information about the APARs that this release addresses, see the Program Directory and the Dynamic Workload Console Release Notes at http://www-01.ibm.com/support/docview.wss?rs=672&uid=swg27045183.

Who should read this publication

This publication is intended for IBM Workload Scheduler for z/OS users who are already familiar with IBM Workload Scheduler. A background knowledge of IBM Workload Scheduler for z/OS is required.

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, see the Accessibility Appendix in the *IBM Workload Scheduler User's Guide and Reference*.

Technical training

Cloud & Smarter Infrastructure provides technical training.

For Cloud & Smarter Infrastructure technical training information, see: http://www.ibm.com/software/tivoli/education

Support information

IBM provides several ways for you to obtain support when you encounter a problem.

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:

- Searching knowledge bases: You can search across a large collection of known problems and workarounds, Technotes, and other information.
- Obtaining fixes: You can locate the latest fixes that are already available for your product.
- Contacting IBM Software Support: If you still cannot solve your problem, and you need to work with someone from IBM, you can use a variety of ways to contact IBM Software Support.

For more information about these three ways of resolving problems, see the appendix about support information in *IBM Workload Scheduler: Troubleshooting Guide*.

Conventions used in this publication

Conventions used in this publication.

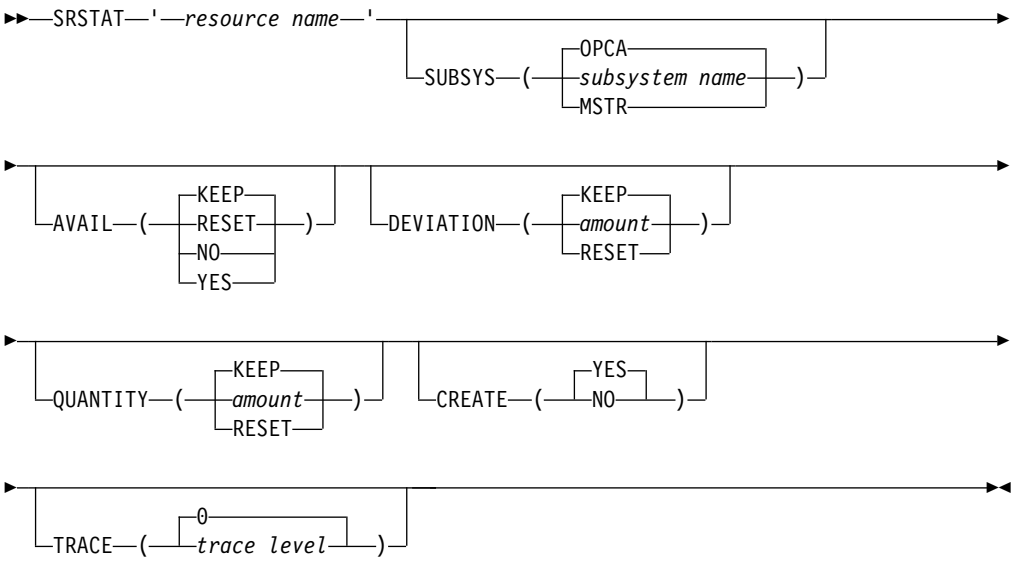
The publication uses several typeface conventions for special terms and actions. Technical changes to the text are indicated by a vertical line to the left of the change. These conventions have the following meanings:

Information type	Style convention	Example
Commands	All capital letters	CREATE
References in the text to fields on panels	All capital letters	QUANTITY
File and directory names, input you should type in panel fields	Monospace	`MYAPPLICATION`
First time new term introduced, publication titles	Italics	*Application*

How to read syntax diagrams

Syntax diagrams help to show syntax in a graphical way.

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



The symbols have these meanings:

The statement begins here.

The statement is continued on the next line.

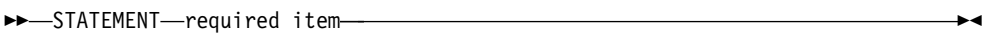
The statement is continued from a previous line.

The statement ends here.

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

These are the conventions used in the diagrams:

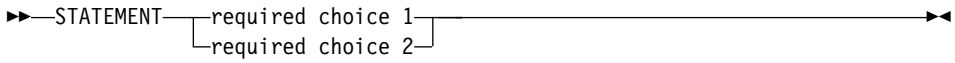
- Required items appear on the horizontal line (main path):



- Optional items appear below the main path:

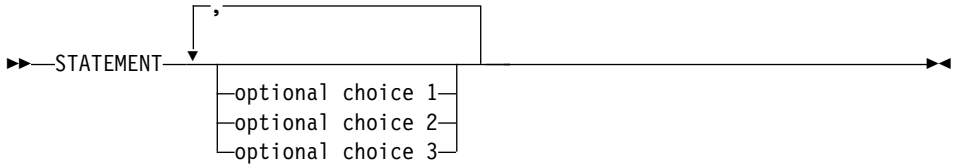
- An arrow returning to the left above the item indicates an item that you can repeat. If a separator is required between items, it is shown on the repeat arrow.

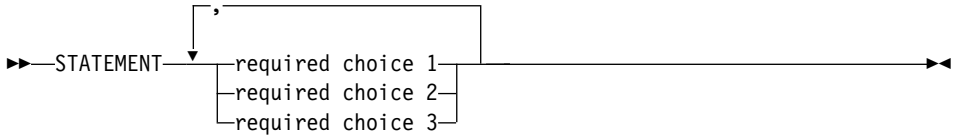
- If you can choose from two or more items, they appear vertically in a stack.
  - If you must choose one of the items, one item of the stack appears on the main path:



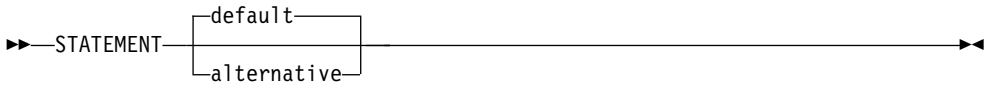
– If choosing one of the items is optional, the entire stack appears below the main path:

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

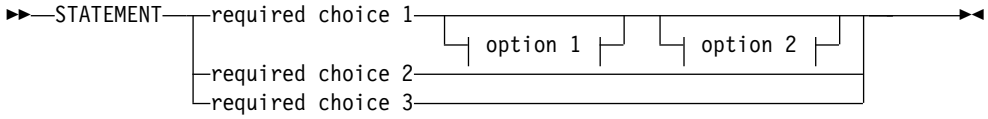




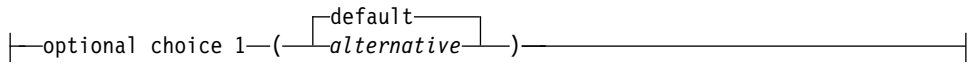
- Parameters that are above the main line are default parameters:



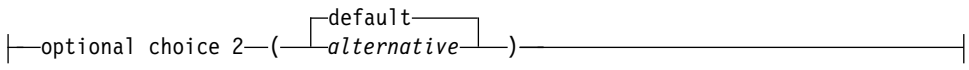
- Keywords appear in uppercase (for example, STATEMENT).
- Parentheses and commas must be entered as part of the command syntax, as shown.
- For complex commands, the item attributes might not fit on one horizontal line. If that line cannot be split, the attributes appear at the bottom of the syntax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



option 1



option 2



Chapter 1. Configuring

This chapter provides a general description of the end-to-end with z-centric capabilities environment and describes the configuration steps required to make it work.

Overview

The different types of workload scheduling you can implement in your z-centric end-to-end environment:

Static scheduling

The IBM Workload Scheduler for z/OS Agents are installed on the distributed systems and are connected to the z/OS® system through the IBM Workload Scheduler for z/OS controller.

Static scheduling including job types with advanced options

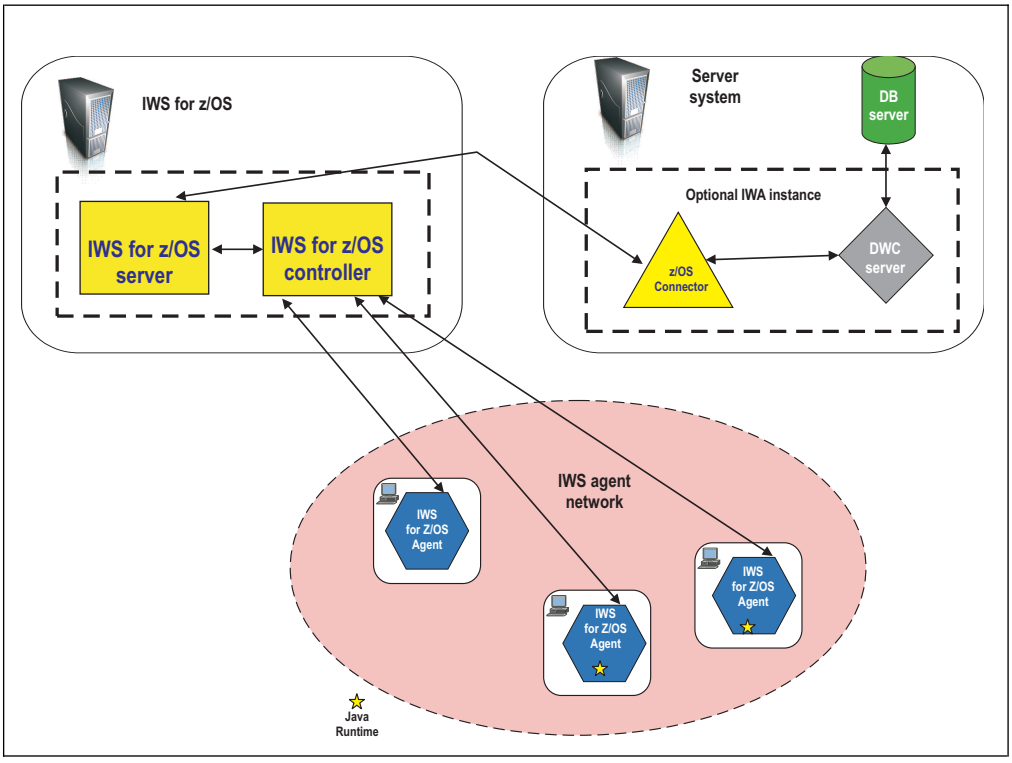


Figure 1. Static end-to-end environment with Java run time installed in which you can run job types with advanced options

The IBM Workload Scheduler for z/OS Agents are installed on the distributed systems with the Java™ run time, and are connected to the z/OS system through the IBM Workload Scheduler for z/OS controller. Figure 1 shows a **static** end-to-end environment in which IBM Workload Scheduler for z/OS Agents:

- Without the Java run time installed, can run existing job types
- With the Java run time installed, can run both existing jobs and job types with advanced options

Dynamic scheduling

The IBM Workload Scheduler for z/OS Agents are installed on the distributed systems with dynamic capabilities, and are connected to the dynamic domain manager. The dynamic domain manager is connected to the z/OS system through the IBM Workload Scheduler for z/OS controller.

Dynamic scheduling including job types with advanced options

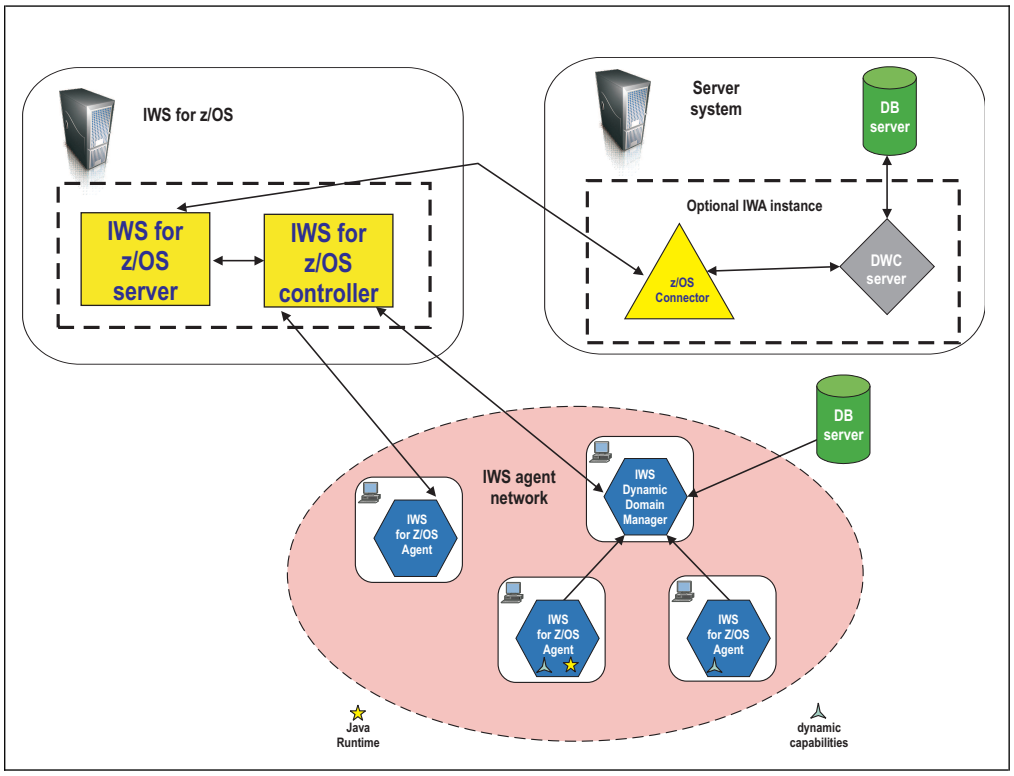


Figure 2. Dynamic end-to-end environment to run also job types with advanced options.

The IBM Workload Scheduler for z/OS Agents are installed on the distributed systems with dynamic capabilities and the Java run time, and are connected to the dynamic domain manager. The dynamic domain manager is connected to the z/OS system through the IBM Workload Scheduler for z/OS controller. Figure 2 shows a **dynamic** end-to-end environment in which IBM Workload Scheduler for z/OS Agents:

- Without the Java run time installed, can run existing job types
- With the Java run time installed, can run both existing jobs and job types with advanced options

By using the z-centric end-to-end scheduling environment, you schedule and control both static and dynamic workload from the mainframe to distributed systems with a low cost of ownership. IBM Workload Scheduler for z/OS acts as a single point of control, providing you with all the mainframe capabilities to manage distributed workload.

IBM Workload Scheduler for z/OS Agents can be installed on the UNIX, IBM® i, or Windows workstations where you run jobs scheduled from IBM Workload Scheduler for z/OS. After installation, you associate each agent to an IBM Workload Scheduler for z/OS workstation definition, so that it can be used as a target for IBM Workload Scheduler for z/OS jobs. The communication between the IBM Workload Scheduler for z/OS Agent and IBM Workload Scheduler for z/OS

controller is direct, through the HTTP or HTTPS protocol. If the connection is interrupted, the HTTP client retries to establish it until it is successful.

Figure 3 shows the process flow to submit and track jobs on IBM Workload Scheduler for z/OS Agent workstations.

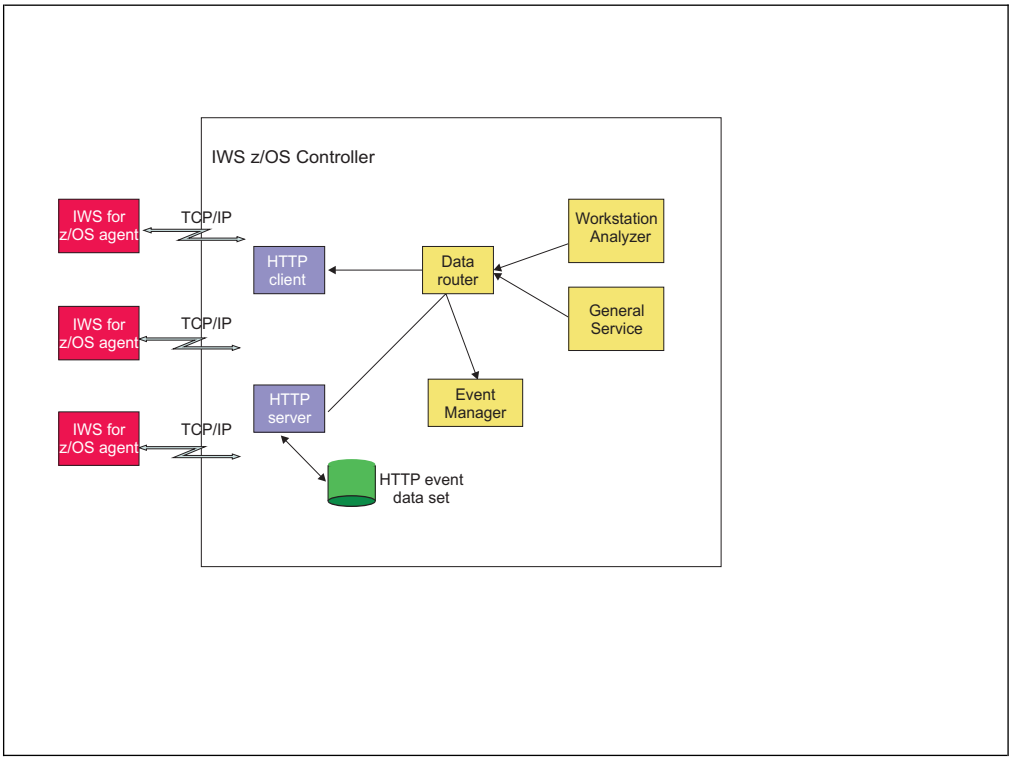


Figure 3. Process flow

Table 1 shows the steps required to set up your end-to-end with z-centric capabilities environment.

Table 1. Installation and configuration steps for setting up an end-to-end with z-centric capabilities environment

To...	**Refer to...**
Plan your end-to-end environment with z-centric capabilities and install the IBM Workload Scheduler for z/OS Agent.	*IBM Workload Scheduler for z/OS: Planning and Installation*
Install the dynamic domain manager and agent.	*IBM Workload Scheduler: Planning and Installation*
Configure the IBM Workload Scheduler for z/OS Agent properties.	"Configuring the IBM Workload Scheduler for z/OS Agent settings" on page 4
Define the IBM Workload Scheduler for z/OS Agent and dynamic domain manager destinations.	"Defining destinations with the ROUTOPTS statement" on page 22
Define the IBM Workload Scheduler for z/OS Agent workstations.	Chapter 5, "Defining and managing IBM Workload Scheduler for z/OS Agent workstations in the database," on page 39
Define dynamic workstations.	Chapter 6, "Defining and managing dynamic workstations," on page 47

Table 1. Installation and configuration steps for setting up an end-to-end with z-centric capabilities environment (continued)

To...	Refer to...
Configure the SSL connection protocol, if you did not change the default configuration of the IBM Workload Scheduler for z/OS Agent.	Chapter 7, "Setting the security features," on page 53
Customize the connection parameters.	"Customizing the HTTP connection with the HTTPOPTS statement" on page 25
Note: If you want to move from an end-to-end with fault tolerance capabilities environment to an end-to-end with z-centric one, refer to "Listing" on page 41.	

Configuring the IBM Workload Scheduler for z/OS Agent settings

The configuration settings of the IBM Workload Scheduler for z/OS 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

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.

Advantages of job types with advanced options

While the existing IBM Workload Scheduler for z/OS job is a generic script or command, you can define and schedule job types with advanced options to perform specific tasks, such as database, file transfer, Java, and web services operations. You can easily define these job types without having specific skills on the applications where the job runs.

While the existing IBM Workload Scheduler for z/OS job is a generic script or command, you can define and schedule job types with advanced options to perform specific tasks, such as database, file transfer, Java, and web services operations. You can easily define these job types without having specific skills on the applications where the job runs.

The job types with advanced options include both those supplied with the product and the additional types implemented through the custom plug-ins.

The following job types with advanced options are available:

File transfer jobs
: Transfer files to and from a server reachable using FTP, SSH, or other protocols.

Web services jobs
Call a web service.

Database jobs
Perform queries, SQL statements, and jobs on a number of databases, including custom databases. You can also create and run stored procedures on DB2, Oracle, and MSSQL databases.

Executable jobs
Run a script or command with advanced options, such as redirecting standard input and standard output to a file.

Java jobs
Run a Java class

MSSQL jobs
Run a Microsoft SQL job.

Access Method jobs
Extend job scheduling functions of IBM Workload Scheduler to other systems and applications using access methods. The access methods communicate with the external system to launch the job and return the status of the job. The following access methods are available:

- Oracle E-Business Suite
- PeopleSoft
- SAP
- MVS
- Custom methods

J2EE jobs
Allow Java applications in the same network to send and receive messages from and to a JMS destination.

IBM i jobs
Run a command on IBM i systems.

Provisioning
Jobs that span physical computers, virtual machines, and private and public cloud environments creating an on-demand environment. This job type integrates with IBM SmartCloud Provisioning.

Remote command
Jobs that can run on remote computers that do not require an agent installation.

You define job types with advanced options by connecting to a z/OS engine with the Dynamic Workload Console. From the Dynamic Workload Console, you open the Workload Designer and select the job type you want to create. When the job definition is saved, it is stored in the JCL library and is available for scheduling from IBM Workload Scheduler for z/OS

The job types with advanced options run on IBM Workload Scheduler for z/OS Agents in both the static configuration (IBM Workload Scheduler for z/OS Agent connected directly to the z/OS controller) and in the dynamic configuration (IBM Workload Scheduler for z/OS Agent connected to the dynamic domain manager).

In addition to configuring job types with advanced options using the Dynamic Workload Console, you can use the related configuration files. For more information, see Appendix C, "Configuring to schedule job types with advanced options," on page 147.

For more information about the procedure for defining job types with advanced options, see the section about creating job types with advanced options in *IBM Dynamic Workload Console User's Guide*. For more information about each job type, see the Dynamic Workload Console online help. For information about how to create jobs using the **JOBREC** statement, see "JOBREC" on page 65.

In addition, you can create custom plug-ins to implement your own job types with advanced options for applications not supported by IBM Workload Scheduler. For more information about how to create custom plug-ins, see *Extending IBM Workload Automation*, *SC14-7623*.

The agents with dynamic capabilities can run the jobs you created for the existing IBM Workload Scheduler workstation types. To run these jobs on the dynamic agents, you only have to change the specification of the workstation where you want the job to run. The major advantage is that you can use the workflows you previously created without additional effort.

Configuring the agent

The configuration settings of the agent are stored in the `JobManager.ini` file. To find out where this file is located, see Where products and components are installed.

In a distributed environment, if a gateway is configured to allow the master domain manager or dynamic domain manager to communicate with a dynamic agent located behind a network boundary, then the gateway configuration settings of the agent are contained in the `JobManagerGW.ini` file. This file is almost identical to the `JobManager.ini` file, however, only parameters in the [ITA], [Env], and [ResourceAdvisorAgent] sections require configuration. For these parameters, definitions are given for both the `JobManager.ini` and `JobManagerGW.ini` files.

These files are 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:

- Event-driven workload automation properties
- Log properties
- Trace properties when the agent is stopped. You can also customize traces when the agent is running using the procedure described in "Configuring trace properties when the agent is running" on page 11.
- Native job executor
- Java job executor
- Resource advisor agent
- System scanner

The log messages are written in the following file:

On Windows operating systems:

<*TWA_home*>\TWS\stdlist\JM\JobManager_message.log

On UNIX and Linux operating systems:

<*TWA_home*>/TWS/stdlist/JM/JobManager_message.log

The trace messages are written in the following file:

On Windows operating systems:

- <*TWA_home*>\TWS\stdlist\JM\ITA_trace.log
- <*TWA_home*>\TWS\stdlist\JM\JobManager_trace.log
- <*TWA_home*>\TWS\JavaExt\logs\javaExecutor0.log

On UNIX and Linux operating systems:

- <*TWA_home*>/TWS/stdlist/JM/ITA_trace.log
- <*TWA_home*>/TWS/stdlist/JM/JobManager_trace.log
- <*TWA_home*>/TWS/JavaExt/logs/javaExecutor0.log

Logging information about job types with advanced options

You can use the logging.properties file to configure the logging process for job types with advanced options, with the exception of the Executable and Access Method job types.

The logging.properties file is located on the IBM Workload Scheduler for z/OS Agent, under *TWA_home*/TWS/JavaExt/cfg/logging.properties.

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 Defines the logging level for the internal logger.

com.ibm.scheduling
Defines the logging level for the job types with advanced options. To log information about job types with advanced options, set this keyword to ALL.

- The path where the logs are written, specified by the following keyword:
  `java.util.logging.FileHandler.pattern`

Not all the properties in the `JobManager.ini` and `JobManagerGW.ini` files can be customized. For a list of the configurable properties, see the following sections:

- "Configuring log message properties [JobManager.Logging.cclog]" on page 9.
- "Configuring trace properties when the agent is stopped [JobManager.Logging.cclog]" on page 10.
- "Configuring common launchers properties [Launchers]" on page 14.
- "Configuring properties of the native job launcher [NativeJobLauncher]" on page 15.
- "Configuring properties of the Java job launcher [JavaJobLauncher]" on page 18.
- "Configuring properties of the Resource advisor agent [ResourceAdvisorAgent]" on page 18.
- "Configuring properties of the System scanner [SystemScanner]" on page 20
- See the section about configuring properties of event-driven workload automation [EventDrivenWorkload] in *Scheduling End-to-end with z-centric Capabilities*.

Configuring proxy properties [ITA_Env]

About this task

To have the IBM Workload Scheduler for z/OS Agent communicate with the z/OS controller through a proxy, edit the `[ITA_Env]` section in the `JobManager.ini` file as follows.

Note: On the controller, ensure that you have set the PROXY keyword in the ROUTOPTS statement.

Depending on the type of connection you are using, add or modify the following property:

SSL-secured connection

https_proxy
The URL of the proxy through which the agent communicates to the controller. The value is

`https_proxy =http://<proxy_wks>:<proxy_wks_port>`

where:

- *<proxy_wks>* is the fully qualified host name of the workstation where the proxy is configured.
- *<proxy_wks_port>* is the port number of the workstation where the proxy is configured.

Non SSL-secured connection

http_proxy
The URL of the proxy through which the agent communicates to the controller. The value is

| `http_proxy =http://<proxy_wks>:<proxy_wks_port>`

| where:

| • *<proxy_wks>* is the fully qualified host name of the workstation where the proxy is configured.

| • *<proxy_wks_port>* is the port number of the workstation where the proxy is configured.

| Restart the agent after the property change.

Configuring log message properties [JobManager.Logging.cclog]

About this task

To configure the logs, edit the [JobManager.Logging.cclog] section in the `JobManager.ini` file. This procedure requires that you stop and restart the IBM Workload Scheduler agent

The section containing the log properties is named:
`[JobManager.Logging.cclog]`

You can change the following properties:

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.loggerhd.fileEncoding
By default, log files for the agent are coded in UTF-8 format. If you want to produce the log in a different format, add this property and specify the required codepage.

JobManager.loggerfl.level
The amount of information to be provided in the logs. The value ranges from 3000 to 7000. Smaller numbers correspond to more detailed logs. The default is 3000.

JobManager.ffdc.maxDiskSpace
Exceeding this maximum disk space, log files collected by the first failure data capture mechanism are removed, beginning with the oldest files first.

JobManager.ffdc.baseDir
The directory to which log and trace files collected by the ffdc tool are copied. Default directory is *<TWA_home>*`\TWS\stdlist\JM\JOBMANAGER-FFDC`.

JobManager.ffdc.filesToCopy
Log and trace files (JobManager_message.log and JobManager_trace.log) collected by the ffdc tool located in *<TWA_home>*`\TWS\stdlist\JM`. For example, `JobManager.ffdc.filesToCopy = "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_message.log" "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_trace.log"`

When a message is logged (JobManager.ffdc.triggerFilter = JobManager.msgIdFilter) that has an ID that matches the pattern

"AWSITA*E" (JobManager.msgIdFilter.msgIds = AWSITA*E), which corresponds to all error messages, then the log and trace files (JobManager.ffdc.filesToCopy = "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_message.log" "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_trace.log") are copied (JobManager.ffdc.className = ccg_ffdc_filecopy_handler) to the directory JOBMANAGER-FFDC (JobManager.ffdc.baseDir = /opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JOBMANAGER-FFDC). If the files copied exceed 10 MB (JobManager.ffdc.maxDiskSpace = 10000000), then the oldest files are removed first (JobManager.ffdc.quotaPolicy = QUOTA_AUTODELETE).

After installing the z-centric agent or dynamic agent on Windows 2012, the JobManager_message.log might not be created. In this case, perform the following procedure:

1. Stop the agent.
2. Create a backup copy of JobManager.ini, and edit the original file by changing the row:

   ```
   JobManager.loggerhd.className = ccg_multiproc_filehandler
   ```

   to

   ```
   JobManager.loggerhd.className = ccg_filehandler
   ```
3. Restart the agent.

Configuring trace properties when the agent is stopped [JobManager.Logging.cclog]

How to configure the trace properties when the agent is stopped.

To configure the trace properties when the agent is stopped, edit the [JobManager.Logging] section in the JobManager.ini file and then restart the IBM Workload Scheduler agent.

The section containing the trace properties is named:

```
[JobManager.Logging.cclog]
```

You can change the following properties:

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
Determines the type of trace messages that are logged. Change this value to trace more or fewer events, as appropriate, or on request from IBM Software Support. Valid values are:

DEBUG_MAX
Maximum tracing. Every trace message in the code is written to the trace logs.

INFO All *informational*, *warning*, *error* and *critical* trace messages are written to the trace. The default value.

WARNING
All *warning*, *error* and *critical* trace messages are written to the trace.

ERROR
All *error* and *critical* trace messages are written to the trace.

CRITICAL
Only messages which cause the agent to stop are written to the trace.

The output trace (`JobManager_trace.log`) is provided in XML format.

After installing the z-centric agent or dynamic agent on Windows 2012, the `JobManager_trace.log` might not be created. In this case, perform the following procedure:

1. Stop the agent.
2. Create a backup copy of `JobManager.ini`, and edit the original file by changing the row:

   ```
   JobManager.trhd.className = ccg_multiproc_filehandler
   ```

   to

   ```
   JobManager.trhd.className = ccg_filehandler
   ```
3. Restart the agent.

Configuring trace properties when the agent is running

Use the **twstrace** command to set the trace on the agent when it is running.

Using the **twstrace** command, you can perform the following actions on the agent when it is running:

- "See command usage and verify version" on page 12.
- "Enable or disable trace" on page 12.
- Set the traces to a specific level, specify the number of trace files you want to create, and the maximum size of each trace file. See "Set trace information" on page 12.
- "Show trace information" on page 13.
- Collect trace files, message files, and configuration files in a compressed file using the command line. See "Collect trace information" on page 13.
- Collect trace files, message files, and configuration files in a compressed file using the Dynamic Workload Console. See the section about retrieving IBM Workload Scheduler agent traces from the Dynamic Workload Console in *Troubleshooting Guide*.

You can also configure the traces when the agent is not running by editing the [JobManager.Logging] section in the `JobManager.ini` file as described in Configuring the agent section. This procedure requires that you stop and restart the agent.

twstrace command

Use the **twstrace** command to configure traces, and collect logs, traces, and configuration files (ita.ini and jobManager.ini) for agents. You collect all the information in a compressed file when it is running without stopping and restarting it.

See command usage and verify version

To see the command usage and options, use the following syntax.

Syntax

twstrace -u | **-v**

Parameters

-u Shows the command usage.

-v Shows the command version.

Enable or disable trace

To set the trace to the maximum or minimum level, use the following syntax.

Syntax

twstrace -enable | **-disable**

Parameters

-enable
Sets the trace to the maximum level. The maximum level is **1000**.

-disable
Sets the trace to the minimum level. The minimum level is **3000**.

Set trace information

To set the trace to a specific level, specify the number of trace files you want to create, and the maximum size the trace files can reach, use the following syntax.

Syntax

twstrace [**-level** <*level_number*>] [**-maxFiles** <*files_number*>] [**-maxFileBytes** <*bytes_number*>]

Parameters

-level <*level_number*>
Sets the trace level. Specify a value in the range from 1000 to 3000, which is also the default value. Note that if you set this parameter to 3000, you have the lowest verbosity level and the fewest trace messages. To have a better trace level, with the most verbose trace messages and the maximum trace level, set it to **1000**.

-maxFiles <*files_number*>
Specify the number of trace files you want to create.

-maxFileBytes *<bytes_number>*
Set the maximum size in bytes that the trace files can reach. The default is **1024000** bytes.

Show trace information

To display the current trace level, the number of trace files, and the maximum size the trace files can reach, use the following syntax.

Syntax

twstrace -level | -maxFiles | -maxFileBytes

Parameters

-level
See the trace level you set.

-maxFiles
See the number of trace files you create.

-maxFileBytes
See the maximum size you set for each trace file

Sample

The example shows the information you receive when you run the following command:

```
twstrace -level -maxFiles -maxFileBytes
AWSITA176I The trace properties are: level="1000",
max files="3", file size="1024000".
```

Collect trace information

To collect the trace files, the message files, and the configuration files in a compressed file, use the following syntax.

Syntax

twstrace -getLogs [**-zipFile** <*compressed_file_name*>] [**-host** <*host_name*>] [**-protocol** {*http | https* } **[-port** <*port_number*>][**-iniFile** <*ini_file_name*>]

Parameters

-zipFile *<compressed_file_name>*
Specify the name of the compressed file that contains all the information, that is logs, traces, and configuration files (ita.ini and jobManager.ini) for the agent. The default is **logs.zip**.

-host *<host_name>*
Specify the host name or the IP address of the agent for which you want to collect the trace. The default is **localhost**.

-protocol *http|https*
Specify the protocol of the agent for which you are collecting the trace. The default is the protocol specified in the **.ini** file of the agent.

-port <*port_number*>
Specify the port of the agent. The default is the port number of the agent where you are running the command line.

-iniFile <*ini_file_name*>
Specify the name of the **.ini** file that contains the SSL configuration of the agent for which you want to collect the traces. If you are collecting the traces for a remote agent for which you customized the security certificates, you must import the certificate on the local agent and specify the name of the **.ini** file that contains this configuration. To do this, perform the following actions:

1. Extract the certificate from the keystore of the remote agent.
2. Import the certificate in a local agent keystore. You can create an ad hoc keystore whose name must be **TWSClientKeyStore.kdb**.
3. Create an **.ini** file in which you specify:
   - **0** in the **tcp_port** property as follows:
     `tcp_port=0`
   - The port of the remote agent in the **ssl_port** property as follows:
     `ssl_port=<ssl_port>`
   - The path to the keystore you created in Step 2 in the **key_repository_path** property as follows:
     `key_repository_path=<local_agent_keystore_path>`

Configuring common launchers properties [Launchers]

About this task

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

`[Launchers]`

You can change the following properties:

BaseDir
The installation path of the IBM Workload Scheduler agent.

CommandHandlerMinThreads
The default is **20**.

CommandHandlerMaxThreads
The default is **100**.

CpaHeartBeatTimeSeconds
The polling interval in seconds used to verify if the **agent** process is still up and running. If the agent process is inactive the product stops also the **JobManager** process. The default is **30**.

DirectoryPermissions
The access rights assigned to the agent for creating directories when running jobs. The default is **0755**. Supported values are UNIX-format entries in hexadecimal notation.

DownloadDir
The name of the directory where the fix pack installation package or upgrade eImage for fault-tolerant agents or dynamic agents is downloaded during the centralized agent update process. If not specified, the following default directory is used:

On Windows operating systems:
<*TWA_home*>\TWS\stdlist\JM\download

On UNIX operating systems:
<*TWA_home*>/TWS/stdlist/JM/download

The centralized agent update process doesn't apply to z-centric agents.

ExecutorsMaxThreads
The default is **400**.

ExecutorsMinThreads
The default is **38**.

FilePermissions
The access rights assigned to the agent for creating files when running jobs. The default is **0755**. Supported values are UNIX-format entries in hexadecimal notation.

MaxAge
The number of days that job logs are kept (in path *TWA_home*/TWS/stdlidst/JM) before being deleted. The default is **30**. Possible values range from a minimum of 1 day.

NotifierMaxThreads
The default is **5**.

NotifierMinThreads
The default is **3**.

SpoolDir
The path to the folder containing the jobstore and outputs. The default is:
value of BaseDir/stdlidst/JM

StackSizeBytes
The size of the operating system stack in bytes. The default is **DEFAULT**, meaning that the **agent** uses the default value for the operating system.

Configuring properties of the native job launcher [NativeJobLauncher]

About this task

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

You can change the following properties:

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 false. This property does not apply to IBM i.

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.

DefaultWorkingDir
Specifies the working directory of native jobs. You can also specify the value for the working directory when creating or editing the job definition in the Workload Designer. When specified in the Workload Designer, this

value overrides the value specified for the **DefaultWorkingDir** property. If you do not specify any working directories, the *<TWS_home>*\bin directory is used.

JobUnspecifiedInteractive
Applies to Windows operating systems only. Specifies if native jobs are to be launched in interactive mode. It can be `true` or `false`. The default is `false`.

KeepCommandTraces
Set to `true` to store the traces of the method invocation for actions performed on a job definition, for example, when selecting from a picklist. These files are stored in the path `/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/r3batch_cmd_exec`. The default setting is `false`.

KeepJobCommandTraces
Set to `true` to store the traces of the method invocation for actions performed on a job instance, for example, viewing a spool list. These files are stored in the .zip file of the job instance. The default setting is `true`.

LoadProfile
Specifies if the user profile is to be loaded. It can be `true` or `false`. The default is `true`.

MonitorQueueName
Specifies the name of the queue where the IBM i jobs are monitored. If you do not specify this property, the default queue (QBATCH) is used.

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.

PostJobExecScriptPathName
The fully qualified path of the script file that you want to run when the job completes. By default, this property is not present in the `JobManager.ini` file. If you do not specify any file path or the script file doesn't exist, no action is taken.

This property applies to dynamic agent and z/OS agent. For details about running a script when a job completes, see *User's Guide and Reference*.

PromotedNice
Used in workload service assurance. This property is not supported on the Agent for z/OS.

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.
- An out of range value (for example -200), prompts the operating system to automatically promote the jobs with the lowest allowed nice value.
- 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.

PromotedPriority
Used in workload service assurance. This property is not supported on the Agent for z/OS.

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. Valid values are:

- `High`
- `AboveNormal` (the default)
- `Normal`
- `BelowNormal`
- `Low` or `Idle`

Note that if you a set a lower priority value than the one non-critical jobs might be assigned, no warning is given.

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:

- *TWS_user* on UNIX and Linux systems
- The local system account on Windows systems

The default is `false`.

RunExecutablesAsIBMiJobs
If you set this property to `true`, you can define IBM i jobs as generic jobs without using the XML definition. Generic jobs are automatically converted to IBM i jobs. As a side effect, generic jobs cannot be run when this parameter is enabled (`RunExecutablesAsIBMiJobs=true`). There is no default value because this property is not listed in the `JobManager.ini` file after the agent installation.

If you set this property to `true`, ensure that the user you used to install the agent has been granted the `*ALLOBJ` special authority.

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 [JavaJobLauncher]

About this task

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

`[JavaJobLauncher]`

You can change the following properties:

JVMDir
The path to the virtual machine used to start job types with advanced options. You can change the path to another compatible Java virtual machine.

JVMOptions
The options to provide to the Java Virtual Machine used to start job types with advanced options. Supported keywords for establishing a secure connection are:

- htttps.proxyHost
- https.proxyPort

Supported keywords for establishing a non-secure connection are:

- Dhttp.proxyHost
- Dhttp.proxyPort

For example, to set job types with advanced options, based on the default JVM http protocol handler, to the unauthenticated proxy server called with name myproxyserver.mycompany.com, define the following option:

```
JVMOptions = -Dhttp.proxyHost=myproxyserver.mycompany.com
 -Dhttp.proxyPort=80
```

Configuring properties of the Resource advisor agent [ResourceAdvisorAgent]

About this task

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

`[ResourceAdvisorAgent]`

You can change the following properties:

BackupResourceAdvisorUrls
The list of URLs returned by the IBM Workload Scheduler master in a distributed environment or by the dynamic domain manager either in a z/OS or in a distributed environment. The agent uses this list to connect to the master or dynamic domain manager.

CPUScannerPeriodSeconds
The time interval that the Resource advisor agent collects resource information about the local CPU. The default value is every 10 seconds.

FullyQualifiedHostname
The fully qualified host name of the agent. It is configured automatically at installation time and is used to connect with the master in a distributed

environment or with the dynamic domain manager in a z/OS or in a distributed environment. Edit only if the host name is changed after installation.

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.

ResourceAdvisorUrl

JobManager.ini
The URL of the master in a distributed environment, or of the dynamic domain manager in a z/OS or in a distributed environment, that is hosting the agent. This URL is used until the server replies with the list of its URLs. The value is `https://$(tdwb_server):$(tdwb_port)/JobManagerRESTWeb/JobScheduler/resource`, where:

$(*tdwb_server***)**
is the fully qualified host name of the master in a distributed environment or of the dynamic domain manager either in a z/OS or in a distributed environment.

$(*tdwb_port***)**
is the port number of the master in a distributed environment or of the dynamic domain manager either in a z/OS or in a distributed environment.

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 `http`). 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 master in a distributed environment or the dynamic domain manager either in a z/OS or in a distributed environment.

In a distributed environment, if **-gateway** is set to either `local` or `remote`, then this is the URL of the dynamic agent workstation where the gateway resides and through which the dynamic agents communicate. The value is `https://$(tdwb_server):$(tdwb_port)/ita/JobManagerGW/JobManagerRESTWeb/JobScheduler/resource`, where:

$(*tdwb_server***)**
The fully qualified host name of the dynamic agent workstation where the gateway resides and through which the dynamic agent communicates with the dynamic workload broker.

$(*tdwb_port***)**
The port number of the dynamic agent workstation where the gateway resides and through which the dynamic agent communicates with the dynamic workload broker.

JobManagerGW.ini
In a distributed environment, if **-gateway** is set to `local`, then **ResourceAdvisorUrl** is the URL of the master or dynamic domain

manager. The value is `https://$(tdwb_server):$(tdwb_port)/JobManagerRESTWeb/JobScheduler/resource`, where:

$(*tdwb_server***)**
The fully qualified host name of the master or dynamic domain manager.

$(*tdwb_port***)**
The port number of the master or dynamic domain manager.

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.

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 master or dynamic domain manager either in a z/OS or in a distributed environment.

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 master in a distributed environment or the dynamic domain manager either in a z/OS or in a distributed environment. 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

If there are no significant changes, the resources are synchronized with the IBM Workload Scheduler master in a distributed environment or with thedynamic domain manager either in a z/OS or in a distributed environment every `NotifyToResourceAdvisorPeriodSeconds` seconds.

Configuring properties of the System scanner [SystemScanner]

About this task

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

```
[SystemScanner]
```

You can change the following properties:

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 properties of event-driven workload automation [EventDrivenWorkload]

About this task

In the `JobManager.ini` file, the section containing the properties of event-driven workload automation is named:

```
[EventDrivenWorkload]
```

Important: Do not modify the values of these properties, however, you can consult them for troubleshooting purposes.

The following properties are contained in this section:

EventProcessorHostname
The host name where the event processor is located.

ThisWorkstation
The name of the agent workstation where the JobManager.ini file, containing these properties, is located.

EventProcessorEifSslPort
The Event Integration Facility (EIF) port number where the event processing server receives events when it is SSL-protected.

EventProcessorEifPort
The EIF port number where the event processing server receives events.

EdwaDesiredStatusRunning
To enable event-driven workload automation set to `true`, otherwise, `false`. The default value is `true`.

CRC The current cyclic redundancy check (CRC) of the event rules of this workstation as reported by the event processor.

DeployedCRC
The CRC of the event rules already deployed.

EventProcessorWorkstation
The name of the workstation on which the event processor is located.

EventProcessorPort
The port number used to contact the event processor.

EventProcessorContext
The context part of the unified resource identifier (URI) of the event processor.

EventProcessorProtocol
The protocol used to contact the event processor. Valid values: `http`, `https`. The default value is `https`.

Regular maintenance

Regular maintenance refers to the mechanisms that are used on your dynamic workstation agents to free up storage space and improve performance.

Unlike fault-tolerant agents where maintenance tasks must be performed manually using the **rmstdlist** utility command, you can have regular maintenance performed on your dynamic agent workstations to keep disk space under control by configuring the following parameters as appropriate.

Table 2. Agent configuration parameters. Configuration parameters for maintenance

File	Parameter	Description
JobManager.ini located in the path *TWA_home*/TWS/ITA/cpa/config	MaxAge	The number of days that job logs are kept (in path *TWA_home*/TWS/stdlidst/JM) before being deleted. The default is 2. Possible values range from a minimum of 1 day.
	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 in the stdlist/JM directory. The default is 3.
	JobManager.ffdc.maxDiskSpace	The maximum disk space reached, by the log files collected by the First Failure Data Capture tool, after which the oldest files are removed.
	JobManager.trhd.maxFileBytes	The maximum size that the log file can reach. The default is 1024000 bytes.
	JobManager.trhd.maxFiles	The maximum number of log files that can be stored. The default is 3.
logging.properties located in the path <TWA_home>/TWS/JavaExt/cfg/ Logs related to jobs with advanced options.	java.util.logging.FileHandler.limit	The maximum amount to write log messages to a file. Default value is 1000000 (bytes)
	java.util.logging.FileHandler.count	The number of output files to cycle through. Default value is 10.

Defining destinations with the ROUTOPTS statement

IBM Workload Scheduler for z/OS Agents and dynamic domain managers are connected to the controller. This means that you use the ROUTOPTS statement to define the connection details of an IBM Workload Scheduler for z/OS Agent and dynamic domain manager to the controller or standby controller.

Set the HTTP or HTTPS keyword to define all your IBM Workload Scheduler for z/OS Agent destinations. For IBM Workload Scheduler for z/OS Agents and dynamic domain managers, a destination is the combination of a name and the fully qualified host name, or IP address, and port number of the IBM Workload Scheduler for z/OS Agent or dynamic domain manager.

HTTP and HTTPS have the same syntax. The difference is that HTTPS defines the connection as SSL-secure. ROUTOPTS is defined in the member of the EQQPARM library specified by the PARM parameter in the JCL EXEC statement of the controller.

The syntax for defining IBM Workload Scheduler for z/OS Agent and dynamic domain managers destinations is the following:

```
ROUTOPTS HTTP|HTTPS(destination,...,destination)
```

where *destination* is indicated as follows:

For IBM Workload Scheduler for z/OS Agents

```
dest name:'hostname or IPaddr of agent'/port of agent[/type][/pulseivl][/proxyname]
```

For more information, see Figure 1 on page 1.

For dynamic domain managers

```
hostname or IPaddress of DDM/port of DDM/B[/pulseivl]
```

For more information, see Figure 2 on page 2.

For master domain managers

```
hostname or IPaddress of MDM/port of MDM/B
```

The destination name is any alphanumeric sequence of up to 8 characters. You can include as many destinations as you want within the parentheses. They must be separated by commas. Because you cannot specify more than 455 lines for each ROUTOPTS statement, you can add further destinations by specifying more than one ROUTOPTS statement.

The *pulseivl* value is optional and defines the frequency, expressed in minutes, with which the IBM Workload Scheduler for z/OS controller checks that the agent or dynamic domain manager is running. Values range from 0 (no heartbeat checking is run) to 1440 minutes. See "Configuring the heartbeat checking interval for z-centric agents and dynamic domain managers" on page 26 for details.

Note: Any *pulseivl* value defined in a `ROUTOPTS HTTP|HTTPS` *destination* statement overrides the global value eventually defined with the `HTTPOPTS PULSEIVL` keyword.

The following example specifies the destinations to two IBM Workload Scheduler for z/OS Agents:

```
ROUTOPTS HTTPS(ITAAC1:'192.27.144.12'/44112, ITAAC2:'192.14.55.231'/61424//10)
```

where a heartbeat checking interval of 10 minutes is set for ITAAC2. Note that if you use *pulseivl*, there must be two slashes (/) before it because the *type* parameter for z-centric agents is null.

The following example specifies the destination to one dynamic domain manager:

```
ROUTOPTS HTTPS(BRKS:'9.168.125.59'/8081/B/5)
```

where a heartbeat checking interval of 5 minutes is set for the dynamic domain manager. For more information about the ROUTOPTS statement, see *Customization and Tuning*.

The *proxyname* value is optional: you specify it when one or more z-centric agents are connected to the controller through a proxy server. You must have defined the *proxyname* in the PROXY parameter of ROUTOPTS.

The following example defines that the z-centric agents ZCENT1 and ZCENT2 are connected to the controller through the proxy server named PRX1 (you defined PRX1 in the PROXY parameter of ROUTOPTS):

```
ROUTOPTS HTTPS(ZCENT1:'192.27.144.12'/8081///PRX1, ZCENT2:'192.14.55.231'/16789//5/PRX1)
         PROXY(PRX1:'195.34.134.10'/3741)
```

You can modify, add, or delete an HTTP or HTTPS destination while IBM Workload Scheduler for z/OS is running. To make the changes immediately effective, without stopping and restarting the controller, issue the MODIFY command /F *procname*,RFRDEST.

Note: The RFRDEST command does not update any changes you make to the PROXY parameter in ROUTOPTS, it updates only the HTTP | HTTPS destinations, including the *proxy name*. For detailed information about RFRDEST, see *Managing the Workload*.

To list the HTTP and HTTPS destinations that are currently used by the controller, enter the following MODIFY command (the list is stored in the MLOG):

```
/F procname,DSPDEST
```

For detailed information about the DSPDEST command, see *Managing the Workload*.

You can add a maximum of 100 destinations, without having to restart the controller. The MODIFY command manages up to a total of 100 new destinations, regardless if you add them at once or at different times. After 100 destinations added, you must stop and restart the controller to have the MODIFY command manage other 100 new destinations.

Note: Deleting a destination does not increment the number of maximum additions allowed.

You can modify or delete an unlimited number of destinations. However, if you modify the destination *name* this operation is considered as adding a new destination.

Chapter 2. Customizing the HTTP connections

The communication between the IBM Workload Scheduler for z/OS Agents and the IBM Workload Scheduler for z/OS controller is direct, through the HTTP or HTTPS protocol.

This chapter includes the following customization topics for the HTTP/HTTPS connections:

- "Customizing the HTTP connection with the HTTPOPTS statement"
- "Customizing the HTTP timeout on the Dynamic Workload Console"

Customizing the HTTP connection with the HTTPOPTS statement

HTTPOPTS statement to customize HTTP connection.

You can optionally use the HTTPOPTS initialization statement to define a number of parameters for IBM Workload Scheduler for z/OS Agents and dynamic domain managers, for example:

- Variable tables to be used when running job types with advanced options
- Job log retrieval parameters

This statement is optional. You do not have to define the TCPIPJOBNAME and the HOSTNAME parameters if you already have a TCPOPTS initialization statement that includes them. Define the job log retrieval parameters if you want to change the defaults.

For a detailed description of the HTTPOPTS statement, see *Customization and Tuning*.

Customizing the HTTP timeout on the Dynamic Workload Console

You can increase the length of the HTTP connection timeout in the Dynamic Workload Console to allow more time for retrieving objects related to the z-centric environment.

If you find that you get message errors, such as:

```
AWSJZC019E A communication error has occurred with IBM Workload Scheduler for z/OS.
The connection wait has exceeded the timeout of  "30,000" milliseconds...
```

when you try to retrieve z-centric objects from the z/OS engine on the Dynamic Workload Console, this implies that the HTTP/HTTPS connections within your z-centric environment might be suffering or that some of your plug-in applications require longer than expected.

To mitigate this problem you can increase the timeout allowed on the Dynamic Workload Console when receiving data from the IBM Workload Scheduler for z/OS controller.

To do so, set the `zosHttpTimeout` option of the `TdwcGlobalSettings.xml` file that configures the global settings of the Dynamic Workload Console.

If the option is not set, it takes the default value of 30,000 milliseconds (ms), which is probably not enough to fulfill your requirements. In this case, add the option in `TdwcGlobalSettings.xml` and set a value larger than 30,000 ms (for example, 90,0000 ms).

For additional information, see the *IBM Workload Automation: Dynamic Workload Console User's Guide*.

Configuring the heartbeat checking interval for z-centric agents and dynamic domain managers

You can configure the IBM Workload Scheduler for z/OS controller to check at intervals if all or specific attached z-centric agents and dynamic domain managers are running. Without the use of this mechanism the controller does not register, until the next job submission occurs, that a workstation has become unavailable. This may impact your scheduling plans.

The use of this feature is optional and configurable by adding the PULSEIVL keyword in the following IBM Workload Scheduler for z/OS initialization statements:

HTTPOPTS
The value for PULSEIVL is valid globally for all the attached z-centric agents and dynamic domain managers.

ROUTOPTS
PULSEIVL is featured as a parameter in the `HTTP/HTTPS destination` keyword and in this case defines the heartbeat checking interval of the specific agent defined in the destination. This option is useful when heartbeat checking is wanted only on the more strategic agents or when a different checking frequency is desired.

The pulse interval of a destination can be updated using the `HTTP refresh destination` command without having to stop the controller.

The value is expressed in minutes and can range from zero (no heartbeat checking is run) to 1440.

For example:

```
HTTPOPTS  ...
          PULSEIVL(60)
          ...

ROUTOPTS HTTPS(DDM1:'162.17.129.32'/65232/B/5)
```

This configuration sets a heartbeat checking interval of 60 minutes for all z-centric agents and dynamic domain managers, but for DDM1 the interval is set to 5 minutes.

See *IBM Workload Scheduler for z/OS: Customization and Tuning* for details on the HTTPOPTS and ROUTOPTS initialization statements and their keywords.

Configuring the connection between the IBM Workload Scheduler for z/OS Agent and controller through a proxy

You can configure that one or more z-centric agents connect with the controller through a proxy server, by defining the proxy properties in the ROUTOPTS statement and `JobManager.ini` file.

About this task

On the controller, set the proxy properties in ROUTOPTS by performing the following steps:

1. Define one or more proxy destinations as follows:

   `ROUTOPTS PROXY(`*`destination`*`, ...,`*`destination`*`)`

   Each *`destination`* has the following syntax:

   `proxy name``:'`*`IP address or hostname`*`'/`*`port`*

   where:

   proxy name
   : The name assigned to the destination, up to 4 alphanumerical characters.

   IP address or hostname
   : The fully qualified host name or IP address used to communicate with the proxy server. It can be up to 52 alphanumeric characters.

   port
   : The port number used to communicate with the proxy server.

2. In the HTTP|HTTPS parameter specify the *`proxy name`* that you defined, as follows:

   `ROUTOPTS HTTP|HTTPS(`*`dest name`*`:'`*`IPaddr or hostname`*`'/`*`port`*`[/`*`type`*`][/`*`pulseivl`*`][/`*`proxy name`*`])`

3. On the z-centric agent, edit the `[ITA_Env]` section of the `JobManager.ini` file as described in "Configuring proxy properties [ITA_Env]" on page 8

The following example shows a ROUTOPTS statement:

```
ROUTOPTS PROXY(PRX1:'192.27.144.13'/44113,PRX2:'192.27.144.14'/44118)  1
         HTTP(ZCENT1:'192.27.144.12'/44112///PRX2,ZCENT2:'192.14.55.231'/61424///PRX2)  2
         HTTPS(ZCENT3:'192.27.144.13'/8913/Z//PRX1)                                     3
```

where:

1 Two proxy destinations, PRX1 and PRX2, are configured.

2 The z-centric agents ZCENT1 and ZCENT2 connect with the controller through proxy PRX2.

3 The z-centric agent ZCENT3 connects with the controller through proxy PRX1, in an SSL-secure connection.

Starting and stopping HTTP subtasks

An HTTP Server subtask and an HTTP Client subtask run the transmission of data between the scheduler and the IBM Workload Scheduler for z/OS Agents on the distributed side. These subtasks automatically start at controller startup.

At anytime you can start or stop a subtask while IBM Workload Scheduler for z/OS is running by using the MODIFY command. You can enter this command from a multiple console support (MCS) console or from a program such as the spool display and search facility (SDSF). In both cases, the terminal or console operator must have the required authority to enter operator commands.

Enter the MODIFY command as follows:

```
/F procname,modifyoption
```

where:

procname
Is the IBM Workload Scheduler for z/OS controller JCL procedure name.

modifyoption
Can be one of the following:

S=*taskname*
Start the specified subtask.

P=*taskname*
Stop the specified subtask.

where *taskname* is either of the following:

HTC The HTTP Client subtask

HTS The HTTP Server subtask

Chapter 3. Defining user IDs and passwords with the USRREC statement

Defining IDs and passwords with USRREC statement. Use the USRREC initialization statement to define the passwords for the users who need to schedule jobs to run on Windows workstations.

You do not need to use the USRREC initialization statement to define the passwords for the users who need to schedule jobs to run on Windows workstations. do this if you run native job types on non-Windows workstations.

If you run job types with advanced options (database, file transfer, web service, xajob, J2EE jms) you can use USSREC regardless of the operating system. The advantage of doing this is that you do not have to hard code the password in the job definition as it will be retrieved from the IBM Workload Scheduler for z/OS database at run time.

If the statement is very long and needs to be split into two lines, use the comma (,) as a continuation character to continue the string to the next line.

USRREC is defined in the member of the EQQPARM library specified by the USRMEM keyword in the HTTPOPTS statement.

You can make changes to the USRREC statement while IBM Workload Scheduler for z/OS is running. To make them immediately effective, without stopping and restarting the controller, enter the following MODIFY operator command:

```
/F procname,RFRUSER
```

where *procname* is the IBM Workload Scheduler for z/OS JCL procedure name. For more information about the MODIFY command, see *Managing the Workload*. For more information about the USRREC statement, see "USRREC."

USRREC

Purpose

This statement defines the passwords for the users who need to schedule native jobs to run on Windows workstations, or who want to use this feature when defining job types with advanced options that will run on workstations based on any operating system.

After you have defined a user and its password, you use these values as follows in the definitions of job types with advanced options:

The value in USRNAM

- In the JOBREC job definition statement, it is specified as the value of the JOBUSR keyword.
- In the definition panel of the job in the Dynamic Workload Console it is specified in the `User Name` field of the `Credentials` section.

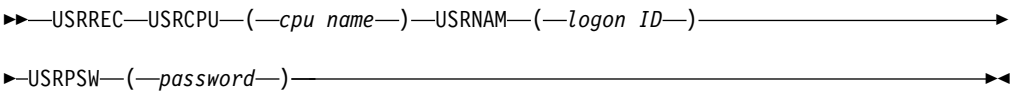
The value in USRPSW

Is resolved at run time if you:

- Specify `Y` for the JOBPWD keyword in the JOBREC job definition statement.
- Select `User` in the `Password type` widget that is displayed by clicking the ellipsis (...) located by the `Password` field in the `Credentials` section of the job definition panel in the Dynamic Workload Console.

USRREC is defined in the member of the EQQPARM library as specified by the USERMEM keyword in the TOPOLOGY statement.

Format



Parameters

USRCPU(*cpu name***)**
The IBM Workload Scheduler workstation on which the user can launch jobs. It consists of four alphanumeric characters, starting with a letter.

USRNAM(*logon ID***)**
The user name of a Windows workstation or the name of the user authorized to run a job type with advanced options. The name is case-sensitive and can be of up to 47 characters.

If you are defining a user that will run native jobs on Windows workstations, note that:

- The name can include a domain name and might be case-sensitive, depending on the Windows version.
- The user must be able to log on to the computer on which IBM Workload Scheduler has launched jobs, and must also be authorized to *Log on as batch*.
- If the name is not unique in Windows, it is considered to be either a local user, a domain user, or a trusted domain user, in that order.
- If you are defining a Windows domain user and the value contains a backslash (\), then the entire character string must be enclosed by single quotes, for example:

  ```
  USRNAM('XXXXX\user1')
  ```

- If you are defining a Windows user in the *username@internet_domain* format that contains the at sign (@), for example `administrator@mywindom.com`, you must enclose the entire character string in single quotes:

  ```
  USRNAM('administrator@mywindom.com')
  ```

USRPSW(*password***)**
The password for the user specified in USRNAM. It can consist of up to 31 characters and must be enclosed in single quotation marks. It is case-sensitive. You can change the password every time you create a Symphony® file (that is, when creating a CP extension).

The password is stored in the USRINFO member in plaintext, meaning that it is *not encrypted*. To encrypt it, run the sample EQQE2EPW JCL provided in the EQQBENCR member of the EQQSAMP library. For details about this sample JCL, see *IBM Workload Scheduler for z/OS: Planning and Installation*.

If the password is for a user that will run native jobs on Windows workstations, note that:

- The password might be case-sensitive, depending on the Windows version.
- If you are defining a Windows domain user and the value contains a backslash (\), then the entire character string must be enclosed by single quotes, for example:

  ```
  USRPSW('XXXXX\password1')
  ```

- If you are defining a password for the Windows user in the *username@internet_domain* format that contains the at sign (@), for example `administrator@mywindom.com`, you must enclose the entire character string in single quotes:

  ```
  USRPSW('passw0rd')
  ```

- As an alternative to defining the password in the USRREC statement, you can define it in a local file on the Windows workstation, by using the user utility, and set LOCALPSW(YES) in the TOPOLOGY statement.

Chapter 4. Defining variables and passwords for local resolution

You can set variables and passwords to be defined and resolved locally on the IBM Workload Scheduler for z/OS agents.

This is particularly useful in the case of passwords because you are not required to specify them in the job definition. The advantage is that, if the password has to change, you do not modify the job definition, but you change it with the `param` command locally on the agents (or on the pool agents) that run or may run the job.

This feature is not restricted to Windows workstations alone. You can use it also on UNIX, as long as you apply it on job types with advanced options.

See the following sections for:

- "Creating and managing variables and passwords on the agents."
- "Specifying variables and passwords in the JOBREC statement" on page 36 in the JOBLIB data set.
- "Specifying local variables and passwords in job types with advanced options" on page 36 to define your job in the Dynamic Workload Console so that the variables or passwords are resolved locally on the agents at run time.

Creating and managing variables and passwords on the agents

Use the `param` command to define and manage user passwords and variables locally on the IBM Workload Scheduler for z/OS agents. The values are resolved at submission time on the agent where the job is submitted.

Authorization

To create, delete, or display variables or passwords, you must have Administrator or root user rights on the workstation that runs the agent or *TWS_user* rights on the agent.

Syntax

param -u | -V |
{-c | -ec} [*file.section.*|*file..*|*section.*] *variable* [*value*] |
[*file.section.*|*file..*|*section.*] *variable* |
{-d | -fd} [*file.section.*|*file..*|*section.*] *variable*

Arguments

-u Displays command usage information and exits.

-V Displays the command version and exits.

-c | -ec
Creates variable or password *variable* and defines its value *value*. The variable or password is placed in a namespace *file* that you can organize in one or more sections named *section*.

If you do not provide a file name *file*, the variable or password is placed in default file `jm_variables` in path *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files` (`/TWA/TWS/ITA/cpa/config/jm_variables_files`) on the agent.

If you do not provide a section name *section*, the variable or password is placed in the main body of the file.

Important: If you are defining a password, you must specify a section named `password` for *variable*. This specifies that *variable* is a password.

If you are creating a variable, *variable* is the variable name and *value* is its value. If you are creating a password, *variable* is the user name and *value* is its password. If you do not enter *value* within the arguments, the command requests interactively to enter a value.

Argument **-c** creates the variable in clear form. Argument **-ec** creates the variable in encrypted form. Passwords are encrypted by default also if you use **-c**.

-d | -fd
Deletes (**-d**) or forces deletion (**-fd**) of a file, section, or variable (password). You can use the following wildcards:

***** Replaces one or more alphanumeric characters.

? Replaces one alphanumeric character.

With **-d** the command asks for confirmation before deleting. With **-fd** it deletes without asking confirmation.

When you delete all the variables in a section, the section is removed from the file. When you delete all the sections and all the variables from a file, the file is removed.

file The name of the file used as a namespace for *variable*. If you do not specify *file*, the command uses the default file `jm_variables` in path *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files` (`/TWA/TWS/ITA/cpa/config/jm_variables_files`).

All the variable namespaces go in path *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files` (`/TWA/TWS/ITA/cpa/config/jm_variables_files`).

section The name of the section within *file* where *variable* is defined. When *variable* is used for a password, it must be placed in a section named `password`. No section name is required to store variables.

value The value for *variable*.

variable
Can be a variable name or a user identification. If it is used for identification, it must be placed in a section named `password` within the namespace file.

Comments

To display a variable or password, a namespace file, or a section, use the command as follows:

param [*file.section.*|*file..*|*section.*] *variable*

where you can use the * and ? wildcards described for the deletion command.

The namespace files, including default `jm_variables`, have no extension.

Variable names are case sensitive.

On IBM i systems, if you use the QP2TERM and the QSH shells, passwords are made visible during the creation process with `param` and are displayed clearly in the shell logs. To guarantee the obfuscation of a password, you need to use the AIXTERM or XTERM shells.

Examples

The command:

```
param -c compassets.hardware.platform1 unix
```

defines variable `platform1` with value `unix` in section `hardware` of the new or existing file named `compassets`. The value is not encrypted.

The command:

```
param -c compassets..platform1 unix
```

defines variable `platform1` with value `unix` in the new or existing file named `compassets`. The value is not encrypted.

The command:

```
param -ec hardware.platform1 unix
```

defines variable `platform1` with value `unix` in section `hardware` in the default file *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files\jm_variables`. The value is encrypted.

The command:

```
param -c compassets.password.jladams san07rew
```

defines variable `jladams` with value `san07rew` in section `password` of the new or existing file named `compassets`. Since `jladams` is defined in section `password`, it is interpreted as a username. The value `san07rew` is encrypted by default since it is interpreted as a password.

The command:

```
param *.*.platform1
```

lists variable `platform1` in all its defined locations. That is:

```
..\TWA\TWS\ITA\cpa\config\jm_variables_files\compassets.hardware.platform1=unix
..\TWA\TWS\ITA\cpa\config\jm_variables_files\compassets..platform1=unix
..\TWA\TWS\ITA\cpa\config\jm_variables_files\jm_variables.hardware.platform1=**
```

The command:

```
param password.*adam*
```

lists all variables including the string `adam` contained in the `password` section of all files. In this case:

```
...\TWA\TWS\ITA\cpa\config\jm_variables_files\compassets.password.jladams=********
```

The command:

```
param -d compassets.password.jladams
```

deletes variable jladams.

The command:

```
param -d compassets.password.*
```

deletes all the variables found in section password and therefore removes this section from file compassets.

The command:

```
param -d compassets.*.*
```

deletes all the contents (variables and sections containing variables) found in file compassets and therefore removes the file.

Specifying variables and passwords in the JOBREC statement

After defining a variable and its value with the param command, to add it within a job definition with the JOBREC statement, use the following format. The values are resolved at submission time on the agent where the job is submitted.

Format

```
//JOBREC
  JOBUSR(username)
  JOBPWD(AGENT)
//END JOBREC
```

where:

JOBUSR(*username*)
The name of the user to access WebSphere Application Server, if required. If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined with the JOBPWD keyword.

- If you are defining a Windows domain user, use the following format:

  ```
  JOBUSR(domainName\user1)
  ```

- If you are defining a Windows user in the *username@internet_domain* format, use the following format:

  ```
  JOBUSR('administrator@mywindom.com')
  ```

JOBPWD(AGENT)
Means that the password is resolved locally on the agent. The password must have been defined on the agent by means of the param command. This feature is available independently of the operating system of the workstation.

Specifying local variables and passwords in job types with advanced options

After defining a variable and its value with the param command, to add it within a job definition so that it is resolved locally on the agent at run time, use the following syntax:

```
${agent:variable_name}
```

After defining a password with the param command, to add it within a job definition so that it is resolved locally on the agent at run time, use the following syntax:

```
${agent:password.user_name}
```

You can nest variables within passwords. If you used a variable to define a user, enter the password as follows within the job definition:

```
${agent:password.${agent:variable_name}}
```

where *variable_name* was previously defined as having value *user_name* with the `param` command.

Example

An IBM Workload Automation administrator needs to define a file transfer job that downloads a file from a remote server to one of the IBM Workload Scheduler for z/OS Agents making up a pool of agents. The administrator wants to parametrize in the job definition:

- The name with which the remote file will be saved locally
- The remote user name and its password

The administrator proceeds in the following way on one of the agents:

1. Defines a variable named `localfile`. The variable is given a value equal to `./npp.5.1.1.Installer.DW.exe` and is created in a new variables file named `FTPvars` (no section). The command to do this is:

   ```
   E:\IBM\TWA\TWS\CLI\bin>param -c FTPvars..localfile ./npp.5.1.1.Installer.DW.exe
   ```

2. Defines a variable named `remoteUser`. The variable is given a value equal to `FTPuser` and is created in the `FTPvars` file (no section). The command to do this is:

   ```
   E:\IBM\TWA\TWS\CLI\bin>param -c FTPvars..remoteUser FTPuser
   ```

3. Defines the password for `FTPuser`. The password value is `tdwb8nxt` and is created in the `password` section of the `FTPvars` file. The command to do this is:

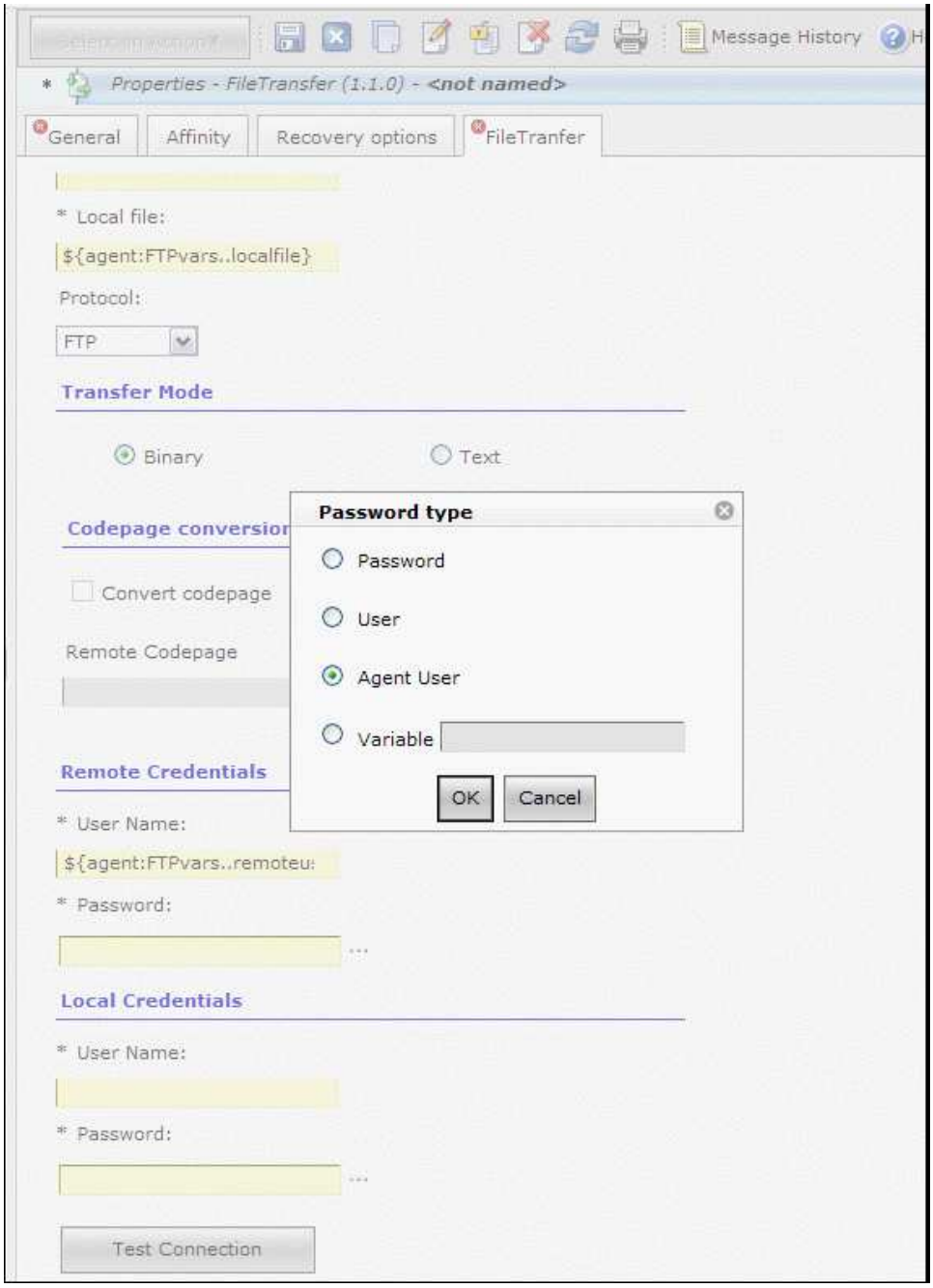
   ```
   E:\IBM\TWA\TWS\CLI\bin>param -c FTPvars.password.FTPuser tdwb8nxt
   ```

4. With a text editor opens file `E:\IBM\TWA\TWS\ITA\cpa\config\jm_variables_files\FTPvars` and checks its contents:

   ```
   localfile = ./npp.5.1.1.Installer.DW.exe
   remoteuser = FTPuser

   [password]
   FTPuser = {aes}XMMYMY2zBHvDEDBo5DdZVmwOJao60pX1K6x2HhRcovA=
   ```

5. Copies file `FTPvars` in the *agent_installation_path*`\TWA\TWS\ITA\cpa\config\jm_variables_files>` path of every other agent defined in the pool.
6. Starts defining the new file transfer job in the Workload - Design windows of Dynamic Workload Console. In the `FileTransfer` window:
   a. Enters `${agent:FTPvars..localfile}` in the `Local file` field.
   b. Enters `${agent:FTPvars..remoteuser}` in the `Remote Credentials →User Name` field.
   c. Clicks the `...` button next to the `Remote Credentials →Password` field. The `Password type` window pops up and the administrator selects `Agent User`.



d. After the administrator clicks the `OK` button for confirmation in the popup window, the `Remote Credentials →Password` field is complete with the `${agent:password.${agent:FTPvars..remoteuser}}` value.

7. Type in all the other fields to complete the job definition.

When the job is run, the entities and the password entered as variables are resolved with the values defined in the `FTPvars` file.

Chapter 5. Defining and managing IBM Workload Scheduler for z/OS Agent workstations in the database

After defining the destinations of your IBM Workload Scheduler for z/OS Agents, you need to define the IBM Workload Scheduler for z/OS Agents as IBM Workload Scheduler for z/OS computer automatic workstations.

You can define the IBM Workload Scheduler for z/OS Agents as IBM Workload Scheduler for z/OS computer automatic workstations by using either of the following:

ISPF The workstation definition and management panels in ISPF include the `Z-CENTRIC AGENT` field. You can take the following actions on IBM Workload Scheduler for z/OS Agent workstation definitions:

- "Creating"
- "Modifying" on page 40
- "Browsing" on page 40
- "Deleting" on page 41
- "Listing" on page 41

Dynamic Workload Console
For details, refer to the Dynamic Workload Console online help.

Creating

Figure 4 shows the CREATING GENERAL INFORMATION ABOUT A WORK STATION panel, used to create general information about an IBM Workload Scheduler for z/OS Agent workstation named `ZCEN`.

```
EQQWCGEP ----- CREATING GENERAL INFORMATION ABOUT A WORK STATION --------
Command ===>

Enter the command R for resources or A for availability or O for end-to-end
options, or enter/change data below:

WORK STATION NAME   ===> ZCEN
DESCRIPTION         ===> z-centric workstation_____
WORK STATION TYPE   ===> C         G  General, C  Computer, P  Printer
                                   R  Remote Engine
REPORTING ATTR      ===> A         A  Automatic, S  Manual start and completion
                                   C  Completion only, N  Non reporting
PRINTOUT ROUTING    ===> SYSPRINT  The ddname of daily plan printout data set
SERVER USAGE        ===> B         Parallel server usage, B, N, P, or C
DESTINATION         ===> ITAAC1    Name of destination
Options: allowed Y or N
 SPLITTABLE         ===> N         JOB SETUP            ===> N
 STARTED TASK, STC  ===> N         WTO                  ===> N
 AUTOMATION         ===> N         FAULT-TOLERANT AGENT ===> N
 WAIT               ===> N         Z-CENTRIC AGENT      ===> Y
 VIRTUAL            ===> N         DYNAMIC              ===> N

REMOTE ENGINE TYPE  ===>           Z z/OS or D Distributed
Defaults:
 TRANSPORT TIME     ===> 00.00     Time from previous work station  HH.MM
 DURATION           ===> 00.05.00   Duration for a normal operation  HH.MM.SS
```

Figure 4. EQQWCGEP - Creating general information about an IBM Workload Scheduler for z/OS Agent workstation

Define an IBM Workload Scheduler for z/OS Agent as a computer automatic workstation and enter the destination name previously defined in the ROUTOPTS statement. The Z-CENTRIC AGENT option excludes all other options.

Modifying

Figure 5 shows the MODIFYING GENERAL INFORMATION ABOUT A WORK STATION panel, used to modify the database definition of an IBM Workload Scheduler for z/OS Agent workstation named ZCEN.

```
EQQWMGEP ---- MODIFYING GENERAL INFORMATION ABOUT A WORK STATION --------------
Command ===>
Enter the command R for resources  A for availability  O for end-to-end options
or D for Destinations above, or enter data below:

Work station name      : ZCEN
DESCRIPTION         ===> z-centric workstation_____
WORK STATION TYPE   ===> C         G  General, C  Computer, P  Printer
REPORTING ATTR      ===> A         A  Automatic, S  Manual start and completion
                                   C  Completion only, N  Non reporting
PRINTOUT ROUTING    ===> SYSPRINT  The ddname of daily plan printout data set
SERVER USAGE        ===> N         Parallel server usage C , P , B or N
DESTINATION         ===> ITAAC1__  Name of destination
Options:  allowed  Y or N
 SPLITTABLE         ===> N         JOB SETUP            ===> N
 STARTED TASK, STC  ===> N         WTO                  ===> N
 AUTOMATION         ===> N         FAULT-TOLERANT AGENT ===> N
 WAIT               ===> N         Z-CENTRIC AGENT      ===> Y
 VIRTUAL            ===> N         DYNAMIC              ===> N
Defaults:
 TRANSPORT TIME     ===> 00.00     Time from previous work station HH.MM
 DURATION           ===> ________  Duration for a normal operation HH.MM.SS
```

Figure 5. EQQWMGEP - Modifying an IBM Workload Scheduler for z/OS Agent workstation description

You can, at any time, modify a *computer automatic* workstation as follows:

- From z-centric to fault-tolerant, and vice versa
- From z-centric to computer automatic, and vice versa

For detailed information about how to modify a fault-tolerant agent workstation to a z-centric one, refer to "Listing" on page 41.

Browsing

Figure 6 on page 41 shows the BROWSING A WORK STATION DESCRIPTION panel, used to browse the database description of an IBM Workload Scheduler for z/OS Agent workstation named ZCEN.

```
EQQWBGEP ------------ BROWSING A WORK STATION DESCRIPTION ----------------------
Command ===>

Enter the command R for resources, A for availability, O for end-to-end options
or D for Destinations above

Work station          : ZCEN
Description           : z-centric workstation

Work station type     : Computer
Reporting attribute   : Automatic
FT Work station       : No
Printout routing      : SYSPRINT
Server usage          : Neither
Destination           : ITAAC1

Splittable            : No     Job setup             : No
Started task STC      : No     WTO                   : No
AUTOMATION            : No     Fault-tolerant agent  : No
WAIT                  : No     z-centric agent       : Yes
VIRTUAL               : No     Dynamic               : No
Transport time        : 00.00
Duration              :
```

Figure 6. EQQWBGEP - Browsing an IBM Workload Scheduler for z/OS Agent workstation description

Deleting

Figure 7 shows the CONFIRMING DELETION OF A WORK STATION DESCRIPTION panel, used to delete the database description of an IBM Workload Scheduler for z/OS Agent workstation named ZCEN.

```
EQQWDGEP ----- CONFIRMING DELETION OF A WORK STATION DESCRIPTION --------------
Command ===>

Enter Y in the command field to confirm deletion or
enter N to reject deletion.

Work station          : ZCEN
Description           : Accts Receivable

Work station type     : Computer
Reporting attribute   : Automatic

Printout routing      : SYSPRINT
Server usage          : Neither
Destination           : ITAAC1
Splittable            : No     Job setup             : No
Started task STC      : No     WTO                   : No
AUTOMATION            : No     Fault-tolerant agent  : No
WAIT                  : No     z-centric agent       : Yes
VIRTUAL               : No     Dynamic               : No
Transport time        : 00.00
Duration              :
```

Figure 7. EQQWDGEP - Deleting an IBM Workload Scheduler for z/OS Agent workstation description.

Listing

Figure 8 on page 42 shows the SPECIFYING WORK STATION LIST CRITERIA panel, used for specifying selection criteria to list IBM Workload Scheduler for z/OS Agent workstation definitions.

```
EQQWWSEP ----------- SPECIFYING WORK STATION LIST CRITERIA --------------------
Command ===>

Specify selection criteria below and press ENTER to create a list.

WORK STATION NAME   ===> ____
DESTINATION         ===> ________
TYPE                ===> ___        G, C, P in any combination, or blank
REPORTING ATTRIBUTE ===> ____       A, S, C, N  in any combination or blank
FT Work station     ===> _          Y, N or blank
AUTOMATION          ===> _          Y, N or blank
WAIT Work station   ===> _          Y, N or blank
VIRT Work station   ===> _          Y, N or blank
z-Centric Agent     ===> Y          Y, N or blank
Dynamic             ===> _          Y, N or blank
```

Figure 8. EQQWWSEP - Specifying list selection criteria for IBM Workload Scheduler for z/OS Agent workstations

Managing the status of the IBM Workload Scheduler for z/OS Agent workstation in the plan

Use the WSSTAT command when you want to:

- Change the status of an IBM Workload Scheduler for z/OS Agent workstation in the current plan.
- Verify if the communication between the IBM Workload Scheduler for z/OS Agent workstation and the controller is active.

You use WSSTAT on IBM Workload Scheduler for z/OS Agents in the same way as you do with any other computer automatic workstation.

The status information is communicated to the controller to indicate that a workstation is in one of the following states:

ACTIVE
The workstation is running and scheduling its own plan.

OFFLINE
Communication is lost between the controller and the IBM Workload Scheduler for z/OS Agent workstation on the system that the workstation represents. The HTTP client retries to establish the connection until it is successful.

FAILED
You manually set the workstation status to FAILED.

When you run the WSSTAT command, you can optionally define restart and routing options for the workload defined on the workstation when you are reporting a status of OFFLINE or FAILED.

You can invoke WSSTAT as a TSO command or by using a batch job which runs program EQQEVPGM. If you invoke WSSTAT as a TSO command, you must allocate the EQQMLIB data set to the address space of the TSO user, either by adding DD statements to the logon procedure, or by using the ALLOC command after TSO logon. In the TSO environment, error messages and trace records are sent directly to the terminal user. Messages are not delivered to indicate successful command execution.

Use of the WSSTAT command can be restricted with fixed resource code RL and subresource RL.WSSTAT. The authority of the requester is verified by the subsystem name identified in the command if an AUTHDEF statement is defined for that subsystem. When SUBSYS(MSTR) is specified, all tracker subsystems defined on the z/OS system where the WSSTAT command is issued attempt to verify the authority of the requester before an event is generated. A requester might be rejected by one subsystem and accepted by another.

The subsystem to which you direct the command need not be active when the command is issued. An event is generated and queued in CSA together with other job-tracking events. If the subsystem is not active when the command is issued, the authority of the requester is verified using the class name specified in the AUTHDEF statement when the subsystem was last started. If the subsystem has not been started since a z/OS IPL, no authority verification can be performed.

Note: If you used the panels to set the status of a workstation to OFFLINE, you cannot reset it to ACTIVE with WSSTAT.

Automatic and manual changes to the status of the IBM Workload Scheduler for z/OS Agent workstation

At startup, the controller does not establish a connection with an IBM Workload Scheduler for z/OS Agent workstation until the first job is submitted on that IBM Workload Scheduler for z/OS Agent. The workstation status is automatically set to ACTIVE with extended status Waiting for first submission.

When the first job is submitted, the communication between the controller and agent is opened and the workstation is set to ACTIVE.

You can manually change the IBM Workload Scheduler for z/OS Agent workstation status in the following ways:

- From ACTIVE to OFFLINE.
- From ACTIVE, Waiting for first submission to ACTIVE. This operation is also useful to check if the communication between the controller and the agent is established.
- From OFFLINE to ACTIVE.

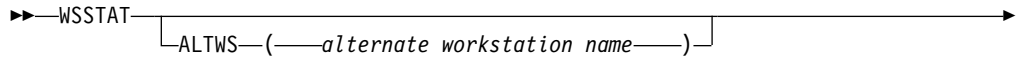
You cannot set the agent workstation to ACTIVE, if it was set to OFFLINE automatically. In this case, message EQQWL17W is issued:

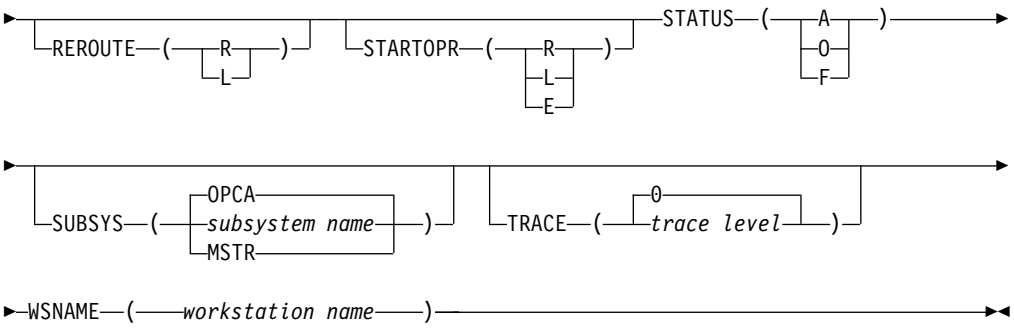
```
WORK STATION WSNAME CANNOT BE VARIED TO ACTIVE STATUS.
A QUERY AGENT INFO REQUEST HAS BEEN SENT TO THE AGENT AT DESTINATION DEST.
```

and the controller tries to establish a connection with the IBM Workload Scheduler for z/OS Agent. If the attempt is successful, the workstation status is set to ACTIVE by the controller.

WSSTAT

Format





Note: These are the only WSSTAT parameters that apply to the IBM Workload Scheduler for z/OS Agent workstation. For a complete list of the WSSTAT parameters, see *IBM Workload Scheduler for z/OS: Managing the Workload*.

Parameters

ALTWS (*alternate workstation name***)**
When the workstation status is set to OFFLINE or FAILED, you can specify the alternate workstation on which to start reroutable operations.

If you omit this parameter, the value defined for the current workstation open interval is used. If the REROUTE parameter specifies L, or if the default specifies no rerouting, the value of ALTWS is ignored.

The parameter is optional.

REROUTE (R | L)
When the workstation status is set to OFFLINE or FAILED, you can specify R for operations to be rerouted to the alternate workstation, or L for no rerouting (to leave the operations on the inactive workstation).

If you omit this parameter, the value defined in either the WSOFFLINE or the WSFAILURE keyword on the JTOPTS initialization statement is used as the default.

The parameter is optional.

STARTOPR (R | E | L)
When the workstation status is set to OFFLINE or FAILED, you can specify what the scheduler must do with operations that are currently in started status at the workstation, where:

R Restart operations automatically on the alternate workstation.
E Set all started operations to ended-in-error.
L Leave the operations in started status.

If you omit this parameter, the value defined in either the WSOFFLINE or the WSFAILURE keyword on the JTOPTS initialization statement is used as the default.

The parameter is optional.

Note: If you select STARTOPR(E), a started job continues to run, because IBM Workload Scheduler for z/OS never cancels jobs that have started.

STATUS (A | O | F)
The status you want to report for the workstation:

A Active
O Offline
F Failed

You must specify either STATUS or one of the alternative parameters.

SUBSYS (*subsystem name* **| MSTR | OPCA)**
The name of the tracker subsystem that WSSTAT is directed to. This parameter can be up to 4 characters long. The first character must be alphabetic while the remaining characters must be alphanumeric. All lowercase characters are converted to uppercase.

If you specify **MSTR**, the command is directed to all tracker subsystems on the z/OS system on which the command is issued.

Note: If the tracker and controller in your configuration run on different subsystems, specify the name of the tracker subsystem in this parameter.

TRACE (*level* **| 0)**
The event tracing indicator. When any nonzero positive number is specified, a trace entry is created for each event generated by WSSTAT. The trace record is written to the message log file identified by ddname EQQMLOG. The record identifies the name of each receiving subsystem. The default value 0 does not generate trace records.

WSNAME (*workstation name***)**
The name of the workstation to be updated. You must specify this parameter.

The WSSTAT parameters are checked for validity and consistency. The validity checks are performed on the tracker where the command is run. The validity check processes parameter names, and length and type of parameter values.

If the input is valid, a *workstation status event* is generated and communicated to the controller. Processing of the event includes a consistency check of the values specified in the parameters. The following consistency checks are made:

- The workstation name exists.
- The alternate workstation exists.
- If the value given in the STATUS parameter is the same as the current status of the workstation, the command is ignored.

Examples

The following two examples show how you can use WSSTAT in TSO, or in a batch job (using the batch program EQQEVPGM).

Example 1 - TSO command

```
ALLOC F(EQQMLIB) DA('OPC.MESSAGE.LIBRARY') SHR REUSE

WSSTAT SUBSYS(OPCB) WSNAME(AS4H) STATUS(O) START(R)
```

In this example, the status of workstation AS4H is set to OFFLINE. Started operations are restarted on the alternate workstation.

Example 2 - Batch job

```
//CHSTATUS JOB (ACCOUNT),'Change WS status',CLASS=A
//STEP1    EXEC PGM=EQQEVPGM
//STEPLIB  DD DSN=OPC.LOAD.MODULE.LIBRARY,DISP=SHR
//EQQMLIB  DD DSN=OPC.MESSAGE.LIBRARY,DISP=SHR
```

```
//EQQMLOG  DD SYSOUT=A
//SYSIN    DD *
WSSTAT SUBSYS(OPCB) WSNAME(AS4H) STATUS(A)
/*
```

In this example, the status of workstation AS4H is set to ACTIVE.

Browsing system information of IBM Workload Scheduler for z/OS Agent workstations

You can view the system information of an IBM Workload Scheduler for z/OS Agent workstation by entering the I row command in the following panels:

- Modifying Workstations in the Current® Plan (EQQMWSLL)
- Browsing Summary of Activities at a Workstation (EQQSWSSP)

Figure 9 shows what type of information is provided in the panel.

```
EQQSWIZC -------- BROWSING IWSZ Z-CENTRIC SYSTEM INFORMATION -------
Command ===>

Work station     : LWE1              Accts Receivable
Destination      : ITAAC1            HTTP
Code page        : IBM-037
Hostname         : AR11.ZAPHOD.COM
Port number      : 44112

System Information:
Operating system name : WINDOWS
Operating system level:
Agent version         : 8.6
Agent fixpack level   : 0

Default JOBREC values
JOBUSR     : user01                              User name
JOBPWD     : Y                                   Password (Y/N/A)
JOBTYPE    : accessmeth1                         Access Method
```

Figure 9. EQQSWIZC - Browsing an IBM Workload Scheduler for z/OS Agent workstation system information

Chapter 6. Defining and managing dynamic workstations

When you add dynamic capabilities to your environment, you can schedule your workload dynamically.

You can add dynamic capabilities to your environment by connecting the z/OS controller to a dynamic domain manager. The dynamic domain manager is installed in the distributed environment and manages IBM Workload Scheduler for z/OS Agents with dynamic capabilities, also installed in the distributed environment.

The type of dynamic capabilities you add to your IBM Workload Scheduler agents depends on the type of workload you plan to run:

- If you plan to run existing IBM Workload Scheduler jobs dynamically, install the IBM Workload Scheduler for z/OS Agents with dynamic capabilities and connect the agents to a dynamic domain manager connected to an IBM Workload Scheduler for z/OS controller.
- If you plan to run both existing IBM Workload Scheduler jobs and job types with advanced options dynamically, install the IBM Workload Scheduler for z/OS Agents with dynamic capabilities and Java run time and connect the agents to a dynamic domain manager connected to an IBM Workload Scheduler for z/OS controller.

In the z/OS environment, you can have the following dynamic workstations:

Broker
: Also known as dynamic domain manager in the distributed environment. An installed component in an IBM Workload Scheduler distributed or z/OS network that is the management hub in a domain where you run both static and dynamic workload. All communications to and from the dynamic agents in the domain are routed through the dynamic domain manager. You can use this workstation to submit jobs by reference. You can define the broker workstation either from ISPF panels or from the Dynamic Workload Console. For more information, see *IBM Workload Scheduler: Planning and Installation*.

Pool
: A workstation that groups a set of IBM Workload Scheduler for z/OS Agents with dynamic capabilities that have similar hardware or software characteristics to submit jobs to. IBM Workload Scheduler balances the jobs among the IBM Workload Scheduler for z/OS Agents with dynamic capabilities within the pool and automatically reassigns jobs to available agents if an agent is no longer available. To create a pool of IBM Workload Scheduler for z/OS Agents with dynamic capabilities, define a workstation of type pool hosted by the workload broker workstation, then select the IBM Workload Scheduler for z/OS Agents with dynamic capabilities you want to add to the pool. You can define the pool only using the Dynamic Workload Console.

Dynamic pool
: A workstation that groups a set of IBM Workload Scheduler for z/OS Agents with dynamic capabilities, which is dynamically defined based on the resource requirements you specify. For example, if you require a workstation with low CPU usage and Windows installed to run your job, you specify these requirements using the Dynamic Workload Console. This

workstation maps all the IBM Workload Scheduler for z/OS Agents with dynamic capabilities in your environment that meet the requirements you specified. The resulting pool is dynamically updated whenever a new suitable IBM Workload Scheduler for z/OS Agent with dynamic capabilities becomes available. You can define the dynamic pool only using the Dynamic Workload Console.

Note: For these types of workstations, the WSOFFLINE keyword of the JTOPTS initialization statement of the controller is forced to the LEAVE value.

To enable dynamic end-to-end communication, perform the following steps:

1. Install the dynamic domain manager in the distributed environment. For more information, see *IBM Workload Scheduler: Planning and Installation*.
2. Establish a connection between the z/OS controller and the dynamic domain manager: in the ROUTOPTS statement, set the DESTINATION keyword to the host name of the dynamic domain manager previously installed, in the following format:

   DESTINATION NAME=*DDM_hostname/DDM:port_number*/**B**
3. Define the dynamic domain manager workstation. Depending on how you plan to use the dynamic domain manager, the procedure varies:

   Submission of jobs by reference
   Define the dynamic domain manager using the following ISPF panels:
   a. CREATING GENERAL INFORMATION ABOUT A WORKSTATION
   b. WORKSTATION END-TO-END OPTIONS

   For more information, see "Creating broker workstations."

   Dynamic submission of jobs to pools and dynamic pools
   Define the dynamic domain manager in the Dynamic Workload Console using the **Create Workstations** panel. For more information, see "Creating pools and dynamic pools" on page 50.
4. Install the IBM Workload Scheduler for z/OS Agents with dynamic capabilities.
5. *Optionally*, define the pools and dynamic pools in the distributed environment using the Dynamic Workload Console.

To schedule jobs from IBM Workload Scheduler for z/OS to a distributed environment, you have the following options:

- Submission to pools and dynamic pools. For more information, see "Submitting jobs to pools or dynamic pools" on page 50.
- Submission of jobs defined on the dynamic domain manager. For more information, see "Submitting jobs by reference" on page 50.

Creating broker workstations

Create broker workstations using the ISPF panels and the Dynamic Workload Console.

You can use broker workstations to submit jobs by reference, as explained in "Submitting jobs by reference" on page 50. To define a broker workstation, perform the following steps:

1. Define a broker workstation as a computer automatic workstation, as described in Figure 10 on page 49 and enter the destination name previously defined in the ROUTOPTS statement. The DYNAMIC option excludes all other options.

```
EQQWCGEP ----- CREATING GENERAL INFORMATION ABOUT A WORK STATION --------
Command ===>

Enter the command R for resources or A for availability or O for end-to-end
options, or enter/change data below:

WORK STATION NAME   ===> DYNW
DESCRIPTION         ===> dynamic workstation_____
WORK STATION TYPE   ===> C         G  General, C  Computer, P  Printer
                                   R  Remote Engine
REPORTING ATTR      ===> A         A  Automatic, S  Manual start and completion
                                   C  Completion only, N  Non reporting
PRINTOUT ROUTING    ===> SYSPRINT  The ddname of daily plan printout data set
SERVER USAGE        ===> B         Parallel server usage, B, N, P, or C
DESTINATION         ===> ITAAC1    Name of destination
Options: allowed Y or N
 SPLITTABLE         ===> N         JOB SETUP            ===> N
 STARTED TASK, STC  ===> N         WTO                  ===> N
 AUTOMATION         ===> N         FAULT-TOLERANT AGENT ===> N
 WAIT               ===> N         Z-CENTRIC AGENT      ===> N
 VIRTUAL            ===> N         DYNAMIC              ===> Y
REMOTE ENGINE TYPE  ===>           Z z/OS or D Distributed

Defaults:
 TRANSPORT TIME     ===> 00.00     Time from previous work station  HH.MM
 DURATION           ===> 00.05.00   Duration for a normal operation  HH.MM.SS
```

Figure 10. EQQWCGEP - Creating general information about a broker workstation

2. Select O to open the WORK STATION END-TO-END OPTIONS panel.
3. Select BROKER=Y.

```
EQQWCEOD ----- WORK STATION END-TO-END OPTIONS--------
Command ===>

Enter/change data below:

WORK STATION NAME   ===> BRK

Default JOBREC values on this workstation:
JOBUSER   ===>                                                  User name
JOBPWD    ===>                                                  Password (Y/N/A)
JOBTYPE   ===>                                                  Access method

Dynamic options:
BROKER    ===> Y
```

Figure 11. EQQWCEOD - Defining end-to-end options

For more information, see the section about creating workstations in *IBM Workload Scheduler for z/OS: Managing the Workload*.

Defining a backup dynamic domain manager

How you can define a backup dynamic domain manager.

To define a backup dynamic domain manager, perform the following steps:

1. Install the backup dynamic domain manager in the distributed environment. For more information, see *IBM Workload Scheduler: Planning and Installation*.
2. Establish a connection between the z/OS controller and the backup dynamic domain manager: in the ROUTOPTS statement, set the DESTINATION keyword to the host name of the backup dynamic domain manager previously installed, in the following format:

   DESTINATION NAME=*backup_DDM_hostname/backup_DDM:port_number*/**B**

3. Define a dynamic workstation with type broker.
4. Define the dynamic workstation as alternate workstation of the workstation corresponding to the dynamic domain manager, the pool, and the dynamic pool.

   You do not need to define pools and dynamic pools associated to the backup dynamic domain manager. When the dynamic domain manager fails and the backup dynamic domain manager takes over, the backup dynamic domain manager automatically inherits the dynamic agents, pools, and dynamic pools defined on the dynamic domain manager.

Creating pools and dynamic pools

You can create pools and dynamic pools using the Dynamic Workload Console.

To create pools and dynamic pools using the Dynamic Workload Console , perform the following steps:

1. Log in to the Dynamic Workload Console.
2. Click **Scheduling Environment > Design > Create Workstations**.
3. Select a z/OS engine from the list and click **Create Workstation**.
4. Specify the workstation attributes using the **General**, **Resources**, and **Open Time Intervals** tabs as appropriate. Depending on the type of workstation you select, some attributes are mandatory.
5. Click **Save**.
6. Create the number and type of workstations you require.
7. Define the new workstations in the z/OS environment as dynamic workstations.
8. In the ROUTOPTS statement, set the DESTINATION keyword to the hostname of the dynamic domain manager.

For more information about workstation characteristics and attributes, see the Dynamic Workload Console online help.

Submitting jobs to pools or dynamic pools

About this task

To submit jobs to a pool or dynamic pool, perform the following steps:

1. Define a JOBREC statement using the appropriate syntax, as explained in "Defining a job in the JOBLIB data set" on page 65. Alternatively, you can create the job from the Dynamic Workload Console.
2. Submit the JOBREC statement to the workstation defined in "Creating pools and dynamic pools." The job is submitted to the workstation you specified. IBM Workload Scheduler dynamically assigns the job to the best available resource.

Submitting jobs by reference

About this task

When you submit a job by reference, you create a job definition in the distributed environment and submit it from IBM Workload Scheduler for z/OS to the distributed environment. In the distributed environment, the job is dynamically assigned to the best available resource. You write in the IBM Workload Scheduler

for z/OS JCL only the job name of the job, without having to write or import the whole job into the JCL. You can submit jobs by reference only on broker workstations.

To submit jobs by reference from IBM Workload Scheduler for z/OS, perform the following steps:

1. Define a job with name **broker_test** using the Tivoli® Dynamic Workload Broker web console. This job contains the operations to be performed:
2. Create a JCL, using the ISPF panels or the Dynamic Workload Console, specifying the name of the job you created in step 1. This references the job in the distributed environment.
3. Submit the JCL. As a result, the **broker_test** job is submitted in the distributed environment.

Chapter 7. Setting the security features

IBM Workload Scheduler for z/OS provides a secure, authenticated, and encrypted connection mechanism for communicating across the network. This mechanism is based on the Secure Sockets Layer (SSL) protocol.

By default, the IBM Workload Scheduler for z/OS Agent is installed with the SSL protocol enabled. According to this default, in the HTTPOPTS statement you must set the connection security for end-to-end communication to enable SSL.

Setting SSL-secure connections for communication using the default certificates

To provide SSL security for an HTTP connection between the z/OS controller and IBM Workload Scheduler for z/OS Agent, set the HTTPS keyword in the ROUTOPTS statement (for details about ROUTOPTS, refer to *IBM Workload Scheduler for z/OS: Customization and Tuning*).

Using security certificates

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

EQQCERCL
The security certificate for the client.

EQQCERSR
The security certificate for the server.

You can decide to use these default certificates or create your own. However, in a production environment, it is recommended that you customize SSL communication with your own certificates.

In both cases, you need to import them into your security system. If you are using RACF®, you are provided with the EQQRCERT sample job that you can run to import the certificates. To run this job, ensure that you use the same user ID that RACF associates with the controller started task.

The EQQRCERT job:

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

Customizing the SSL connection between the agents and the z/OS controller when using your certificates

Customizing the SSL connection between the agents and the z/OS controller connected to it when using your certificates.

About this task

The IBM Workload Scheduler for z/OS Agents and the z/OS controller use HTTPS to communicate. The communication process uses the default SSL certificates that come with the product. If you want to use your own certificates, attained by customizing the z/OS controller certificates, you need to customize also the agent certificates and the configuration file. To enable SSL communication, perform the following steps:

1. Generate a `.kdb` CMS key store file. This file must contain a private key trusted by the z/OS controller to which the agent is registered, and the z/OS controller public key to allow the agent to trust it.
2. Save the password of the key store in a stash file that has the same name as the file that you generated in step 1 and give it extension `.sth`.
3. Open the `ita.ini` agent configuration file and set the values specific for your environment to the following properties:

   ```
   cert_label=<label_agent_private_key>
   key_db_name=<file_name>
   key_repository_dir=<directory>
   tcp_port=0
   ssl_port=<ssl_port_value>
   ```

   Where:

   label_agent_private_key
   Is the label of the agent private key that you want to use to communicate. The default is **client**.

   file_name
   Is the name of the file, without its extension. The default value is **TWSClientKeyStore**.

   directory
   Is the name of the directory that contains the files generated in step 1 and in step 2. The default path is `/opt/IBM/TWA_<TWS_user>/TWS/ITA/cpa/ita/cert`.

   tcp_port_value
   Is the TCP/IP port value. Specify **0**.

   ssl_port_value
   Is the *tcp_port_value*. For example, if the TCP/IP port value was 31114, specify 31114.
4. Use the following command to stop the agent:

   ```
   ShutDownLwa
   ```
5. Use the following command to re-start the agent:

   ```
   StartUpLwa
   ```

After you complete the procedure, depending on the SSL storing certificate method you use, import the certificates in a RACF KEYRING or in a keystore created in the UNIX System services. Depending on the method you use refer either to the RACF or the Unix System services documentation.

Enabling FIPS compliance over IBM Workload Scheduler for z/OS server SSL secured connection

Federal Information Processing Standard Security Requirements for Cryptographic Modules, referred to as FIPS 140-2, is a standard published by the National Institute of Standards and Technology (NIST). Organizations can require compliance to the FIPS 140-2 standard to provide protection for sensitive or valuable data to cryptographic-based security systems.

System SSL was designed to meet the Federal Information Processing Standard - FIPS 140-2 Level 1 criteria.

System SSL can run in either "FIPS mode" or "non-FIPS mode". By default, System SSL runs in "non-FIPS" mode.

IBM Workload Scheduler for z/OS uses the System SSL configuration. To run IBM Workload Scheduler for z/OS in "FIPS mode", you must enable FIPS compliance over System SSL connections.

To implement FIPS 140-2 Level 1 standards perform the following actions:

1. Ensure that FIPS-compliance over a SSL connection is enabled on the controller, as described in *IBM z/OS Cryptographic Services System Secure Sockets Layer Programming manual, Chapter 4. System SSL and FIPS 140-2* .
2. On the controller, set ENABLEFIPS to YES in the HTTPOPTS initialization statement. For more information about the HTTPOPTS statement, see the *IBM Workload Scheduler for z/OS: Customization and Tuning* manual.
3. On the z-centric agent, ensure that the FIPS-compliance is enabled. For more information about how to configure SSL to be FIPS-compliant, see the *IBM Workload Scheduler: Administration Guide* .
4. On the z-centric agent ensure that:
   - SSL is configured, as described in “Customizing the SSL connection between the agents and the z/OS controller when using your certificates” on page 53
   - In the `ita.ini` file, the `ssl_port` is set and that `fips_enable = 1`.

Encrypting the user password

You define user IDs and passwords for Windows users by using the USRREC statement in a PARMLIB member (the default member name is `USRINFO`). For details, see *Customization and Tuning*. To encrypt the user password, run the sample EQQE2EPW JCL provided in the EQQBENCR member of the EQQSAMP library. For details about this sample JCL, see *Planning and Installation*.

Chapter 8. Defining and managing jobs

How to define and manage jobs within the z-centric end-to-end scheduling solution.

Defining a job from the Dynamic Workload Console

How to define a new job definitions using the Dynamic Workload Console.

About this task

From the Workload Designer, panel of the Dynamic Workload Console you can define multiple types of jobs, running on both distributed and z/OS environments. Some predefined job types are organized into categories including native jobs, and job types with advanced options, which are specific job types that you use to perform operations on external applications. In addition to the existing ones, you can define your own job types and add them to this list. Starting with IBM Workload Scheduler for z/OS version 8.6, it is possible to create JOBLIB members containing z-centric job types using the Dynamic Workload Console. These jobs leverage the job type plug-ins that are included in the z-centric agent Java extensions.

These jobs created using the Dynamic Workload Console are saved in XML format in the EQQJBLIB and should not be edited directly. If they need to be modified, changes should be made using the Dynamic Workload Console.

Also, if a USERID is provided in the job definition, it is not possible to modify the user using either the IBM Workload Scheduler for z/OS user exit EQQUX001 or the workstation end-to-end options. Only the user specified when the job is defined from the Dynamic Workload Console will be considered when the job runs.

Currently, the following job categories and types are available:

Table 3. Job Types

Category	Job Type	Description	Can you create this job type with JOBREC?
Native	Windows	Jobs that run on Windows operating systems.	NO
	UNIX	Jobs that run on UNIX operating systems. Jobs that run on limited fault-tolerant agents for IBM i.	NO
	Other	Jobs that run on extended agents. For information about customized task types for supported vendor acquired applications, see Scheduling Applications with IBM Workload Automation.	NO
	z/OS	Jobs that run the specified command in the JCL tab on a JCL system.	NO
	Executable	Jobs that run script or command with advanced options, such as redirecting standard input and standard output to a file.	YES
	Remote Command	Jobs that run on remote computers where no IBM Workload Scheduler agent installation is present.	NO
	IBM i	Jobs that run a command on IBM i systems.	YES
ERP	SAP Job on XA Workstations	Jobs that run on an SAP extended agent. This includes the three types of SAP R/3 job definitions: • Standard R/3 job • BW Process Chain job • BW InfoPackage job	NO
	SAP Job on Dynamic Workstations	Jobs that run on dynamic agent workstations, pools, dynamic pools, and z-centric agents. The following types of SAP job definition are available: • Standard R/3 job • BW Process Chain job • BW InfoPackage job	NO
	SAP PI Channel	Jobs that run SAP Process Integration (PI) Channel jobs to control communication channels between the Process Integrator and a backend SAP R/3 system.	YES
	SAP BusninessObject Business Intelligence (BI)	Jobs that enable automation, monitor, and control of workflows containing SAP BusinessObjects BI reports (Crystal and Webi reports).	NO
	Oracle E-Business Suite	Jobs that enable automation, monitor, and control of workflows containing Oracle E-Business Suite jobs.	NO
Cloud	Workload Broker	Jobs that manage the lifecycle of a dynamic workload broker job. For information about how to use dynamic workload broker, see IBM Workload Scheduler Scheduling Workload Dynamically.	NO
	Provisioning	Jobs that span physical computers, virtual machines, and private and public cloud environments creating an on-demand environment. This job type integrates with IBM SmartCloud Provisioning.	YES
	Salesforce	Jobs that enable automation, monitor, and control of Salesforce activities and data.	NO

Table 3. Job Types (continued)

Category	Job Type	Description	Can you create this job type with JOBREC?
File Transfer and Coordination	Shadow Distributed	Jobs that run locally and map other jobs running in remote IBM Workload Scheduler for z/OS distributed environments.	NO
	Shadow z/OS	Jobs that run locally and map other jobs running in remote IBM Workload Scheduler for z/OS environments.	NO
	File Transfer	Jobs that run program to transfer files to and from a server reachable using FTP, SSH, or other protocols.	YES
	Sterling Connect: Direct	Jobs that run Sterling Connect: Direct programs to transfer one or more files from one primary node to a secondary node.	NO
	Hadoop Distributed File System	Jobs that define, schedule, monitor, and manage file transfer programs between your workstation and the Hadoop Distributed File System server.	NO
Database and Integrations	Database	Jobs that perform queries, SQL statements, and jobs on a number of databases, including custom databases. You can also create and run stored procedures on DB2, Oracle, and Microsoft SQL Server, Netezza, Hive, BigSQL databases.	YES
	MS SQL	Jobs that run a Microsoft SQL Server job.	YES
	IBM WebSphere MQ	Jobs that enable communications among applications that run in different distributed environment at different times. Communications are based on the following message exchange patterns: • Request/Response. • Publish on queues or topics.	NO
	Web Services	Jobs that run a Web service.	YES
	RESTful Web Services	Jobs that send requests via HTTP methods (PUT, POST, GET, HEAD) to Web resources.	NO
	J2EE	Jobs that allow Java applications in the same network to send and receive messages from and to a JMS destination.	YES
	Java	Jobs that run a Java class.	YES
	JSR 352 Java Batch	Jobs that allow to run Java Batch applications that implement the JSR 352 standard programming specification.	NO
	MQTT	Jobs that allow to run `publish` and `subscribe` actions on topics managed by an MQTT message broker.	NO

Table 3. Job Types (continued)

Category	Job Type	Description	Can you create this job type with JOBREC?
Business Analytics	Cognos® Reports	Jobs that run IBM Cognos reports, interactive reports, query, and report views.	YES
	Informatica PowerCenter	Jobs that schedule Informatica PowerCenter workflows and track their outcomes. for more details, see *IBM Workload Scheduler Plug-in for Informatica PowerCenter: User's Guide*.	YES
	InfoSphere DataStage	Jobs that run IBM InfoSphere DataStage jobs.	YES
	Hadoop Map Reduce	Jobs that define, schedule, monitor, and manage the execution of Hadoop Map Reduce jar files.	NO
	Apache Oozie	Jobs that define, schedule, monitor, and manage the execution of Apache Oozie workflows and of the following Hadoop jobs: • MapReduce • Pig • Hive • Sqoop For more information, see *Scheduling Applications with IBM Workload Automation*.	NO
	BigInsights	Jobs that define, schedule, monitor, and manage BigInsights Workbook data sheets or Applications.	NO
OSLC	OSLC Automation	Jobs that invoke any OSLC provider that is implementing the OSLC Automation Specification. Automation resources define automation plans, automation requests and automation results of the software development, test and deployment lifecycle.	NO
	OSLC Provisioning	Jobs that invoke any OSLC provider, such as IBM Workload Scheduler and IBM SmartCloud Orchestrator, that implements the OSLC Provisioning Specification. Provisioning resources define provisioning plans, provisioning requests and provisioning results of the software development, test and deployment lifecycle.	NO

Note:

1. The access methods and application plug-ins needed to run the jobs listed above are packaged with IBM Workload Scheduler. Entitlement to use some of them requires that you purchase a separate chargeable component in addition to IBM Workload Scheduler. For a list of the chargeable components that must be purchased separately, see the *IBM Workload Scheduler: Administration Guide*.
2. According to your IBM Workload Scheduler license, IBM License Metric Tool helps you maintain your license compliance. By using License Metric Tool, you can generate reports that summarize your license consumption. The generated reports are maintained on the License Metric Tool server and should be periodically reviewed and signed, creating a history for audit purposes in the process. If you are contacted by a third-party software compliance auditor who plans to visit your enterprise to carry out a software audit, ensure that all reports are up-to-date and signed, and then supply copies of reports that cover the time periods that the auditor requests. For more detailed information about how to use the IBM License Metric Tool, see the *IBM Workload Scheduler: Administration Guide*.

Note:

- To create job types with advanced options, ensure you have completed the prerequisite steps described in "Prerequisite steps to create job types with advanced options" before performing the following procedure.

- Before defining a Provisioning job, you must have stored the Provisioning HTTP server certificate and defined it as explained in "Prerequisite steps to create Provisioning jobs".
- Before you can define OSLC Automation and Provisioning job definitions, you must perform some prerequisite steps as explained in "Prerequisites steps to create OSLC Automation and OSLC Provisioning jobs".

To create a new job definition from the Dynamic Workload Console, perform the following procedure:

Procedure

1. From the Dynamic Workload Console portfolio, click **Administration** >**Workload Design** >**Manage Workload Definitions**.
2. Specify a z/OS engine name.
3. In the Working List pane, select **New**.
4. Select the category and type of job you want to create.
5. In the properties panel, specify the attributes for the job definition you are creating. For all the details about available fields and options, see the online help by clicking the "**?**" on the top-right corner.
6. Click **Save** to save the job definition in the database.

Prerequisite steps to create job types with advanced options

How to define a new job definitions using the Dynamic Workload Console.

About this task

Perform the following steps before you define and schedule job types with advanced options.

Procedure

1. **Install a number of dynamic agents and add the Java run time**

   To install dynamic agents, run the installation program. You can install the dynamic agent during the full installation of IBM Workload Scheduler or in a stand-alone installation of just the agent. During the installation, you have the option of adding the Java run time to run job types with advanced options, both those types supplied with the product and the additional types implemented through the custom plug-ins.

   Follow the installation wizard to complete the installation.

   See the section about installation options in *Planning and Installation Guide* for descriptions of the installation parameters and options.
2. **Organize the dynamic agents in pools and dynamic pools.**

   Pools and dynamic pools help you organize the environment based on the availability of your workstations and the requirements of the jobs you plan to run.

   a. From the navigation toolbar, click **Administration > Workload Environment Design > Create Workstations**.

   b. Select a distributed or z/OS engine. The workstations you can create vary depending on the engine type you select.

   c. Select the workstation type you want to create.

      - To create a pool, define the dynamic agents you want to add to the pool and the workload broker workstation where the pool is hosted.

- To create a dynamic pool, specify the requirements that each dynamic agent must meet to be added to the dynamic pool.

3. **Grant the required authorization for defining job types with advanced options.**

   The IBM Workload Scheduler administrator has to grant specific authorizations in the security file to allow the operators to create job types with advanced options.

   - Distributed In the distributed environment, perform the following steps:
     a. Navigate to the *TWA_home*/TWSdirectory from where the **dumpsec** and **makesec** commands must be run.
     b. Run the **dumpsec** command to decrypt the current security file into an editable configuration file.

        For more information, see the section about **dumpsec** in IBM Workload Scheduler Administration.
     c. Add display and run access to the workstation, as follows:
        - If the operation is performed on the IBM Workload Scheduler Connector, display and run access is required on the CPU corresponding to the workstation where the job is created.
        - If the operation is performed on the workstation where the job runs, display access is required on the workload broker workstation.

        For more information, see the section about configuring the security file in IBM Workload Scheduler Administration.
     d. Close any open **conman** user interfaces using the **exit** command.
     e. Stop any connectors on systems running Windows operating systems.
     f. Run the **makesec** command to encrypt the security file and apply the modifications.

        For more information, see the section about **makesec** in IBM Workload Scheduler Administration.
     g. If you are using local security, the file is immediately available on the workstation where it has been updated.
        1) If you are using a backup master domain manager, copy the file to it.
        2) 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.
        3) Run **JnextPlan** to distribute the Symphony file that corresponds to the new security file.

   - z/OS In the z/OS environment, perform the following steps:
     a. Define the fixed resource that owns the subresource and give universal read access to it:

        ```
        RDEFINE (CLASS_NAME) FIXED_RESOURCE UACC(READ)
        ```
     b. Give user *USER_ID* update access to the *FIXED_RESOURCE* fixed resource:

        ```
        PERMIT FIXED_RESOURCE ID(USER_ID) ACCESS(UPDATE) CLASS(CLASS_NAME)
        ```
     c. Define a RACF resource, JSO*RACF_RESOURCE*, to RACF and give universal read access to it:

        ```
        RDEFINE (OPCCLASS) JSORACF_RESOURCE UACC(READ)
        ```

        JSO is the 3-character code that RACF uses for JS.OWNER.
     d. Give user *USER_ID* update access to JSO*RACF_RESOURCE*:

```
PERMIT JSORACF_RESOURCE ID(USER_ID) ACCESS(UPDATE) CLASS(CLASS_NAME)
```

4. Define the job types with advanced options as described in Creating job definitions.

   Distributed You can define job types with advanced options also using the **composer** command.

   For more information, see the section about job definition in the IBM Workload Scheduler User's Guide and Reference

   z/OS You can define job types with advanced options also using the **JOBREC** command.

   For more information, see "JOBREC" on page 65.

Prerequisite steps to create Provisioning jobs

How to define a new Provisioning job definition using the Dynamic Workload Console.

About this task

To create a Provisioning job definition, you must first complete the prerequisite steps listed below.

Procedure

1. Install IBM SmartCloud® Provisioning. To access the version supported by the product, generate the Data Integration report from the IBM Software Product Compatibility Reports web site, and select the **Supported Software** tab.
2. Obtain the SmartCloud HTTP server certificate and save it in a directory for later use. The Provisioning administrator can provide the certificate, or you can retrieve the certificate performing the following steps in your browser. The following example is based on Mozilla Firefox:
   a. Log in to Provisioning server providing Provisioning credentials.
   b. To download the certificate, click: **Tools**>**Options**>**Advanced**>**Encryption**>**View Certificates**
   c. Select **IBM**> **IBM WebSphere® Cloudburst Appliance** and click **Export**. A file, named `IBMWebSphereCloudBurstAppliance.crt (X509 Certificate - PEM)` is created.
3. Browse to the directory where a JRE is installed, for example: `C:\Program Files\IBM\TWS\javaExt\JRE\JRE`
4. Create a new truststore by launching the following command: `keytool -genkeypair -alias certificatekey -keyalg RSA -validity 7 -keystore keystore.jks`,

   where, *keystore.jks* is the file path to the keystore.
5. Add IBM SmartCloud certificate to the truststore by launching the following command: `keytool -import -file certificate_directory\ IBMWebSphereCloudBurstAppliance.crt -alias scp -keystore trustore_directory\keystore.jks`,
6. Open the *TWA_HOME*`\TWS\ITA\cpa\config\JobManager.ini` file, and locate `JavaJobLauncher` section, `JVMOptions` row.
7. Add the following instructions to the row: `"-Djavax.net.ssl.trustStore=DIRECTORY_TRUSTSTORE/keystore.jks -Djavax.net.ssl.trustStorePassword=TRUSTSTORE_PASSWORD"` For example:

   ```
   JVMOptions = -Djavax.net.ssl.trustStore=C:/myUtils/keystore.jks
   -Djavax.net.ssl.trustStorePassword=passw0rd
   ```

8. To complete the procedure, stop and restart the agent.

Prerequisite steps to create OSLC Automation and OSLC Provisioning jobs

How to define a new OSLC Automation and OSLC Provisioning job definition by using the Dynamic Workload Console.

About this task

To create an OSLC Automation or OSLC Provisioning job definition, you must first complete the prerequisite steps listed hereafter.

Note: Before performing the following procedure, ensure that you installed the Jazz for Service Management Registry Services from the Installation Manager.

Procedure

1. Obtain the Registry Services server certificate and save it in a directory that you will later use. Registry Services administrator can provide the certificate, or, with Firefox browser, for example, you can retrieve the certificate performing the following steps:
   a. Log in to a Registry Services (for example, `https://hostname:16311/oslc/pr`)
   b. Download the certificate by clicking in the browser toolbar: **Tools**>**Options**>**Advanced**>**Encryption**>**View Certificates**
   c. Select **IBM**>***Registry_Services_hostname:port*** and click **Export**. A file is created with the name that you specify, for example `myserver:16311`.
2. Browse to the directory where a JRE is installed, for example: `C:\Program Files\IBM\TWA_<TWS_user>\TWS\JavaExt\jre\jre\bin`
3. Create a new truststore by launching the following command: `keytool -genkeypair -alias certificatekey -keyalg RSA -validity 7 -keystore trustore_directory\keystore.jks`

   Note: Ensure that the *trustore_directory* is not created in the `javaExt\JRE` path.
4. Add the IBM Registry Services certificate to the truststore by launching the following command: `keytool -import -file certificate_directory\certificate_name -alias oslc -keystore trustore_directory\keystore.jks`
5. Open the *TWA_HOME*`\TWS\ITA\cpa\config\JobManager.ini` file, and locate `JavaJobLauncher` section, `JVMOptions` row.
6. Add the following instructions to the row: `"-Djavax.net.ssl.trustStore=`*DIRECTORY_TRUSTSTORE*`/keystore.jks -Djavax.net.ssl.trustStorePassword=`*TRUSTSTORE_PASSWORD*`"`. For example:

   ```
   JVMOptions = -Djavax.net.ssl.trustStore=C:/myUtils/keystore.jks
   -Djavax.net.ssl.trustStorePassword=passw0rd
   ```
7. Stop and restart the agent.
8. Create the `OSLCAutomation.properties` and `OSLCProvisioning.properties` files, respectively for the OSLC Automation and OSLC Provisioning jobs, and locate them in `<TWA_Home>/TWS/JavaExt/cfg/`.

   Specify the service provider catalogs (or Registry Services) that you will later use to create the job in the following format:

   ServiceProviderCatalogName=RegistryServicesURI

9. On the master domain manager, extract the security certificate from the keystore and add it to the Jazz for Service Management truststore. The file paths and names are as follows:

   Master domain manager keystore
   `<TWA_Home>/WAS/TWSProfile/etc/TWSServerKeyFile.jks`

   Jazz for Service Management truststore
   `<JazzSM_Home>/profile/config/cells/JazzSMNode01Cell/nodes/JazzSMNode01Cell/trust.p12`

10. On the workstation where you installed Jazz for Service Management, extract the security certificate from the keystore and add it to the master domain manager truststore. The file paths and names are as follows:

    Jazz for Service Management keystore
    `<JazzSM_Home>/profile/config/cells/JazzSMNode01Cell/nodes/JazzSMNode01Cell/key.p12`

    Master domain manager truststore
    `<TWA_Home>/WAS/TWSProfile/etc/TWSServerTrustFile.jks`

11. Close and restart the WebSphere Application Server on the master domain manager and on Jazz for Service Management.

Defining a job in the JOBLIB data set

You define the jobs to be run on IBM Workload Scheduler for z/OS Agent workstations in a job definition member of the JOBLIB data set. In the JOBLIB member, you can either define the job contents or invoke a script or an executable that is stored locally on the IBM Workload Scheduler for z/OS Agent workstation.

If you do not specify any content in the job definition member, the error code JCLI is issued.

To define the job contents, the following syntax rules apply:

- The lines that start with //* OPC, //*%OPC, or //*>OPC are used for variable substitution and automatic recovery. They are removed before the script is downloaded onto the IBM Workload Scheduler for z/OS Agent.
- Each line starts between column 1 and column 80.
- The last backslash (\) in a line is considered the character of continuation.
- Blanks at the end of the line are automatically removed.

To define the job properties, use the "JOBREC" statement. In the "JOBREC" statement, you can define different job types.

JOBREC

To define the properties of a job to be run on IBM Workload Scheduler for z/OS Agent workstations, use the following format:

```
//JOBREC
keyword(value)
keyword(value)
//END JOBREC
```

The following syntax rules apply:

- The value of each keyword can continue over one or more lines.
- Each statement must be on a separate line.

- The value of each keyword must be enclosed within parentheses. This means that every character enclosed within the parentheses is considered to be part of the keyword value, including blanks and single quotes.
- The script or executable must follow the //END JOBREC line.
- If you specify the same keyword more than once, only the last keyword specified is considered valid.
- You can insert any comment text by using //* as the starting characters.
- Unsupported keywords inserted in the statement are ignored without warning.
- Keywords are optional, unless otherwise indicated.
- Keywords and values are case-sensitive.

Note: When the job is defined using a JOBREC and specifying the JOBTYPE keyword, or when it is defined using the Dynamic Workload Console, any USERID specified in the job definition is not used as the user under which the job runs on the z-centric agent. Rather, the specified USERID is needed to perform logon to the remote back end processes, such as, database servers or web servers, with which the job must interact.

In contrast, with native z-centric jobs defined directly in the EQQJBLIB without any JOBREC, or with a JOBREC but without the JOBTYPE keyword, the USERID provided by the JOBUSER keyword of the JOBREC, or by EQQUX001, is used by the z-centric agent as the ID under which it submits the job on the server where the agent runs.

JOBREC is supported by z-centric Agent on IBM i systems, provided that you set `RunExecutableAsIBMiJobs=true` in the `JobManager.ini` file.

In the JOBREC statement, you can define different job types.

Supported job types and their related keywords are listed in Table 4.

Table 4. Jobs and their keywords

Job to be run	Keywords	Section
database	STATEMENT(*id*) (required) CMDTYPE(*id*) (optional) DBNAME (required) DBTYPE (required) JOBPWD (optional) JOBTYPE (required) JOBUSR (optional) POLLINGIVL(*id*) (optional, applies only to the MSSQL database type) PORT (required) SERVER (required) SYNCTYPE(*id*) (optional, applies only to the MSSQL database type) WSNAME (optional)	"Syntax diagram for database job types" on page 67

Table 4. Jobs and their keywords (continued)

Job to be run	Keywords	Section
file transfer	JOBPWD (required) JOBTYPE (required) JOBUSR (required) LOCALFILE (required) PROTOCOL (optional) REMOTEFILE (required) SERVER (required) TRANSFERTYPE (required) WSNAME (optional)	"Syntax diagram for file transfer job types" on page 74
web service	JOBPWD (optional) JOBTYPE (required) JOBUSR (optional) OPNAME (required) PARM(*id*) (optional) URL (required) WSNAME (optional)	"Syntax diagram for web service job types" on page 78
java	CLASSNAME (required) JARPATH (optional) JAVAPARM(*id*) (optional) JOBTYPE (required)	"Syntax diagram for Java job type" on page 80
xajob	ENVVAR(*id*) (optional) JOBPWD (optional) JOBTYPE (required) JOBUSR (optional) WSNAME (optional)	"Syntax diagram for xajob type" on page 81
j2ee jms	AUTHALIAS (optional) CONNFACTORY (required) DESTINATION (required) INVOKEYTYPE (optional) JOBPWD (optional) JOBTYPE (required) JOBUSR (optional) MESSAGE (optional) WSNAME (optional)	"Syntax diagram for J2EE jms job type" on page 83

Syntax diagram for database job types

The database job type:

If you set JOBTYPE to `(/database)`, you can choose among the following database types:

- DB2®
- Oracle
- MSSQL

The syntax for DB2 and Oracle is identical, but the syntax for MSSQL supports MSSQL-specific operations. For details, see "Syntax diagram for DB2 and Oracle database job types" on page 68 and "Syntax diagram for MSSQL database job type" on page 70.

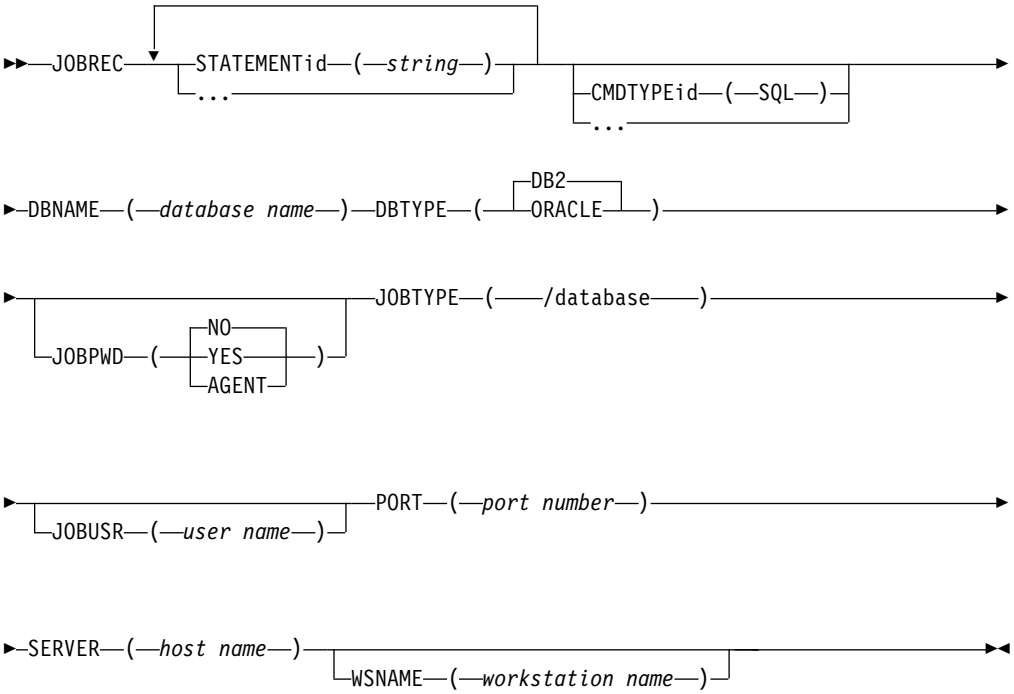
Syntax diagram for DB2 and Oracle database job types

Purpose

Syntax diagram for the DB2 and Oracle database job types. The following syntax rules apply:

- You can define multiple statements in the same job by specifying a series of STATEMENT*id* keywords, where *id* is a unique numeric value, including null and zero. For example, you can define STATEMENT, STATEMENT0, STATEMENT1, STATEMENT2, and so on.
- If you set DBTYPE to `DB2` or `Oracle`, you can set CMDTYPE to `SQL`.
- You can specify any number of CMDTYPE keywords, according to the corresponding STATEMENT keyword. The STATEMENT keyword must be specified before the corresponding CMDTYPE keyword. For example, you can specify `CMDTYPE3` only if you have previously specified `STATEMENT3`.

Format



Parameters

STATEMENT*id***(***string***)**

The string defining the SQL query or job. This keyword is required. You can define multiple statements in the same job by specifying a series of STATEMENT*id* keywords, where *id* is a unique numeric value, including null and zero. For example, you can define STATEMENT, STATEMENT0, STATEMENT1, STATEMENT2, and so on.

When running stored procedures, ensure that the returned result is in tabular format. Results in any other format are not supported.

CMDTYPE*id*(SQL)
The type of command to be run. You can specify a series of CMDTYPE*id* keywords, where *id* is a unique numeric value, including null and zero. The *id* of the CMDTYPE keyword must match the *id* of the related STATEMENT keyword. For example, you can specify `CMDTYPE3` only if you specify `STATEMENT3`.

If you set DBTYPE to `(/DB2)` or to `(/ORACLE)`, you can set CMDTYPE to `SQL`.

DBNAME(*database name*)
The name of the database. This keyword is required.

DBTYPE(DB2|ORACLE)
The database type. This keyword is required.

JOBPWD(YES|NO|AGENT)
Specifies if the user name entered in JOBUSR or set by using the job-submit exit EQQUX001 is associated with a password.

If you set JOBPWD to `YES`, IBM Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement (for details, see *IBM Workload Scheduler for z/OS: Customization and Tuning*).

If you set JOBPWD to `AGENT`, this means that the password is resolved locally on the agent. The password must have been defined on the agent by means of the param command. This feature is available independently of the operating system of the workstation.

Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to `NO` if the user works with UNIX workstations and if no password is required.

JOBTYPE(/xajob/*access method name*|/j2ee/jms|/web service|/file transfer|/database|/java)
To run a database job, specify **/database**. For information about the other types of job, see the specific section for each job type. This keyword is required.

/database
Set this value to submit a database job. Before you can run a query on a database, download the JDBC drivers for your database client to each IBM Workload Scheduler for z/OS Agent on which you want to submit database jobs. Specify the path to the database client jar files in the `DatabaseJobExecutor.properties` file, located in the `JavaExt\cfg` directory in your IBM Workload Scheduler installation directory. Define the `jdbcDriversPath` property to point to the JDBC jar files directory, for example, `jdbcDriversPath=c:\\mydir\\jars\\jdbc`. The JDBC jar files must be located in the specified directory or its subdirectories. Ensure that you have list permissions for the directory and its subdirectories.

JOBUSR(*user name*)
The user name for accessing the database.

If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined (see the JOBPWD keyword).

- If you are defining a Windows domain user, use the following format:
  `JOBUSR(domainName\user1)`
- If you are defining a Windows user in the *username@internet_domain* format, use the following format:
  `JOBUSR('administrator@mywindom.com')`

To specify the user name, you can also use the job-submit exit EQQUX001. This user name overrides the value specified for JOBUSR. In turn, the value specified for JOBUSR overrides the value specified for USRNAM in the USRREC statement.

PORT(*port number***)**
The port number for the database job. This keyword is required.

SERVER(*host name***)**
The host name of the server where the file transfer is to be performed or where the database is located. This keyword is required if you set JOBTYPE to (`/file transfer`) or to (`/database`).

WSNAME(*workstation_name***)**
The name of the workstation from which user name and password must be retrieved. User name and password can be specified in the statement, using the JOBUSR and JOBPWD keywords, or can be associated to a workstation with the USRREC statement. With the WSNAME keyword, you can specify a workstation where the user name and password are stored, which is different from the workstation where the job runs.

Examples

The following is an example of a JOBREC statement to select all records from table SYSTOOLS.POLICY:

```
//JOBREC
JOBTYPE(/database)
STATEMENT20(SELECT * FROM SYSTOOLS.POLICY)
DBNAME(DBWEB)
DBTYPE(DB2)
SERVER(9.168.99.87)
PORT(50000)
JOBUSR(Administrator)
JOBPWD(YES)
CMDTYPE20(SQL)
//END JOBREC
```

The following is an example of a JOBREC statement to select columns country_id and country_name FROM table countries:

```
//JOBREC
JOBTYPE(/database)
STATEMENT1(SELECT country_id, country_name FROM countries)
DBNAME(SAMPLE)
DBTYPE(ORACLE)
SERVER(9.168.115.37)
PORT(70000)
CMDTYPE1(SQL)
JOBUSR(ZCENTUMB)
JOBPWD(YES)
//END JOBREC
```

Syntax diagram for MSSQL database job type

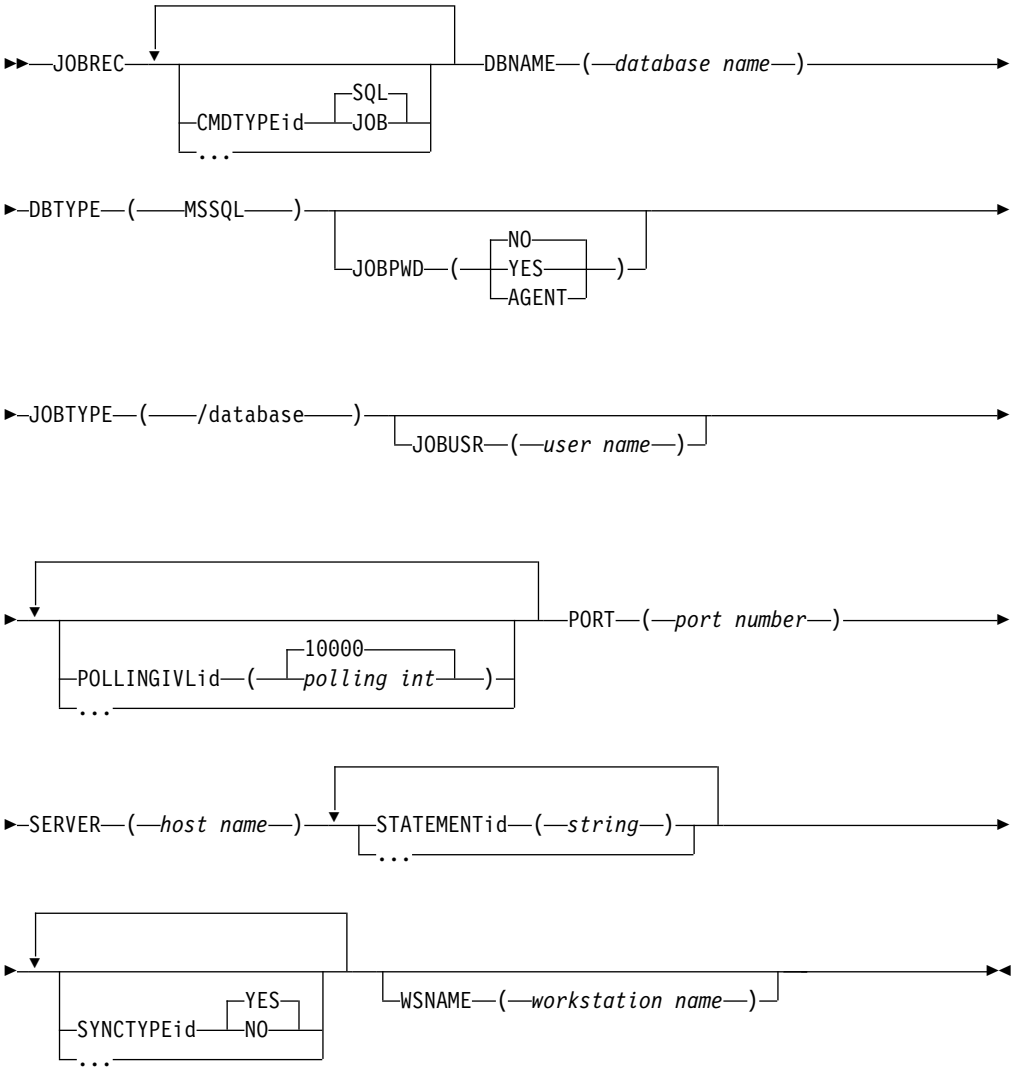
Purpose

This section provides detailed information about the syntax diagram for the MSSQL database job type. The following syntax rules apply:

- You can define multiple statements in the same job by specifying a series of STATEMENT*id* keywords, where *id* is a unique numeric value, including null and zero. For example, you can define STATEMENT, STATEMENT0, STATEMENT1, STATEMENT2, and so on.

- If you set DBTYPE to (/MSSQL), you can set CMDTYPE to either JOB or SQL and specify the related job or SQL query in the STATEMENT keyword. If you set CMDTYPE to JOB, you can specify a job saved on the server; if you set CMDTYPE to SQL, you can specify a statement in the native language of the database that you selected.
- If you set CMDTYPE to JOB, you can optionally define the SYNCTYPE and POLLINGIVL keywords. This information is not visible in the syntax diagram.
- If you set CMDTYPE to SQL, you cannot define the SYNCTYPE and POLLINGIVL keywords. This information is not visible in the syntax diagram.
- You can specify any number of CMDTYPE, SYNCTYPE, and POLLINGIVL keywords, according to the corresponding STATEMENT keyword. The STATEMENT keyword must be specified before the corresponding CMDTYPE, SYNCTYPE, and POLLINGIVL keywords. For example, you can specify CMDTYPE3, SYNCTYPE3, and POLLINGIVL3 only if you have previously specified STATEMENT3.

Format



Parameters

CMDTYPE*id***(SQL|JOB)**
The type of command to be run. You can specify a series of CMDTYPE*id* keywords, where *id* is a unique numeric value, including null and zero. The *id* of the CMDTYPE keyword must match the *id* of the related STATEMENT keyword. For example, you can specify CMDTYPE3 only if you specify STATEMENT3.

If you set DBTYPE to (/MSSQL), you can set CMDTYPE to either JOB or SQL and specify the related job or SQL query in the STATEMENT keyword. If you set CMDTYPE to JOB, you can specify a job saved on the server; if you set CMDTYPE to SQL, you can specify a statement in the native language of the database that you selected.

DBNAME(*database name***)**
The name of the database. This keyword is required.

DBTYPE(MSSQL)
The database type. This keyword is required.

JOBPWD(YES|NO|AGENT)
Specifies if the user name entered in JOBUSR or set by using the job-submit exit EQQUX001 is associated with a password.

If you set JOBPWD to YES, IBM Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement (for details, see *IBM Workload Scheduler for z/OS: Customization and Tuning*).

If you set JOBPWD to AGENT, this means that the password is resolved locally on the agent. The password must have been defined on the agent by means of the param command. This feature is available independently of the operating system of the workstation.

Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to NO if the user works with UNIX workstations and if no password is required.

JOBTYPE(/xajob/*access method name***|/j2ee/jms|/web service|/file transfer|/database|/java)**
To run a database job, specify **/database**. For information about the other types of job, see the specific section for each job type. This keyword is required.

/database
Set this value to submit a database job. Before you can run a query on a database, download the JDBC drivers for your database client to each IBM Workload Scheduler for z/OS Agent on which you want to submit database jobs. Specify the path to the database client jar files in the DatabaseJobExecutor.properties file, located in the JavaExt\cfg directory in your IBM Workload Scheduler installation directory. Define the jdbcDriversPath property to point to the JDBC jar files directory, for example, jdbcDriversPath=c:\\mydir\\jars\\jdbc. The JDBC jar files must be located in the specified directory or its subdirectories. Ensure that you have list permissions for the directory and its subdirectories.

JOBUSR(*user name***)**
The user name for accessing the database.

If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined (see the JOBPWD keyword).

- If you are defining a Windows domain user, use the following format:

```
JOBUSR(domainName\user1)
```

- If you are defining a Windows user in the *username@internet_domain* format, use the following format:

```
JOBUSR('administrator@mywindom.com')
```

To specify the user name, you can also use the job-submit exit EQQUX001. This user name overrides the value specified for JOBUSR. In turn, the value specified for JOBUSR overrides the value specified for USRNAM in the USRREC statement.

POLLINGIVL*id***(***polling int***|10000)**
The polling interval in milliseconds. This keyword is valid only if you set CMDTYPE to (/JOB). You can specify a series of POLLINGIVL*id* keywords, where *id* is a unique numeric value, including null and zero. The *id* of the POLLINGIVL keyword must match the *id* of the related STATEMENT keyword. For example, you can specify POLLINGIVL3 only if you specify STATEMENT3.

PORT(*port number***)**
The port number for the database job. This keyword is required.

SERVER(*host name***)**
The host name of the server where the file transfer is to be performed or where the database is located. This keyword is required if you set JOBTYPE to (/file transfer) or to (/database).

STATEMENT*id***(***string***)**
The string defining the SQL query or job. This keyword is required. You can define multiple statements in the same job by specifying a series of STATEMENT*id* keywords, where *id* is a unique numeric value, including null and zero. For example, you can define STATEMENT, STATEMENT0, STATEMENT1, STATEMENT2, and so on.

When running stored procedures, ensure that the returned result is in tabular format. Results in any other format are not supported.

SYNCTYPE*id***(YES|NO)**
Specifies if synchronization is required. This keyword is valid only if you set CMDTYPE to (/JOB). You can specify a series of SYNCTYPE*id* keywords, where *id* is a unique numeric value, including null and zero. The *id* of the SYNCTYPE keyword must match the *id* of the related STATEMENT keyword. For example, you can specify SYNCTYPE3 only if you specify STATEMENT3.

WSNAME(*workstation_name***)**
The name of the workstation from which user name and password must be retrieved. User name and password can be specified in the statement, using the JOBUSR and JOBPWD keywords, or can be associated to a workstation with the USRREC statement. With the WSNAME keyword, you can specify a workstation where the user name and password are stored, which is different from the workstation where the job runs.

Examples

The following is an example of a JOBREC statement to run job SPMTEST_BACKUP without synchronization and with a polling interval of 100 milliseconds, and to select all records from the DBO.TWSRECORDS table:

```
//JOBREC
JOBTYPE(/database)
DBNAME(SCVTEST)
DBTYPE(MSSQL)
```

```
SERVER(SQLCON008)
PORT(1278)
STATEMENT1(SPMTEST_BACKUP)
CMDTYPE1(JOB)
SYNCTYPE1(NO)
POLLINGIVL1(100)
STATEMENT2(SELECT * FROM DBO.TWSRECORDS)
CMDTYPE2(SQL)
JOBUSR(PWR_USER)
JOBPWD(YES)
//END JOBREC
```

Syntax diagram for file transfer job types

File transfer job type.

Syntax diagram for file transfer job types

Purpose

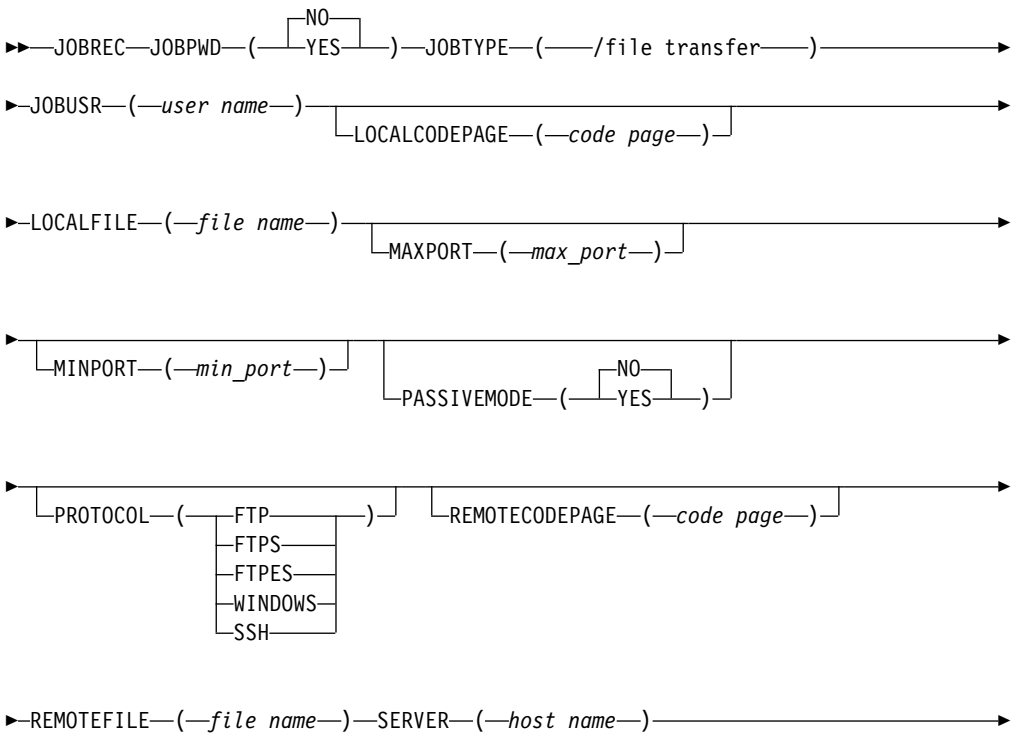
If you set JOBTYPE to (/file transfer), you can choose between the following transfer types for transferring binary and text files:

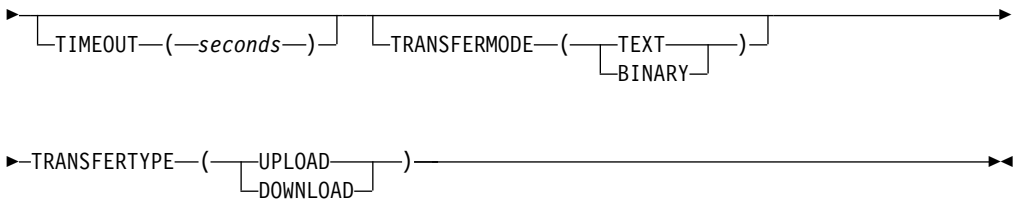
- Upload
- Download

This section provides detailed information about the syntax diagram for both the upload and download job types. The following syntax rules apply:

- The syntax of the selected protocol applies.

Format





Parameters

The MINPORT and the MAXPORT options define the port range to use on the client side of TCP data connections. When the active mode is enabled, the MINPORT and the MAXPORT options restrict the port numbers sent by the PORT command. These options are meant to accommodate highly restrictive firewall rules.

If you do not specify the MINPORT and the MAXPORT options, the operating system determines the port numbers to be used. If you specify one of these parameters, the other parameter is required.

JOBPWD(YES|NO)
Specifies that the user name set in JOBUSR or by using the job-submit exit EQQUX001 is associated with a password.

If you set JOBPWD to YES, IBM Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement (for details, see the section about defining user IDs and passwords with the USRREC statement in this manual).

Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to NO if the user works with UNIX workstations.

JOBTYPE(/xajob/*access method name*|/j2ee/ejb|/j2ee/jms|/web service|/file transfer|/database|/java)
To run a file transfer job, specify **/file transfer**. For information about the other types of job, see the specific section for each job type. This keyword is required.

/file transfer
Set this value to submit a file transfer job.

JOBUSR(*user name*)
The user name for accessing the FTP, FTPS, FTPES, SSH, or Windows server.

If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined (see the JOBPWD keyword).

- If you are defining a Windows domain user, use the following format:
  `JOBUSR(domainName\user1)`
- If you are defining a Windows user in the *username@internet_domain* format, use the following format:
  `JOBUSR('administrator@mywindom.com')`

To specify the user name, you can also use the job-submit exit EQQUX001. This user name overrides the value specified for JOBUSR. In turn, the value specified for JOBUSR overrides the value specified for USRNAM in the USRREC statement.

LOCALCODEPAGE(*code page***)**
The name of the code page used on the local system. This keyword is required only when transferring text files from a distributed to a z/OS environment and vice versa. If you specify this parameter, the REMOTECODEPAGE parameter is required.

LOCALFILE(*file name***)**
The name of the local file that you want to transfer. When uploading, this is the source file, when downloading, this is the target file. This keyword is required.

MAXPORT(*max_port***)**
The maximum port value to use on the client side of TCP data connections. For example, if you specify MAXPORT=8009, IBM Workload Scheduler for z/OS restricts the port number to be lesser than or equal to the port 8009. The default value is 1035.

MINPORT(*min_port***)**
The minimum port value to use on the client side of TCP data connections. For example, if you specify MINPORT=8000, IBM Workload Scheduler for z/OS restricts the port number to be greater than or equal to the port 8000.

PASSIVEMODE(YES|NO)
Specifies whether the server is passive or active in establishing connections for data transfers. If you set this option to NO, the server establishes the data connection with the client (active mode), if you set this option to YES, the client establishes the data connection with the server (passive mode). The default value is NO.

PROTOCOL(FTP|FTPS|FTPES|WINDOWS|SSH)
The protocol to be used for the file transfer. Allowed values are:

FTP A standard network protocol used to exchange files over a TCP/IP-based network, such as the Internet.

FTPS An extension to the File Transfer Protocol (FTP) that adds support for the Transport Layer Security (TLS) and the Secure Sockets Layer (SSL) cryptographic protocols. Specifically, the supported protocol is TLS security for FTP sessions with **implicit** secure TLS login and with private security level for the data connection.

FTPES
An extension to the File Transfer Protocol (FTP). Specifically, the supported protocol is TLS security for FTP sessions with **explicit** TLS security and private security level for the data connection.

WINDOWS
The Microsoft file sharing protocol. Specify the root shared directory in the REMOTEFILE keyword. Specify the address of the workstation hosting the shared directory in the SERVER keyword.

SSH A network protocol that provides file access, file transfer, and file management functions over any data stream.

REMOTECODEPAGE(*file name***)**
The name of the code page used on the remote system. This keyword is required only when transferring text files from a distributed to a z/OS environment and vice versa. If you specify this parameter, the LOCALCODEPAGE parameter is required. If you want to use a custom code page, define the REMOTECODEPAGE parameter as follows:

```
REMOTECODEPAGE(USER:custom_code_page)
```

where *custom_codepage* is the code page defined by the user.

For example, to use the tcpip.ftpd.ftpxlbin.frence3 custom code page, define the REMOTECODEPAGE parameter as follows:

```
REMOTECODEPAGE(USER:tcpip.ftpd.ftpxlbin.frence3)
```

REMOTEFILE(*file name***)**
The name of the remote file that you want to transfer. When uploading, this is the target file, when downloading, this is the source file. This keyword is required.

SERVER(*host name***)**
The host name of the server where the file transfer is to be performed or where the database is located. This keyword is required if you set JOBTYPE to (/file transfer) or (/database).

TIMEOUT(*seconds***)**
Specifies the number of seconds to be used for the file transfer operation. The default value is 60 seconds.

TRANSFERMODE(TEXT|BINARY)
The mode of file transfer that you want to perform, either text or binary.

TRANSFERTYPE(UPLOAD|DOWNLOAD)
The type of file transfer that you want to perform; either upload or download. This keyword is required.

Examples

The following is an example of a JOBREC statement to transfer a file using the FTP protocol:

```
//JOBREC
JOBTYPE(/file transfer)
TRANSFERTYPE(DOWNLOAD)
SERVER(server address)
LOCALFILE(C:\file1.txt)
REMOTEFILE(/file1.txt)
PROTOCOL(FTP)
JOBUSR(username)
JOBPWD(YES)
//END JOBREC
```

The following is an example of a JOBREC statement to transfer a file using the SSH protocol:

```
//JOBREC
JOBTYPE(/file transfer)
TRANSFERTYPE(DOWNLOAD)
SERVER(server address)
LOCALFILE(C:\file1.txt)
REMOTEFILE(/home/provaftp/file1.txt)
PROTOCOL(SSH)
JOBUSR(username)
JOBPWD(YES)
//END JOBREC
```

The following is an example of a JOBREC statement to transfer a file using the WINDOWS protocol:

```
//JOBREC
JOBTYPE(/file transfer)
TRANSFERTYPE(UPLOAD)
SERVER(server address)
```

```
LOCALFILE(C:\file3.txt)
REMOTEFILE(shared directory\file88.txt)
PROTOCOL(WINDOWS)
JOBUSR(username)
JOBPWD(YES)
//END JOBREC
```

The following example shows a JOBREC statement to transfer a native z/OS file from a z/OS workstation to a Windows workstation:

```
//JOBREC
JOBTYPE(/file transfer)
TRANSFERTYPE(DOWNLOAD)
TRANSFERMODE(TEXT)
SERVER(9.168.101.41)
LOCALFILE(d:\MyFTPRemoteTextFile.txt)
REMOTEFILE('TWSZ.REMOTE.TEXTFILE')
LOCALCODEPAGE(ISO8859-1)
REMOTECODEPAGE(IBM-037)
JOBUSR(root)
JOBPWD(YES)
PROTOCOL(FTP)
//END JOBREC
```

The following example shows a passive mode transfer with a timeout of 10 seconds. The system determines which port to use between 1034 and 1035:

```
//JOBREC
JOBTYPE(/file transfer)
TRANSFERTYPE(DOWNLOAD)
TRANSFERMODE(TEXT)
SERVER(9.168.107.126)
LOCALFILE(d:\MyFTPRemoteTextFile11.txt)
REMOTEFILE('TWS851.REMOTE.TEXTFILE')
LOCALCODEPAGE(ISO8859-1)
REMOTECODEPAGE(IBM-037)
TIMEOUT(10)
MINPORT(1034)
MAXPORT(1035)
PASSIVEMODE(YES)
JOBUSR(LNDN_USER)
JOBPWD(YES)
PROTOCOL(FTPES)
//END JOBREC
```

Syntax diagram for web service job types

The web service job type.

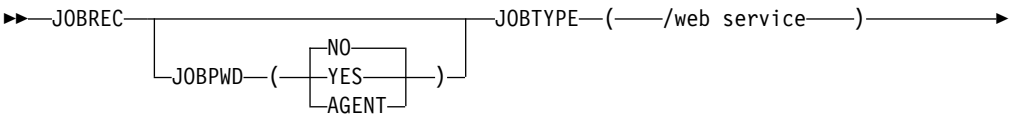
Syntax diagram for web service job types

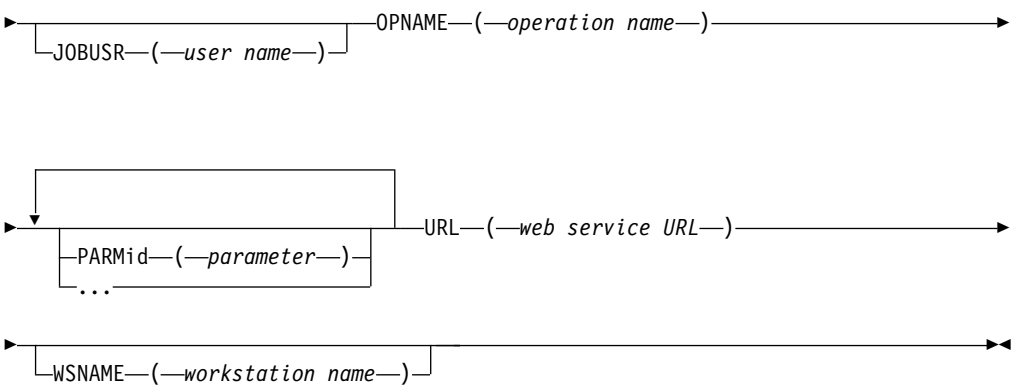
Purpose

If you set JOBTYPE to (/web service), you can run an existing web service:

This section provides detailed information about the syntax diagram for the web service job type.

Format





Parameters

JOBPWD(YES|NO|AGENT)
Specifies if the user name entered in JOBUSR or set by using the job-submit exit EQQUX001 is associated with a password.

If you set JOBPWD to YES, IBM Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement (for details, see *IBM Workload Scheduler for z/OS: Customization and Tuning*).

If you set JOBPWD to AGENT, this means that the password is resolved locally on the agent. The password must have been defined on the agent by means of the param command. This feature is available independently of the operating system of the workstation.

Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to NO if the user works with UNIX workstations and if no password is required.

JOBTYPE(/xajob/*access method name*|/j2ee/jms|/web service|/file transfer|/database|/java)
To run a web service job, specify **/web service**. For information about the other types of job, see the specific section for each job type. This keyword is required.

/web service
Set this value to submit a web service job.

JOBUSR(*user name*)
The user name for accessing the web service, if required.

If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined (see the JOBPWD keyword).

- If you are defining a Windows domain user, use the following format:
  `JOBUSR(domainName\user1)`
- If you are defining a Windows user in the *username@internet_domain* format, use the following format:
  `JOBUSR('administrator@mywindom.com')`

To specify the user name, you can also use the job-submit exit EQQUX001. This user name overrides the value specified for JOBUSR. In turn, the value specified for JOBUSR overrides the value specified for USRNAM in the USRREC statement.

OPNAME(*operation name***)**
The name of the operation to run. This keyword is required.

PARM*id***(***parameter***)**
The parameter to be used when running the script or command. You can specify a series of PARM*id* keywords, where *id* is a unique numeric value, including null and zero.

URL(*web service URL***)**
The URL of the web service to be called. This keyword is required.

WSNAME(*workstation_name***)**
The name of the workstation from which user name and password must be retrieved. User name and password can be specified in the statement, using the JOBUSR and JOBPWD keywords, or can be associated to a workstation with the USRREC statement. With the WSNAME keyword, you can specify a workstation where the user name and password are stored, which is different from the workstation where the job runs.

Examples

The following is an example of a JOBREC statement to call a web service that calculates the sum of the specified parameters:

```
//JOBREC
 JOBTYPE(/web service)
 PARM1(10)
 PARM2(20)
 JOBUSR(administrator)
 JOBPWD(YES)
 OPNAME(getSum)
 URL(http://manchester.testlab.uk.service.com:9467/Sum/services/Sum/wsdl/Sum.wsdl)
//END JOBREC
```

Syntax diagram for Java job type

The Java job type.

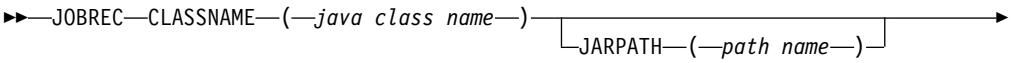
Syntax diagram for Java job type
Purpose

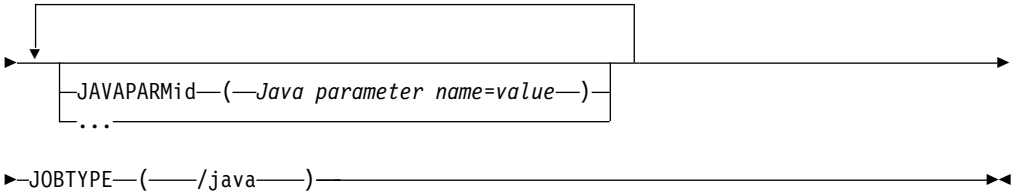
If you set JOBTYPE to (/java), you can run an existing Java class:

This section provides detailed information about the syntax diagram for the Java job type. The following syntax rules apply:

- Use the UNIX format for the value of the JARPATH keyword.

Format





Parameters

CLASSNAME(*java class name***)**
The name of the Java class to run. This keyword is required.

JARPATH(*path name***)**
The path to the jar file.

JAVAPARM*id***(***Java parameter name=value***)**
The Java parameter used by the Java class. You can define multiple parameters in the same job by specifying a series of JAVAPARM*id* keywords, where *id* is a unique numeric value, including null and zero. For example, you can define JAVAPARM1, JAVAPARM2, and so on.

Use the following format, which is case sensitive:

```
JAVAPARMid(Java parameter name=value)
```

Where:

Java parameter name
The name of the Java parameter.

value The value associated with the Java parameter.

JOBTYPE(/xajob/*access method name***|/j2ee/jms|/web service|/file transfer|/database|/java)**
To run a Java job, specify **/java**. For information about the other types of job, see the specific section for each job type.

/java Set this value to submit a Java job. Ensure that you have list permissions for the folder specified in the jarPath property.

Examples

The following is an example of a JOBREC statement to run a class with name mypackage.base:

```
//JOBREC
 JOBTYPE(/java)
 CLASSNAME(mypackage.base)
 JAVAPARM1(name=backup)
 JARPATH(c:/jarPath2)
//END JOBREC
```

Syntax diagram for xajob type

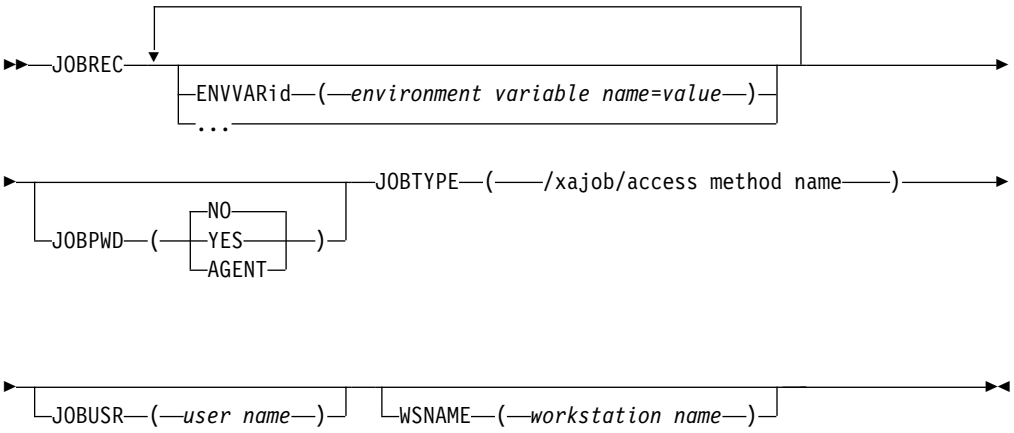
The xajob transfer job type.

Syntax diagram for xajob job type
Purpose

If you set JOBTYPE to /xajob/*access method name*, you can run a job on an extended agent:

This section provides detailed information about the syntax diagram for the xajob job type.

Format



Parameters

ENVVAR*id***(***environment variable name=value***)**
The environment variable that the task launcher must set before submitting the job. You can specify a series of ENVVAR*id* keywords, where *id* is a unique numeric value, including null and zero.

Use the following format, which is case sensitive:

```
ENVVARid(environment variable name=value)
```

Where:

environment variable name
The name of the environment variable.

value The value associated with the environment variable.

JOBPWD(YES|NO|AGENT)
Specifies if the user name entered in JOBUSR or set by using the job-submit exit EQQUX001 is associated with a password.

If you set JOBPWD to YES, IBM Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement (for details, see *IBM Workload Scheduler for z/OS: Customization and Tuning*).

If you set JOBPWD to AGENT, this means that the password is resolved locally on the agent. The password must have been defined on the agent by means of the param command. This feature is available independently of the operating system of the workstation.

Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to NO if the user works with UNIX workstations and if no password is required.

JOBTYPE(/xajob/*access method name***|/j2ee/jms|/web service|/file transfer|/database|/java)**

To run an xajob, specify **/xajob/***access method name*. For information about the other types of job, see the specific section for each job type. This keyword is required.

/xajob/*access method name*

The name of the access method hosted by the extended agent on which you want to submit the job. For example, for a job to be submitted on a SAP extended agent, this value might be /xajob/r3batch. If you set this keyword, the JOBCMD keyword value is ignored.

JOBUSR(*user name***)**

The name of the user submitting the job.

If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined (see the JOBPWD keyword).

- If you are defining a Windows domain user, use the following format:

  JOBUSR(*domainName*\user1)
- If you are defining a Windows user in the *username@internet_domain* format, use the following format:

  JOBUSR('administrator@mywindom.com')

To specify the user name, you can also use the job-submit exit EQQUX001. This user name overrides the value specified for JOBUSR. In turn, the value specified for JOBUSR overrides the value specified for USRNAM in the USRREC statement.

WSNAME(*workstation_name***)**

The name of the workstation from which user name and password must be retrieved. User name and password can be specified in the statement, using the JOBUSR and JOBPWD keywords, or can be associated to a workstation with the USRREC statement. With the WSNAME keyword, you can specify a workstation where the user name and password are stored, which is different from the workstation where the job runs.

Examples

The following is an example of a JOBREC statement to un a job using the r3batch access method. The content of the JOBTYPE keyword is case sensitive:

```
//JOBREC
JOBTYPE(/xajob/r3batch)
ENVVAR10(VAL=10)
//END JOBREC
-JOB SAPJOB -USER SAPUSER
```

Syntax diagram for J2EE jms job type

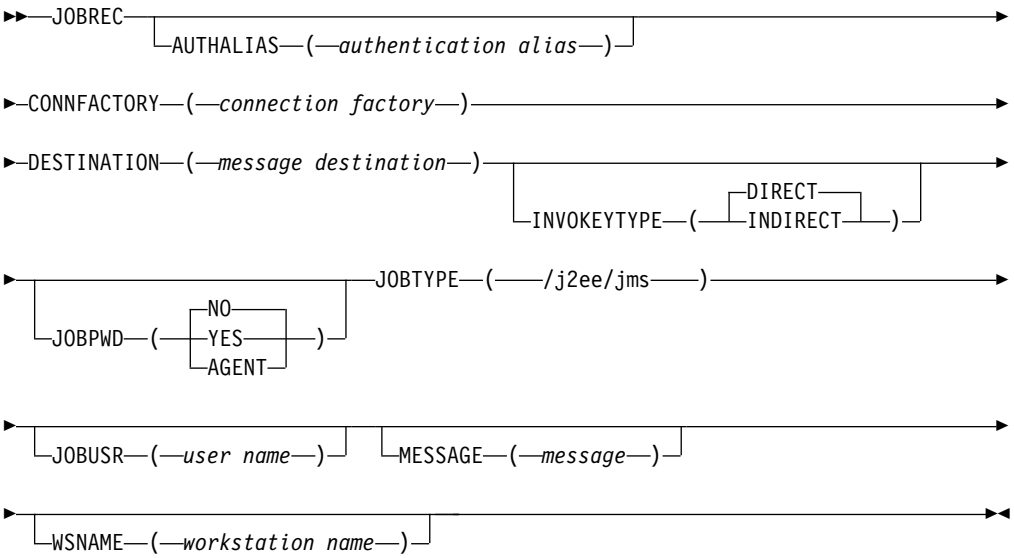
The J2EE jms job type.

Syntax diagram for J2EE jms job type
Purpose

If you set JOBTYPE to (/j2ee/jms), you can run an existing j2ee jms class:

This section provides detailed information about the syntax diagram for the j2ee jms job type.

Format



Parameters

AUTHALIAS(*authentication alias***)**
The Java Authentication and Authorization Service (JAAS) authentication alias. This keyword is valid only if you set INVOKEYTYPE to (DIRECT).

CONNFACTORY(*connection factory***)**
The administered object that a client uses to create a connection to the Java Messaging Service (JMS) provider. This keyword is required.

DESTINATION(*message destination***)**
The administered object that encapsulates the identity of a message destination, which is where messages are delivered and consumed. This keyword is required.

INVOKEYTYPE(INDIRECT|DIRECT)
The type of scheduling. This keyword is valid only if you set JOBTYPE to (/j2ee/jms). Allowed values are:

INDIRECT
IBM Workload Scheduler for z/OS uses an existing WebSphere Application Server scheduling infrastructure already configured on an external WebSphere Application Server.

DIRECT
IBM Workload Scheduler for z/OS immediately forwards the job to the components of the external WebSphere Application Server instance.

JOBPWD(YES|NO|AGENT)
Specifies if the user name entered in JOBUSR or set by using the job-submit exit EQQUX001 is associated with a password.

If you set JOBPWD to YES, IBM Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement (for details, see *IBM Workload Scheduler for z/OS: Customization and Tuning*).

If you set JOBPWD to AGENT, this means that the password is resolved locally on the agent. The password must have been defined on the agent by means of the param command. This feature is available independently of the operating system of the workstation.

Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to NO if the user works with UNIX workstations and if no password is required.

JOBTYPE(/xajob/*access method name***|/j2ee/jms|/web service|/file transfer|/database|/java)**

To run a j2ee jms job, specify **/j2ee/jms**. For information about the other types of job, see the specific section for each job type. This keyword is required.

/j2ee/jms

Set this value to submit a Java Messaging Service (JMS) job.

JOBUSR(*user name***)**

The user name for accessing WebSphere Application Server, if required.

If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined (see the JOBPWD keyword).

- If you are defining a Windows domain user, use the following format:

  ```
  JOBUSR(domainName\user1)
  ```

- If you are defining a Windows user in the *username@internet_domain* format, use the following format:

  ```
  JOBUSR('administrator@mywindom.com')
  ```

To specify the user name, you can also use the job-submit exit EQQUX001. This user name overrides the value specified for JOBUSR. In turn, the value specified for JOBUSR overrides the value specified for USRNAM in the USRREC statement.

MESSAGE(*message***)**

The object that is sent from one application to another.

WSNAME(*workstation_name***)**

The name of the workstation from which user name and password must be retrieved. User name and password can be specified in the statement, using the JOBUSR and JOBPWD keywords, or can be associated to a workstation with the USRREC statement. With the WSNAME keyword, you can specify a workstation where the user name and password are stored, which is different from the workstation where the job runs.

Examples

The following is an example of a JOBREC statement to submit a JMS job:

```
//JOBREC
JOBTYPE(/j2ee/jms)
INVOKEYTYPE(DIRECT)
CONNFACTORY(jms/MyCF)
DESTINATION(jms/MyQueue)
MESSAGE(This is my message)
JOBUSR(administrator)
JOBPWD(YES)
//END JOBREC
```

Syntax diagram for native job type

This section describes the native job type.

Syntax diagram for script and executable job type
Purpose

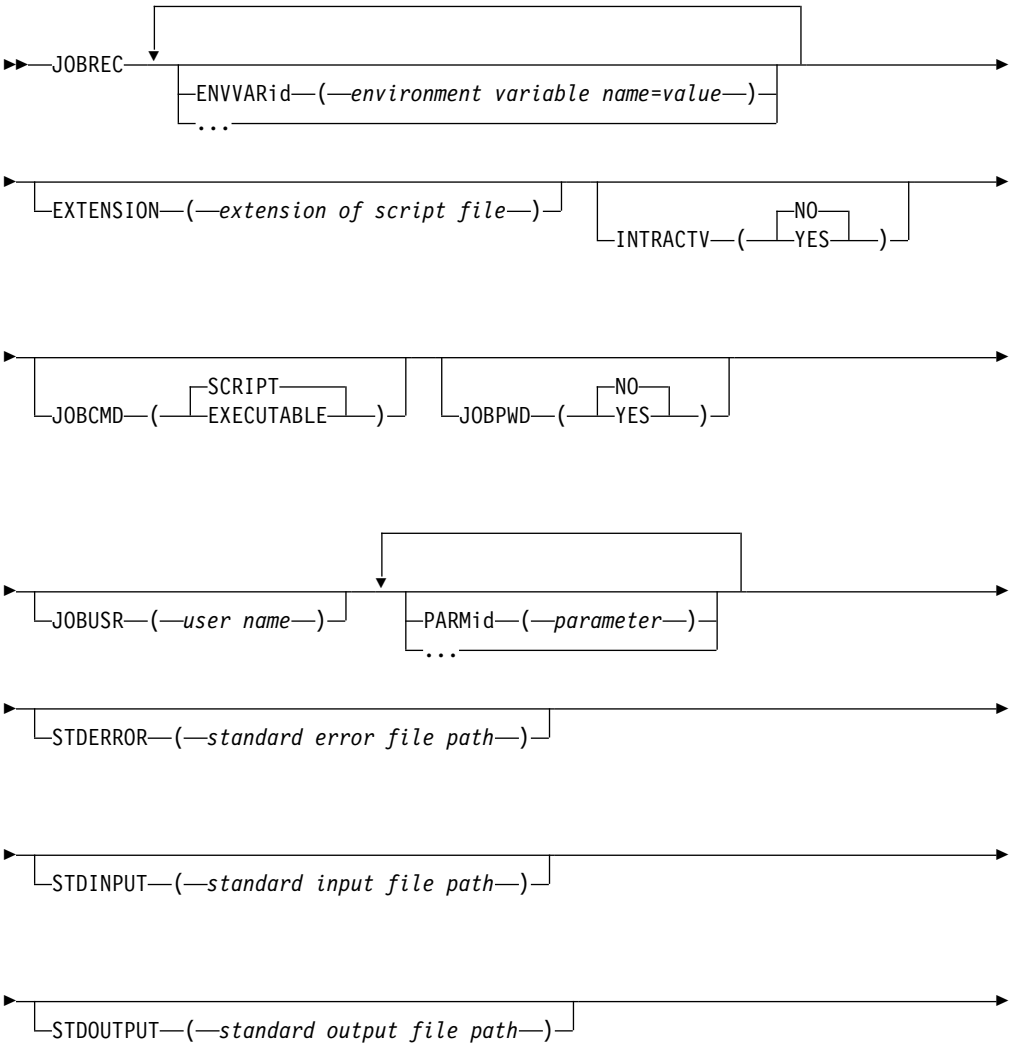
If you set JOBCMD to (script) or (executable), you can choose between the following native job types:

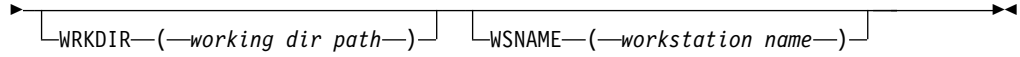
- Script
- Executable

This section provides detailed information about the syntax diagram for the script and executable job types. The following syntax rules apply:

- The JOBCMD keyword is required for executable and optional for script job types.

Format





Parameters

ENVVAR*id***(***environment variable name=value***)**
The environment variable that the task launcher must set before submitting the job. You can specify a series of ENVVAR*id* keywords, where *id* is a unique numeric value, including null and zero.

Use the following format, which is case sensitive:

```
ENVVARid(environment variable name=value)
```

Where:

environment variable name
The name of the environment variable.

value The value associated with the environment variable.

EXTENSION(*extension of script file***)**
The extension (bat, vbs, etc.) of the script file launched by the job. This keyword is valid only when JOBCMD is set to SCRIPT.

INTRACTV(YES|NO)
Specifies if a Windows job runs interactively on the Windows desktop. This keyword is valid only for jobs running on Windows IBM Workload Scheduler for z/OS Agent workstations.

JOBCMD(EXECUTABLE|SCRIPT)
The content of the job definition member. Valid values are:

SCRIPT
The job definition member contains the command or script to be run. You can add the EXTENSION keyword to specify the file extension.

EXECUTABLE
The job definition member invokes the job that is to be run, which is stored locally on the IBM Workload Scheduler for z/OS Agent workstation. If you did not set any value for the WRKDIR keyword, you must specify the fully qualified path to the job.

JOBPWD(YES|NO)
Specifies that the user name set in JOBUSR or by using the job-submit exit EQQUX001 is associated with a password. If you set JOBPWD to YES, IBM Workload Scheduler for z/OS searches for the user password in the USRPSW keyword of the USRREC statement (for details, see *IBM Workload Scheduler for z/OS: Customization and Tuning*).

Typically, the password is required for users who schedule jobs to run on Windows workstations. Set JOBPWD to NO if the user works with UNIX workstations.

JOBUSR(*user name***)**
The user name for running the script or executable, if required.

If the user schedules jobs to run on Windows workstations, ensure that a user password is also defined (see the JOBPWD keyword).

- If you are defining a Windows domain user, use the following format:

```
JOBUSR(domainName\user1)
```

- If you are defining a Windows user in the *username@internet_domain* format, use the following format:

```
JOBUSR('administrator@mywindom.com')
```

To specify the user name, you can also use the job-submit exit EQQUX001. This user name overrides the value specified for JOBUSR. In turn, the value specified for JOBUSR overrides the value specified for USRNAM in the USRREC statement.

PARM*id***(***parameter***)**
The parameter to be used when running the script or command. You can specify a series of PARM*id* keywords, where *id* is a unique numeric value, including null and zero.

STDERROR(*standard error file path***)**
The fully qualified name of the file where the standard error must be redirected.

STDINPUT(*standard input file path***)**
The fully qualified name of the file to be used as standard input.

STDOUTPUT(*standard output file path***)**
The fully qualified name of the file to be used as standard output.

WRKDIR(*working dir path***)**
The fully qualified path to the directory where the script or command is to be run.

WSNAME(*workstation_name***)**
The name of the workstation from which user name and password must be retrieved. User name and password can be specified in the statement, using the JOBUSR and JOBPWD keywords, or can be associated to a workstation with the USRREC statement. With the WSNAME keyword, you can specify a workstation where the user name and password are stored, which is different from the workstation where the job runs.

Examples

The following is an example of a JOBREC statement to invoke the script ALLOCATE.BAT stored locally on the IBM Workload Scheduler for z/OS Agent workstation:

```
//JOBREC
WRKDIR(C:\)
STDERROR(D:\MYDIR\ERROR.TXT)
STDINPUT(D:\MYDIR\INPUT.TXT)
STDOUTPUT(D:\MYDIR\OUTPUT.TXT)
STDOUTPUT(D:\MYDIR\NEWT.TXT)
JOBCMD(EXECUTABLE)
PARM(300)
PARM2(ELEMENT)
//END JOBREC
D:\MYDIR\ALLOCATE.BAT
```

The following is an example of a JOBREC statement to run the command DIR on the directory C:\PROGRAM on the IBM Workload Scheduler for z/OS Agent workstation:

```
//JOBREC
WRKDIR(C:\PROGRAM)
JOBCMD(SCRIPT)
EXTENSION(vbs)
//END JOBREC
DIR
```

Using variable substitution

The variable substitution mechanism that is used in IBM Workload Scheduler for z/OS native job types works the same way when scheduling jobs on IBM Workload Scheduler for z/OS Agents. The variable substitution mechanism changes when you define job types with advanced options, such as Java jobs that run a Java class, or file transfer jobs that transfer files to and from a server, to name a few. These types of jobs are created and edited using the Dynamic Workload Console.

When you define a variable for job types with advanced options, including those defined in a JOBREC using the JOBTYPE keyword, the variable must be preceded by the $ (dollar) sign and enclosed in {} (curly brackets), as follows:

`${variable_name}`

If you want to nest variables, use the following syntax:

`${variable_1_name ${variable_2_name}}`

In this case, variable_2 is resolved first and provides the value for variable_1.

With job types with advanced options you also have the option to define and use variables locally on the IBM Workload Scheduler for z/OS Agents. To define a variable on the agent, use the param command. After defining the variable, to add it within a job definition so that it is resolved locally on the agent at run time, use the following syntax:

`${Agent:variable_name}`

User passwords can be similarly parameterized for local resolution. See Chapter 4, “Defining variables and passwords for local resolution,” on page 33 for details.

Dependent variables and promptable variables are not supported in job types with advanced options.

For more information about variable substitution, see *IBM Workload Scheduler for z/OS: Managing the Workload*.

Defining file dependencies to check for file changes

You can use the **filewatch** utility to check for file system changes on files and directories, for example, when you want to make sure that a file exists before running a job that processes that file. By defining a job that runs the **filewatch** utility, you can implement file dependency, that is, a relationship between a file and an operation in which specific activity on the file determines the starting of the operation.

Note: The **filewatch** utility is not available for Limited Fault-Tolerant Agent for IBM i systems. For these systems, see the *Limited Fault-Tolerant Agent IBM i* manual.

To run the **filewatch** utility you can use variable substitution, which occurs when:

- The controller submits the job associated to a JOBLIB member. For details about the scheduler job tailoring, see *Managing the Workload*.
- The IBM Workload Scheduler for z/OS Agent submits the job. For details about submitting a job, see "Defining a job in the JOBLIB data set" on page 65.

Syntax

filewatch -v | -u | -?

filewatch -c[ondition] ***condval*** **-f[ilename]** *file_path* **-dea[dline]** *deadline*
[**-i[nterval]** *interval*]
[**-dat[abase]** *log_extension*]
[**-r[eturncode]** *rc*]
[**-t[race]** *trace level*]

The arguments are not positional. You can use an abbreviated format for all the arguments. Generally you can truncate the arguments to any position following the first character, except for `deadline` and `database` that require at least three characters.

-v Returns the command version and exits.

-u Returns command usage information and exits.

-? Same as **-u**

-condition
The condition to be checked. Valid values are:

wcr | waitCreated
Waits until the file exists. If the file already exists, **filewatch** exits immediately. If `-filename` argument specifies a directory, the process waits until the directory exists and contains a new file.

wmr | waitModificationRunning
Waits until the file size or modification time changes. If `-filename` argument specifies a directory, the process waits until the size or earlier file modification time changes, when a file is created, modified, or deleted.

wmc | waitModificationCompleted
Checks that the file size or modification time stopped changing, meaning that **filewatch** waits for three search intervals without any change. If `-filename` argument specifies a directory, the process checks the size or the earlier file modification time change, for example if the number of directory files and the earlier file modification time does not change within three search intervals.

wmrc | waitModificationRunningCompleted
Waits until the file size or modification time changes and stops changing, meaning that, after the first change, **filewatch** waits for three search intervals without any further change. If `-filename` argument specifies a directory, the process checks the size or the earlier file modification time change, for example if the number of directory files and the earlier file modification time does not change within three search intervals.

wdl | waitDelete
Stops running when the file is deleted. If `-filename` argument specifies a directory, the process waits until a file is deleted from the directory.

-filename
The file path to be processed. You can embed blank or special characters, by using double quotation marks. Wildcard characters are not supported. To include more files in the monitoring process, you can store those files in a specific directory and use a file path specifying that directory. When filewatch is used on a Windows server, and the file path specifies a directory, the path name must not include a trailing forward slash "/" character.

-deadline
The deadline period, expressed in seconds. The allowed formats are:

- An integer in the range 0 to 31536000 (the upper value corresponds to one year). To have **filewatch** performing an indefinite loop, specify 0.
- *hh:mm:ss*, in the range 00:00:01 to 24:00:00, to select a time within the same day when **filewatch** started.

-interval
The file search interval, expressed in seconds. Specify an integer in the range:

- 5–3600 when specifying **wcr** or **wdl** as condition value.
- 30–3600 otherwise.

The default is 60.

-database
Optional extension of the log database, that is the database where **filewatch** stores the file status. If you specify this value, **filewatch** updates the *TWA_home*/TWS/filewatchdb.*log_extension* database, otherwise *TWA_home*/TWS/filewatchdb is updated.

For details about the log database, see "Maintaining the filewatch log database" on page 92.

-returncode
The exit return code, if the condition is not successfully checked by the deadline. Specify an integer in the range 0 to 256. The returncode value is ignored if you specify 0 as deadline value. The default is 4.

-trace Trace level for internal logging and traces. Possible values are:

0 To receive error messages only.

1 Indicates the *fine* level, to receive the most important messages with the lowest volume.

2 Indicates the *finer* level, to activate entry and exit traces.

3 Indicates the *finest* level, to receive the most detailed tracing output.

The default value is 0.

You find the trace output in the log of the job that run **filewatch**.

Maintaining the filewatch log database

The log database is used to store **filewatch** activity data. This log is located in the *TWA_home*/TWS directory. The default name is filewatchdb.

When processing a file, **filewatch** uses it to store records with information about:

- File name.
- File status. The allowed values are Exist, Created, Running, Completed, RunningCompleted, or Deleted.
- Date and time of file creation at the entry update time.
- Date and time of the last file modification at the entry update time.
- Date and time of the last entry update.

Note: The log contains only one entry for each file.

To maintain the log database, use the **filewatchdb** utility.

Syntax

filewatchdb -v | -u | -?

filewatchdb -c[ondition] *condval*
[-f[ilename] *file_path*
[-dat[abase] *log_extension*]

The arguments are not positional.

Arguments

-v Returns the command version and exits.

-u Returns command usage information and exits.

-? Same as **-u**

-condition
The condition to be checked. Valid values are:

bld | build
Rebuild the log database.

dlt | delete
Delete the log database record corresponding to the filename argument value.

gls | getLastStatusChange
Gets and returns the log database record corresponding to the filename argument value.

-filename
Use this argument to delete the entry corresponding to the specified value. You can embed blank or special characters, by using double quotation marks. Use * as wildcard character. This argument is required if you specify dlt (delete) or gls (getLastStatusChange) as condition value.

-database
Optional extension of the log database to be accessed. If you specify this value, **filewatchdb** accesses the *TWA_home*/TWS/filewatchdb.*log_extension* database, otherwise *TWA_home*/TWS/filewatchdb is accessed.

Checking for file changes with filemonitor

You can use the **filemonitor** utility to monitor for file changes (files that were either created or modified) within a time interval. This could be useful when, for example, you want to make sure that a file exists before running a job that processes that file. By defining a job that runs the **filemonitor** utility, you can implement file dependency, that is, a relationship between a file and an operation in which specific activity on the file determines the starting of the operation.

Note:

1. The **filemonitor** utility is available on the z-centric agents with IBM Workload Scheduler V9.4, Fix Pack 1 installed.
2. To issue **filemonitor** from the command line, ensure that you set up the environment with the following command from `<TWA_home>`:

   On UNIX and Linux

   ```
   ../twa_env.sh
   ```

   On Windows

   ```
   twa_env.cmd
   ```
3. To issue **filemonitor** from a job, use the command `filemonitorlauncher`. The same arguments valid for **filemonitor** apply to `filemonitorlauncher`.

Syntax

filemonitor -V | -U

filemonitor -path *path_to_monitor* [**-exitOnPathToMonitorNotFound**]
-event {**fileCreated** | **fileModified**} [**-modificationCompletedTime** *seconds*]
[**-repositoryName** *repository_name*]
[**-repositoryPath** *repository_path*]
[**-recursive**]
[**-outputFile** *output_filename*]
[**-scanInterval** *scan_interval*]
[**-maxEventsThreshold** *max_events*]
[**-minFileSize** *min_file_size*]
[**-timeout** *seconds*]
[**-preserveEventsOnDelete** *seconds*]

filemonitor -reset
[**-repositoryName** *repository_name*]
[**-repositoryPath** *repository_path* - **generateEventsOnFirstScan**]

Arguments

Note: If you set the same argument more than once, the last value specified is applied and no error message is reported.

-V Displays the command version and exits.

-U Displays command usage information and exits.

-path *path_to_monitor*

The path where the files to be processed are located. You can specify blank or special characters within double quotation marks. Wildcard characters are supported. To include more files in the monitoring process, store all the files in the directory set with the **-path** argument. Paths containing spaces

must be enclosed in double quotes. Universal Naming Convention (UNC) paths are also supported with the following syntax types:

- `\\server_name\share_name\directory_name\...`
- `\\?\UNC\server_name\share_name\directory_name`
- `\\?\path_name`

[-exitOnPathToMonitorNotFound]
Optionally, specify this argument to have the command exit if the specified path is not found.

-event {fileCreated | fileModified} [-modificationCompletedTime *seconds***]**
The event type to be monitored. This argument is required when you specify -**path**.

Supported types are **fileCreated** and **fileModified**. For both types, you can set the **-modificationCompletedTime**, which is the time interval, in seconds, that is used to determine when the event is sent.

-event fileCreated
As soon as the file is created, the event, `FileCreated`, is sent.

-event fileModified
As soon as the file is modified, the event, `ModificationCompleted`, is sent.

-event fileCreated -modificationCompletedTime *seconds*
When a file is created, the event is not sent immediately, but only after the interval of time specified by -**modificationCompletedTime***seconds* has elapsed, and during which no subsequent changes were made to the file, which includes the file being deleted and recreated with the same name.

-event fileModified -modificationCompletedTime *seconds*
When a file is modified, the event is not sent immediately, but only after the interval of time specified by -**modificationCompletedTime***seconds* has elapsed, and during which no subsequent changes were made to the file.

-repositoryName *repository_name*

Optionally, specify a database where to log the status of the retrieved files. The default is `filemonitor.db`.

-repositoryPath *repository_path*
The path to the **filemonitor** database. The default is `<TWA_home>/TWS/stdlist/JM/filemonitor`. Paths containing spaces must be enclosed within double quotes. Wildcard characters are not allowed.

-generateEventsOnFirstScan
All the files retrieved during the first scan performed by **filemonitor** are considered as created or modified, and can generate events. This argument is available only if you specify the **repositoryPath** argument.

-recursive
Optional. The monitoring process includes subdirectories.

-outputFile *output_filename*
Optional. An output file where to store the retrieved events. Ensure that the directory where the output file is to be created is already existing. The command output is also printed to standard output and stored in the job properties, if you launch the **filemonitor** command from a job. Wildcard characters are not allowed.

-scanInterval *scan_interval*
Optional. A period of time in seconds between two consecutive checks on the files being created or modified. The default value is 300 seconds. The supported range is 1-3600 seconds.

-maxEventsThreshold *max_events*
Optional. The maximum number of events to be returned. The default value is **1**. If you specify **all**, all events are returned.

-minFileSize *min_file_size*
Optional. The minimum size in bytes that files must reach to be included in the scan. The default value is **0**.

-timeout *seconds*
Optional. The maximum time, in seconds, that **filemonitor** runs. If you do not specify this parameter, **filemonitor** runs indefinitely.

-preserveEventsOnDelete
Optional. Returns events on the specified file, also if the file was deleted in the meantime. If you do not specify this argument, when a file is deleted all events preceding the file deletion, if any, are discarded.

-reset Resets the information collected. With this argument you can optionally specify a **-repositoryName** and **-repositoryPath**.

Configuring trace properties for filemonitor

To configure the trace properties for **filemonitor**, edit the [FileMonitor.Logging] section in the `<TWA_Home>/TWS/ITA/cpa/config/FileMonitor.ini` file, and restart the **filemonitor** utility.

The section containing the trace properties is named:
`[FileMonitor.Logging.cclog]`

FileMonitor.trhd.fileName
The name of the trace file.

FileMonitor.trhd.maxFileBytes
The maximum size that the trace file can reach. The default is 1024000 bytes.

FileMonitor.trhd.maxFiles
The maximum number of trace files that can be stored. The default is 3.

FileMonitor.trfl.level
Determines the type of trace messages that are logged. Change this value to trace more or fewer events, as appropriate, or on request from Software Support. Valid values are:

DEBUG_MAX
Maximum tracing. Every trace message in the code is written to the trace logs.

INFO All *informational*, *warning*, *error* and *critical* trace messages are written to the trace. The default value.

WARNING
All *warning*, *error* and *critical* trace messages are written to the trace.

ERROR
All *error* and *critical* trace messages are written to the trace.

CRITICAL
Only messages which cause the agent to stop are written to the trace.

The output trace (by default, `FileMonitor_trace.log`) is provided in XML format, and is located in `<TWA_Home>/TWS/stdlist/JM`.

Return Codes

0 The operation completed successfully.

4 **Filemonitor** stopped running, because timeout expired. No results were returned.

-1 An error occurred. Search the trace log (by default, `FileMonitor_trace.log`) for additional details.

Comments

If one or more files were created or modified in between subsequent invocations, the modifications are detected. However, files already detected in a previous run are not listed again in subsequent invocations. Wildcard characters are supported in both file names and directory names.

Example

In the following example, the **filemonitor** command checks every 2 minutes for all files created in the `C:\temp\logs` path and having a minimum size greater than 1024 bytes. The check is performed on all sub folders and the results are stored in `C:\backup\logs\reports.txt`:

```
filemonitor -path "C:\temp\logs" -event fileCreated -recursive
-outputFile "C:\backup\logs\reports.txt"
-scanInterval 120 -maxEventsThreshold all -minFileSize 1024
```

Promoting critical jobs

An overview of the mechanism that is behind the promotion of critical jobs, or of their predecessors, running on IBM Workload Scheduler for z/OS Agents, including pools and dynamic pools.

The promotion of a critical job is the process whereby the execution of a job in the critical path, that is about to miss its deadline, is provided with more resources. Thus, it is redirected to a more powerful or faster dynamic agent, to idle workstations in a pool or dynamic pool, to an agent allocated to the execution of critical work.

As soon as IBM Workload Scheduler for z/OS determines that a job flagged as critical is not going to meet its deadline, it automatically updates the definition of the job, and of as many of its predecessors as necessary (that is, all the jobs in the critical path of the critical job), with the `tws.job.promoted` and `tws.job.resourcesForPromoted` promotion variables. These variables set the dynamic workload broker component to perform the load balancing required to guarantee that the job is run on time.

If the critical job is to be processed in a dynamic pool, the variables must be added manually to the job definition. See *"Promoting jobs scheduled on dynamic pools"* in the *IBM Workload Scheduler: User's Guide and Reference* for details.

To control the priority allocation of system resources to the jobs in the critical network that must be promoted, you must configure the PromotedNice or PromotedPriority property of the `native job launcher` section in the `JobManager.ini` file of your IBM Workload Scheduler for z/OS Agents.

When a native job is promoted, the `TWS_PROMOTED_JOB` environment variable is automatically added with value set to YES in the job definition. You can possibly take advantage of the value (YES or NO) of this variable in the script/program run by the native job, to implement a prioritization mechanism.

Killing IBM Workload Scheduler for z/OS Agent operations

The `kill` action ends an operation in the current plan that has already started. You can kill a running IBM Workload Scheduler for z/OS Agent operation from any of the following interfaces:

- "ISPF"
- "PIF"
- "BCIT" on page 98
- "OCL" on page 98
- "Dynamic Workload Console" on page 98

This action can be taken only on `STARTED` operations that are in the `EXECUTING` status, so that their job ID is known.

The operation number (except for the OCL instructions where the application number is the required parameter) is required to identify the operation that is to be killed.

ISPF

About this task

To kill an IBM Workload Scheduler for z/OS Agent operation, perform the following steps:

1. Select option 3 (OPERATIONS) from the Modifying the Current Plan panel (fast path 5.3).
2. Specify the list criteria for the operation you want to kill.

   You see the Modifying Operations in the Current Plan panel (EQQMOPRL). This panel lists the operations in the current plan that fulfill the list criteria you specified.
3. Enter the `K` (Kill) row command next to the row that lists the operation.

The IBM Workload Scheduler for z/OS Agent notifies the controller when the operation is killed.

PIF

To kill an IBM Workload Scheduler for z/OS Agent operation, use the following request:

```
MODIFY CPOP OPNO=operation_number OPCMD=KJ
```

For more detailed information, refer to *Driving IBM Workload Scheduler for z/OS*.

BCIT

To kill an IBM Workload Scheduler for z/OS Agent operation, use the following instruction:

```
ACTION=MODIFY,RESOURCE=CPOP,OPNO=operation_number,OPCMD=KJ
```

For more detailed information, refer to *Driving IBM Workload Scheduler for z/OS*.

OCL

To kill an IBM Workload Scheduler for z/OS Agent operation, use the following instruction:

```
KILLJOB APPL(application_number) OPNO(operation_number)
```

For more detailed information, refer to *Driving IBM Workload Scheduler for z/OS*.

Dynamic Workload Console

About this task

To kill an IBM Workload Scheduler for z/OS Agent operation, perform the following steps:

1. Click **IBM Workload Scheduler for z/OS→Workload→Monitor→Monitor Jobs**.
2. Select **All Jobs in plan** or another predefined task name.
3. Choose an engine name, or specify connection properties, and click **OK**.
4. Select a job and click **More Actions→Kill**.

Chapter 9. Administering an IBM i dynamic environment

On overview on how to administer the IBM Workload Scheduler IBM i dynamic environment.

To begin scheduling jobs with advanced options on IBM i agents, the agents must be configured.

Scheduling on IBM i

About this task

When scheduling a job on IBM i systems, the job launches a native command that can be either a system or a user command. The native command consists of `SBMJOB` system command, which launches a batch job. The native command starts one or more batch jobs. The batch jobs can be monitored only if they are started by the native command. The IBM i Agent Monitor can monitor a maximum of 130 batch jobs.

You can specify the name of the queue where the monitoring agent component runs by using the **`MonitorQueueName`** property in the native job launcher section of the `JobManager.ini` file. If you do not specify this property, the default queue (QBATCH) is used.

For more information, see the section about configuring properties of the native job launcher [NativeJobLauncher] in *Administration Guide*.

You can use the Dynamic Workload Console and **conman showjobs** command line to check whether an IBM i job is waiting for a reply to a message. An IBM i job that is waiting for a message reply is in the SUSP (suspended) status. This status indicates that the job is running while waiting for input. When the input is received, the job status changes to EXEC (executing).

For more information about job statuses, see the section about status description and mapping for distributed jobs in *Dynamic Workload Console User's Guide*.

If an IBM i job is waiting for a reply, you can view the message text and the related reply. This information is written into the correspondent IBM Workload Scheduler job log so that the IBM Workload Scheduler operator knows the exact message the IBM i job is waiting for.

When an IBM i job is waiting for a reply to a message, you can reply to the message directly from the **Monitor Workload** of the Dynamic Workload Console. The job in SUSP (suspended) status requires your attention on additional information to be displayed. A pop-up window shows the message that is waiting for your reply. Reply to the message in the pop-up window, then select one of the following actions:

Forward action
To forward your reply. A message in the pop-up window confirms that your reply was sent successfully.

Cancel action
To cancel your reply. The pop-up window is closed.

Note: For a correct display of the pop-up window that shows the message waiting for your reply, your dynamic domain manager must be at version 9.3.0.2.

You can even define standard rules to automate the reply to the waiting messages. When defining an IBM i job, by using the Dynamic Workload Console or the **composer** command line, specify the list of messages for which you want to set up an automated reply. For each message, specify:

Message Id
The message identifier.

Message Text
The message text.

Message Reply
The automated reply that you want to define.

For more information, see the section about defining an IBM i job in *User's Guide and Reference*.

The agent joblog and TWSASPOOLS environment variable

About this task

By default, all information about the running of jobs is stored in the agent joblog. Most of this information usually consists of spool files. To select the spool file types that you want included in the agent joblog, use the **TWSASPOOLS** system variable, which works at IBM i agent level for any job to be submitted.

The **TWSASPOOLS** system variable forces the IBM i agent to either ignore all spool files or include one or more of them.

On the IBM i agent, create a new system level environment variable named TWSASPOOLS and set it to a list of the spool file types that are to be included. The list must begin with the **SPOOLS**: token.

For example, to force the IBM i agent to ignore all spool files, create the TWSASPOOLS variable as follows.

```
ADDENVVAR ENVVAR(TWSASPOOLS) VALUE(SPOOLS:) LEVEL(*SYS)
```

where the list after the SPOOL: token is empty. In this case, any agent joblog report for the IBM i agent is limited to the activity report that the Agent Monitor produces to trace its submission and monitoring action, and to the IBM i joblog of the Agent Monitor, which is always added at the end of the agent joblog.

To allow the IBM i agent to include only the QPRINT and the QPJOBLOG spool file types, that is, any spool files produced by **printf** instructions inside any ILE-C program and any produced joblog, create the TWSASPOOLS as follows:

```
ADDENVVAR ENVVAR(TWSASPOOLS) VALUE('SPOOLS: QPRINT QPJOBLOG') LEVEL(*SYS)
```

If the TWSASPOOLS variable already exists, change it as follows:

```
CHGENVVAR ENVVAR(TWSASPOOLS) VALUE('SPOOLS: QPRINT QPJOBLOG') LEVEL(*SYS)
```

If any VALUE parameter is set to an incorrect string, the IBM i agent ignores the TWSASPOOLS environment variable option. You can create and change the TWSASPOOLS environment variable while with the IBM i agent active, but no workload activity must be running.

Child job monitoring on IBM i agents

About this task

When you submit a command on an IBM i agent, the command might start one or more batch jobs. The IBM i agent monitors these batch jobs, which are referred to as child jobs.

When searching and monitoring any child jobs that are started, the IBM i agent uses a high percentage of its processing time.

If you know that your job scheduling does not start any child jobs or you have no interest in monitoring child jobs, you can instruct the IBM i agent to not search and monitor child jobs, and hence improve the performance of the agent.

You can exclude child job monitoring either at the agent level for all the commands or at the job definition level for a single command. If you want child job monitoring only for some specific submitted commands, you can set this option at the job definition level for a single command.

You can perform one or both of the following procedures to exclude or include child job monitoring:

Exclude child jobs from job monitoring at the agent level

By default child jobs are monitored. You can exclude child jobs from job monitoring for all submitted commands by creating the `TWS_NOCHILDS` system environment variable using the following IBM i system command:

```
ADDENVVAR ENVVAR(TWS_NOCHILDS) LEVEL(*SYS)
```

If the IBM i agent finds the `TWS_NOCHILDS` on the IBM i system, it does not monitor child jobs for any submitted command.

Exclude or include child jobs from job monitoring at the job definition level

You can exclude or include child jobs from job monitoring for a specific job by using `:NOCHILDS` or `:CHILDS` as ending tokens of the command string for the specific command.

- If you add the `:NOCHILDS` end token at the end of the native command you are submitting, the IBM i agent ignores any child jobs that are started by the command.
- If you add the `:CHILDS` end token at the end of the command you are submitting, the IBM i agent finds and monitors all the child jobs that are started by the command.

Note: The setting at job definition level overrides the setting at agent level.

The **`SBMJOB`** system command, when submitted, always starts a batch job. Do not try to exclude the job monitoring, because if the IBM i agent finds the `SBMJOB` command in the job definition , it removes and ignores the `:CHILDS` or `:NOCHILDS` ending token in the job definition and also ignores the setting of the `TWS_NOCHILDS` system variable.

Examples

To monitor any child jobs that are started when the `PAYROLL` program is run, define the following command in the job definition:

- If the `TWS_NOCHILDS` system variable is defined on the IBM i system:

```
CALL PGM(MYLIB/PAYROLL) :CHILDS
```

- If the TWS_NOCHILDS system variable is not defined on the IBM i system:

```
CALL PGM(MYLIB/PAYROLL)
```

To not monitor any child jobs that are started when MYSCHEDULE program is run, define the following command in the job definition:

- If the TWS_NOCHILDS system variable is not defined on the IBM i system:

```
CALL PGM(MYLIB/MYSCHEDULE) :NOCHILDS
```

- If the TWS_NOCHILDS system variable is defined on the IBM i system:

```
CALL PGM(MYLIB/MYSCHEDULE)
```

Note: The SBMJOB command always starts a child jobs. The IBM i agent monitors the child job even if you define a SBMJOB command as in the following job definition:

```
SBMJOB CMD(CALL PGM(MYLIB/USERPGM)) :NOCHILDS
```

Information about child job monitoring on IBM i agent joblog

About this task

If you include the child job monitoring on IBM i agent as described in the section "Child job monitoring on IBM i agents" on page 101, you can see information related to child job monitoring on the IBM i agent joblog.

Otherwise the information about the child job monitoring is not present in the joblog.

Examples

This example shows the information related to child job monitoring included at job level for the IBMI35C job on the agent D400. The joblog shows the final status of 4 child jobs started from the submitted native command:

```
============================================================
=                        JOBLOG HEADER
=
= Occurrence name : AS400PGMCHILD
= Occurrence IA   : 2012/10/29 21:23
= Job name        : IBMI35C
= Work station    : D400
= Operation number: 002
= Start time      : 2012/10/31 10:23
= End time        : 2012/10/31 10:23
= Process ID      : 375602
= Duration        : 0.00.10
= Status          : ERROR
= Return code     : 30
= Hostname        : NC117025.ROMELAB.IT.IBM.COM
=
============================================================

The Dynamic Agent submitter-monitor job is qualified as:
   JobName=DYNAMICMON JobUser=ZOSDYN JobNumber=375602
Here follows the user command string
   <CALL PGM(MINERMA/SBM4JOBS) :CHILDS>
2012/10/31 10:23:27.893 - Dynamic Agent job submitted the User Command
   CALL PGM(MINERMA/SBM4JOBS)
The FOLLOWING 4 JOBS STARTED under the submitted User Command
  JobName=ZOSDYN     JobUser=ZOSDYN     JobNumber=375607
  JobName=ZOSDYN     JobUser=ZOSDYN     JobNumber=375609
  JobName=ZOSDYN     JobUser=ZOSDYN     JobNumber=375611
```

```
  JobName=ZOSDYN     JobUser=ZOSDYN     JobNumber=375612
Message CPF1241 (Success) received on MsgQueue ZOSDYN    QUSRSYS
   for the job ZOSDYN    ZOSDYN    375607
Message CPF1240 (Abend) received on MsgQueue ZOSDYN    QUSRSYS
   for the job ZOSDYN    ZOSDYN    375609
Message CPF1241 (Success) received on MsgQueue ZOSDYN    QUSRSYS
   for the job ZOSDYN    ZOSDYN    375611
Message CPF1241 (Success) received on MsgQueue ZOSDYN    QUSRSYS
   for the job ZOSDYN    ZOSDYN    375612
*** END codes gathered by the Monitor job ***
 > END Status Code (Status): 0
 > PROGRAM Return Code (Prc): 0
 > USER Return Code (Urc): 0
   Urc was retrieved through SYSAPI

2012/10/31 10:23:37.904 - Dynamic Agent job ended monitoring the User Command
*** Return Code for submitted Command is 30 ***
*** job(s) started under User Command abended ***
```

This example shows the joblog for the IBMI35C job on the agent P400 when child job monitoring is excluded at job level:

```
================================================================
=                         JOBLOG HEADER
=
= Occurrence name : AS400PGMNOCHILD
= Occurrence IA   : 2012/11/07 03:49
= Job name        : IBMI35NC
= Work station    : P400
= Operation number: 002
= Start time      : 2012/11/07 16:48
= End time        : 2012/11/07 16:48
= Process ID      : 491329
= Duration        : 0.00.05
= Status          : COMPLETED
= Return code     : 0
= Hostname        : NC117025.ROMELAB.IT.IBM.COM
=
================================================================

The Dynamic Agent submitter-monitor job is qualified as:
   JobName=DYNAMICMON JobUser=ZOSDYN JobNumber=491329
Here follows the user command string
   <CALL PGM(MINERMA/SBM4JOBS) :NOCHILDS>
2012/11/07 16:48:50.631 - Dynamic Agent job submitted the User Command
   CALL PGM(MINERMA/SBM4JOBS)
As per user choice, NO job started under the submitted command will be monitored
*** END codes gathered by the Monitor job ***
 > END Status Code (Status): 0
 > PROGRAM Return Code (Prc): 0
 > USER Return Code (Urc): 0
   Urc was retrieved through SYSAPI

2012/11/07 16:48:55.685 - Dynamic Agent job ended monitoring the User Command
*** Return Code for submitted Command is 0 ***
*** User Command ended successfully ***
```

The agent return code retrieval

About this task

The IBM i programming model was originally based on an early object orientation model in which programs communicated through message passing, rather than using return codes. The introduction of the Integrated Language Programming

(ILE) model lead to the definitions of common areas to exchange data as return codes in the same job environment: the user return codes and the system end codes.

For information about user return codes, see "Controlling the job environment with the user return code."

When the IBM i agent verifies that a submitted command or job is completed, it assigns a return code to the job based on the job status of the completed job. The return code is set depending on the completion message of the command or job. If the command or job completes successfully, the return code is set to 0. If the command or job does not complete successfully, the return code is set to the value of the severity of the message related to the exception that caused the abnormal end of the job. The IBM i agent can also set the return code to the value of the user return code when it is returned by the submitted command. If retrieved, the user return code is used as the value to set the return code.

The return code value assigned to the job is included in the IBM i agent joblog for the job and sent back to the scheduler user interface (WEB UI or z/OS ISPF panels) as return code, for compatibility reasons with agents on other operating systems.

Controlling the job environment with the user return code

About this task

With the introduction of the IBM i ILE model, it is possible to retrieve a value returned by a called program inside the same job.

When the Agent Monitor verifies that a submitted command is completed, it retrieves the following end of job codes using an IBM i System API:

End status code or <Status> (0 if successful)

It indicates if the system issued a controlled cancellation of the job. Possible values are:

1 the subsystem or the job itself is canceled.

0 the subsystem or the job itself is not canceled.

blank the job is not running.

Program return code or <Prc> (0000 if successful)

It specifies the completion code of the last program (such as a data file utility program, or an RPG or COBOL program, invoked by the job).

If the job includes no program, the program return code is 0.

User return code or <Urc> (0000 if successful)

It specifies the user-defined return code set by ILE high-level language constructs. For example, the return code of a program written in C language.

It represents the most recent return code set by any thread within the job.

If the submitted command is a call to a user ILE program returning a value on exiting, this value is found in the Urc end of job code.

You can decide how to control the job environment of your submitted jobs by preparing the commands to be submitted as CALLs to your ILE programs, where the internal flow is controlled and the end status is decided through proper exit values. If a user program ends in error for an incorrect flow control, without returning a value, the Agent Monitor does not set the Return Code as user return code (Urc), but follows the criteria described in "The agent return code retrieval" on page 103.

The following example shows an ILE C user program where two batch jobs are launched and a value of 10 is returned to the caller, regardless of the completion status of the batch jobs.

```
=========================================================================
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
void main(int argc, char *argv[])
{
   int   EnvVarRC=0;
   printf("issuing SBMJOB CMD(CALL MYLIB/DIVBY0)...\n");
   system("SBMJOB CMD(CALL MYLIB/DIVBY0)");
   printf("issuing SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT))...\n");
   system("SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT)) LOG(4 0 *SECLVL)");
   exit(10);
   return;
}
=========================================================================
```

Alternative method to set the user return code

About this task

In some IBM i environments, the system API retrieving the user return code (Urc) from the Agent Monitor code does not retrieve the correct value for Urc. It is therefore not recommended that you use any IBM i system APIs to retrieve the user return code. To receive a value returned by a called program, it is better to provide, instead, a parameter to receive the value.

Even if the Agent Monitor can retrieve the user return code using system API, an alternative user return code retrieval method was implemented in the Agent Monitor code. The alternative retrieval method has the following logic. The `USERRC` job environment variable is created and set to the *INI* value before submitting the user command. When the command ends, the Agent Monitor retrieves its user return code using the system APIs, but it also verifies if the `USERRC` job environment variable was updated at user program level. If a value different from *INI* is found, this is considered as the user return code and the value retrieved using the system APIs is ignored because the user program modified the value of `USERRC` job environment variable.

The change of the `USERRC` variable at user program level requires the `USERRC` value change before exiting from the application user code. In the ILE C case, you can do this using the **`putenv`** statement, where the user return code is set to be returned.

The following example shows how the user code returns the user return code using the IBM i agent reserved job environment variable `USERRC`. This code was obtained from the code of the example in "Controlling the job environment with the user return code" on page 104 by replacing the **`exit`** with the **`putenv`** statement.

```
=========================================================================
#include <stdio.h>
#include <stdlib.h>
```

```
#include <string.h>
void main(int argc, char *argv[])
{
   int   EnvVarRC=0;
   printf("issuing SBMJOB CMD(CALL MYLIB/DIVBY0)...\n");
   system("SBMJOB CMD(CALL MYLIB/DIVBY0)");
   printf("issuing SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT))...\n");
   system("SBMJOB CMD(WRKACTJOB OUTPUT(*PRINT)) LOG(4 0 *SECLVL)");
   EnvVarRC = putenv("USERRC=10");
   return;
}
==============================================================================
```

Configuring the agent on IBM i systems

An overview on how to configure the agent on IBM i systems.

The configuration settings of the agent are contained in the `JobManager.ini` file and in the `JobManagerGW.ini` file (for the path of these files, see Where products and components are installed).

The configuration files are 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 properties
- Trace properties when the agent is stopped. You can also customize traces when the agent is running using the procedure described in "Configuring trace properties when the agent is running" on page 11.
- Native job executor
- Java job executor
- Resource advisor agent
- System scanner

On IBM i systems, the log messages are written in the following file:

```
<TWA_home>/TWS/stdlist/JM/JObManager_message.log
```

On IBM i systems, the trace messages are written in the following files:

```
<TWA_home>/TWS/stdlist/JM/ITA_trace.log
<TWA_home>/TWS/stdlist/JM/JobManager_trace.log
<TWA_home>/TWS/stdlist/JM/javaExecutor0.log
```

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

- "Configuring log message properties [JobManager.Logging.cclog]" on page 9.
- "Configuring trace properties when the agent is stopped [JobManager.Logging.cclog]" on page 10.
- "Configuring common launchers properties [Launchers]" on page 14.
- "Configuring properties of the native job launcher [NativeJobLauncher]" on page 15.
- "Configuring properties of the Java job launcher [JavaJobLauncher]" on page 18.
- "Configuring properties of the Resource advisor agent [ResourceAdvisorAgent]" on page 18.

- "Configuring properties of the System scanner [SystemScanner]" on page 20

Note: In the `JobManager.ini` file and in the `JobManagerGW.ini` file you must refer to Java 64 bit version.

Configuring log message properties [JobManager.Logging.cclog]

About this task

To configure the logs, edit the [JobManager.Logging.cclog] section in the `JobManager.ini` file. This procedure requires that you stop and restart the IBM Workload Scheduler agent

The section containing the log properties is named:
`[JobManager.Logging.cclog]`

You can change the following properties:

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.loggerhd.fileEncoding
By default, log files for the agent are coded in UTF-8 format. If you want to produce the log in a different format, add this property and specify the required codepage.

JobManager.loggerfl.level
The amount of information to be provided in the logs. The value ranges from 3000 to 7000. Smaller numbers correspond to more detailed logs. The default is 3000.

JobManager.ffdc.maxDiskSpace
Exceeding this maximum disk space, log files collected by the first failure data capture mechanism are removed, beginning with the oldest files first.

JobManager.ffdc.baseDir
The directory to which log and trace files collected by the ffdc tool are copied. Default directory is `<TWA_home>\TWS\stdlist\JM\JOBMANAGER-FFDC`.

JobManager.ffdc.filesToCopy
Log and trace files (JobManager_message.log and JobManager_trace.log) collected by the ffdc tool located in `<TWA_home>\TWS\stdlist\JM`. For example, `JobManager.ffdc.filesToCopy = "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_message.log" "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_trace.log"`

When a message is logged (JobManager.ffdc.triggerFilter = JobManager.msgIdFilter) that has an ID that matches the pattern "AWSITA*E" (JobManager.msgIdFilter.msgIds = AWSITA*E), which corresponds to all error messages, then the log and trace files (JobManager.ffdc.filesToCopy = "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_message.log" "/opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JobManager_trace.log") are copied

(JobManager.ffdc.className = ccg_ffdc_filecopy_handler) to the directory JOBMANAGER-FFDC (JobManager.ffdc.baseDir = /opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/JOBMANAGER-FFDC). If the files copied exceed 10 MB (JobManager.ffdc.maxDiskSpace = 10000000), then the oldest files are removed first (JobManager.ffdc.quotaPolicy = QUOTA_AUTODELETE).

After installing the z-centric agent or dynamic agent on Windows 2012, the JobManager_message.log might not be created. In this case, perform the following procedure:

1. Stop the agent.
2. Create a backup copy of JobManager.ini, and edit the original file by changing the row:

   ```
   JobManager.loggerhd.className = ccg_multiproc_filehandler
   ```

   to

   ```
   JobManager.loggerhd.className = ccg_filehandler
   ```
3. Restart the agent.

Configuring trace properties when the agent is stopped [JobManager.Logging.cclog]

How to configure the trace properties when the agent is stopped.

To configure the trace properties when the agent is stopped, edit the [JobManager.Logging] section in the JobManager.ini file and then restart the IBM Workload Scheduler agent.

The section containing the trace properties is named:

```
[JobManager.Logging.cclog]
```

You can change the following properties:

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
Determines the type of trace messages that are logged. Change this value to trace more or fewer events, as appropriate, or on request from IBM Software Support. Valid values are:

DEBUG_MAX
Maximum tracing. Every trace message in the code is written to the trace logs.

INFO All *informational*, *warning*, *error* and *critical* trace messages are written to the trace. The default value.

WARNING
All *warning*, *error* and *critical* trace messages are written to the trace.

ERROR
All *error* and *critical* trace messages are written to the trace.

CRITICAL
Only messages which cause the agent to stop are written to the trace.

The output trace (`JobManager_trace.log`) is provided in XML format.

After installing the z-centric agent or dynamic agent on Windows 2012, the `JobManager_trace.log` might not be created. In this case, perform the following procedure:

1. Stop the agent.
2. Create a backup copy of `JobManager.ini`, and edit the original file by changing the row:

   ```
   JobManager.trhd.className = ccg_multiproc_filehandler
   ```

   to

   ```
   JobManager.trhd.className = ccg_filehandler
   ```
3. Restart the agent.

Configuring trace properties when the agent is running

Use the **twstrace** command to set the trace on the agent when it is running.

Using the **twstrace** command, you can perform the following actions on the agent when it is running:

- "See command usage and verify version" on page 12.
- "Enable or disable trace" on page 12.
- Set the traces to a specific level, specify the number of trace files you want to create, and the maximum size of each trace file. See "Set trace information" on page 12.
- "Show trace information" on page 13.
- Collect trace files, message files, and configuration files in a compressed file using the command line. See "Collect trace information" on page 13.
- Collect trace files, message files, and configuration files in a compressed file using the Dynamic Workload Console. See the section about retrieving IBM Workload Scheduler agent traces from the Dynamic Workload Console in *Troubleshooting Guide*.

You can also configure the traces when the agent is not running by editing the [JobManager.Logging] section in the `JobManager.ini` file as described in Configuring the agent section. This procedure requires that you stop and restart the agent.

twstrace command

Use the **twstrace** command to configure traces, and collect logs, traces, and configuration files (ita.ini and jobManager.ini) for agents. You collect all the information in a compressed file when it is running without stopping and restarting it.

See command usage and verify version

To see the command usage and options, use the following syntax.

Syntax

twstrace -u | -v

Parameters

-u Shows the command usage.

-v Shows the command version.

Enable or disable trace

To set the trace to the maximum or minimum level, use the following syntax.

Syntax

twstrace -enable | -disable

Parameters

-enable
Sets the trace to the maximum level. The maximum level is **1000**.

-disable
Sets the trace to the minimum level. The minimum level is **3000**.

Set trace information

To set the trace to a specific level, specify the number of trace files you want to create, and the maximum size the trace files can reach, use the following syntax.

Syntax

twstrace [**-level** <*level_number*>] [**-maxFiles** <*files_number*>] [**-maxFileBytes** <*bytes_number*>]

Parameters

-level <*level_number*>
Sets the trace level. Specify a value in the range from 1000 to 3000, which is also the default value. Note that if you set this parameter to 3000, you have the lowest verbosity level and the fewest trace messages. To have a better trace level, with the most verbose trace messages and the maximum trace level, set it to **1000**.

-maxFiles <*files_number*>
Specify the number of trace files you want to create.

-maxFileBytes <*bytes_number*>
Set the maximum size in bytes that the trace files can reach. The default is **1024000** bytes.

Show trace information

To display the current trace level, the number of trace files, and the maximum size the trace files can reach, use the following syntax.

Syntax

twstrace -level | -maxFiles | -maxFileBytes

Parameters

-level
See the trace level you set.

-maxFiles
See the number of trace files you create.

-maxFileBytes
See the maximum size you set for each trace file

Sample

The example shows the information you receive when you run the following command:

```
twstrace -level -maxFiles -maxFileBytes
```

```
AWSITA176I The trace properties are: level="1000",
max files="3", file size="1024000".
```

Collect trace information

To collect the trace files, the message files, and the configuration files in a compressed file, use the following syntax.

Syntax

twstrace -getLogs [**-zipFile** <*compressed_file_name*>] [**-host** <*host_name*>] [**-protocol** {*http | https* } [**-port** <*port_number*>][**-iniFile** <*ini_file_name*>]

Parameters

-zipFile <*compressed_file_name*>
Specify the name of the compressed file that contains all the information, that is logs, traces, and configuration files (ita.ini and jobManager.ini) for the agent. The default is **logs.zip**.

-host <*host_name*>
Specify the host name or the IP address of the agent for which you want to collect the trace. The default is **localhost**.

-protocol *http|https*
Specify the protocol of the agent for which you are collecting the trace. The default is the protocol specified in the **.ini** file of the agent.

-port <*port_number*>
Specify the port of the agent. The default is the port number of the agent where you are running the command line.

-iniFile <*ini_file_name*>
Specify the name of the **.ini** file that contains the SSL configuration of the agent for which you want to collect the traces. If you are collecting the traces for a remote agent for which you customized the security certificates, you must import the certificate on the local agent and specify the name of the **.ini** file that contains this configuration. To do this, perform the following actions:

1. Extract the certificate from the keystore of the remote agent.

2. Import the certificate in a local agent keystore. You can create an ad hoc keystore whose name must be **TWSClientKeyStore.kdb**.
3. Create an **.ini** file in which you specify:
   - **0** in the **tcp_port** property as follows:
     `tcp_port=0`
   - The port of the remote agent in the **ssl_port** property as follows:
     `ssl_port=<ssl_port>`
   - The path to the keystore you created in Step 2 on page 14 in the **key_repository_path** property as follows:
     `key_repository_path=<local_agent_keystore_path>`

Configuring common launchers properties [Launchers]

About this task

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

`[Launchers]`

You can change the following properties:

BaseDir
The installation path of the IBM Workload Scheduler agent.

CommandHandlerMinThreads
The default is **20**.

CommandHandlerMaxThreads
The default is **100**.

CpaHeartBeatTimeSeconds
The polling interval in seconds used to verify if the **agent** process is still up and running. If the agent process is inactive the product stops also the **JobManager** process. The default is **30**.

DirectoryPermissions
The access rights assigned to the agent for creating directories when running jobs. The default is **0755**. Supported values are UNIX-format entries in hexadecimal notation.

DownloadDir
The name of the directory where the fix pack installation package or upgrade eImage for fault-tolerant agents or dynamic agents is downloaded during the centralized agent update process. If not specified, the following default directory is used:

On Windows operating systems:
`<TWA_home>\TWS\stdlist\JM\download`

On UNIX operating systems:
`<TWA_home>/TWS/stdlist/JM/download`

The centralized agent update process doesn't apply to z-centric agents.

ExecutorsMaxThreads
The default is **400**.

ExecutorsMinThreads
The default is **38**.

FilePermissions
The access rights assigned to the agent for creating files when running jobs. The default is **0755**. Supported values are UNIX-format entries in hexadecimal notation.

MaxAge
The number of days that job logs are kept (in path *TWA_home*/TWS/stdlidst/JM) before being deleted. The default is **30**. Possible values range from a minimum of 1 day.

NotifierMaxThreads
The default is **5**.

NotifierMinThreads
The default is **3**.

SpoolDir
The path to the folder containing the jobstore and outputs. The default is:
value of BaseDir/stdlidst/JM

StackSizeBytes
The size of the operating system stack in bytes. The default is **DEFAULT**, meaning that the **agent** uses the default value for the operating system.

Configuring properties of the native job launcher [NativeJobLauncher]

About this task

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

You can change the following properties:

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 false. This property does not apply to IBM i.

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.

DefaultWorkingDir
Specifies the working directory of native jobs. You can also specify the value for the working directory when creating or editing the job definition in the Workload Designer. When specified in the Workload Designer, this value overrides the value specified for the **DefaultWorkingDir** property. If you do not specify any working directories, the <*TWS_home*>\bin directory is used.

JobUnspecifiedInteractive
Applies to Windows operating systems only. Specifies if native jobs are to be launched in interactive mode. It can be true or false. The default is false.

KeepCommandTraces
Set to true to store the traces of the method invocation for actions performed on a job definition, for example, when selecting from a picklist.

These files are stored in the path /opt/IBM/TWA_<TWS_user>/TWS/stdlist/JM/r3batch_cmd_exec. The default setting is false.

KeepJobCommandTraces
Set to true to store the traces of the method invocation for actions performed on a job instance, for example, viewing a spool list. These files are stored in the .zip file of the job instance. The default setting is true.

LoadProfile
Specifies if the user profile is to be loaded. It can be true or false. The default is true.

MonitorQueueName
Specifies the name of the queue where the IBM i jobs are monitored. If you do not specify this property, the default queue (QBATCH) is used.

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.

PostJobExecScriptPathName
The fully qualified path of the script file that you want to run when the job completes. By default, this property is not present in the JobManager.ini file. If you do not specify any file path or the script file doesn't exist, no action is taken.

This property applies to dynamic agent and z/OS agent. For details about running a script when a job completes, see *User's Guide and Reference*.

PromotedNice
Used in workload service assurance. This property is not supported on the Agent for z/OS.

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.
- An out of range value (for example -200), prompts the operating system to automatically promote the jobs with the lowest allowed nice value.
- 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.

PromotedPriority
Used in workload service assurance. This property is not supported on the Agent for z/OS.

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. Valid values are:

- `High`
- `AboveNormal` (the default)
- `Normal`
- `BelowNormal`
- `Low` or `Idle`

Note that if you a set a lower priority value than the one non-critical jobs might be assigned, no warning is given.

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:

- *TWS_user* on UNIX and Linux systems
- The local system account on Windows systems

The default is `false`.

RunExecutablesAsIBMiJobs

If you set this property to `true`, you can define IBM i jobs as generic jobs without using the XML definition. Generic jobs are automatically converted to IBM i jobs. As a side effect, generic jobs cannot be run when this parameter is enabled (`RunExecutablesAsIBMiJobs=true`). There is no default value because this property is not listed in the `JobManager.ini` file after the agent installation.

If you set this property to `true`, ensure that the user you used to install the agent has been granted the `*ALLOBJ` special authority.

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 [JavaJobLauncher]

About this task

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

`[JavaJobLauncher]`

You can change the following properties:

JVMDir

The path to the virtual machine used to start job types with advanced options. You can change the path to another compatible Java virtual machine.

JVMOptions

The options to provide to the Java Virtual Machine used to start job types with advanced options. Supported keywords for establishing a secure connection are:

- htttps.proxyHost
- https.proxyPort

Supported keywords for establishing a non-secure connection are:

- Dhttp.proxyHost
- Dhttp.proxyPort

For example, to set job types with advanced options, based on the default JVM http protocol handler, to the unauthenticated proxy server called with name myproxyserver.mycompany.com, define the following option:

```
JVMOptions = -Dhttp.proxyHost=myproxyserver.mycompany.com
 -Dhttp.proxyPort=80
```

Configuring properties of the Resource advisor agent [ResourceAdvisorAgent]

About this task

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

```
[ResourceAdvisorAgent]
```

You can change the following properties:

BackupResourceAdvisorUrls

The list of URLs returned by the IBM Workload Scheduler master in a distributed environment or by the dynamic domain manager either in a z/OS or in a distributed environment. The agent uses this list to connect to the master or dynamic domain manager.

CPUScannerPeriodSeconds

The time interval that the Resource advisor agent collects resource information about the local CPU. The default value is every 10 seconds.

FullyQualifiedHostname

The fully qualified host name of the agent. It is configured automatically at installation time and is used to connect with the master in a distributed environment or with the dynamic domain manager in a z/OS or in a distributed environment. Edit only if the host name is changed after installation.

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.

ResourceAdvisorUrl

JobManager.ini

The URL of the master in a distributed environment, or of the dynamic domain manager in a z/OS or in a distributed environment, that is hosting the agent. This URL is used until the server replies with the list of its URLs. The value is `https://$(tdwb_server):$(tdwb_port)/JobManagerRESTWeb/JobScheduler/resource`, where:

$(*tdwb_server***)**
is the fully qualified host name of the master in a distributed environment or of the dynamic domain manager either in a z/OS or in a distributed environment.

$(*tdwb_port***)**
is the port number of the master in a distributed environment or of the dynamic domain manager either in a z/OS or in a distributed environment.

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 `http`). 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 master in a distributed environment or the dynamic domain manager either in a z/OS or in a distributed environment.

In a distributed environment, if **`-gateway`** is set to either `local` or `remote`, then this is the URL of the dynamic agent workstation where the gateway resides and through which the dynamic agents communicate. The value is `https://$(tdwb_server):$(tdwb_port)/ita/JobManagerGW/JobManagerRESTWeb/JobScheduler/resource`, where:

$(*tdwb_server***)**
The fully qualified host name of the dynamic agent workstation where the gateway resides and through which the dynamic agent communicates with the dynamic workload broker.

$(*tdwb_port***)**
The port number of the dynamic agent workstation where the gateway resides and through which the dynamic agent communicates with the dynamic workload broker.

JobManagerGW.ini
In a distributed environment, if **`-gateway`** is set to `local`, then **`ResourceAdvisorUrl`** is the URL of the master or dynamic domain manager. The value is `https://$(tdwb_server):$(tdwb_port)/JobManagerRESTWeb/JobScheduler/resource`, where:

$(*tdwb_server***)**
The fully qualified host name of the master or dynamic domain manager.

$(*tdwb_port***)**
The port number of the master or dynamic domain manager.

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.

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 master or dynamic domain manager either in a z/OS or in a distributed environment.

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 master in a distributed environment or the dynamic domain manager either in a z/OS or in a distributed environment. 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

If there are no significant changes, the resources are synchronized with the IBM Workload Scheduler master in a distributed environment or with thedynamic domain manager either in a z/OS or in a distributed environment every `NotifyToResourceAdvisorPeriodSeconds` seconds.

Configuring properties of the System scanner [SystemScanner]

About this task

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

`[SystemScanner]`

You can change the following properties:

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 to schedule job types with advanced options

About this task

In addition to defining job types with advanced options using the Dynamic Workload Console or the **composer** command, you can use the related configuration files. The options you define in the configuration files apply to all job types with advanced options of the same type. You can override these options when defining the job using the Dynamic Workload Console or the **composer** command.

Configuration files are available on each dynamic agent in *TWA_home*/TWS/JavaExt/cfg for the following job types with advanced options:

Table 5. Configuration files for job types with advanced options

Job type	File name	Keyword
• Database job type • MSSQL Job	DatabaseJobExecutor.properties	Use the `jdbcDriversPath` keyword to specify the path to the JDBC drivers. Define the keyword so that it points to the JDBC jar files directory, for example: `jdbcDriversPath=c:\\mydir\\jars\\jdbc` The JDBC jar files must be located in the specified directory or its subdirectories. Ensure you have list permissions on the directory and its sub subdirectories. **Note:** For the MSSQL database, use version 4 of the JDBC drivers.
Java job type	JavaJobExecutor.properties	Use the `jarPath` keyword to specify the path to the directory where the jar files are stored. This includes all jar files stored in the specified directory and all sub directories.
J2EE job type	J2EEJobExecutorConfig.properties	For more information about the J2EE job type, see Configuring to schedule J2EE jobs.

Customizing the SSL connection between IBM i agents and the z/OS controller when using your certificates

Customizing the SSL connection between IBM i IBM Workload Scheduler for z/OS Agents and the z/OS controller when using your certificates.

About this task

The communication between IBM i agents and the z/OS controller to which they are registered to is, by default, in http. If you are using your own certificates, to enable an https communication you must customize the agent certificates and the configuration file by performing the following steps:

1. Generate a random file.
2. Generate a PEM file containing the private key of the agent and call it `ita_prv<`*`suffix`*`>.pem`. The IBM Workload Scheduler default PEM file is called **ita_prvtws.pem**.
3. Save the password of the agent private key in a stash file (`.sth file`).

4. Generate another PEM file and call it `ita_pub<suffix>.pem`. It must contain the certificate for the agent private key.
5. Create a copy of the file created in Step 4 and call it `ita_cert<suffix>.pem`.
6. Generate another PEM file and call it `ita_ca_cert<suffix>.pem`. This file must contain the certificate of both the agent and the z/OS controller or the dynamic domain manager to which the agent is connected.
7. Open the `ita.ini` agent configuration file and set the values appropriate for your environment in the following properties:

   ```
   password_file=<stash_file_fullpath>
   random_file=<random_file_fullpath>
   cert_label=<label_agent_private_key>
   key_db_name=<suffix>
   key_repository_dir=<directory_ita_*<suffix>.pem>
   tcp_port=0
   ssl_port=<ssl_port_value>
   ```

   Where:

   stash_file_fullpath
   Specify the fully qualified path to the stash file that contains the agent private key password. This is the file that you created in the Step 3 on page 119. The default value is `/opt/IBM/TWA_<TWS_user>/TWS/ITA/cpa/ita/cert/password.sth`.

   random_file_fullpath
   Specify the fully qualified path to the random file. This is the file that you created in the Step 1 on page 119. The default value is `/opt/IBM/TWA_<TWS_user>/TWS/ITA/cpa/ita/cert/TWS.rnd`.

   label_agent_private_key
   Specify the label of the agent private key. The default is **client**.

   suffix Specify the suffix you used in the names of all the files that you generated. The default product value is **tws**.

   directory_ita_*<suffix>.pem
   Specify the directory that contains all the .pem files that you generated. The default directory is `/opt/IBM/TWA_<TWS_user>/TWS/ITA/cpa/ita/cert`.

   tcp_port_value
   Specify **0** as TCP/IP port value.

   ssl_port_value
   Specify the *tcp_port_value*. For example, if the TCP/IP port value was 31114, specify 31114.
8. Stop the IBM i agent by using the following command:

   ```
   ShutDownLwa
   ```
9. Start the IBM i agent by using the following command:

   ```
   StartUpLwa
   ```

After you complete the procedure, depending on the SSL storing certificate method you use, import the certificates in a RACF KEYRING or in a keystore created in the UNIX System services. Depending on the method you use refer either to the RACF or the Unix System services documentation.

Chapter 10. Monitoring and recovering jobs

This chapter contains the following sections.

IBM Workload Scheduler for z/OS Agent job status notifications

Notifications about the change of status of running jobs are returned to the HTTP server subtask by the Tivoli agent. Table 6 shows the possible status changes for jobs running on IBM Workload Scheduler for z/OS Agents.

Table 6. Job status change notifications

Notification	New operation status	Explanation
Submitted (POST executed)	S	Post executed (from local HTTP client)
Submitted	S	Agent launched the job
Canceled	E (+ CAN)	Job canceled by command
Executing	S	Job started execution
Failed_Execution	E (+ error code)	Job ended in error
Succeeded_Execution	C	Job completed successfully
Failed_Submission	E (+ OSUB)	Job submission failed
Unknown_Execution	E (+ CCUN)	Agent lost track of the job

Setting up and running automatic recovery for IBM Workload Scheduler for z/OS Agent jobs

IBM Workload Scheduler for z/OS supports the use of automatic recovery statements in IBM Workload Scheduler for z/OS Agent scripts.

When you create or edit an IBM Workload Scheduler for z/OS Agent script, start the lines where you specify the automatic recovery statements with `//* OPC`, `//*%OPC`, or `//*>OPC`. These lines are removed before the script is downloaded onto the IBM Workload Scheduler for z/OS Agents.

You can run the following recovery actions on IBM Workload Scheduler for z/OS Agent jobs:

- Restart the current occurrence at the failed operation, with or without JCL changes.
- Restart the current occurrence at another operation.
- Add occurrences of special recovery applications. Make the restart of the failed occurrence dependent on the completion of the recovery occurrences.
- Release a dependent occurrence.

Data set cleanup is not available.

Automatic-recovery-control statement for IBM Workload Scheduler for z/OS Agent scripts

Each automatic-recovery-control statement describes an error situation and the recovery actions for it.

Use these rules to code RECOVER statements:

- Each statement must begin in a new 80-byte logical record.

  The symbols `//*%OPC` must appear in bytes 1 to 7 and be followed by at least one blank.

  `//*%OPC RECOVER`
  Identifies a RECOVER statement

  `//*%OPC`
  Identifies a RECOVER continuation statement

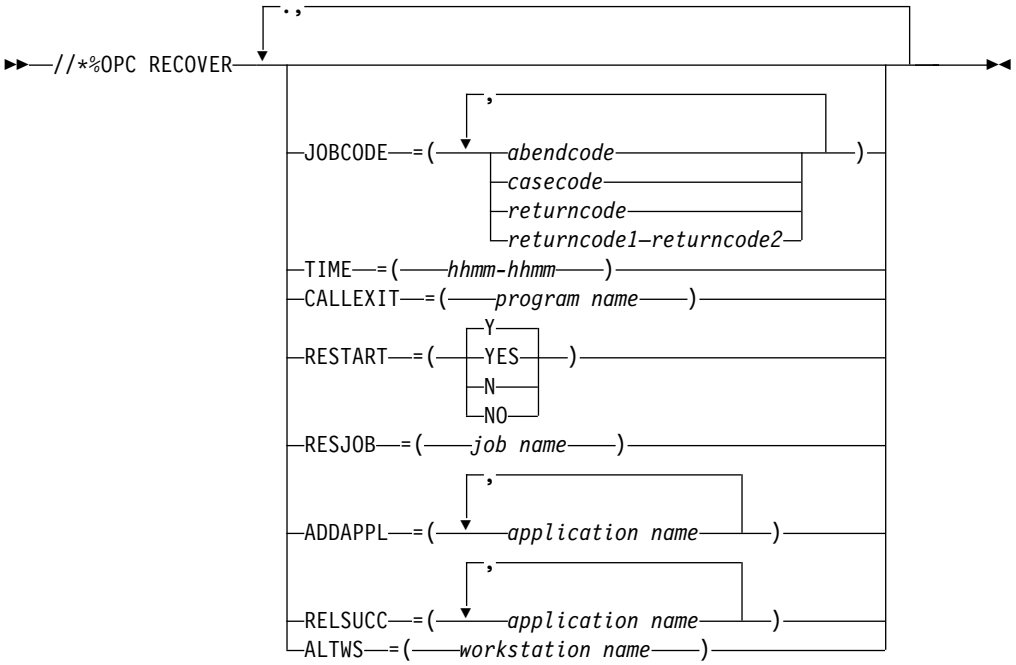
  The automatic recovery function also inserts informational statements in the JCL:

  `//* OPC`
  Identifies a message statement

  `//*>OPC`
  Identifies a comment statement
- You cannot use variables anywhere in the RECOVER statement.
- The parameters are optional; you can code them in any sequence.
- Each parameter consists of a keyword followed by an equals sign and variable information.
- Parameters are separated by commas.
- You cannot code the same keyword more than once on the same statement.
- If you code only one parameter value, you do not need to enclose it in parentheses; for example, `JOBCODE=PCHK`.
- Bytes 72 to 80 are ignored by the automatic recovery function.
- When the total length of fields on a control statement exceeds 71 bytes, continue the statement using the following continuation conventions:
  - Interrupt the field after a complete or partial parameter, including the comma that follows it, before byte 72.
  - Code the identifying continuation characters `//*%OPC` followed by at least one blank in bytes 1 to 7 of the statement that follows.
  - Continue the interrupted operand in any position from bytes 9 to 16.
- Limit the RECOVER statement to one card (avoid continuation).

RECOVER statement syntax

The syntax of the RECOVER statement follows. These are the only RECOVER parameters that apply to the IBM Workload Scheduler for z/OS Agent workstation. For a complete list of the RECOVER parameters, see *IBM Workload Scheduler for z/OS: Managing the Workload*.



Statement parameters

All the parameters for the RECOVER statement are optional. For IBM Workload Scheduler for z/OS Agent jobs, the following are the available RECOVER statement categories: selection and recovery action.

Selection parameters:

Selection parameters for error situations managed by the recover statement.

These selection parameters specify the error situations that the RECOVER statement manages:

JOBCODE
Restricts the RECOVER statement to be valid only for those job completion codes and return codes specified.

TIME Restricts the RECOVER statement to be valid only in the time range specified.

Note: Review the EXCLUDECC and EXCLUDERC keywords of the AROPTS statement, which specify codes for which no automatic recovery is done. For information about the AROPTS statement, see *IBM Workload Scheduler for z/OS: Customization and Tuning*.

Recovery action parameters:
The following recovery action parameters specify the actions to take for recovery:

RESTART
Specifies if the occurrence is to be restarted.

RESJOB
Specifies the name of the job from which the occurrence must be rerun.

ADDAPPL
Specifies an application or a list of applications to be added as occurrences in the current plan.

RELSUCC
Specifies the application ID of a successor occurrence, or a list of IDs.

ALTWS
Specifies the name of an alternate workstation on which to run the operation.

Selection parameters

When a job ends in error, this information is available:
- Job name
- Abend code, if job abended
- Return code, if job did not abend

The selection parameters use this information:

JOBCODE
The code can be an abend code, an error code set by IBM Workload Scheduler for z/OS or JCC, a case code, a return code, or a return code range. The values are those given as the error code on the Handling Operations Ended in Error panel in the Modify Current Plan panel.

The JOBCODE keyword values are:

S*xxx* Specifies a system abend code.

U*xxx* Specifies a user abend code.

xxxx Specifies a case code or an error code set by IBM Workload Scheduler for z/OS, either directly or by using the job completion checker.

n Specifies a return code.

x-y Specifies a return code range, where *x* and *y* represent positive decimal values.

The codes specified in the JOBCODE parameter are tested against the job code as set by IBM Workload Scheduler for z/OS when the job ends.

You can also specify abend codes in generic form. Therefore, an asterisk (*) can represent any character, or any group of characters, in those positions where it is placed. A code can contain more than one asterisk only if each asterisk is separated from the next by another character.

Note: The JOBCODE=* notation covers all possible user and system abend codes; it does not cover return codes. To cover all possible return codes, specify a return code range in the form:

```
JOBCODE=x-y
```

where *x* and *y* represent positive decimal values.

A return code cannot be greater than 4095. To specify a range of values, set *x* to the lower value and *y* to the higher value. To specify all return codes greater than or equal to a certain value, set *x* to that value and *y* to 4095.

TIME The time is specified in the form *hhmm*, where *hh* is the hour from 00 to 24, and *mm* is the minute from 00 to 60. This is the time when the recovery is automatic.

For example:

TIME=0700-1600
No recovery actions occur between 4 p.m. and 7 a.m.

TIME=2200-0800
IBM Workload Scheduler for z/OS does automatic recovery only between 22.00 and 8.00.

TIME=0000-2400
Recovery can be automatic at any time.

TIME=0000-0000
Recovery is not started unless there is manual intervention.

The recovery actions for a job that remains in the ended-in-error list can be manually started up later. Such requests override any TIME value specified.

Default: The recovery specification is for the time range specified by the STARTIME and ENDTIME keywords of the AROPTS automatic recovery options statement. Refer to *IBM Workload Scheduler for z/OS: Customization and Tuning* for more details.

Action parameters

The following parameters specify the action that IBM Workload Scheduler for z/OS must take when the recovery statement is invoked.

RESTART
RESTART=Y causes the job to be rerun, either from the failing operation or, if you specified RESJOB, from an earlier operation within the occurrence.

RESTART=N prevents the restart. It can be used with ADDAPPL when the recovery actions are to be done by a separate application. It can also be used to select cases for which no recovery should be performed or when testing the recovery procedure. The operation remains in ended-in-error status.

Default: RESTART=Y.

RESJOB
The RESJOB parameter handles failures at an occurrence level, not within the failing job itself.

The occurrence of the application is rerun from the first preceding computer workstation operation, whose name matches the job name specified in the RESJOB parameter. If the job name specified cannot be found in a computer operation preceding the failed operation in the same occurrence, no automatic recovery occurs and the job remains in the error handling list with an extended status code indicating an automatic recovery error.

The indicated operation must be a predecessor to the failed operation or be the failed operation itself.

Note: External successors cannot be handled automatically. Therefore the set of operations selected for rerun that can be completed at failure time must not have any external successors.

The rerun operation must be defined on an automatically reporting computer workstation.

The parameter is ignored if RESTART=N is specified.

Default: Rerun from the failed operation.

ADDAPPL

Here you specify a list of applications. Occurrences of these applications are added to the current plan if this RECOVER statement is invoked.

- If RESTART=N is specified, the applications are independent of the failed occurrence.
- If RESTART=Y is specified, the recovery applications are added to the current plan as predecessors to the failing operation or, if RESJOB is specified, to the operation where restart is being attempted.

Added applications are independent of each other. A maximum of 40 application occurrences can be added.

For example, suppose a database update fails. A rerun of the failed job is necessary but must be postponed to a later time. However, a database restore job must be run to repair the database before the online users require the database. This recovery situation can be specified as:

```
//*%OPC RECOVER JOBCODE=SCHK,RESTART=N,ADDAPPL=A301RORG
```

This means that, on case code SCHK, do not rerun the failing operation. However, add an occurrence of the application A301RORG to the current plan, without a dependency to the failing operation and leave the failing operation in ended-in-error status.

Default: No application occurrences are added.

Note:

1. When automatic recovery adds an occurrence to the current plan, input arrival and deadline times are not taken from the application description. Instead the occurrence is given an input arrival of the time the add is run, according to the time on the z/OS system where the controller is started and the deadline is set for 8 hours after input arrival. If an occurrence of that application already exists with this input arrival time, then one minute is added to the time until a time is reached when the occurrence can be included. If the added occurrence includes time-dependent operations with specific input arrival times, then the operations are started at the specified time.
2. An occurrence that has been added to the current plan by automatic job recovery does not become the predecessor to an occurrence that is added later by daily planning, even if normal dependency criteria are met.

RELSUCC

This parameter specifies which external successors to the failing operation are allowed to run even if their predecessor operation has ended in error.

The external successors to the failing operation are checked, and the dependencies between the failed operation and the specified successors are deleted at recovery time.

The effect is that this predecessor (the failed operation) is reported as complete to the external successor, and the successor-predecessor chaining is removed. The external successor becomes ready if its other predecessors are completed. The dependency does not exist when the failed occurrence is rerun.

Even if one successor is released, other successors might be waiting for the failed occurrence to complete. These might be successors not yet in the current plan. Assume that W is a weekly application and D is a daily application that is dependent on W. If W fails and there is a RECOVER

statement causing the release of that day's D, the occurrence of D the next day also waits for W to complete, but without any automatic release.

You can specify a maximum of 40 application IDs.

Default: None.

ALTWS

Specifies the name of an alternate workstation on which to run the operation. The ALTWS parameter overrides the alternate workstation defined in the workstation description. You can use this parameter, for example, with the TIME parameter to specify alternate workstations for an operation, depending on the time of day.

Default: None.

Chapter 11. Running a script when a job completes

In many scenarios, when a job completes, you might want to run one or more actions, by using the information related to the job completion. For this purpose, you can write a script file and store it in a directory of the agent file system. The script is run every time that a job completes, either successfully or unsuccessfully. The script runs with the same credentials as the agent user that is running the job.

Note: The agent user must be authorized to access the script file and its directory.

To provide IBM Workload Scheduler with the path of the script file, you must modify the `JobManager.ini` file as follows:

1. Locate the `JobManager.ini` file on the local agent instance where the script will run. The file is located in the `TWA_home/TWS/ITA/cpa/config` directory on the agent.
2. In the [NativeJobLauncher] section of the file, define the value of the **PostJobExecScriptPathName** property, with the fully qualified path of the script file that you want to run when a job completes.
3. Save the changes to the file.

If you do not specify any file path or the script file doesn't exist, no action is taken when the job completes. For details about customizing the **PostJobExecScriptPathName** property, see *Administration Guide*.

The following job variables can be used in the script:

- JOB_ID
- JOB_ALIAS
- JOB_SPOOL_DIR
- JOB_STATUS
- JOB_RETURN_CODE
- JOB_DURATION
- JOB_START_TIME
- JOB_END_TIME

The script is run for any of the following job final statuses:

- SUCCEEDED_ EXECUTION
- UNKNOWN
- CANCELLED
- FAILED_ EXECUTION

In the `JobManager_message.log`, you are notified via a message if the job started successfully, or if any error prevented the job from starting. To analyze the output of the script execution, you can check the `out.log` file in the `post_script` subdirectory of the job `SpoolDir`.

Chapter 12. Viewing the job logs from the host

An overview on how to view or retrieve the job logs issued in the z-centric environment and how to attach a header with job run information on top of each log.

There are two ways of retrieving the job logs:

- You can manually request to browse the log of a specific job (on demand) or have automatically displayed the logs of jobs that ended in error. To browse the log you can use any of the available user interfaces (ISPF dialogs, Dynamic Workload Console, or programing interfaces).

  To specify the job log retrieval policy (on demand, or automatic for ended-in-error), define the JOBLOGRETRIEVAL keyword in the HTTPOPTS initialization statement. See *IBM Workload Scheduler for z/OS: Customization and Tuning*.
- You can activate the Output collector started task to have the logs of all jobs, independently of their outcome, sent to the JES spool to be picked up by an external output management product.

On all the job logs originated from the z-centric environment you can customize IBM Workload Scheduler for z/OS so that a header is added automatically at the top of the job log with the information necessary to classify the output (occurrence name, occurrence IA, job name, workstation name, operation number, start time, end time) and information about the run (process ID, return code, duration, status, hostname).

Remember: to have the job log header mechanism functioning for workload run on dynamic agents, the dynamic agents must be running at least version 8.6.0.1 of IBM Workload Scheduler.

Customizing the job log header

A sample job log header template is downloaded on IBM Workload Scheduler for z/OS at installation time. You can customize it to fit it to your preferences.

When you customize the JLOGHDRTEMPL keyword of the HTTPOPTS initialization statement, you command the system to add a header containing run-related information to every job log released in the z-centric environment.

The header format can be personalized. The information displayed in the header is added depending on how you customize the header template sample that comes with the installation of IBM Workload Scheduler for z/OS and which predefined variables that describe the job and its run are featured in the header.

When you create the sample job JCL with the EQQJOBS installation aid, the following job log header template sample is added to the samples library that you specified in the EQQJOBS panels:

```
============================================================
=                         JOBLOG HEADER
=
= Occurrence name : ${occname}
= Occurrence IA   : ${IAYYYY}/${IAMM}/${IADD} ${IAhh}:${IAmm}
= Job name        : ${jobname}
```

```
= Work station    : ${wsname}
= Operation number: ${opernum}
= Start time      : ${SYYYY}/${SMM}/${SDD} ${Shh}:${Smm}
= End time        : ${EYYYY}/${EMM}/${EDD} ${Ehh}:${Emm}
= Process ID      : ${processid}
= Duration        : ${duration}
= Status          : ${status}
= Return code     : ${returncode}
= Hostname        : ${hostname}
=
===============================================================
```

You can then edit and copy the sample to a PARM library that must be pointed to by the JLOGHDRTEMPL keyword of the HTTPOPTS initialization statement.

The properties of the predefined variables provided in the sample are described in the following table. When you customize the sample, you can select the variables that you deem useful or retain them all.

Table 7. The predefined variables of the sample job log header template.

Variable	Name	Length	Value is...
Occurrence name	occname	16	sent at job submission
Occurrence input arrival time	IAYYYY IAMM IADD IAhh IAmm		sent at job submission
Job name	jobname	8	sent at job submission
Workstation	wsname	4	sent at job submission
Operation number	opernum	2	sent at job submission
Start time	SYYYY SMM SDD Shh Smm		local on agent
End time	EYYYY EMM EDD Ehh Emm		local on agent
Process ID	processid		local on agent
Duration	duration		local on agent
Status	status		local on agent
Return code	returncode		local on agent; NOERROR actions are ignored and not truncated to 4 characters
Hostname	hostname		local on agent; hostname of the agent where the job ran

The syntax for the job log header variables is consistent with the syntax used for variable substitution in the z-centric environment, that is:

${*variable_name*}

Important: The template sample is in the IBM-37 codepage. The brace characters ({ and }) are mapped differently depending on the codepage. If you use a different codepage to view the template, you are very likely to need to correct the variables syntax.

Only simple variable substitution is allowed, no compound variables are supported. If the variable name is incorrect, the variable is left unresolved.

You can also use variables to compose the run dates. The dates are based on the local time of the controller.

Collecting job logs with Output collector

An overview on the mechanism that automatically retrieves the logs of jobs and dynamic jobs run on IBM Workload Scheduler for z/OS Agents and copies them to the JES spool so that they can be processed by an external output management product.

The standard log management process for jobs running in the z-centric environment is to be requested manually, with an exception for the logs of jobs that ended in error that can be received semi-automatically. In addition to this, or as an alternative, you can activate the Output collector started task, which automatically retrieves the log of every job run in the z-centric environment and sends it to the JES spool so that it can be processed by an external output management product.

Every time a job or a dynamic job completes or terminates on an IBM Workload Scheduler for z/OS Agent, the Output collector started task receives an event from the IBM Workload Scheduler for z/OS controller (which manages all communication with the agents and the dynamic domain managers in the z-centric environment). The event contains the information necessary for Output collector to identify the job and where it run. The output collector then retrieves the job log from the agent, or the dynamic domain manager if the job is dynamic, and copies it to a SYSOUT in JES (using a specific SYSOUT class) to make it available to an output management product.

Activation of this feature is optional. If you activate it, it automatically collects the logs of all jobs run in the z-centric environment, regardless of whether they complete successfully or terminate in error. If you do not activate it, you can still configure your system to either request logs manually or to receive those of jobs ended in error.

The following figure describes the job log retrieval process with Output collector.

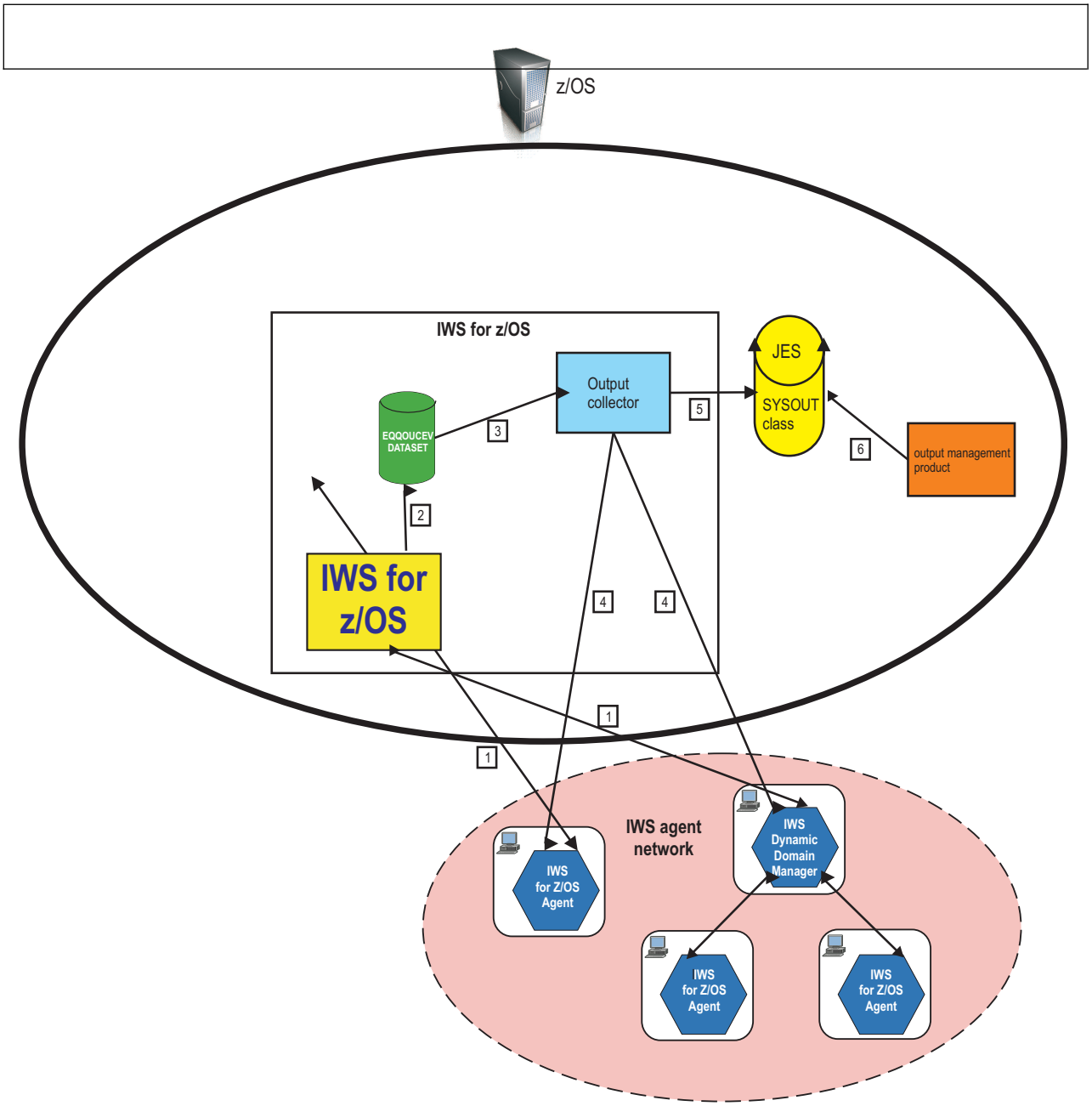


Figure 12. Job log retrieval with Output collector.

Where:

1. The controller submits and tracks jobs and gets the job logs upon request.
2. Every time a job completes or terminates, the controller writes an event in the EQQOUCEV data set. The event contains identification of the job and the name of the agent that ran it.
3. Output collector reads the event in EQQOUCEV.
4. Based on the information found in the event record, Output collector retrieves the job log from the agent (or the dynamic domain manager if the job is dynamic).
5. Output collector copies the job log to a SYSOUT in JES using a specific SYSOUT class.

6. The external output management product can get the job log for analysis, accounting, and other operations.

The controller and Output collector use the EQQOUCEV and EQQOUCKP data sets to share the information concerning completed or terminated jobs. The communication process is based upon events. Every time a job completes or terminates, the controller queues an event for Output collector with the information necessary to identify the job and the agent that run it in a new record in EQQOUCEV. Output collector reads the record, checkpoints it in EQQOUCKP, dispatches it to the proper thread, and marks the event as processed moving to the next-to-read index in the data set header. EQQOUCKP is used to checkpoint the incoming requests to prevent their loss in case of unplanned closures.

To write a job log in the JES spool, Output collector allocates a SYSOUT data set with DDNAME equal to the job name, writes the job log in it, and then closes it. This implies that all the SYSOUTS will have the started task job name and job id and will differ only in the DDNAME and job log header.

Attention: The following event may take place while Output collector is processing more job instances with the same name at the same time, even if on different agents: if Output collector is retrieving a particularly sizable log from an agent and the agent goes offline, Output collector stops the retrieval until the link with the agent is established again. As soon as the agent is online again, Output collector resumes retrieving the log. The anomaly is that, if in this lapse of time Output collector is requested to retrieve any other logs with the same name, perhaps from other agents (remember that all logs are named after their jobs), the blocked state is extended also to these requests. As soon as conditions are set for resuming the retrieval of the first log, the other logs are also processed. This does not apply to other job logs having different names: Output collector in this time span continues to process them normally.

Activating and configuring Output collector

This is a summary about how to activate and configure the Output collector started task.

To activate Output collector, run the EQQJOBS installation aid and go to panel EQQJOBSC of option `1 - Create sample job JCL`. In the EQQJOBSC panel you specify that Output collector will be running and you enter the names of the SYSOUT class and of the WRITER task that Output collector uses to copy the logs to the JES spool. This creates the sample started task procedure and initial parameters for running Output collector and allocates the data sets (EQQOUCEV and EQQOUCKP) used for receiving job log information from the controller. See *IBM Workload Scheduler for z/OS: Planning and Installation* for a detailed description.

To fully configure Output collector, you must act on the following initialization statements:

HTTPOPTS
Add the following keywords:

OUTPUTCOLLECTOR
Specify that the feature will be running.

JLOGHDRTEMPL
Specify the name of the EQQPARM library member that contains

the job log header. This keyword is not exclusive to Output collector, but can be used for all the job logs produced in the z-centric environment.

These keywords are independent of the JOBLOGRETRIEVAL keyword, which enables users to request job logs manually through the usual IBM Workload Scheduler for z/OS interfaces.

OPCOPTS

Optionally, add the **OUTCOL** keyword to specify the name of the Output collector started task to the controller. In a sysplex environment this enables the standby controller to continue the interaction with Output collector.

OUCOPTS

Add this initialization statement and its keywords to define:

- The names of the JES SYSOUT class and of the WRITER task used by Output collector (if they were not defined with EQQJOBS).
- The name of the EQQPARM library member that contains the OPCOPTS, ROUTOPTS, and HTTPOPTS statements.
- The number of threads (range) that Output collector can open.
- The maximum number of job logs that Output collector can retrieve concurrently from an agent.
- The name of the controller for which Output collector is started.

In a sysplex configuration the Output collector started task must reside in the same image where the controller is.

See *IBM Workload Scheduler for z/OS: Customization and Tuning* for a detailed description.

Refreshing destinations from Output collector

You can use the MODIFY operator commands to display and refresh HTTP/HTTPS destinations also from Output collector.

You can refresh or display IBM Workload Scheduler for z/OS Agent destinations while Output collector is running, without having to stop and restart Output collector to make the changes effective. To do so, run the TSO commands:

- `/F procname, DSPDEST` to display
- `/F procname, RFRDEST` to add, modify, or delete

where in *procname* you can enter the JCL procedure name for Output collector instead of the one for the controller.

If you run `/F procname, RFRDEST` with the *procname* for the controller, the refresh operation is run on the controller and propagated also to Output collector.

Be aware that any changes you make using the procedure name for the controller apply also to Output collector, while they are not extended to the controller if you use the procedure name for Output collector.

Chapter 13. Troubleshooting

Troubleshooting agent problems

The following problems could be encountered:

On AIX operating systems the concurrent submission of one hundred or more jobs on the same agent can result in a core dump or in a resource temporarily unavailable message

On AIX operating systems, the concurrent submission of one hundred or more jobs on the same agent can result in a memory dump or in a resource temporarily unavailable message

On AIX operating systems if you submit concurrently one hundred or more jobs on the same agent you can receive a core memory dump or the following message:

```
resource temporarily unavailable
```

Cause and solution:

This problem is due to insufficient memory and the process number per user allocated to run the jobs concurrently. To solve this problem, verify the value of the following configuration settings and change them as follows:

Ulimit settings

The submission of a significant number of Java jobs requires a large amount of memory. Change the value for data, stack, and memory limits according to the number of jobs you want to submit. The submission of a significant number of native jobs requires a high number of file descriptors and processes. Change the values for nofiles and processes according to the number of jobs you want to submit. The following example gives possible setting values to submit 100 jobs concurrently:

```
time(seconds)        unlimited
file(blocks)         2097151
data(kbytes)         131072
stack(kbytes)        32768
memory(kbytes)       32768
coredump(blocks)     2097151
nofiles(descriptors) 4000
threads(per process) unlimited
processes(per user)  unlimited
```

Process number per user

To submit a high number of jobs concurrently you must have a high value for the **maxuproc** setting. Use the **lsattr -E -l sys0 -a maxuproc** command to verify the number of concurrent processes that a user can create. Use the **chdev -l sys0 -a maxuproc=<*value*>** command to change the value for the **maxuproc** setting. For example, to submit 100 jobs concurrently use the following command:

```
chdev -l sys0 -a maxuproc=500
```

Appendix A. Configuring to schedule J2EE jobs

To schedule J2EE jobs, complete the following configuration tasks:

- Configure the J2EE executor on every IBM Workload Scheduler for z/OS Agent where you want to submit J2EE jobs.
- Configure the J2EE Job Executor Agent on an external WebSphere Application Server

Configuring the J2EE executor

To be able to dynamically schedule J2EE jobs, you must configure the following property files on every IBM Workload Scheduler for z/OS Agent where you want to submit J2EE jobs:

- `J2EEJobExecutorConfig.properties`
- `soap.client.props`

These files are configured with default values at installation time. The following sections describe the values you can customize.

J2EEJobExecutorConfig.properties file

The `J2EEJobExecutorConfig.properties` file is located on the IBM Workload Scheduler for z/OS Agent, under *TWA_home*`/TWS/JavaExt/cfg`.

Table 8 shows a description of the file keywords.

Table 8. J2EEJobExecutorConfig.properties file keywords

Keyword	Specifies...	Default value	Must be customized
wasjaas.default	The path to the IBM WebSphere configuration file (`wsjaas_client.conf`) used to authenticate on the external WebSphere Application Server using JAAS security.	*TWA_home*/TWS/JavaExt/cfg/wsjaas_client.conf or *TWA_home*\TWS\JavaExt\cfg\wsjaas_client.conf	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)	wasadmin,{xor}KD4sPjsyNjE\= (ID=wasadmin and password=wasadmin in {xor} encrypted format)	Yes, see "Running {xor} encryption on your password" on page 141 to learn how to encrypt your password.

Table 8. J2EEJobExecutorConfig.properties file keywords (continued)

Keyword	Specifies...	Default value	Must be customized

connector.indirect	The name of the communication channel with WebSphere Application Server. Selecting an indirect invoker means that dynamic workload broker leverages an existing WebSphere Application Server scheduling infrastructure already configured on a target external WebSphere Application Server. When creating the job definition, you can specify whether 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:
		• `indirect` keyword
		• Name of the scheduler:
		`sch/MyScheduler`
		• `soap` keyword
		• Host name of the external WebSphere Application Server instance:
		`washost.mydomain.com`
		• SOAP port of the WebSphere Application Server instance:
		`8880`
		• Path to the `soap.client.props` file:
		TWA_home`/TWS/JavaExt/cfg/soap.client.props`
		• Credentials keyword:
		`mycred`
			You must customize the following:
			• The scheduler name. Replace the `sch/MyScheduler` string with the JNDI name of the IBM WebSphere scheduler you plan to use.
			• The host name of the external WebSphere Application Server instance.
			• The SOAP port of the external WebSphere Application Server instance.

connector.direct	The name of the direct communication channel without using the WebSphere Application Server scheduler. Selecting a direct invoker means that dynamic workload broker immediately forwards the job to the external WebSphere Application Server instance component (JMS). When creating the job definition, you can specify whether 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:
		• `direct` keyword
		• The following string:
		`com.ibm.websphere.naming.WsnInitialContextFactory`
		• The following string:
		`corbaloc:iiop:washost.mydomain.com:2809`
			You must customize the following:
			• 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 trustore file (this file must be copied to this local path from the WebSphere Application Server instance).	*TWA_home*/TWS/JavaExt/cfg/DummyClientTrustFile.jks	You can change the path (*TWA_home*/TWS/JavaExt/cfg), provided you copy the trustore path from the external WebSphere Application Server to this path.

Table 8. J2EEJobExecutorConfig.properties file keywords (continued)

Keyword	Specifies...	Default value	Must be customized
trustStore.password	The password for the WebSphere Application Server trustore file.	WebAs	Yes

Running {xor} encryption on your password

About this task

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 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:
   string`={xor}`*your_encrypted_password*
5. Copy your encrypted password, inclusive of the `{xor}` characters, and paste it where required onto your property files.

For example, 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 soap.client.props file

The `soap.client.props` file is located on the IBM Workload Scheduler for z/OS Agent, under *TWA_home*`/TWS/JavaExt/cfg/soap.client.props`.

Following installation, this file is as follows:

```
#-------------------------------------------------------------------------------
# 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
```

To enable SOAP client security, follow these steps:

1. Change `com.ibm.SOAP.securityEnabled` to `true`.
2. Customize:
   - `com.ibm.SOAP.loginUserid` with the true WebSphere Application Server administrator userid.
   - `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 141.

Configuring the J2EE Job Executor Agent

To set up the environment on the external WebSphere Application Server, Version 7.0, for the J2EE Job Executor Agent, perform the tasks in this section.

Create a Service Integration Bus

About this task

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 panel is displayed.
3. Click **New** to display the Buses configuration window.

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

Configure the Default Messaging Service

About this task

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 panel, type 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 the *`Basic_SIB_port_number`* and *`Secure_SIB_port_number`* can be found by expanding **Servers**, then selecting *your_application_server_name*, then selecting **Messaging engine inbound transports** under the **Server Messaging** heading.
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 type in the fields as follows:

   ```
   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 type the fields as follows:

   ```
   Name=MyActSpec
   JNDI name=eis/MyActSpec
   Bus name=MyBus
   Destination type=Queue
   Destination JNDI name=jms/MyQueue
   ```

   Click **OK**.

Configure the Java security

About this task

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 type the fields below as follows:

   ```
   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

About this task

1. In the left panel, go to **Resources → JDBC..→ JDBCProviders**. In the resulting right panel, 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 type in the fields as follows:

   ```
   Name=MyScheduler XA DataSource
   JNDI name=jdbc/SchedulerXADS
   Database name=${USER_INSTALL_ROOT}/databases/Schedulers/${SERVER}/SchedulerDB;
                 create=true
   ```
5. On top of the page, click the **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

About this task

1. In the left panel, go to **Resources → Asynchronous beans → Work managers** and click **New**.
2. Type in the fields as follows:

   ```
   Name=SchedulerWM
   JNDI name=wm/SchedulerWM
   ```
3. Click **OK**.

Create and configure a scheduler

About this task

1. In the left panel, go to **Resources → Schedulers** and click **New**.
2. Type in the fields as follows:

   ```
   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.

Appendix B. Security order of precedence used for the execution of J2EE tasks

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

1. Using the Java Authentication and Authorization Service (JAAS) security context on the thread at the time the task was created.
2. Using the `setAuthenticationAlias` method on the `TaskInfo` object.
3. Using a specified security identity on a `BeanTaskInfo` task `TaskHandler` method.

The authentication methods are sorted 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** win over any credentials specified in the tasks themselves.

Appendix C. Configuring to schedule job types with advanced options

About this task

In addition to defining job types with advanced options using the Dynamic Workload Console or the **JOBREC** statement, you can use the related configuration files. The options you define in the configuration files apply to all job types with advanced options of the same type. You can override these options when defining the job using the Dynamic Workload Console or the **JOBREC** statement.

Configuration files are available on each dynamic agent in *TWA_home*/TWS/JavaExt/cfg for the following job types with advanced options:

Table 9. Configuration files for job types with advanced options

Job type	File name	Keyword
• Database job type • MSSQL Job	DatabaseJobExecutor.properties	Use the `jdbcDriversPath` keyword to specify the path to the JDBC drivers. Define the keyword so that it points to the JDBC jar files directory, for example: `jdbcDriversPath=c:\\mydir\\jars\\jdbc` The JDBC jar files must be located in the specified directory or its subdirectories. Ensure you have list permission for the directory and its sub subdirectories. **Note:** The MSSQL database requires version 4 of the JDBC drivers for both Database and MSSQL Job job types.
Java	JavaJobExecutor.properties	Use the `jarPath` keyword to specify the path to the directory where the jar files are stored. This includes all jar files stored in the specified directory and all sub directories.
J2EE	J2EEJobExecutorConfig.properties	For more information about the J2EE job type, see Appendix A, "Configuring to schedule J2EE jobs," on page 139.

Logging information about job types with advanced options

You can use the `logging.properties` file to configure the logging process for job types with advanced options, with the exception of the Executable and Access Method job types.

The `logging.properties` file is located on the IBM Workload Scheduler for z/OS Agent, under *TWA_home*`/TWS/JavaExt/cfg/logging.properties`.

Following 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 Defines the logging level for the internal logger.

  com.ibm.scheduling
  Defines the logging level for the job types with advanced options. To log information about job types with advanced options, set this keyword to ALL.

- The path where the logs are written, specified by the following keyword:

  `java.util.logging.FileHandler.pattern`

Appendix D. Managing return codes

By default, all nonzero error codes are considered as an error. Optionally, you can define the error codes that, for job-tracking purposes, are not considered as errors. To do this, set the following parameters:

NOERROR
Defines a list of error codes that, for job-tracking purposes, are treated as normal completion codes. When setting this value on an IBM Workload Scheduler for z/OS Agent, beware that the maximum length allowed for a return code is 4 digits. If the IBM Workload Scheduler for z/OS Agent returns an error code higher than `9999`, it is truncated at the fourth rightmost digit. For example, `12345` is considered as `2345`.

For negative error codes, specify a five-character sequence starting with the minus (-) symbol, for example `-0008`.

ERRRES
Defines a list of error codes that, for job-tracking purposes, result in an automatic reset of an operation. The operation is reset to status A (arriving) and contains the message Error, automatically reset in its operation details panel. An error code can be either of the following:

- 4-digit job or started-task return code (*nnnn*)
- System abend code (S*xxx*)
- User abend code (U*xxx*)
- IBM Workload Scheduler for z/OS-defined code

The HIGHRC parameter is supported for positive RC on IBM Workload Scheduler for z/OS Agents and agents. Jobs ending with negative RC are always considered in error. The negative RC are handled via NOERROR. For a detailed description of NOERROR and ERRRES, see *IBM Workload Scheduler for z/OS: Customization and Tuning*.

The following is a list of the return codes for each job type with advanced options:

```
Database jobs:
RC =  0 -> Job completed successfully
RC = -1 -> SQL statement was run with an exit code different from 1
RC = -2 -> MSSQL Job error
RC = -3 -> SQL statement did not run because of an error in the statement

File transfer jobs:
RC =  0 -> The file transfer completed successfully

RC = -1 -> The file transfer is not performed. The job fails with the following
error code: AWKFTE007E

Explanation: An error occured during the file transfer operation

Possible reasons: Remote file not found or permission denied

RC = -2 -> The file transfer is not performed. The job fails with the following
error code: AWKFTE020E

Explanation: Only for SSH or Windows protocols. An error was returned
while attempting to convert the code page

Possible reasons: For SSH or Windows protocols, the code page is
```

automatically detected and converted. In this case, there is an error in the code page of the file to be transferred, which is not compliant with the code page of the local system

RC = -3 -> The file transfer is not performed.The job fails with the following error code: AWKFTE015E

Explanation: An error occurred during the file transfer operation

Possible reasons: Local file is not found

RC = -4 -> The file transfer is performed with the default code page. The job fails with the following error code: AWKFTE023E

Explanation: The specified codepage conversion has not been performed. File transfer has been performed with default code pages

Possible reasons: The specified code page is not available

IBM i jobs:
Return code = user return code when retrieved
Return code = 0 -> job completed successfully
Return code > -1 -> job completed unsuccessfully

Web services jobs:
RC = 0 -> Job completed successfully
RC = -1 -> The server hostname contained in the Web Service URL is unknown
RC = -2 -> Web Service invocation error

Appendix E. Modifying the workstation from fault-tolerant agent to IBM Workload Scheduler for z/OS Agent

About this task

To modify a fault-tolerant agent workstation that uses centralized job scripts to an IBM Workload Scheduler for z/OS Agent workstation, perform the following steps.

Note:

- For jobs whose JOBREC includes an RCCONDSUC keyword, see Appendix D, "Managing return codes," on page 149 for information about how to perform return code management for jobs scheduled on an IBM Workload Scheduler for z/OS Agent.
- To modify a fault-tolerant agent workstation that uses non-centralized job scripts to an IBM Workload Scheduler for z/OS Agent workstation, you can follow the same procedure. Additionally, you must convert the information contained in the EQQSCLIB data set into information for the JOBLIB data set, by using the appropriate "JOBREC" on page 65 statement keywords.

Procedure

1. Define the IBM Workload Scheduler for z/OS Agent destination by setting the ROUTOPTS statement. For details, refer to "Defining destinations with the ROUTOPTS statement" on page 22.
2. Optionally, customize the HTTP connection with the HTTPOPTS statement. For details, refer to "Customizing the HTTP connection with the HTTPOPTS statement" on page 25.
3. From the MODIFYING GENERAL INFORMATION ABOUT A WORK STATION panel, modify the workstation database definition by setting Z-CENTRIC AGENT to `Y`. For details, refer to "Modifying" on page 40.
4. To prevent that message EQQ3036W is issued in the MLOG, comment out the CPUREC statement related to the fault-tolerant workstation you are modifying.

Results

At the next daily planning run, the changes take effect and message EQQ0371I is issued in EQQMLOG:

```
FTA OR Z-CENTRIC AGENT INFO HAS BEEN CHANGED FOR WORKSTATION WSNAME
```

If on the workstation modified there were operations with status S (started), the status is changed to I (interrupted) and message EQQ0372W is issued in EQQMLOG:

```
STATUS OF OPERATION ADID IA OPNO HAS BEEN CHANGED FROM STARTED TO INTERRUPTED
AS A CONSEQUENCE OF WORKSTATION WSNAME CHANGE
```

The DP batch job ends with RC=4.

Appendix F. Defining access methods for agents

Access methods are used to extend the job scheduling functions of IBM Workload Scheduler to other systems and applications. They run on:

Extended agents
: They are logical workstation related to an access method hosted by a physical IBM Workload Scheduler workstation (not another extended agent). More than one extended agent workstation can be hosted by the same IBM Workload Scheduler workstation and use the same access method. The extended agent runs on fault-tolerant agents defined using a standard IBM 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 IBM Workload Scheduler submits a job to an external system.

  Jobs are defined for an extended agent in the same manner as for other IBM Workload Scheduler workstations, except that job attributes are dictated by the external system or application.

  Information about job running execution is sent to IBM Workload Scheduler from an extended agent using the job `stdlist` file. A method options file can specify alternate logins to launch jobs and check *opens* file dependencies. For more information, see the *User's Guide and Reference*.

  A physical workstation can host a maximum of 255 extended agents.

dynamic agents and IBM Workload Scheduler for z/OS agents
: They communicate with external systems to start the job and return the status of the job. To run access methods on external applications using dynamic agents, you define a job of type **access method**.

Access methods are available on the following systems and applications.
- SAP R/3
- z/OS
- Custom methods
- unixssh
- unixrsh
- Local UNIX (fault-tolerant agents only)

The UNIX access methods included with IBM Workload Scheduler, are described in the related section in *Administration Guide*.

If you are working with dynamic agents, for information about defining IBM Workload Scheduler workstations, see the section that explains how to define workstations in the database in *User's Guide and Reference*. For information about writing access methods, see the section about the access method interface in *User's Guide and Reference*.

More information about access methods is found in *Scheduling Applications with IBM Workload Automation*.

Access method interface

The interface between IBM Workload Scheduler and an access method consists of information passed to the method on the command line, and of messages returned to IBM Workload Scheduler in **stdout**.

Method command line syntax

The IBM Workload Scheduler host runs an access method using the following command line syntax:

methodname **-t** *task options* **--** *taskstring*

where:

methodname
Specifies the file name of the access method. All access methods must be stored in the directory: *TWS_home*/methods

-t *task* Specifies the task to be performed, where *task* is one of the following:

LJ Launches a job.

MJ Manages a previously launched job. Use this option to resynchronize if a prior **LJ** task ended unexpectedly.

CF Extended agents only. Checks the availability of a file. Use this option to check file opens dependencies.

GS Extended agents only. Gets the status of a job. Use this option to check job follows dependencies.

options Specifies the options associated with the task. See "Task options" for more information.

taskstring
A string of up to 255 characters associated with the task. See "Task options."

Task options

The task options are listed in Table 10. An X means that the option is valid for the task.

Table 10. Method command task options

Task	-c	-n	-p	-r	-s	-d	-l	-o	-j	-q	-w	-S	Task String
LJ	X	X	X	X	X	X	X	X	X			X	*ljstring*
MJ	X	X	X	X	X	X	X	X	X				*mjstring*
CF	X	X	X							X			*cfstring*
GS	X	X	X	X		X					X		*gsstring*

-c *xagent*,*host*,*master*
Specifies the names of the agent, the host, and the master domain manager separated by commas.

-n *nodename*
Specifies the node name of the computer associated with the agent, if any. This is defined in the extended agent's workstation definition **Node** field.

-p *portnumber*
Specifies the TCP/IP port number associated with the agent, if any. This is defined in the agent workstation definition **TCP Address** field.

-r *currentrun*,*specificrun*
Specifies the current run number of IBM Workload Scheduler and the specific run number associated with the job separated by a comma. The current and specific run numbers might be different if the job was carried forward from an earlier run.

-s *jstream*
Specifies the name of the job's job stream.

-d *scheddate*,*epoch*
Specifies the job stream date (*yymmdd*) and the epoch equivalent, separated by a comma.

-l *user* Specifies the job's user name. This is defined in the job definition **Logon** field.

-o *stdlist*
Specifies the full path name of the job's standard list file. Any output from the job must be written to this file.

-j *jobname*,*id*
Specifies the job's name and the unique identifier assigned by IBM Workload Scheduler, separated by a comma. The name is defined in the job definition **Job Name** field.

-q *qualifier*
Specifies the qualifier to be used in a test command issued by the method against the file.

-w *timeout*
Specifies the amount of time, in seconds, that IBM Workload Scheduler waits to get a reply on an external job before sending a SIGTERM signal to the access method. The default is 300.

-S *new name*
Specifies that the job is rerun using this name in place of the original job name. Within a job script, you can use the `jobinfo` command to return the job name and run the script differently for each iteration.

-- *ljstring*
Used with the **LJ** task. The string from the **Script File** or **Command** field of the job definition.

-- *mjstring*
Used with the **MJ** task. The information provided to the IBM Workload Scheduler by the method in a message indicating a job state change **%CJ** (for additional details on messages indicating job state change, see "Method response messages" on page 156) following to an **LJ** task. Usually, this identifies the job that was launched. For example, a UNIX method can provide the process identification (PID) of the job it launched, which is then sent by the IBM Workload Scheduler as part of an **MJ** task.

-- *cfstring*
Used with the **CF** task. For a file `opens` dependency, the string from the **Opens Files** field of the job stream definition.

-- *gsstring*
Used with the **GS** task. Specifies the job whose status is checked. The format is as follows:

followsjob[,*jobid*]

where:

followsjob
The string from the **Follows Sched/Job** list of the job stream definition.

jobid An optional job identifier received by IBM Workload Scheduler in a **%CJ** response to a previous **GS** task.

Method response messages

Methods return information to IBM Workload Scheduler in messages written to **stdout**. Each line starting with a percent sign (%) and ending with a new line is interpreted as a message. The messages have the following format:

%CJ *state* [*mjstring* | *jobid*]

%JS [*cputime*]

%RC *rc*

%UT [*errormessage*]

where:

CJ Changes the job state.

state The state to which the job is changed. All IBM Workload Scheduler job states are valid except HOLD and READY. For the **GS** task, the following states are also valid:

ERROR
An error occurred.

EXTRN
Status is unknown.

mjstring
A string of up to 255 characters that IBM Workload Scheduler will include in any **MJ** task associated with the job.

jobid A string of up to 64 characters that IBM Workload Scheduler will include in any **GS** task associated with the job.

JS [*cputime***]**
Indicates successful completion of a job and provides its elapsed run time in seconds.

RC *rc* *rc* is a number that is interpreted by IBM Workload Scheduler as the return code of the extended agent job. The return code is taken into account only if a return code condition was specified in the definition of the extended agent job. Otherwise, it is ignored and the successful completion of the extended agent job is indicated by the presence of message **%JS** *[cputime]*.

Likewise, if the method does not send the **%RC** message, then the successful completion of the extended agent job is indicated by the presence of message **%JS** *[cputime]*.

UT [*errormessage***]**
Indicates that the requested task is not supported by the method. Displays a string of up to 255 characters that IBM Workload Scheduler will include in its error message.

Method options file

For extended, agents, and IBM Workload Scheduler for z/OS Agent you can use a method options file to specify login information and other options.

An options file is a text file located in the methods directory of the IBM Workload Scheduler installation, containing a set of options to customize the behavior of the access method. The options must be written one per line and have the following format (with no spaces included):

option=value

All access methods use two types of options files:

Extended agents

Global options file
A common configuration file created by default for each access method installed, whose settings apply to all the extended agent workstations defined for that method. When the global options file is created, it contains only the **LJuser** option, which represents the operating system user ID used to launch the access method. You can customize the global options file by adding the options appropriate to the access method.

Local options file
A configuration file that is specific to each extended agent workstation within a particular installation of an access method. The name of this file is *XANAME_accessmethod*.opts, where:

XANAME
Is the name of the extended agent workstation. The value for *XANAME* must be written in uppercase alphanumeric characters. Double-byte character set (DBCS), Single-byte character set (SBCS), and Bidirectional text are not supported.

accessmethod
Is the name of the access method.

If you do not create a local options file, the global options file is used. Every extended agent workstation, except for z/OS, must have a local options file with its own configuration options.

For example, if the installation of the access method includes two extended agent workstations, CPU1 and CPU2, the names of the local options files are respectively CPU1_*accessmethod*.opts and CPU2_*accessmethod*.opts.

IBM Workload Scheduler reads the options file, if it exists, before running an access method. For extended agents, if the options file is modified after IBM Workload Scheduler is started, the changes take effect only when it is stopped and restarted.

IBM Workload Scheduler for z/OS Agents and agents

Global options file
A common configuration file created by default for each access method installed, whose settings apply to all the agent workstations defined for that method. When the global options file is created, it contains only the **LJuser** option, which represents the operating system user ID used to run the access method. You can customize the global options file by adding the options appropriate to the access method.

The name of the global options file is *accessmethod*.opts, where access method is the name of the method you are creating.

Local options file
A configuration file that is specific to each access method. The name of this file is *optionsfile_accessmethod*.opts,

In a distributed environment:

- If you are defining a job to run the access method by using the Dynamic Workload Console it is the options file you specified in the **New** > **Job definition** > **ERP** > **Access Method** XA Task tab.
- If you are defining the access method by using **composer** it is the options file you specified in the **target** attribute of the job definition.

If you do not create a local options file, the global options file is used.

In a z/OS environment:

- If you are defining a job to run the access method by using the Dynamic Workload Console it is the options file you specified in the **New** > **ERP** > **Access Method** XA Task tab.
- If you are defining the access method by using the **JOBREC** statement it is the name of the workstation where the access method runs.

If you do not create a local options file, the global options file is used.

If you do not specify an option in the *options_file_accessmethod*.opts file the product uses the value specified for that option in the global option file. If you do not specify them neither in the *options_file_accessmethod*.opts file nor in the global option file the product issues an error message.

The options file must have the same path name as its access method, with an .opts file extension. For example, the Windows path name of an options file for a method named netmeth is

TWS_home\methods\netmeth.opts

IBM Workload Scheduler reads the options file, if it exists, before running an access method.

The options recognized by IBM Workload Scheduler are as follows:

LJuser=*username*
Specifies the login to use for the **LJ** and **MJ** tasks. The default is the login from the job definition. See "Launch job task (LJ)" and "Manage job task (MJ)" on page 160.

CFuser=*username*
Extended agents only. Specifies the login to use for the **CF** task. The default for UNIX is **root**, and for Windows is the user name of the account in which the product was installed. See awsrgcheckfiletask.dita.

GSuser=*username*
Specifies the login to use for the **GS** tasks. The default for UNIX is **root**, and for Windows is the user name of the account with which IBM Workload Scheduler was installed. See Get status task (GS) extended agents only.

GStimeout=*seconds*
Specifies the amount of time, in seconds, IBM Workload Scheduler waits for a response before killing the access method. The default is **300** seconds.

nodename=*node_name*
Specifies the host name or IP address if required by the method you are defining. For the **unixssh** access method, the host name or IP address to connect to the remote engine.

PortNumber=*port_number*
Specifies the port number if required by the method you are defining. For the **unixssh** access method, the port to connect to the remote engine.

For IBM Workload Scheduler for z/OS Agents and agents, you can specify the node name and port number also in the `JobManager.ini` file.

If you do not specify them in the *options_file_accessmethod*`.opts` file the product uses the value specified in the global option file. If you do not specify them neither in the *options_file_accessmethod*`.opts` file nor in the global option file the product uses the value specified in the *option_file* stanza of the `JobManager.ini` file.

Note: If the extended agent host is a Windows computer, these users must be defined as IBM Workload Scheduler user objects.

Running methods

The following subsections describe the interchange between IBM Workload Scheduler and an access method.

Launch job task (LJ)

About this task

The **LJ** task instructs the extended agent method to launch a job on an external system or application. Before running the method, IBM Workload Scheduler establishes a run environment. The **LJuser** parameter is read from the method options file to determine the user account with which to run the method. If the parameter is not present or the options file does not exist, the user account specified in the **Logon** field of the job's definition is used. In addition, the following environment variables are set:

HOME
The login user's home directory.

LOGNAME
The login user's name.

PATH For UNIX, it is set to`/bin:/usr/bin`. For Windows, it is set to`%SYSTEM%\SYSTEM32`.

TWS_PROMOTED_JOB
Set to `YES`, when the job (a mission-critical job or one of its predecessors) is promoted.

TZ The time zone.

If the method cannot be run, the job is placed in the FAIL state.

Once a method is running, it writes messages to its **stdout** that indicate the state of the job on the external system. The messages are summarized in Table 11.

Table 11. Launch job task (LJ) messages

Task	Method Response	IBM Workload Scheduler Action
LJ and **MJ**	**%CJ** *state* [*mjstring*]	Sets job state to *state*. Includes *mjstring* in any subsequent **MJ** task.
	%JS [*cputime*]	Sets job state to SUCC.
	Exit code=non-zero	Sets job state to ABEND.
	%UT [*errormessage*] and Exit code=2	Sets job state to ABEND and displays *errormessage*.

A typical sequence consists of one or more **%CJ** messages indicating changes to the job state and then a **%JS** message before the method exits to indicate that the job ended successfully. If the job is unsuccessful, the method must exit without writing the **%JS** message. A method that does not support the **LJ** task, writes a **%UT** message to **stdout** and exits with an exit code of **2**.

Manage job task (MJ)

About this task

The **MJ** task is used to synchronize with a previously launched job if IBM Workload Scheduler determines that the **LJ** task ended unexpectedly. IBM Workload Scheduler sets up the environment in the same manner as for the **LJ** task and passes it the *mjstring*. See “Launch job task (LJ)” on page 159 for more information.

If the method locates the specified job, it responds with the same messages as an **LJ** task. If the method is unable to locate the job, it exits with a nonzero exit code, causing IBM Workload Scheduler to place the job in the ABEND state.

Killing a job

About this task

While an **LJ** or **MJ** task is running, the method must trap a SIGTERM signal (signal **15**). The signal is sent when an operator issues a **kill** command from IBM Workload Scheduler console manager. Upon receiving the signal, the method must attempt to stop (**kill**) the job and then exit without writing a **%JS** message.

Appendix G. Collecting job metrics

You can run the following SQL queries on the Workload Scheduler data base to retrieve the number of jobs run by IBM Workload Scheduler over a period of time. One query determines the number of jobs run by specific workstations, while the other query determines the number of jobs run on the entire IBM Workload Scheduler domain. You can run the queries from the command line interface of your database or you can add them in the Dynamic Workload Console to create your custom SQL report tasks.

Job metrics queries for DB2 for zOS

Use the following SQL query to find the number of jobs run on specific workstations:

```
SELECT year(job_run_date_time) AS Year, month(job_run_date_time) AS Month,
cast (count(job_run_date_time) AS INT) AS JobNbr FROM mdl.job_history_v
WHERE workstation_name IN ('WKS_1', 'WKS_2', 'WKS_N')
GROUP BY year(job_run_date_time), month(job_run_date_time)
```

where 'WKS_1', 'WKS_2', 'WKS_N' are the names of the workstations that ran the jobs you want counted.

Use the following SQL query to find the number of jobs run on the entire IBM Workload Scheduler domain:

```
SELECT year(job_run_date_time) AS Year, month(job_run_date_time) AS Month,
cast (count(job_run_date_time) AS INT) AS JobNbr FROM mdl.job_history_v
GROUP BY year(job_run_date_time), month(job_run_date_time)
```

Job metrics queries for DB2

Use the following SQL query to find the number of jobs run on specific workstations:

```
SELECT year(job_run_date_time) AS Year, month(job_run_date_time) AS Month,
cast (count(job_run_date_time) AS INT) AS JobNbr FROM mdl.job_history_v
WHERE workstation_name IN ('WKS_1', 'WKS_2', 'WKS_N') or
(workstation_name = '-' and JOB_STREAM_WKS_NAME_IN_RUN in('WKS_1', 'WKS_2', 'WKS_N') )
GROUP BY year(job_run_date_time), month(job_run_date_time)
```

where 'WKS_1', 'WKS_2', 'WKS_N' are the names of the workstations that ran the jobs you want counted.

Use the following SQL query to find the number of jobs run on the entire IBM Workload Scheduler domain:

```
SELECT year(job_run_date_time) AS Year, month(job_run_date_time) AS Month,
cast (count(job_run_date_time) AS INT) AS JobNbr FROM mdl.job_history_v
GROUP BY year(job_run_date_time), month(job_run_date_time)
```

Job metrics queries for Oracle database

Use the following SQL query to find the number of jobs run on specific workstations:

```
SELECT EXTRACT(year FROM job_run_date_time) AS Year,
EXTRACT(month FROM job_run_date_time) AS Month,
cast (count(job_run_date_time) AS INT) AS JobNbr FROM job_history_v
WHERE workstation_name IN ('WKS_1', 'WKS_2', 'WKS_N')
or (workstation_name = '-' and JOB_STREAM_WKS_NAME_IN_RUN in('WKS_1', 'WKS_2', 'WKS_N'))
GROUP BY EXTRACT(year FROM job_run_date_time), EXTRACT(month FROM job_run_date_time);
```

where 'WKS_1', 'WKS_2', 'WKS_N' are the names of the workstations that ran the jobs you want counted.

Use the following SQL query to find the number of jobs run on the entire IBM Workload Scheduler domain:

```
SELECT EXTRACT(year FROM job_run_date_time) AS Year,
EXTRACT(month FROM job_run_date_time) AS Month,
cast (count(job_run_date_time) AS INT) AS JobNbr FROM job_history_v
GROUP BY EXTRACT(year FROM job_run_date_time), EXTRACT(month FROM job_run_date_time);
```

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document 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 document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (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.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

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 jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may 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 websites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

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 document 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.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

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 is for planning purposes only. The information herein is subject to change before the products described become available.

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 actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

ITIL is a Registered Trade Mark of AXELOS Limited.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Index

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

X

Z

Product Number: 5698-T08

Printed in USA