

**IBM® Z Workload Scheduler**  
**Scheduling End-to-end with Fault Tolerance Capabilities**  
**Version 9.5 (Revised March 2024)**

## Note

Before using this information and the product it supports, read the information in [Notices on page cxxxvi](#).

This edition applies to version 9, release 5, modification level 0 of IBM® Z Workload Scheduler (program number 5698-T09) and to all subsequent releases and modifications until otherwise indicated in new editions.

# Contents

List of Figures.....	vi	Tuning global and local options in distributed network.....	98
List of Tables.....	vii	Tuning network timeouts.....	99
About this publication.....	viii	Selecting end-to-end server messages.....	101
What is new in this release.....	viii	Rules for customizing the CCLog properties file.....	101
Who should read this publication.....	viii	Sample TWSCCLog.properties customization.....	102
Accessibility .....	viii	<b>Chapter 4. Migrating.....</b>	<b>104</b>
Technical training.....	ix	Migrating from a tracker agent to a distributed agent.....	104
Support information.....	ix	Using the tracker jobs migration tool.....	105
Conventions.....	ix	Migrating backwards.....	106
How to read syntax diagrams.....	ix	Migrating from distributed to end-to-end with fault tolerance capabilities environment.....	106
<b>Chapter 1. Overview.....</b>	<b>13</b>	Migrating the agents.....	107
End-to-end with fault tolerance capabilities environment.....	13	Migrating the job scripts.....	107
The IBM Z Workload Scheduler end.....	15	Migrating the database objects.....	108
The IBM Workload Scheduler end.....	15	Migrating from centralized to non-centralized jobs.....	108
Distributed agents.....	15	Changing the topology of the network.....	108
IBM Workload Scheduler domains.....	16	<b>Chapter 5. Managing end-to-end scheduling.....</b>	<b>111</b>
End-to-end scheduling with fault tolerance capabilities architecture.....	16	Defining fault-tolerant workstation jobs.....	111
End-to-end with fault tolerance capabilities functions.....	21	Defining file dependencies to check for file changes.....	112
Integration with IBM Workload Scheduler.....	22	Syntax.....	112
The Symphony file.....	22	Arguments.....	112
Basic server configuration example.....	23	Maintaining the filewatch log database.....	114
Functional comparison between end-to-end and single scheduling environments.....	26	Usage Notes.....	116
Functional comparison with IBM Z Workload Scheduler tracker agents.....	26	Examples.....	116
Functional comparison with IBM Workload Scheduler.....	26	Creating and activating the current plan.....	117
Terminology mapping.....	28	Creating and activating the new current plan....	118
<b>Chapter 2. Installing.....</b>	<b>30</b>	Renewing the Symphony file.....	119
Installing the IBM Z Workload Scheduler end.....	30	Recovering from missing or corrupted Symphony file.....	120
Installation prerequisites.....	30	Managing Windows users and passwords locally.....	122
Installation checklist.....	30	Fault-tolerant workstations and replanning.....	124
Installation details.....	30	Controlling the distributed plan.....	125
Installing the IBM Workload Scheduler end.....	64	Conventions used for the end-to-end scheduling plan.....	126
Installation checklist.....	65	Controlling the plan with conman.....	126
Installation details.....	65	Controlling the plan with the Dynamic Workload Console.....	128
<b>Chapter 3. Customizing.....</b>	<b>72</b>	Killing jobs on standard agents.....	129
Defining centralized and non-centralized scripts.....	72	Sizing event files on USS.....	131
Centralized scripts.....	72	Enabling time zones.....	133
Non-centralized scripts.....	73	Switching to a backup domain manager.....	133
Configuring the SCRPTLIB.....	73	Collecting job metrics.....	134
Setting the security features.....	85		
Setting strong authentication and encryption.....	85		
Setting for work across firewalls.....	97		

Job metrics queries for DB2 for zOS.....	134
Job metrics queries for DB2.....	134
Job metrics queries for Oracle database.....	134
Notices.....	cxxxvi
Index.....	140

# List of Figures

Figure 1: End-to-end with fault-tolerant capabilities environment..... 14

Figure 2: Example of end-to-end with fault tolerance capabilities configuration.....18

Figure 3: A basic configuration example..... 24

Figure 4: EQQJOBS8 - Create sample job JCL.....31

Figure 5: EQQJOBSA - Generate IBM Z Workload Scheduler batch-job skeletons..... 33

Figure 6: Configuration parameters used for end-to-end scheduling with fault tolerance capabilities..... 41

Figure 7: EQQWMLSL - List of workstation descriptions.... 62

Figure 8: EQQWCGEP - Creating general information about a workstation..... 63

Figure 9: Defining a job for end-to-end scheduling.....111

Figure 10: EQQDPLNP - Producing daily plans..... 119

Figure 11: EQQMWSLL - Modifying work stations in the current plan.....124

Figure 12: EQQMWSTP - Modifying the status of a fault-tolerant workstation in the current plan.....125

Figure 13: EQQMOPRL - Modifying operations in the current plan..... 130

Figure 14: EQQREINP - Operation recovery info..... 130

## List of Tables

Table 1: z/OS® memory requirements for directly connected agents and mailman servers.....	19
Table 2: Guidelines for choosing agent types.....	20
Table 3: Dynamic Workload Console terminology mapping.....	28
Table 4: Sample JCL for end-to-end scheduling with fault tolerance capabilities generated by the EQQJOBS8 dialog.....	32
Table 5: CPUREC parameters.....	68
Table 6: DOMREC parameters.....	69
Table 7: USRREC parameters.....	70
Table 8: Settings for fault-tolerant workstations.....	70
Table 9: Comparison operators.....	79
Table 10: Logical operators.....	80
Table 11: Comparison operators.....	83
Table 12: Logical operators.....	83
Table 13: Files for local options.....	90
Table 14: Type of communication depending on the securitylevel value.....	91
Table 15: Recommended localopts settings on distributed agents.....	98
Table 16: Network timeout settings in localopts.....	100
Table 17: Authorizations needed to run the users utility...	122
Table 18: Conman commands for controlling the plan.....	127

## About this publication

*Scheduling End-to-end with Fault Tolerance Capabilities* explains how to set up IBM Z Workload Scheduler and IBM Workload Scheduler to generate an end-to-end scheduling environment. While it exhaustively describes the required installation and customization tasks, this first edition provides only miscellaneous notes on the subject of scheduling end-to-end. 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:

- *Managing the Workload* for information about scheduling from the IBM Z Workload Scheduler interfaces.
- *Dynamic Workload Console User's Guide* for information about scheduling from the Dynamic Workload Console.
- *Messages and Codes* for message information.
- *Troubleshooting Guide* for troubleshooting information.
- *Memo to Users* for interoperability information among component versions.
- [IBM Z Workload Scheduler: 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 *IBM Workload Automation: Overview*, section *Summary of enhancements*.

For information about the APARs that this release addresses, see the IBM Workload Scheduler Release Notes at [IBM Workload Scheduler Release Notes](#) and the Dynamic Workload Console Release Notes at [Dynamic Workload Console Release Notes](#). For information about the APARs addressed in a fix pack, refer to the readme file for the fix pack.

New or changed content is marked with revision bars.

## Who should read this publication

This publication is intended for IBM Z Workload Scheduler users who are already familiar with IBM® IBM Workload Scheduler. A background knowledge of IBM Z Workload Scheduler is required. Background knowledge of IBM Workload Scheduler is also recommended.

## 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.

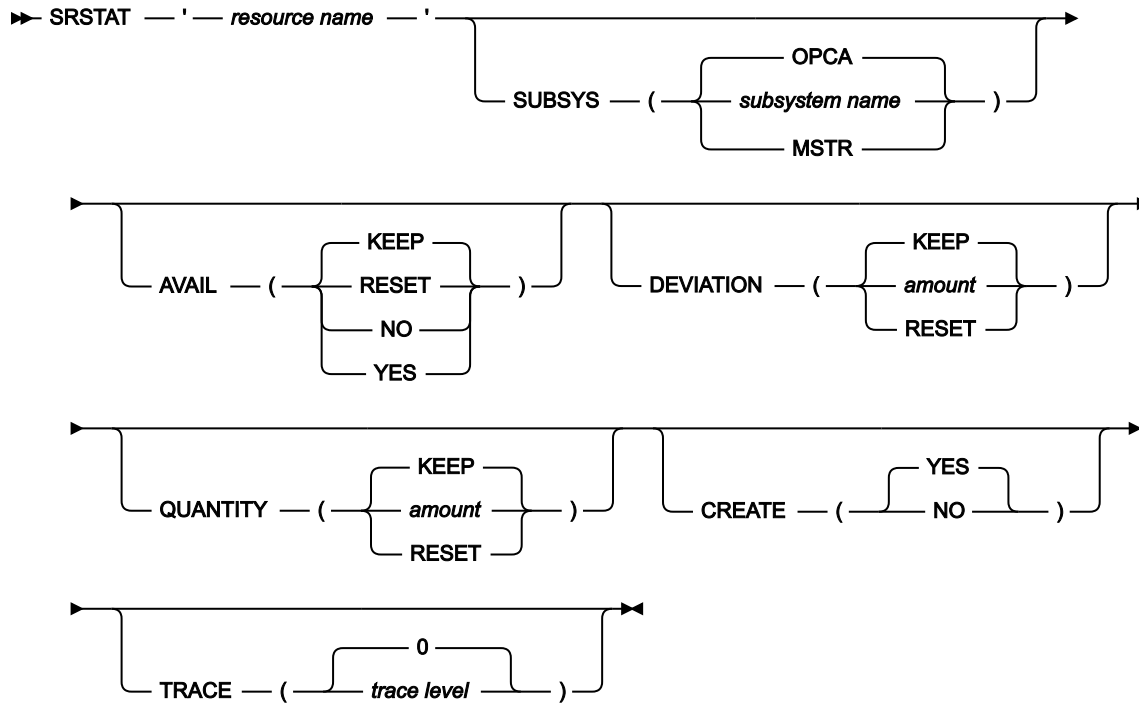
The publication uses several typeface conventions for special terms and actions. Technical changes to the text are indicated by a vertical line to the left of the change. These conventions have the following meanings:

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	<i>Application</i>

## 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:



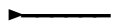
The symbols have these meanings:



The statement begins here.



The statement is continued on the next line.



The statement is continued from a previous line.

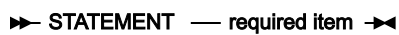


The statement ends here.

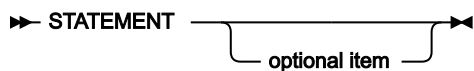
Read the syntax diagrams from left to right and from top to bottom, following the path of the line.

These are the conventions used in the diagrams:

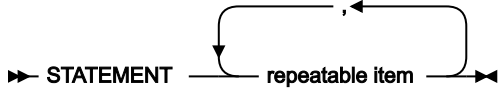
- Required items appear on the horizontal line (main path):



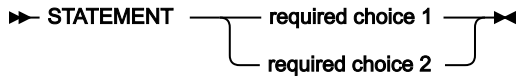
- Optional items appear below the main path:



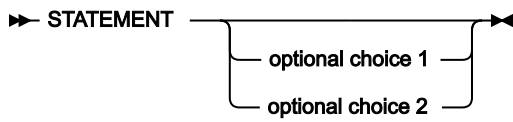
- An arrow returning to the left above the item indicates an item that you can repeat. If a separator is required between items, it is shown on the repeat arrow.



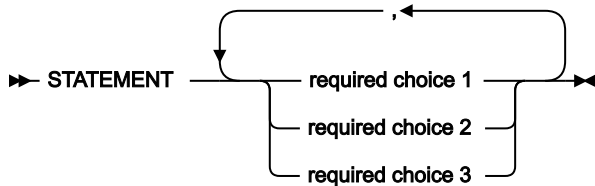
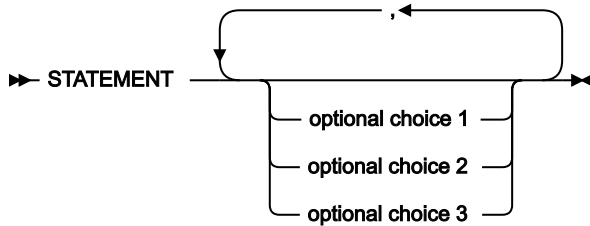
- If you can choose from two or more items, they appear vertically in a stack.
  - If you must choose one of the items, one item of the stack appears on the main path:



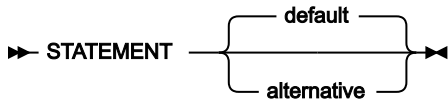
- If choosing one of the items is optional, the entire stack appears below the main path:



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



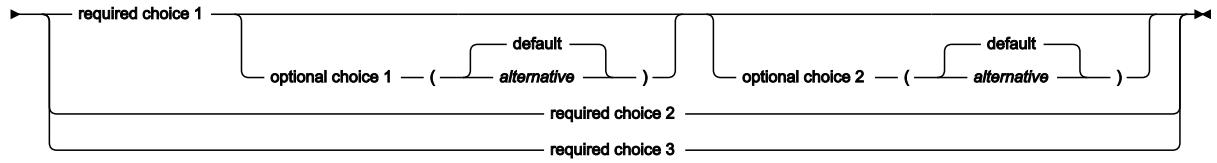
- Parameters that are above the main line are default parameters:



- 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:

► STATEMENT ►



# Chapter 1. Overview

An overview of end-to-end scheduling with fault tolerance capabilities.

The end-to-end scheduling with fault tolerance capabilities is an additional configuration of IBM Z Workload Scheduler and IBM Workload Scheduler. By configuring this environment, you can use the two schedulers together to schedule workloads from your z/OS® system and run jobs on different platforms. This means that from the mainframe you can prepare and run scheduling plans with jobs that will be running on operating systems other than z/OS®. You do this by connecting your IBM Workload Scheduler domains, fault-tolerant workstations, or standard agents to your IBM Z Workload Scheduler controller. In this way, you can schedule and monitor jobs in the distributed environment from your scheduler on z/OS®, either through ISPF panels or through the Dynamic Workload Console, a web user interface.

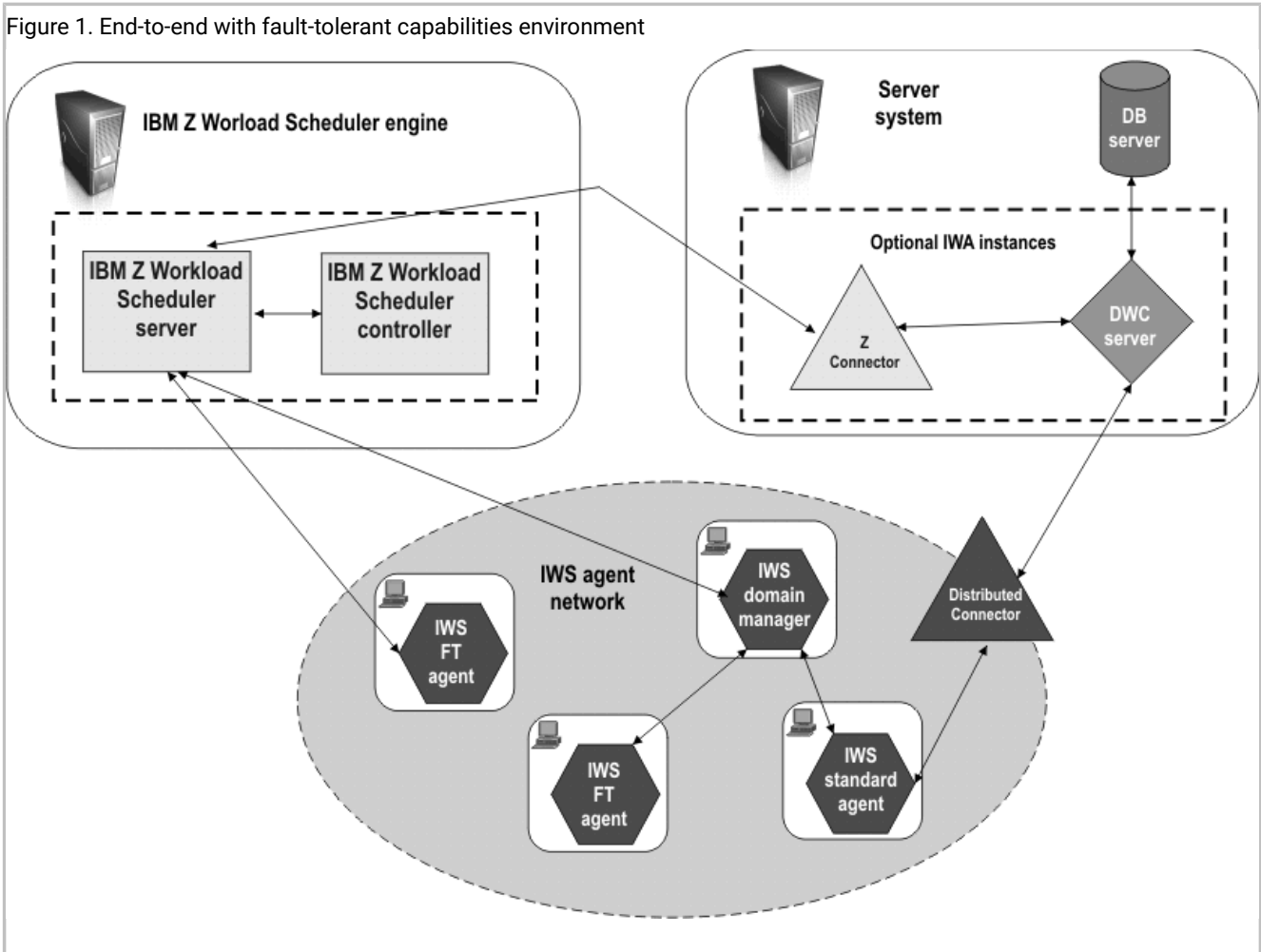
This configuration increases the scheduling options available to you, giving you additional choice in designing both your physical and your organizational workload scheduling structure. You must install and configure both IBM Z Workload Scheduler and IBM Workload Scheduler to do this type of scheduling.

## The end-to-end with fault tolerance capabilities environment

End-to-end with fault tolerance capabilities environment.

For end-to-end scheduling, IBM Z Workload Scheduler and IBM Workload Scheduler work together to schedule jobs. The architecture of the interaction between the two environments includes elements from both schedulers.

[Figure 1: End-to-end with fault-tolerant capabilities environment on page 14](#) shows two IBM Workload Scheduler agents installed on the distributed system as fault-tolerant agents. In an end-to-end with fault tolerance capabilities environment, the IBM® Z Workload Scheduler server component becomes a required component. It is also possible to use the Dynamic Workload Console as a graphical user interface to agents on both the distributed and z/OS® systems. To use Dynamic Workload Console, some optional components are required.



This environment increases the scheduling possibilities of your IBM Z Workload Scheduler system, because you can attach an IBM Workload Scheduler network, or individual IBM Workload Scheduler agents, and so have:

- Multiple platforms and operating systems other than z/OS® capable of running workload that is managed by IBM Z Workload Scheduler
- The possibility to have an unlimited number of linked IBM Workload Scheduler agents or sub-domains

To accomplish this, you must:

1. Configure the end-to-end scheduling with fault tolerance capabilities environment on IBM Z Workload Scheduler.
2. Connect one or more IBM Workload Scheduler standard agents, fault-tolerant agents, or domain managers to the IBM Z Workload Scheduler engine.
3. Define the IBM Z Workload Scheduler engine as the master domain manager on the IBM Workload Scheduler domain managers.

## The IBM Z Workload Scheduler end

The main characteristics of the fault-tolerant end-to-end scheduling on the IBM Z Workload Scheduler system is the addition of the end-to-end server, a started task on the mainframe that works closely with the controller.

The main characteristics of the fault-tolerant end-to-end scheduling are listed as follows:

The end-to-end server:

- Acts as an intermediary between the IBM Z Workload Scheduler controller and the IBM Workload Scheduler network
- Acts as the IBM Workload Scheduler master domain manager
- Is the focal point for all communication with the IBM Workload Scheduler distributed network
- Receives the portion of the IBM Z Workload Scheduler current plan that contains the distributed jobs and job streams
- Runs the IBM Workload Scheduler code in USS (UNIX™ System Services)
- Communicates with the IBM Z Workload Scheduler engine using inbound and outbound queues

In fact, the end-to-end server replicates most of the IBM Workload Scheduler native processes.

## The IBM Workload Scheduler end

The following is a description of the IBM Workload Scheduler distributed environment.

### Distributed agents

A distributed agent is a computer that is part of an IBM Workload Scheduler network on which you can schedule jobs from IBM Z Workload Scheduler.

Distributed agents are:

#### **Domain Manager**

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

#### **Backup Domain Manager**

A fault-tolerant agent capable of assuming the responsibilities of its domain manager.

#### **Fault-tolerant Agent (FTA)**

A workstation capable of resolving local dependencies and of launching its jobs in the absence of a domain manager connection.

#### **Standard Agent**

A workstation that launches jobs only under the direction of its domain manager.

#### **Extended Agent**

A logical workstation you use to launch and control jobs on other systems and applications, such as JES2 or JES3, or IBM Z Workload Scheduler, CA7, Peoplesoft, Oracle Applications, and SAP R/3.

Distributed agents replace tracker agents in IBM Z Workload Scheduler. Using the distributed agents you can schedule on non-z/OS systems with a more reliable, fault-tolerant, and scalable agent.

In the IBM Z Workload Scheduler plan, the logical representation of a distributed agent is called a fault-tolerant workstation.

## Distributed agent installation

The complete installation process for a distributed agent is explained in detail in the IBM Workload Scheduler *Planning and Installation Guide*.

Remember to do the following, when you install a distributed agent that you are going to use in the fault-tolerant end-to-end scheduling:

- Use a four-letter name when you define the distributed agent on IBM Workload Scheduler. This name should be the same as the name of the corresponding fault-tolerant workstation in the IBM Z Workload Scheduler network.

## IBM Workload Scheduler domains

An IBM Workload Scheduler network consists of at least one domain, the master domain, in which the master domain manager is the management hub.

Additional domains can be used to divide a widely distributed network into smaller, locally managed groups.

In a single domain configuration, the master domain manager maintains communications with all of the workstations in the IBM Workload Scheduler network.

In a multi-domain configuration, the master domain manager communicates with the workstations in its domain, and subordinate domain managers. The subordinate domain managers, in turn, communicate with the workstations in their domains and subordinate domain managers. Using multiple domains reduces the amount of network traffic by reducing the communications between the master domain manager and other computers. Multiple domains also provide fault-tolerance by limiting the problems caused by losing a domain manager to a single domain. To limit the effects further, you can designate backup domain managers to take over if their domain managers fail.

When you define a new domain, you must identify the parent domain and the domain manager. The parent domain is the domain directly above the new domain in the domain hierarchy. All communications to and from a domain are routed through the parent domain manager.

The domain topology is described in a PARMLIB (DD name is EQQPARM) member with the workstation definitions; they are read by the batch job that creates the symphony file.

## End-to-end scheduling with fault tolerance capabilities architecture

End-to-end scheduling with fault tolerance capabilities architecture.

The end-to-end scheduling with fault tolerance capabilities is implemented by directly connecting one or more IBM Workload Scheduler agents or domain managers to IBM Z Workload Scheduler.



The domain managers function as the broker systems for the entire distributed network by resolving all dependencies for their subordinate managers and agents. They send their updates (in the form of events) to IBM Z Workload Scheduler so that it can update the plan accordingly.

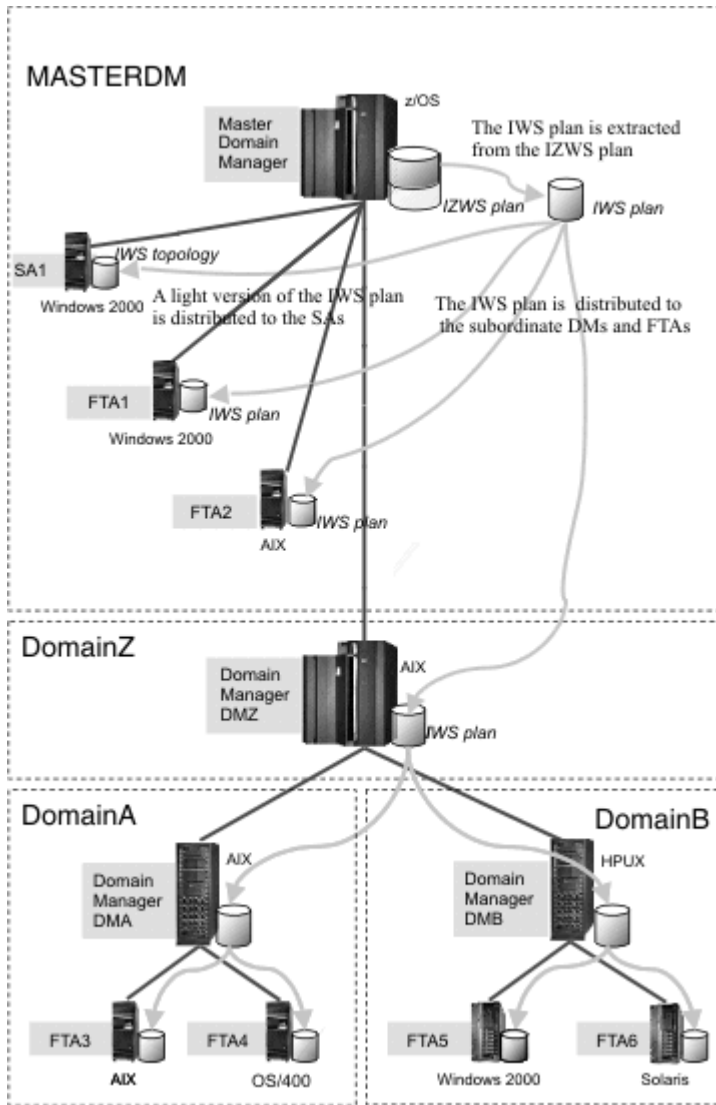
IBM Z Workload Scheduler handles its own jobs and notifies the domain managers of all the status changes of the IBM Z Workload Scheduler jobs that involve the IBM Workload Scheduler plan.

In this configuration the domain managers, and all the distributed agents, recognize IBM Z Workload Scheduler as the master domain manager and notify it of all the changes occurring in their own plan.

Also, the agents are not permitted to interfere with the IBM Z Workload Scheduler jobs, because they are viewed as running on the master that is the only node that manages them.

[Figure 2: Example of end-to-end with fault tolerance capabilities configuration on page 18](#) shows an example of a fault-tolerant end-to-end configuration. It also describes the distribution of the Symphony file from the master domain manager to directly connected agents and to domain managers and their subordinate agents.

Figure 2. Example of end-to-end with fault tolerance capabilities configuration



Before the start of a production period, the master domain manager creates a production control file, named Symphony. This file contains:

- Jobs to be run on IBM Workload Scheduler distributed agents
- z/OS® (mainframe) jobs that are predecessors to IBM Workload Scheduler distributed jobs
- Job streams that have at least one job in the Symphony file

IBM Workload Scheduler is then restarted in the network, and the master domain manager sends a copy of the new Symphony file to each of its agents and subordinate domain managers. The domain managers, in turn, send copies to their agents and subordinate domain managers. This enables the distributed agents throughout the network to continue processing even if the network connection to their domain managers is down.

Stopping the network is necessary to allow for the generation and distribution of the new Symphony file. Optionally, you can configure the end-to-end environment with fault tolerance capabilities to reduce the downtime of each agent to that strictly necessary to receive the new plan by gradually stopping the network. For details, see the `POSTPONE` keyword of the `TOPOLOGY` on page 51 statement.

The end-to-end process is viewed by IBM Z Workload Scheduler as a subtask with a task ID named `TWS`. It is activated by the `TPLGYSRV` keyword of the `OPCOPTS` statement. It handles events to and from fault-tolerant workstations (using the IBM Z Workload Scheduler server).

## Direct agent connections

Standard agents can be connected directly to `OPCMASTER`. The jobs that are scheduled to run on these workstations are added to the Symphony file and managed by the `batchman` component of the end-to-end server.

Because standard agents are not fault-tolerant, when they are linked directly to `OPCMASTER` and active, the controller decides when their jobs must start. This allows for lighter configurations when your business needs do not require fault-tolerance on the distributed part and you want to keep the architecture of your distributed agents at a simple level.

Non-full status fault-tolerant agents also can be connected directly to `OPCMASTER`. They can be used together with `mailman` servers on the end-to-end server to improve performance. In fact, you can allocate dedicated mailman servers to critical standard or fault-tolerant agents that require extra communication resources (this feature is available also for agents not linked directly to the end-to-end server). For details about how to configure extra mailman servers, see the `CPUSERVER` parameter of the `CPUREC` statement in *Customization and Tuning*.

There is no set limit on the maximum number of direct connections supported. The number of direct connection and mailman server definitions and the required memory allocation on the end-to-end server have been found to increase in a straight line. Benchmarking tests run on z/OS® version 1.7 have yielded the following results:

**Table 1. z/OS® memory requirements for directly connected agents and mailman servers**

Number of directly connected fault-tolerant and standard agents	Number of mailman servers	Size of the memory region used on the end-to-end server (MB)
100	10	85
200	20	144

The average number of mailman servers in the test configurations was one for every ten agents. The effective number in your configuration must be calibrated to respond to real needs, such as:

- Compensating for particularly slow TCP/IP connections
- Giving an extra resource to agents that run particularly critical workloads
- Reducing the impact on the network caused by connection losses between the end-to-end server mailman and a particular unstable agent

Mailman servers should not be abused of, since they are additional processes with dedicated message queues that are bound to increase processing and disk I/O.

## Choosing the agents

What are the advantages of connecting agents directly over setting up domains? And when should standard agents be preferred to fault-tolerant ones?

The following table provides a set of guidelines to help you decide how to set up the distributed part of the end-to-end with fault tolerance capabilities network.

**Table 2. Guidelines for choosing agent types**

Using these agent types	is best when you want to...
Standard agents in direct connection to the end-to-end server	<ul style="list-style-type: none"> <li>• Quickly migrate tracker agents.</li> <li>• Be sure that the status displayed in ISPF (menu 5.5) is the real one, since no intermediate agents can hinder the transmission of events to the end-to-end server. This is particularly important when you schedule jobs with centralized scripts, since the scripts are not downloaded to the agent unless it is properly linked.</li> </ul> <p style="margin-left: 40px;">This applies also when a job uses special resources. If the agent is offline, the resource is allocated, but the job does not start until the agent is available again.</p> <ul style="list-style-type: none"> <li>• Manage jobs (with either centralized or non-centralized scripts) from the mainframe. An additional advantage is due to the fact that downloading the Symphony file takes less time, since standard agents use a reduced version of this file.</li> </ul>
Fault-tolerant agents in direct connection to the end-to-end server	<ul style="list-style-type: none"> <li>• Take advantage of fault-tolerance. In addition, you can use <code>conman</code> locally to monitor how the plan is running.</li> <li>• Run primarily jobs with non-centralized scripts and can do without centralized control from the mainframe.</li> <li>• Be sure that the status displayed in ISPF (menu 5.5) is the real one, since no intermediate agents can hinder the transmission of events to the end-to-end server.</li> </ul>
Domain managers with domains	<ul style="list-style-type: none"> <li>• Set up a complex network with many agents.</li> <li>• Rationalize the network topology by lines of business.</li> <li>• Guarantee more dependable fault-tolerance.</li> <li>• Use backup domain managers.</li> <li>• Make sure that problems affecting a part of the network are not extended to the entire network.</li> </ul>

**Table 2. Guidelines for choosing agent types (continued)**

Using these agent types	is best when you want to...
	<ul style="list-style-type: none"> <li>• Reduce to the least possible the network <i>down</i> time while the Symphony file is downloaded. To do so, you can use the <code>TOPOLOGY POSTPONE</code> keyword in conjunction with the <code>mm start tomserver</code> local option on the domain managers.</li> </ul>

## End-to-end with fault tolerance capabilities functions

Some actions that can be performed with end-to-end with fault tolerance capabilities.

With the current versions of IBM Z Workload Scheduler and IBM Workload Scheduler, you can use end-to-end with fault tolerance capabilities scheduling to perform the following actions:

- Use centralized scripts for the distributed agents. The scripts are stored in the `JOBLIB` data set and you use them to perform variable substitution, automatic recovery, JCL editing, and job setup (as for the jobs in the `JOBLIB`). However, this implies a loss of fault-tolerance and requires downloading the script to the agent every time the job is submitted. You need to weigh the benefits and disadvantages of using centralized scripts.
- Use syntax to recover non-centralized jobs on distributed agents automatically when the jobs abend. This recovery is similar to the recovery in the distributed environment.
- Use variable substitution for jobs that run on fault-tolerant workstations and that do not use centralized scripts. IBM Workload Scheduler supplied-variable and user-defined variable tables are supported.
- On small networks, connect fault-tolerant and standard agents directly to IBM Z Workload Scheduler without using intermediate domain managers.
- When needed, use more `mailman` server sub-processes for the workstations directly connected to IBM Z Workload Scheduler.
- Enable firewall support and SSL authentication.
- Specify a success condition for each job that determines whether a job is considered successful or abended.
- Specify or modify the deadline time for a job or a job stream.
- Use new log files for netman, batchman, mailman, the writer, and the translator.
- Connect multiple domain managers to the controller. This allows greater flexibility, scalability, and improved performance.
- Make an application or an operation dependent on the existence of one or more files before it can begin to run.
- Activate and deactivate the submission of jobs scheduled in an end-to-end with fault tolerance capabilities environment.
- Enable or disable the auditing trail of a plan.
- Change the status of distributed agents and switch their domain manager.
- Use the ISPF panels to produce an APAR tape that collects end-to-end data.
- Use the Tracker Jobs Migration Tool to migrate from tracker agents to distributed agents.

## Integration with IBM Workload Scheduler

The end-to-end scheduling with fault tolerance capabilities is based on the Common Agent Technology and it enables IBM Z Workload Scheduler to be the master of an IBM Workload Scheduler distributed network. You configure this by connecting one or more IBM Workload Scheduler agents or domain managers directly to IBM Z Workload Scheduler.

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

The fault-tolerant agents can independently continue with scheduling when communications with IBM Z Workload Scheduler are interrupted due to network problems. At the same time, the distributed agents are prevented from acting on IBM Z Workload Scheduler jobs because these are viewed as running on the master, that is the only node that is authorized to operate on those jobs.

Standard agents are not fault-tolerant. Their ability to run jobs depends on a working connection with their domain manager. A standard agent that is linked directly to `OPCMASTER`, must have a working connection with the end-to-end server to be able to run jobs.

A new type of CPU, named fault-tolerant workstation, is added to IBM Z Workload Scheduler to logically define in the database and in the plan the distributed agents that will be running jobs for IBM Z Workload Scheduler.

## The Symphony file

In a native IBM Workload Scheduler network, the Symphony file, also known as the production control file, is the daily plan. The Symphony file contains the scheduling information needed by the production control process named `batchman`. It is typically created during the pre-production phase before the start of each new planning period by a process named `Jnextplan` on the master domain manager.

At the start of each new planning period, the IBM Workload Scheduler network is restarted and the master domain manager sends a copy of the new Symphony file to each of its automatically linked agents and subordinate domain managers. The domain managers, in turn, send copies to their automatically linked agents and subordinate domain managers.

Once the network is started, scheduling messages such as job starts and completions are passed from the agents to their domain managers, through the parent domain managers to the master domain manager. The master domain manager then broadcasts the messages throughout the hierarchical tree to update the Symphony files of domain managers and fault-tolerant agents running in Full Status mode.

In an end-to-end with fault tolerance capabilities environment, the IBM Z Workload Scheduler engine, in its role as master domain manager of the distributed network, creates the Symphony file from the IBM Z Workload Scheduler current plan. In fact, the server parses the current plan, extracts all the job streams that were defined to run on fault-tolerant workstations, and copies them into the Symphony file of the next production cycle. It then sends the plan down to the first-level domain managers and to the directly connected fault-tolerant agents. In turn, each of these domain managers sends the plan to all of the subordinate workstations in its domain.

Standard agents receive a light version of the Symphony file containing only the following definitions:

- Domains
- workstations
- users

The information on the jobs to run is sent directly by their domain manager.

Standard agents that are directly connected to IBM Z Workload Scheduler receive a light version of the Symphony file containing only domain, workstation, and user definitions. The information on the jobs to run is sent directly by the IBM Z Workload Scheduler controller.

If you have only standard agents connected directly to IBM Z Workload Scheduler, and no fault-tolerant ones, the Symphony file each agent receives is further reduced to only the identification of `OPCMaster` and of the agent itself.

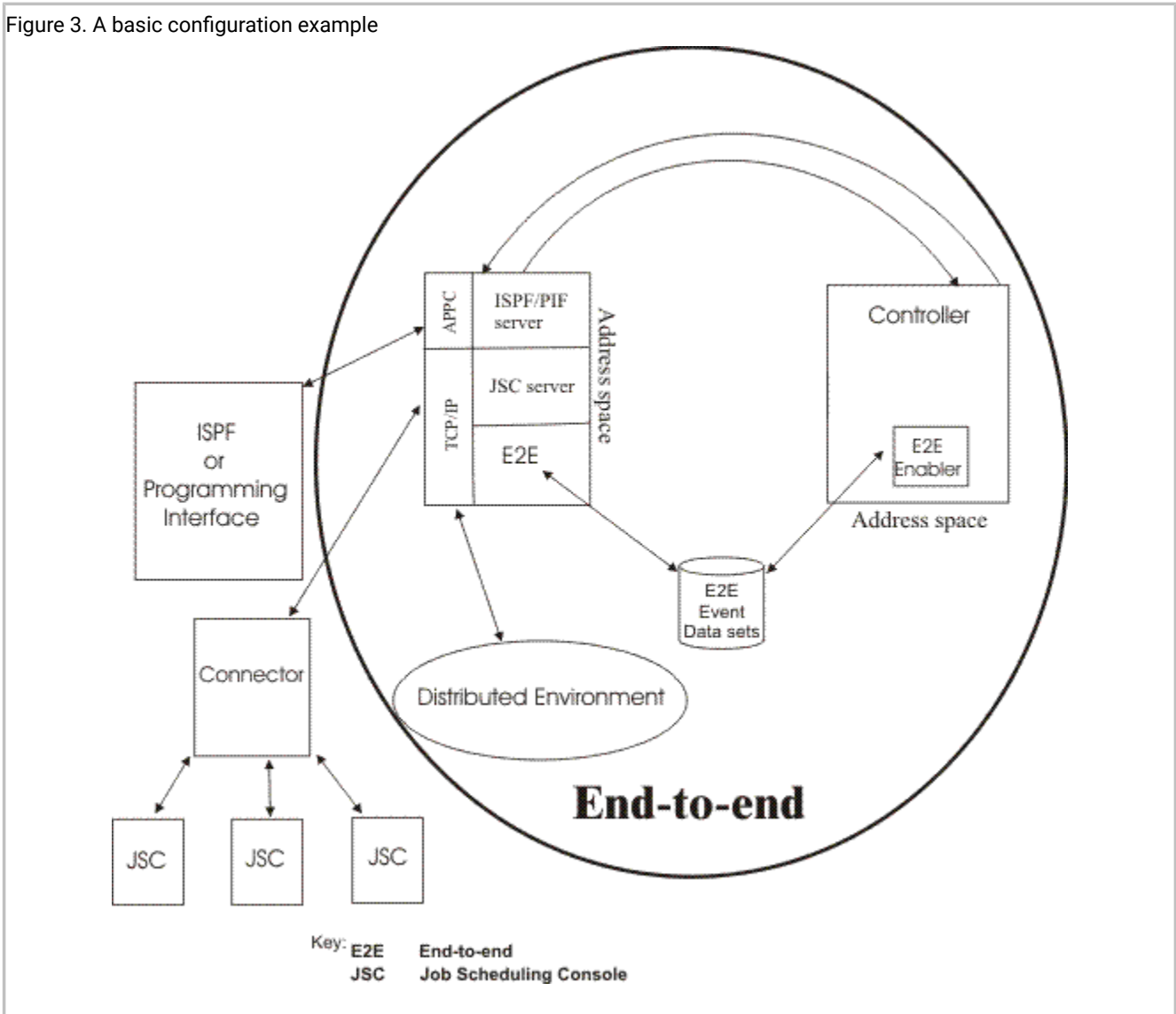
During the production cycle, IBM Workload Scheduler reports the status of running and completed jobs back to the current plan for monitoring in the IBM Z Workload Scheduler engine. With this mechanism, you can define dependencies among IBM Z Workload Scheduler and IBM Workload Scheduler jobs.

## Basic server configuration example

Example of a basic server configuration of IBM Z Workload Scheduler. IBM Z Workload Scheduler connects distributed agents to the server through TCP/IP.

The controller transmits work to the distributed agents through TCP/IP, and the same connection is used to pass event information back. The server is connected to standard agents, fault-tolerant agents, and to first level domain managers in the distributed network. TCP/IP is also used to connect the Dynamic Workload Console to the server through the connector. The server connects to the remote interfaces, either Programming Interfaces or remote ISPF interface users, using Advanced Program-to-Program Communication (APPC). [Figure 3: A basic configuration example on page 24](#) shows a simple configuration using mixed protocols and minimal parameter customization.

Figure 3. A basic configuration example



[EQQPARM members for a sample configuration on page 24](#) shows the initialization statements you can use to create an end-to-end with fault tolerance capabilities configuration consisting of:

- A fault-tolerant agent named FTW1 that is directly connected to the end-to-end server. Note that it is defined as non-full status.
- A standard agent named SA1 that is directly connected to the end-to-end server. Note that its `CPUHOST` name is set to `OPCMaster`.
- A domain manager named FTW2 that connects other agents and sub-domains (not shown in this example).

### EQQPARM members for a sample configuration

#### EQQSERP

```
SERVOPTS SUBSYS(OPCA)
          PROTOCOL(E2E)
```



```
TPLGYPRM(TPLGY)
INIT CALENDAR(DEFAULT)
```

**TPLGY**

```
TOPOLOGY TPLGYMEM(TPLGYDOM)
BINDIR('/usr/lpp/TWS8.3.0')
WRKDIR('/var/TWS/OPCA')
USRMEM(TPLGYUSR)
CODEPAGE(IBM-280)
```

**TPLGYDOM**

```
DOMREC DOMAIN(DOM0)
DOMMNGR(FTW2)
DOMPARENT(MASTERDM)
```

```
CPUREC CPUNAME(FTW1)
CPUOS(UNIX)
CPUNODE('xxx.xx.xxx.x')
CPUDOMAIN(MASTERDM)
CPUTYPE(FTA)
CPUTCPIP(31111)
CPUFULLSTAT(OFF)
CPUAUTOLNK(ON)
CPULIMIT(SYSTEM)
FIREWALL(NO)
CPUTZ('EUT')
```

```
CPUREC CPUNAME(SA1)
CPUOS(WNT)
CPUNODE('xxx.xx.xxx.x')
CPUDOMAIN(MASTERDM)
CPUHOST(OPCMaster)
CPUTYPE(SAGENT)
CPUTCPIP(31111)
CPUAUTOLNK(ON)
CPULIMIT(SYSTEM)
FIREWALL(NO)
CPUTZ('EUT')
```

```
CPUREC CPUNAME(FTW2)
CPUOS(WNT)
CPUNODE('xxx.xx.xxx.x')
CPUDOMAIN(DOM0)
CPUTYPE(FTA)
CPUTCPIP(31111)
CPUFULLSTAT(ON)
CPUAUTOLNK(ON)
CPULIMIT(SYSTEM)
FIREWALL(NO)
CPUTZ('EUT')
```

```
CPUREC .....
```

**TPLGYUSR**

```
USRREC USRCPU(SA1)
USRNAM('tws83sa1')
USRPSW('tws83sa1')
```

```

USRREC  USRCPU(FTW2)
        USRNAM('tws83ftw2')
        USRPSW('tws83ftw2')

USRREC  ...

```

## Functional comparison between end-to-end and single scheduling environments

Functional comparison between end-to-end and single scheduling environments

The functional differences you encounter when scheduling in the end-to-end with fault tolerance capabilities environment as compared to scheduling solely with the IBM Z Workload Scheduler tracker agents or with the IBM Workload Scheduler distributed environment.

### Functional comparison with IBM Z Workload Scheduler tracker agents

lists the functional enhancements that derive from replacing the tracker agents with fault-tolerant workstations:

- The distributed agents can be fault-tolerant. Fault-tolerance is defined as the capability to run the assigned workload despite a loss of connection from its domain manager. After a fault-tolerant workstation has received its daily Symphony, it can carry on with running its part of the plan even if it is disconnected from its domain manager (and provided it remains linked to other workstations with which it shares dependencies).
- Fault-tolerant workstations can host extended agent definitions. Using extended agents, you can schedule workload on other products and subsystems such as R/3, PeopleSoft, Oracle, CA7, and JES from the end-to-end with fault tolerance capabilities environment.
- Fault-tolerant workstations can perform job recovery actions in an autonomous way even when disconnected.
- You can:
  - Connect multiple domain managers to the IBM Z Workload Scheduler controller, also taking advantage of the multi-tier domain structure of IBM Workload Scheduler. This allows greater flexibility and scalability.
  - Enable firewall support and SSL authentication.
  - Browse the scheduling activity at the engine, domain, or agent level.
  - Match user names and job runs.
  - Use the return code mapping function to specify a success condition for each job that determines whether a job is considered successful or abended.
  - Make an application or an operation dependent upon the existence of one or more files before it can begin running. Use the **filewatch** utility, as described in [Defining file dependencies to check for file changes on page 112](#).

### Functional comparison with IBM Workload Scheduler

The following is a list of additional functionality that is available in the end-to-end with fault tolerance capabilities configuration when compared with a native IBM Workload Scheduler environment:

- Security can be handled by RACF®.
- Special (global) resources are available also to the fault-tolerant workstations.

- `In use` and `Waiting` queues are available for resources.
- The capability to take advantage of the IBM Z Workload Scheduler program interface (PIF) and of the batch command interface to automate workload management is extended to fault-tolerant workstations.
- More powerful periods, calendars, and run cycles are supported.
- Use of variable tables for variable substitution also for job streams involving fault-tolerant workstations.
- Operator Instructions.
- ETT on z/OS® jobs, data sets, resources.
- Edit JCL, Job setup (only for centralized jobs).

The following is a list of native IBM Workload Scheduler functionality that is not supported in the end-to-end with fault tolerance capabilities configuration:

- The following conman commands are not supported:
  - `adddep`
  - `altpri`
  - `cancel`
  - `deldep`
  - `release`
  - `rerun`
  - `submit docommand`
- Elapsed daily plans (Symphony) are not archived, and it is not possible to choose and work with a past plan.
- Workstation classes are not supported.
- The workstation `ignore` attribute is not implemented for fault-tolerant workstations. Because of this limitation, together with a lack of support for availability intervals, you cannot take workstations temporarily out of the IBM Workload Scheduler network, unless the workstations are deleted or renamed.
- Prompts, as defined in native IBM Workload Scheduler, do not exist in the end-to-end with fault tolerance capabilities configuration.
- The `limit` attribute for job streams is not supported.
- The `until` time dependency provided in IBM Workload Scheduler is not mapped exactly to the deadline-time concept provided in end-to-end scheduling with fault tolerance capabilities. In particular, the `onuntil` actions provided by IBM Workload Scheduler are more powerful and granular than their equivalent in the end-to-end with fault tolerance capabilities environment.
- Passwords of IBM Workload Scheduler users on Windows™ are not encrypted when scheduling end-to-end with fault tolerance capabilities. They are kept in clear text in the end-to-end partitioned data sets (that can be placed under RACF® security).
- File dependencies are provided with a different mechanism in end-to-end with fault tolerance capabilities. The concept of files and files dependencies provided in IBM Workload Scheduler is simulated by using a job running the **filewatch** utility and setting a dependency on that particular job.
- In end-to-end scheduling with fault tolerance capabilities, you can change the priority of a job stream, but not of individual jobs.
- The `cancel` command for jobs and job streams is mapped to the `delete` function in end-to-end.

## Terminology mapping

Because of legacy reasons, the two scheduling engines and the Dynamic Workload Console use different names for the same scheduling objects or attributes.

**Table 3. Dynamic Workload Console terminology mapping**

IBM Workload Scheduler	IBM Z Workload Scheduler	Dynamic Workload Console	Explanation
Schedule	Application	Job stream	A sequence of jobs, including the resources and workstations that support them, and scheduling information.
Not applicable	Application group	Job stream group	A grouping of job streams that provides scheduling information, such as a calendar, a free-day rule, and run cycles that can be inherited by all the job streams that have been created using the template.
Symphony	Current plan	Production plan	A detailed plan of system activity. The plan encompasses all job and job stream instances and the resources and workstations involved in running them.
External dependency	External dependency	External job dependency	A job from one job stream that is a predecessor for a job in another job stream.
Not applicable	In-effect date for run cycles	Valid from starting	The first date that a run cycle is valid.
Start Time	Input arrival time	Earliest start	The time when a job or job stream is planned to be ready for processing.
Exclusionary run cycle	Negative run cycle	Exclusive run cycle	Specifies when a job stream must not run.
Schedule	Occurrence	Job stream	A job stream that is scheduled for a specific run date in the plan.
Engine	Controller	Engine	The component that runs on the controlling system, and that contains the tasks that manage the plans and databases.
Job	Operation	Job	A unit of work that is part of a job stream and that is processed at a workstation.
Job identifier	Operation number	Job identifier	The number that identifies a job.
Job (in the plan)	Operations in the current plan	Job	A job scheduled for a specific run date in the plan.

**Table 3. Dynamic Workload Console terminology mapping (continued)**

<b>IBM Workload Scheduler</b>	<b>IBM Z Workload Scheduler</b>	<b>Dynamic Workload Console</b>	<b>Explanation</b>
Not applicable	Out-of-effect date for run cycles	Valid to ending	The last date that a run cycle is valid.
Not applicable	Run cycle with offsets	Offset-based run cycle	Includes a user-defined period and an offset, such as the third day in a 90-day period.
Not applicable	Run cycle with rules	Rule-based run cycle	Includes a rule, such as the first Friday of March or the second workday of the week.
Not applicable	Task	Job	A job performed at a computer workstation.

## Chapter 2. Installing

This chapter describes how to install the mainframe and the distributed components of the end-to-end with fault tolerance capabilities network.

### Installing the IBM Z Workload Scheduler end

The following checklist assumes that there is already an IBM Z Workload Scheduler installation. The checklist summarizes the tasks you must perform to set up IBM Z Workload Scheduler for end-to-end scheduling with fault tolerance capabilities.

#### Installation prerequisites

Before starting the actions described in the following section, you must have added the fault-tolerant end-to-end feature using SMP/E on your IBM Z Workload Scheduler installation.

#### Installation checklist

##### About this task

To activate the end-to-end scheduling with fault tolerance capabilities in IBM Z Workload Scheduler, follow these steps:

1. Run EQQJOBS and specify Y for the END-TO-END WITH FAULT TOLERANCE field. For details, see [Step 1. Specifying fault-tolerant end-to-end scheduling information in EQQJOBS on page 30](#).
2. Generate batch-job skeletons. For details, see [Step 2. Generating batch-job skeletons on page 32](#).
3. Allocate the data set running the generated EQQPCS06 sample. For details, see [Step 3. Running the generated sample to allocate the end-to-end data sets on page 33](#).
4. Create and customize the work directory by running the generated EQQPCS05 sample. For details, see [Step 4. Creating and customizing the work directory on page 35](#).
5. Create JCL procedures for address spaces. For details, see [Step 5. Creating JCL procedures for address spaces on page 39](#).
6. Customize the PARMLIB members. For details, see [Step 6. Customizing the PARMLIB members on page 40](#).
7. Define the fault-tolerant workstations in the IBM Z Workload Scheduler workstation description database. For details, see [Step 7. Defining the fault-tolerant workstations on page 62](#).
8. Verify your installation. For details, see [Step 8. Verifying your installation on page 63](#).

#### Installation details

The following sections explain the installation steps in detail.

### Step 1. Specifying fault-tolerant end-to-end scheduling information in EQQJOBS

##### About this task

Enter the information specific to the end-to-end scheduling with fault tolerant capabilities in the EQQJOBS8 dialog (this is part of the step describing how to use the EQQJOBS installation aid, which is documented in *IBM® Z Workload Scheduler: Planning and Installation*). [Figure 4: EQQJOBS8 - Create sample job JCL on page 31](#) shows dialog EQQJOBS8.

Figure 4. EQQJOBS8 - Create sample job JCL

```

EQQJOBS8 ----- Create sample job JCL -----
Command ==>

END-TO-END WITH FAULT TOLERANCE: Y          (Y= Yes, N= No)
Installation Directory  ==> /usr/lpp/TWS/V8R2M0_____
                        ==> _____
                        ==> _____
Work Directory         ==> /var/TWS/inst_____
                        ==> _____
                        ==> _____
User for OPC address space ==> UID_
Refresh CP group      ==> GID__

RESTART AND Cleanup (Data Store) Y          (Y= Yes ,N= No)
Reserved destination  ==> OPCX_____
Connection type       ==> SNA          (SNA/XCF)
SNA Data Store luname ==> I9PC45AA (only for SNA connection )
SNA FN task luname    ==> I9PC45RA (only for SNA connection )
Xcf Group             ==> _____ (only for XCF connection )
Xcf Data Store member ==> _____ (only for XCF connection )
Xcf FL task member    ==> _____ (only for XCF connection )

Press ENTER to create sample job JCL

```

### END-TO-END WITH FAULT TOLERANCE

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

#### Installation Directory

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

#### Work Directory

Specify where the end-to-end subsystem files are. Replace */inst* with a name that uniquely identifies your subsystem. Each subsystem that will use the fault-tolerant workstations must have its own work directory. Only the server and the daily planning batch jobs go in the work directory. See [Step 4. Creating and customizing the work directory on page 35](#).

#### User for OPC address space

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

#### Refresh CP group

This information is used to create the procedure to build the directory with the right ownership. To create the new Symphony file, the user ID used to run the daily planning batch job must belong to the group that

you specify in this field. Make sure that also the user ID associated with the server address space (the one specified in User for OPC address space field) belongs to this group or has this group as supplementary group.

After entering the information, press `Enter`, and several members are generated in the output library that you specified. These members, which are described in [Table 4: Sample JCL for end-to-end scheduling with fault tolerance capabilities generated by the EQQJOBS8 dialog on page 32](#), will be used at various stages in the installation.

**Table 4. Sample JCL for end-to-end scheduling with fault tolerance capabilities generated by the EQQJOBS8 dialog**

Member	Description of job
EQQE2EP	Sample initial parameters for server and batch to define if the end-to-end scheduling with fault tolerance capabilities is active
EQQORST	Resets the USS environment for the end-to-end scheduling with fault tolerance capabilities
EQQPCS05	Allocates files used by the end-to-end server to enable fault-tolerant workstations
EQQPCS06	Allocates VSAM data sets for integration with the end-to-end scheduling with fault tolerance capabilities
EQQSERP	Sample initial parameters for a server
EQQSER	Sample started task procedure for a server
EQQSLCHK	JCL that runs a syntactic check on a SCRIPT library member (see <a href="#">Determining errors in the SCRPTLIB configuration on page 84</a> ). This requires that PTFs <code>UK02507</code> and <code>UK02508</code> are installed.

## Step 2. Generating batch-job skeletons

### About this task

Several controller functions, such as daily planning, are performed by batch jobs that are submitted from the IBM Z Workload Scheduler dialog. To generate the skeleton JCL for these jobs you must select option 2 from the EQQJOBS main menu. Proceeding from there you get to dialog EQQJOBSA, which is relevant to end-to-end scheduling with fault tolerance capabilities.

[Figure 5: EQQJOBSA - Generate IBM Z Workload Scheduler batch-job skeletons on page 33](#) shows dialog EQQJOBSA.



Figure 5. EQQJOBSA - Generate IBM Z Workload Scheduler batch-job skeletons

```

EQQJOBSA ----- Generate OPC batch-job skeletons -----
Command ==>

Specify if you want to use the following optional features:

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

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

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

Press ENTER to generate OPC batch-job skeletons

```

Specify Y in the `END-TO-END WITH FAULT TOLERANCE` field to indicate that you want batch-job skeleton members generated for working with fault-tolerant workstations. When you press `Enter`, the batch-job skeleton members are generated keeping fault-tolerant end-to-end scheduling into account.

### Step 3. Running the generated sample to allocate the end-to-end data sets

#### About this task

Allocate the files and directories required to use the end-to-end scheduling with fault tolerance capabilities. The server starts multiple tasks and processes using the UNIX™ System Services (USS) on z/OS®. The end-to-end server accesses this in a z/OS® File System (ZFS) cluster. Sample JCL EQQPCS06 is generated by the EQQJOBS8 dialog.

Run EQQPCS06 to allocate the following files:

#### End-to-end script library (EQQSCLIB)

This script library data set includes members containing the commands or the job definitions for fault-tolerant workstations. This data set identifies the SCRPTLIB to whose members every job running on a fault-tolerant workstation must be associated (unless the job uses a centralized script). It is required in the controller if you want to use the end-to-end scheduling with fault tolerance capabilities.

Do not compress members in this Partition Data Set (PDS). For example, do not use the ISPF PACK ON command, because IBM Z Workload Scheduler does not use ISPF services to read it.

#### End-to-end data set for centralized script support (EQQTWSCS)

IBM Z Workload Scheduler uses the end-to-end centralized script data set to temporarily store a script when it is downloaded from the JOBLIB data set to the agent for its submission.

If you want to use centralized script support when scheduling end-to-end, you need to use the EQQTWSCS DD statement in the controller and server started tasks. The data set must be a partitioned extended data set.

### End-to-end input and output events data sets (EQQTWSIN and EQQTWSOU)

These data sets are required by every IBM Z Workload Scheduler address space that uses the end-to-end scheduling with fault tolerance capabilities. They record the descriptions of events related to operations running on fault-tolerant workstations and are used by both the end-to-end enabler task and the translator process in the server of the scheduler.

The data sets are device-dependent and can have only primary space allocation. Do not allocate any secondary space. They are automatically formatted by IBM Z Workload Scheduler the first time they are used.



**Note:** An *SD37* abend code is produced when IBM Z Workload Scheduler formats a newly allocated data set. Ignore this error.

`EQQTWSIN` and `EQQTWSOU` are wrap-around data sets. In each data set, the header record is used to track the number of *read* and *write* records. To avoid the loss of event records, a writer task does not write any new records until more space is available when all the existing records have been read.

The amount of space that you need to define for each data set requires some attention. Because the two data sets are also used for joblog retrieval, the limit for the joblog length is half the maximum number of records that can be stored in the input events data set. 10 cylinders are sufficient for most installations.

The maximum length of the events logged in these two data sets, including the joblogs, is 120 bytes. However, it is possible to allocate the data sets with a longer logical record length. Using record lengths greater than 120 bytes does not produce either advantages or problems. The maximum allowed value is 32000 bytes; greater values cause the end-to-end task to terminate. In both data sets there must be enough space for at least 1000 events (the maximum number of joblog events is 500). Use this as a reference if you plan to define a record length greater than 120 bytes. So, when a record length of 120 bytes is used, the space allocation must be at least 1 cylinder. The data sets must be unblocked and the block size must be the same as the logical record length. A minimum record length of 160 bytes is necessary for the `EQQTWSOU` data set to choose how to build the job name in the symphony file (for details, see the `TWSJOBNAME` parameter of the `JTOPTS` statement in *Customization and Tuning*).

To improve performance, define the data sets on a device with plenty of space availability. If you run programs that use the `RESERVE` macro, try to allocate the data sets on a device that is not, or only slightly, reserved.

Initially, you might need to test your system to get an idea of the number and type of events that are created at your installation. Once you have gathered enough information, you can then reallocate the data sets. Before you reallocate a data set, ensure that the current plan is entirely up-to-date. You must also stop the end-to-end sender and receiver task on the controller and the translator thread on the server that use this data set. `EQQTWSIN` and `EQQTWSOU` must not be allocated as multivolume.



**Note:** Do not move these data sets once they have been allocated. They contain device-dependent information and cannot be copied from one type of device to another, or moved around on the same volume. An end-to-end event data set that is moved will be re-initialized. This causes all events in the



data set to be lost. If you have DFHSM or a similar product installed, you should specify that end-to-end event data sets are not migrated or moved.

### Current plan backup copy data set to create the Symphony file (EQQSCPDS)

During the creation of the current plan, the SCP data set is used as a CP backup copy for the production of the Symphony file. This VSAM is used when the end-to-end with fault tolerance capabilities environment is active. It should be allocated with the same size as the CP1/CP2 and NCP VSAM data sets.

## Step 4. Creating and customizing the work directory

### About this task

To activate the support of the fault-tolerant end-to-end scheduling, allocate the HFS or zFS files that this environment uses and, before you run the EQQISMKD job, mount them. To mount the files at system restart, you might consider to modify the BPXPRMxxx member of the PARMLIB accordingly.

To create the HFS or zFS directories and files, on every IBM Z Workload Scheduler controller that uses the fault-tolerant end-to-end environment run the EQQPCS05 sample job. To run the EQQPCS05 sample job, ensure you have one of the following authorities:

- A UNIX™ System Service user ID (USS UID) of 0.
- A BPX.SUPERUSER FACILITY class profile in RACF®.
- The user ID specified in the eqqUID field in the EQQPCS05 sample job. (Ensure that this user belongs to the group you set in eqqGID.)

For the EQQPCS05 sample, if the GID or the UID were not specified in `EQQJOBS`, you can specify them in the `STDENV DD` before running sample (for details, see [Defining users and default groups assigned to server and controller on page 36](#)). Make sure that you use a unique UID with a nonzero value; for additional information about this requirement, refer to INFO APAR II1423.

The user must also have the `/bin/sh` login shell defined in his OMVS section of the RACF® profile. Make sure that the login shell is set as a system default or use the following TSO command to define it:

```
ALTUSER username OMVS(PROGRAM('/bin/sh'))
```

To check the current settings:

1. Run the following TSO command:

```
LISTUSER username OMVS
```

2. Look in the `PROGRAM` line of the OMVS section.

After running `EQQPCS05`, you find the following files in the work directory:

### **localopts**

Defines the attributes of the local workstation (`OPCMASTER`) for batchman, mailman, netman, and writer processes and for SSL. For information about customizing this file, see [Customizing on page 72](#).

Local options that have no effect in an end-to-end with fault tolerance capabilities environment are indicated and commented out in `EQQPCS05`.

### **mozart/globalopts**

Defines the attributes of the IBM Workload Scheduler network (`OPCMASTER` ignores them).

### **Netconf**

Netman configuration files.

### **TWSCCLog.properties**

Defines attributes for the logging function.

You also find the following sub-directories in the work directory:

- `mozart`
- `pobox`
- `stdlist`
- `stdlist/logs` contains the log files of the USS processes

## Defining users and default groups assigned to server and controller

Users assigned to run the server or the daily planning batch job within the end-to-end with fault tolerance capabilities environment, must have access to Unix System Services (USS). You can use RACF® commands to define users to Unix System Services; attributes are kept in the OMVS segment of a user's profile and can be specified in addition to other existing attributes.

To use USS, a user must be defined the following USS attributes:

- User identifier (UID) in his user profile
- Group identifier (GID) in the group profile of his default group

All users defined to access USS must have a UID and a GID, regardless of the fact that they run the server or the daily planning batch job. An incorrect definition of a user on RACF® might cause serious errors and unpredictable behavior in the system.

The RACF® documentation provides the following guidelines to define users and groups for access to USS:

- Assign only to one user a UID equal to zero. You might choose to assign the UID with value 0 to multiple RACF® user IDs. However, you should seek to minimize the attribution of superuser authority in your installation. You can accomplish this by setting USS user limits, and by managing superuser privileges through UNIXPRIV profiles.
- Although the same GID can be assigned to multiple RACF® groups, this is not recommended. If you assign the same GID to multiple groups, control at the individual group level is lost because the GID is used in USS security checks. RACF® groups having the same GID are treated as a single group during USS security checks.
- RACF® does not require the UID to be unique. The same value can be assigned to multiple users but this is not recommended because individual user control would be lost. However, if you want a set of users to have exactly the same access to the OpenEdition resources, you might decide to assign the same UID to more than one user. Users with the same UID assignment are treated as a single user during USS security checks.

To avoid problems on Unix System Services (and consequently on IBM Z Workload Scheduler) when assigning a user identifier to a user, make sure that also the user's default group has an assigned group identifier. A user with a UID and a GID for its default group can use Unix System Services functions and access USS files based on the assigned UID and GID values. If no UID and GID are available as described, the user has no access to USS functions.

## Configuring for end-to-end scheduling with fault tolerance capabilities in a SYSPLEX environment

In a configuration with a controller and no stand-by controllers, define the end-to-end server work directory in a file system mounted under the system-specific ZFS. Then, configure the Byte Range Lock Manager (BRLM) server in a distributed form (see following considerations about BRLM). In this way the server will not be affected by the failure of other systems in the sysplex.

In a configuration with an active controller and several stand-by controllers make sure that all the related end-to-end servers running on the different systems in the Sysplex have read-write access to the same work directory.

The shared ZFS capability can be used: all file systems that are mounted by a system participating in shared ZFS are available to all participating systems. When allocating the work directory in a shared ZFS, you can decide to define it in a file system mounted under the system-specific ZFS or in a file system mounted under the sysplex root. A system-specific file system becomes unreachable if the system is not active. So, to make good use of the takeover process, define the work directory in a file system mounted under the sysplex root and defined as automove.

The Byte Range Lock Manager (BRLM) locks some files in the work directory. The BRLM can be implemented:

- With a central BRLM server running on one member of the sysplex and managing locks for all processes running in the sysplex.



**Note:** This is no longer supported once all systems in a sysplex are at the z/OS® V1R6 or later level.

- In a distributed form, where each system in the sysplex has its own BRLM server responsible for handling lock requests for all regular files in a file system which is mounted and owned locally (refer to APARs OW48204 and OW52293).

If the system where the BRLM runs experiences a scheduled or unscheduled outage, all locks held under the old BRLM are lost; to preserve data integrity, further locking and I/O on any opened files is prevented until files are closed and reopened. Moreover, any process locking a file is terminated (you must be sure that your OS/390® service level includes PTF UW75787 for V2R9 systems, or UW75786 for V2R10 systems).

To avoid this kind of error in the end-to-end server, before starting a scheduled shut down procedure for a system, you must stop the end-to-end server if either or both of the following conditions occurs:

- The work directory is owned by the closing system
  - The **df -v** command on OMVS displays the owners of the mounted file systems
- The system hosts the central BRLM server
  - The console command `DISPLAY OMVS,O` can be used to display the name of the system where the BRLM runs. If the BRLM server becomes unavailable, then the distributed BRLM is implemented. In this case the end-to-end server needs to be stopped only if the system that owns the work directory is stopped.

The server can be restarted after a new system in the sharing has taken the ownership of the file system and/or a new BRLM is established by one of the surviving systems.

To minimize the risk of filling up the IBM Workload Scheduler internal queues while the server is down, you should schedule the closure of the system when the workload is low.

A separate file system data set is recommended for each stdlist directory mounted in R/W on `/var/TWS/inst/stdlist`, where *ins* varies depending on your configuration.

When you calculate the size of a file, consider that you need 10 MB for each of the following files:

- Intercom.msg
- Mailbox.msg
- pobox/tomaster.msg
- pobox/CPUDOMAIN.msg.

You need 512 bytes for each record in the Symphony, Symold, Sinfonia, and Sinfeld files. Consider one record for each CPU, schedule, and job or recovery job.

You can specify the number of days that the trace files are kept on the file system using the parameter TRCDAYS in the TOPOLOGY statement.

## Applying maintenance to the HFS file

To apply maintenance to an HFS file, create a file with a new data set name by using the DFDSS DUMP/RESTORE facility and copy the content of the HFS file that you want to maintain, to it. Mount the created HFS file at a service point (for example, /SERVICE), apply the maintenance, and test it. If the test completes successfully, unmount the original HFS file and mount the tested file at the original mount point. In this way, you prevent the file from losing any APF authorization.

If an error occurs while copying the HFS file, a message is displayed warning you that the APF authorization was lost. You can reassign the APF authorization by issuing the following command:

```
extattr +a /dirname
```

where `dirname` is, by default, `/usr/lpp/TWS/TWS810/bin`.

For more detailed information about how to apply maintenance to the files, see *z/OS® UNIX™ System Services Planning* (GA22-7800).

## Step 5. Creating JCL procedures for address spaces

### About this task

Perform this task for an IBM Z Workload Scheduler tracker, controller, and server started task. You must define a started task procedure or batch job for each IBM Z Workload Scheduler address space.

The EQQJOBS dialog generates several members in the output sample library that you specified when running the EQQJOBS installation aid program. These members contain started task JCL that is tailored with the values you entered in the EQQJOBS dialog. Tailor these members further, according to the data sets you require. For guidance on adequate memory to allocate for the end-to-end server started task, see [Table 1: z/OS memory requirements for directly connected agents and mailman servers on page 19](#).

Because the end-to-end server started task uses TCP/IP communication, you should do the following:

1. Modify the JCL of EQQSER (the end-to-end server started task) so that it has access to the C run time libraries, either as STEPLIB (include the CEE.SCEERUN in the STEPLIB concatenation) or by LINKLIST (the CEE.SCEERUN is in the LINKLIST concatenation).

Use the end-to-end server TOPOLOGY TCPIPJOBNAME() parameter to specify the TCP/IP started task name that is used by the end-to-end server. This parameter can be used if you have multiple TCP/IP stacks or if the TCP/IP started task name is different from TCPIP.

2. Set up a server started task to handle end-to-end scheduling with fault tolerance capabilities. You can use the same server also to communicate with the Dynamic Workload Console. In fact, the server can also handle APPC communication if configured for this. In IBM Z Workload Scheduler version 8.2, the type of communication that should be handled by the server started task is defined in the new SERVOPTS PROTOCOL() parameter.

In the PROTOCOL() parameter, you can specify any combination of:

#### APPC

The server should handle APPC communication.

#### JSC

The server should handle Dynamic Workload Console communication.

**E2E**

The server should handle end-to-end communication. The IBM Z Workload Scheduler controller uses the end-to-end server to communicate events to the fault-tolerant agents. The end-to-end server starts multiple tasks and processes using the UNIX™ System Services.

When you do this, be aware that:

- The IBM Z Workload Scheduler controller and the end-to-end server use TCP/IP services. Therefore it is necessary that you define a USS segment for the controller and for the end-to-end server started task user IDs. No special authorization is required; it can be defined in USS with any user ID.
- Although it is possible to handle end-to-end scheduling with fault tolerance capabilities, JSC communication, and APPC communication with the same server started task, you should have a server started task dedicated for end-to-end scheduling with fault tolerance capabilities (SERVOPTS PROTOCOL(E2E)). In this way you do not have to also stop the other server processes when you restart the Dynamic Workload Console server.
- The server started task is important for handling JSC and end-to-end with fault tolerance capabilities communication. You should set the end-to-end and the Dynamic Workload Console server started tasks as non-swappable and define them with at least the same dispatching priority as the IBM Z Workload Scheduler controller.

## Step 6. Customizing the PARMLIB members

### About this task

The IBM Z Workload Scheduler initialization statements are kept in a parameter library that is identified by the EQQPARM DD statement in the JCL procedure. When you start IBM Z Workload Scheduler, a member in the library containing the initialization statements of the end-to-end server is read.

To use the end-to-end scheduling with fault tolerance capabilities, you must set the initialization statements of the PARMLIB library data set as follows:

1. Define CPU configuration and domain organization by using the CPUREC and DOMREC statements in a PARMLIB member (the default member name is TPLGINFO). For details, see [Defining the distributed agents and domains on page 42](#).

A special case of a multiple domain configuration is when *every* fault-tolerant agent is a domain manager, that is, every FTA has CPUFULLSTATUS(ON). Avoid this configuration, because it is inefficient, in that every message for every event on every FTA must be sent to all the CPUFULLSTATUS FTAs. If you use this configuration, you must increase the REGION SIZE to an appropriate value, which you obtain by tuning your network environment.

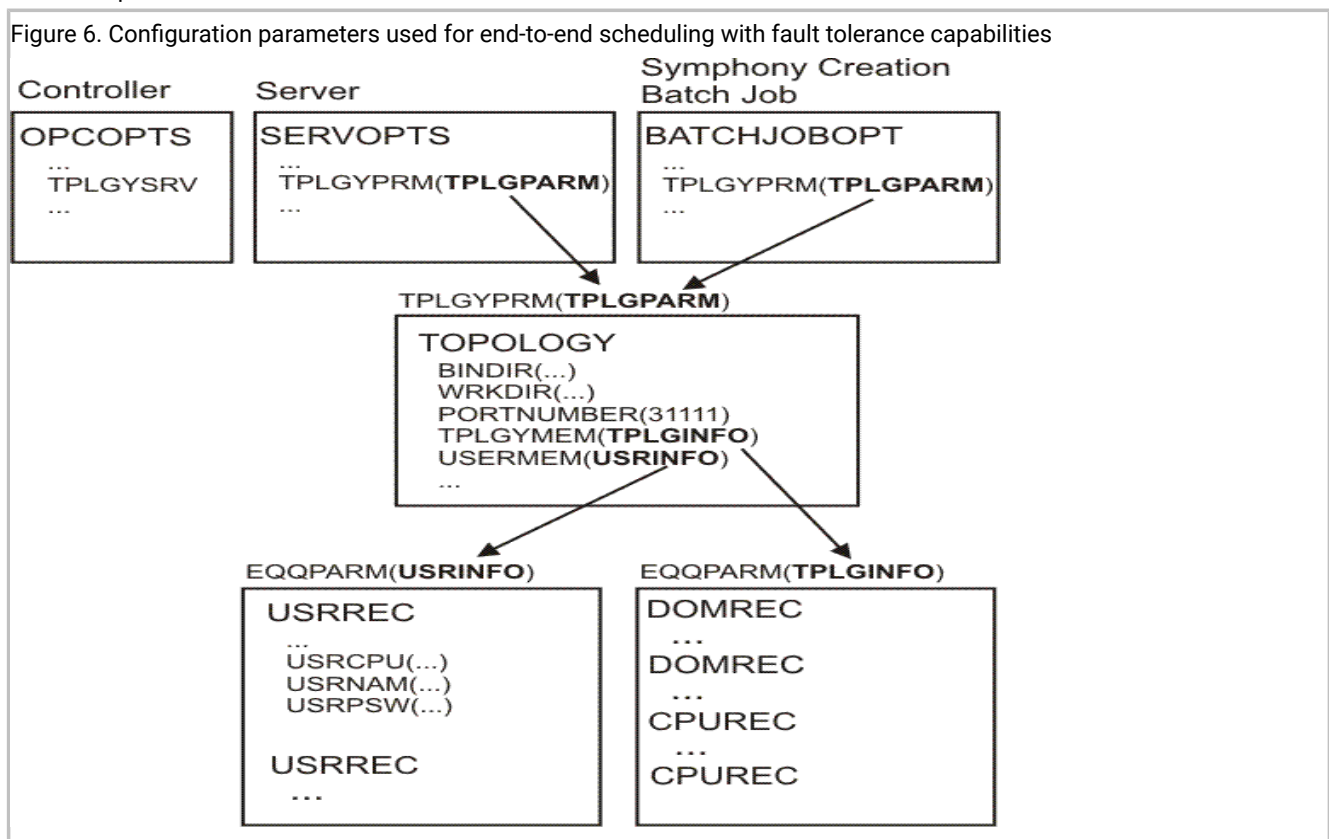
For a description about how to set up the distributed network, see [Choosing the agents on page 20](#).

2. Define user IDs and passwords for Windows™ users by using the USRREC statement in a PARMLIB member (the default member name is USRINFO). Alternatively, you can define the Windows™ user IDs and passwords in a file stored locally in the workstation. In this case, you must set LOCALPSW(YES) in the [TOPOLOGY on page 51](#) statement. For details, see [Defining user IDs and passwords for Windows users on page 49](#).



3. Define the end-to-end with fault tolerance capabilities configuration by using the [TOPOLOGY](#) on page 51 statement in a PARMLIB member (the default member name is TPLGPARM). In this statement, specify the following:
  - For the TPLGYMEM keyword, write the name of the member used in step 1 on page 40.
  - For the USRMEM keyword, write the name of the member used in step 2 on page 40.
 See [Defining the end-to-end with fault tolerance capabilities configuration with the TOPOLOGY statement on page 51](#) for details.
4. Add the TPLGYSRV keyword to the OPCOPTS statement to specify the server name to be used for end-to-end communication. For details, see [Specifying the name of the end-to-end server with the TPLGYSRV keyword on page 59](#).
5. Add the TPLGYPRM keyword to the SERVOPTS statement to specify the member name used in step 3 on page 41. This step activates end-to-end communication in the server. For details, see [Activating end-to-end with fault tolerance capabilities communication in the server with the TPLGYPRM keyword on page 59](#).
6. Add the TPLGYPRM keyword to the BATCHOPT statement to specify the member name used in step 3 on page 41. This step activates the end-to-end with fault tolerance capabilities environment in the Daily Planning batch programs. See [Activating the end-to-end scheduling with fault tolerance capabilities in the Daily Planning batch programs on page 60](#) for details.
7. Add the TWSJOBNAME parameter in the JTOPTS statement to define the rules used to generate the job name in the Symphony file. For details, see [Defining standards for generating job names in the Symphony file on page 60](#).

Figure 6: Configuration parameters used for end-to-end scheduling with fault tolerance capabilities on page 41 shows the relationship between the new initialization statements and members.



The following sections explain the steps in detail.

## Defining the distributed agents and domains

In this step you create a member containing a description of the IBM Workload Scheduler network topology with domains and agents. Remember that the agents of the distributed network are named fault-tolerant workstations when scheduling end-to-end.

Use the CPUREC and DOMREC initialization statements to describe the distributed IBM Workload Scheduler topology for the IBM Z Workload Scheduler server and plan programs. The IBM Z Workload Scheduler fault-tolerant workstations are mapped to physical IBM Workload Scheduler agents or workstations using the CPUREC statement. The DOMREC statement is used to describe the domain topology in the distributed IBM Workload Scheduler network.

You might decide to wait until you have physically installed the agents before starting this task, but the definitions must be in place before you start scheduling.

## Using the CPUREC statement

### CPUREC

#### **Purpose**

This statement begins an IBM Workload Scheduler workstation (CPU) definition. You must specify one CPUREC for each IBM Workload Scheduler workstation you intend to add to the end-to-end with fault tolerance capabilities network, with the exception of the controller that acts as master domain manager of the network.

CPUREC is defined in the member of the EQQPARM library, as specified by the TPLGYMEM keyword in the TOPOLOGY statement.



## Parameters

### **CPUNAME**(*cpu name*)

The name of the IBM Workload Scheduler workstation consisting of up to four alphanumeric characters, starting with a letter.

### **CPUOS**

The host CPU operating system related to the IBM Workload Scheduler workstation. The valid entries are: AIX®, HPUX, POSIX, UNIX™, WNT, and OTHER.

### **CPUNODE**(*node name*)

The node name or the IP address of the CPU. Fully-qualified domain names up to 52 characters are accepted.

### **CPUTCPIP**(*port number***31111**)

The TCP port number of NETMAN on this CPU. It consists of five numerals and, if omitted, uses the default value, 31111.

### **CPUDOMAIN** (*domain name*)

The name of the IBM Workload Scheduler domain of the CPU. It can be up to 16 characters long and must start with a letter. It can be alphanumeric and contain dashes (-) and underscores (\_).

For standard and fault-tolerant agents directly connected to the end-to-end server, this name must be set to `MASTERDM`.

### **CPUHOST** (*cpu name*)

The name of the host CPU of the agent. It is required for standard and extended agents. The host is the IBM Workload Scheduler CPU with which the standard or the extended agent communicates and where its access method resides.

For standard agents directly connected to the end-to-end server, this name must be set to `OPCMaster`.



**Note:** The host cannot be another standard or extended agent.

### **CPUACCESS** (*access method*)

The name of the access method. It is valid for extended agents and must be the name of a file that resides in the IBM Workload Scheduler `home/methods` directory on the host CPU of the agent.

For details about the access methods, refer to *IBM Workload Scheduler: Administration Guide*.

### **CPUTYPE** (**SAGENT**|**XAGENT**|**FTA**)

The CPU type specified as one of the following:

#### **FTA**

Fault-tolerant agent, including domain managers, and backup domain managers.

**SAGENT**

Standard agent. This includes also IBM Workload Scheduler agents that connect to Dynamic Workload Broker environments.

**XAGENT**

Extended agent.

The default type is FTA.

**CPUAUTOLNK(OFF|ON)**

Autolink is most effective during the initial startup sequence of each plan. Then, a new Symphony file is created and all the CPUs are stopped and restarted.

For a fault-tolerant agent or standard agent, specify ON so that, when the domain manager starts, it sends the new Production Control file (Symphony) to start the agent and open communication with it.

For the domain manager, specify ON so that, when the agents start, they open communication with the domain manager.

Specify OFF to initialize an agent when you submit a link command manually from the Modify Current Plan dialogs.



**Note:** If the extended agent workstation is manually set to Link, Unlink, Active, or Offline, the command is sent to its host CPU.

**CPUFULLSTAT (ON|OFF)**

It applies only to fault-tolerant agents that are not attached directly to the end-to-end server. If you specify ON for a directly connected fault-tolerant agent, the value is forced to OFF. If you specify OFF for a domain manager, the value is forced to ON.

Specify ON for the link from the domain manager to operate in Full Status mode. In this mode, the agent is kept updated about the status of jobs and schedules running on other CPUs in the network.

Specify OFF for the agent to receive status information only about the jobs and schedules on other CPUs that affect its own jobs and schedules. This can improve the performance by reducing network traffic.

To keep the Production Control file for an agent at the same level of detail as its domain manager, set CPUFULLSTAT and CPURESDEP to ON. Always set these modes to ON for backup domain managers.

**CPURESDEP (ON|OFF)**

It applies only to fault-tolerant agents. If you specify OFF for a domain manager, the value is forced to ON.

Specify ON to have the agent's Production Control process operate in Resolve All Dependencies mode. In this mode, the agent tracks dependencies for all its jobs and schedules, including those running on other CPUs.



**Note:** Note that CPUFULLSTAT must also be ON so that the agent is informed about the activity on other CPUs.

Specify OFF if you want the agent to track dependencies only for its own jobs and schedules. This reduces CPU usage by limiting processing overhead.

To keep the Production Control file for an agent at the same level of detail as its domain manager, set CPUFULLSTAT and CPURESDEP to ON. Always set these modes to ON for backup domain managers.

### **CPUSERVER(*server ID*)**

It applies to fault-tolerant and to standard agents, regardless of whether they are attached directly to `OPCMaster` or to a domain manager. Omit this option for domain managers.

This keyword can be a letter or a number (A-Z or 0-9) and identifies a server (Mailman) process on the domain manager that sends messages to the agent. The IDs are unique to each domain manager, so you can use the same IDs for agents in different domains without conflict. If more than 36 server IDs are required in a domain, consider dividing it into two or more domains.

If a server ID is not specified, messages to a fault-tolerant or standard agent are handled by a single Mailman process on the domain manager. Entering a server ID causes the domain manager to create an additional Mailman process. The same server ID can be used for multiple agents. The use of servers reduces the time required to initialize agents, and generally improves the timeliness of messages. However, it increases expenditure in local computer resources. As a guide, additional servers should be defined to prevent a single Mailman from handling more than eight agents.

### **CPULIMIT( *value* | 1024)**

Specifies the number of jobs that can run at the same time in the workstation. The default value is 1024. The accepted values are integers from 0 to 1024. If you specify 0, no jobs are launched on the workstation.

For fault-tolerant and standard agents having a direct connection to the end-to-end server, changes in this value come into effect during the extension or replanning of the current plan, not during the renewal of the Symphony file.

### **CPUTZ( *timezone* | UTC)**

Specifies the local time zone of the distributed agent. It must match the time zone of the operating system on which the distributed agent runs. If the time zone does not match that of the agent, the message `AWSBHT128I` is displayed in the log file of the distributed agent. The default is UTC (universal coordinated time).

To avoid inconsistency between the local date and time of the jobs and of the Symphony creation, use the `CPUTZ` keyword to set the local time zone of the fault-tolerant workstation. If the Symphony creation date is later than the current local date of the fault-tolerant workstation, Symphony is not processed.

In the end-to-end with fault tolerance capabilities environment, time zones are disabled by default when installing or upgrading IBM Z Workload Scheduler. If the CPUTZ keyword is not specified, time zones are disabled. For additional information on how to set the time zone in an end-to-end with fault tolerance

capabilities network, see [Enabling time zones on page 133](#). If the fault-tolerant workstation has the option of "Daylight saving changes" automatically set, the values that you must put in the CPU TZ parameter are the following, specified under the `../zoneinfo/` bin directory:

- Africa/....
- America/....
- Antarctica/....
- Arctic/....
- Asia/....
- Atlantic/....
- Australia/....
- Brazil/....
- Canada/....
- Chile/....
- Europe/....
- Indian/....
- Mexico/....
- Mideast/....
- Pacific/....
- US/....

where you can replace `....` with the files that you find in every subdirectory previously mentioned (it depends on your geographical region). If the fault-tolerant-workstation does not have the option "Daylight saving changes" automatically set, see [Enabling time zones on page 133](#).

#### **CPUUSER** (*default user*`(tws)`)

Specifies the default user for the workstation. The maximum length is 47 characters. The default value is `tws`. If the operating system used by this CPU treats the user ID as case sensitive, then this parameter must be coded in the correct case. The value of this option is used only if you have not defined the user in the `JOBUSR` option of the `SCRPTLIB` `JOBREC` statement or using the `EQQUX001` job-submit exit (for centralized scripts).

If you use this keyword to specify the name of the user who submits the specified script or command on a Windows™ fault-tolerant workstation, you can:

- Associate the user name to the Windows™ workstation in the `USRREC` initialization statement.
- Define the user name and password in a file locally stored in the Windows™ workstation and set `LOCALPSW(YES)` in the [TOPOLOGY on page 51](#) statement.



**Note:** IBM Z Workload Scheduler looks for the `USRREC` statement first, then for the local file, if you set `LOCALPSW(YES)`.

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:

```
CPUUSER('XXXXX\user1')
```

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

```
CPUUSER('administrator@mywindow.com')
```

### **SSLLEVEL(ON|OFF|ENABLED|FORCE)**

Must have one of the following values:

#### **ON**

The workstation uses SSL authentication when it connects with its domain manager. The domain manager uses the SSL authentication when it connects with a domain manager of a parent domain. However, it refuses any incoming connection from its domain manager if the connection does not use SSL authentication.

#### **OFF**

The workstation does not support SSL authentication for its connections. This is the default.

#### **ENABLED**

The workstation uses SSL authentication only if another workstation requires it.

#### **FORCE**

The workstation uses SSL authentication for all of its connections. It refuses any incoming connection if it is not SSL.

If this attribute is set to off or omitted, the workstation is not intended to be configured for SSL. In this case, any value for **SSLPORT** will be ignored. You must also set the **nm ssl port** local option to 0 to be sure that this port is not opened by netman.

### **SSLPORT(SSL port number|31113)**

Defines the port used to listen for incoming SSL connections. This value must match the one defined in the **nm SSL port** local option of the workstation. It must be different from the **nm port** local option that defines the port used for normal communications. If **SSLLEVEL** is specified but **SSLPORT** is missing, 31113 is used as the default value. If not even **SSLLEVEL** is specified, the default value of this parameter is 0 on fault-tolerant workstations, which indicates that no SSL authentication is required.

### **FIREWALL(YES| NO)**

Specifies if the communication between a workstation and its domain manager must cross a firewall. If you set the **FIREWALL** keyword for a workstation to YES, it means that a firewall exists between that particular workstation and its domain manager, and that the link between the domain manager and the workstation (which can itself be another domain manager) is the only allowed link between the respective domains. Also, for all the workstations having this option set to YES, the commands to start (*start wkstation*) or stop (*stop wkstation*) the workstation or to get the standard list (*showjobs*) are transmitted through the domain hierarchy instead of opening a direct connection between the master (or domain manager) and the workstation. The



default value for **FIREWALL** is NO, meaning that there is no firewall boundary between the workstation and its domain manager.

To specify that an extended agent is behind a firewall, set the **FIREWALL** keyword for the host workstation. The host workstation is the IBM Workload Scheduler workstation with which the extended agent communicates and where its access method resides.

## Using the DOMREC statement

### DOMREC

#### Purpose

This statement begins a domain definition. You must specify one `DOMREC` for each IBM Workload Scheduler domain you want to add to the end-to-end with fault tolerance capabilities network, with the exception of the master domain. The domain name used for the master domain is `MASTERDM`. The master domain consists of the controller, which acts as the master domain manager, and of any fault-tolerant and standard agents that are directly attached to it.

`DOMREC` is defined in the member of the `EQQPARM` library, as specified by the `TPLGYMEM` keyword in the `TOPOLOGY` statement.

#### Format

```
►► DOMREC — DOMAIN — ( — domain name — ) — DOMMNGR — ( — domain manager name — ) — DOMPARENT — ( —  

  ► — parent domain — ) ►◄
```

#### Parameters

##### **DOMAIN**(*domain name*)

The name of the domain, consisting of up to 16 characters starting with a letter. It can contain dashes and underscores.

##### **DOMMNGR**(*domain manager name*)

The IBM Workload Scheduler workstation name of the domain manager. It must be a fault-tolerant agent running in full status mode.

##### **DOMPARENT**(*parent domain*)

The name of the parent domain.

## Defining user IDs and passwords for Windows™ users

## USRREC

### Purpose

This statement defines the passwords for the users who need to schedule jobs to run on Windows™ workstations. Omit it if your scheduling environment does not include these workstations.

USRREC is defined in the member of the EQQPARM library as specified by the USERMEM keyword in the [TOPOLOGY on page 51](#) statement.

### Format

► USRREC — USRCPU — ( — *cpu name* — ) — USRNAM — ( — *logon ID* — ) — USRPSW — ( — *password* — ) ►

### Parameters

#### USRCPU(*cpuname*)

The IBM Workload Scheduler workstation on which the user can launch jobs. It consists of four alphanumeric characters, starting with a letter. It is valid only on Windows™ workstations.

#### USRNAM(*logon ID*)

The user name of a Windows™ workstation. It can include a domain name and can consist of up to 47 characters.

Windows™ user names 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 user 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@mywindow.com`, you must enclose the entire character string in single quotes:

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

#### USRPSW(*password*)

The user password for the user of a Windows™ workstation. It can consist of up to 31 characters and must be enclosed in a single quotation mark. The password might be case-sensitive, depending on the Windows™ version. Do not specify this keyword if the user does not need a password. You can change the password every time you create a Symphony file (that is, when creating a CP extension).

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@mywindow.com`, you must enclose the entire character string in single quotes:

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

The password is stored in the USRINFO member in plaintext, meaning that *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 Z Workload Scheduler: Planning and Installation*.

Alternatively, if you do not want to set some or all the user IDs and passwords in the USRREC statement, define them in a local file on the Windows™ workstation, by using the user utility, and set LOCALPSW(YES) in the [TOPOLOGY on page 51](#) statement.

## Defining the end-to-end with fault tolerance capabilities configuration with the TOPOLOGY statement

The TOPOLOGY initialization statement is used to define parameters related to the IBM Workload Scheduler topology, such as the port number for netman, the installation path in USS, and the path to the server working directory in USS.

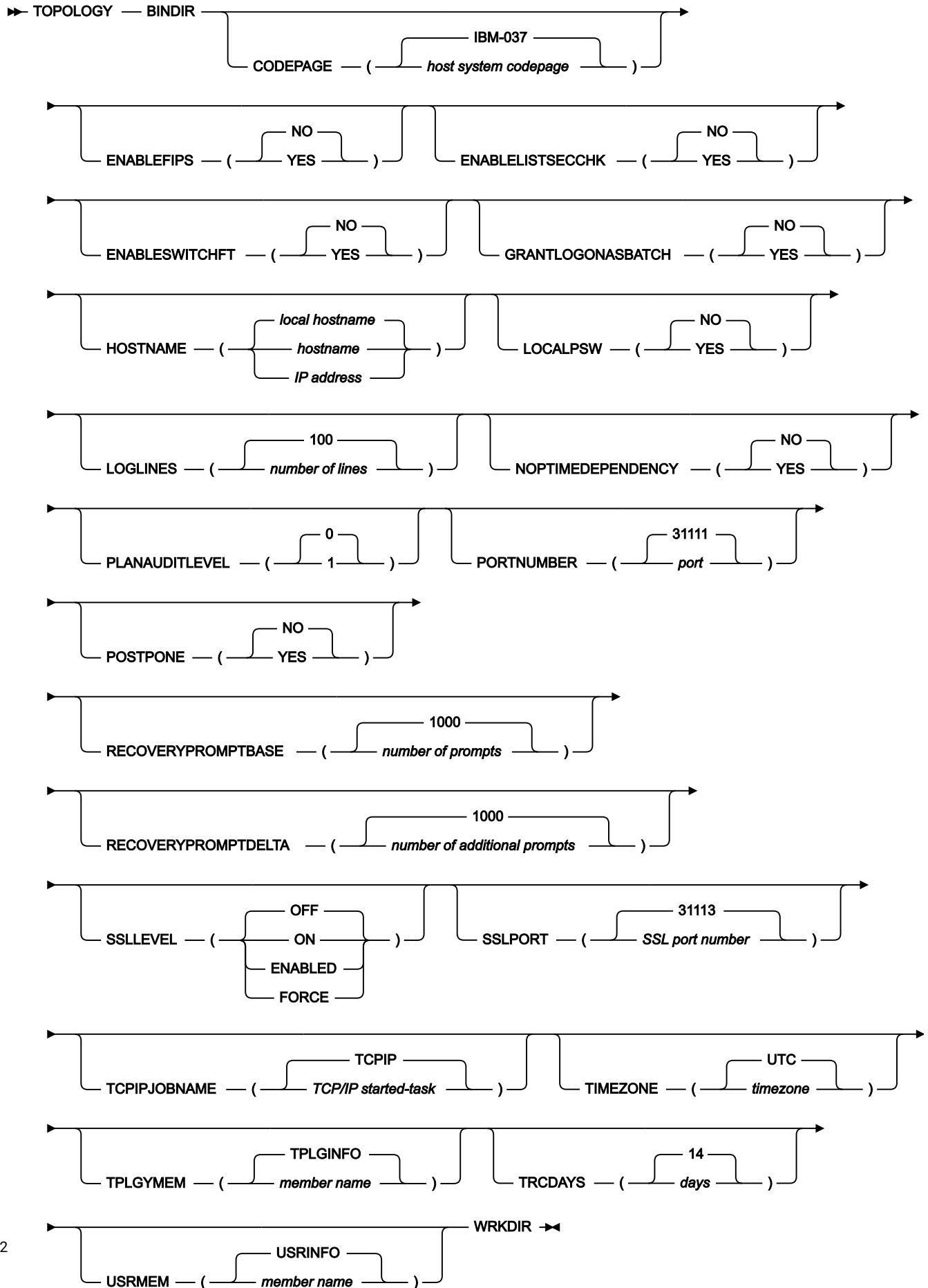
The TPLGYMEM parameter of the TOPOLOGY statement specifies the name of the member with the DOMREC and CPUREC statements. The USRMEM parameter specifies the name of the member with the USRREC initialization statement.

For details about these statements, see *Customization and Tuning*.

## TOPOLOGY

### Purpose

This statement includes all the parameters related to the end-to-end scheduling with fault tolerance capabilities and is defined in the member of the EQQPARM library, as specified by the TPLGYPRM parameter in the BATCHOPT and SERVOPTS statements.



## Parameters

### **BINDIR**(*directory name*)

Specifies the base file system directory where binaries, catalogs, and other files are installed and shared among subsystems.

The specified directory must be the same as the directory where the binaries are, without the final *bin*. For example, if the binaries are installed in `/usr/lpp/TWS/V8R2M0/bin` and the catalogs are in `/usr/lpp/TWS/V8R2M0/catalog/C`, the directory must be specified in the BINDIR keyword as follows: `/usr/lpp/TWS/V8R2M0`.

### **CODEPAGE**(*host system codepage*|IBM-037)

Specifies the name of the host codepage and applies to the end-to-end scheduling with fault tolerance capabilities. You can provide the IBM® – xxx value, where xxx is the EBCDIC code page. The default value, IBM® – 037, defines the EBCDIC codepage for US English, Portuguese, and Canadian French. This is a list of the EBCDIC code pages to choose from:

#### **IBM® – 939**

Japan Extended

#### **IBM® – 937**

Taiwan

#### **IBM® – 935**

China

#### **IBM® – 933**

Korea

#### **IBM® – 975**

Greece

#### **IBM® – 971**

Iceland

#### **IBM® – 970**

Latin 2

#### **IBM® – 838**

Thai

#### **IBM® – 500**

International

#### **IBM® – 424**

Israel

**IBM® – 297**

France

**IBM® – 285**

U.K.

**IBM® – 284**

Spain - Latin America

**IBM® – 280**

Italy

**IBM® – 278**

Sweden - Finland

**IBM® – 277**

Denmark - Norway

**IBM® – 274**

Belgium

**IBM® – 273**

Germany

**IBM® – 1388**

China

**IBM® – 1122**

Estonia

**IBM® – 1112**

Baltic

**IBM® – 1047**

Open Systems

**IBM® – 1026**

Latin 5 (Turkey)

**IBM® – 1025**

Cyrillic

Make sure that this keyword and the `Host Code-page` session parameter of the 3270 emulator from which you operate with IBM Z Workload Scheduler are set to the same value. This is to avoid problems in the translation of certain characters between IBM Z Workload Scheduler and the fault-tolerant workstations. The `Host Code-page` parameter specifies the table that maps EBCDIC codes from the Host to appropriate ANSI codes on the personal computer.

**ENABLEFIPS(NO|YES)**

Indicates whether the SSL communication should comply with FIPS standards. Specify YES to have a FIPS compliant SSL communication. This keyword is ignored if the SSL communication is not enabled.

For more information about FIPS compliance, see *Planning and Installation*.

**ENABLELISTSECCHK(YES|NO)**

This is a security option that controls which objects in the plan the user is permitted to list when running a Dynamic Workload Console query or a conman `show` command. If set to YES, objects in the plan returned from a query or `show` command are shown to the user only if the user has been granted the list permission in the security file. The default value is NO. Change the value to YES if you want to check for the list permission in the security file.

**ENABLESWITCHFT(YES|NO)**

This option enables the `switch fault tolerance` feature. With this option a complete fault tolerance is granted when a domain manager is switched. This option prevents events being lost when a switch domain manager command is performed. See the IBM Workload Scheduler documentation for details about this feature.

**GRANTLOGONASBATCH(YES|NO)**

This is for jobs running on Windows™ platforms only. If set to yes, the log on users for Windows™ jobs are automatically granted the right to log on as batch job. If set to NO, or omitted, the right must be granted manually to each user or group. The right cannot be granted automatically for users running jobs on a backup domain controller, so you must grant those rights manually.

**HOSTNAME (*hostname|IP address|local hostname*)**

Specifies the hostname or the IP address used by the server in the end-to-end with fault tolerance capabilities environment. The default is the hostname returned by the operating system.

If you change the value, you also need to restart the end-to-end server and renew the Symphony file.

You can define a virtual IP address for each server of the active controller and the standby controllers. If you use a dynamic virtual IP address in a sysplex environment, when the active controller fails and the standby controller takes over the communication, the distributed agents automatically switch the communication to the server of the standby controller.

To change the `HOSTNAME` of a server perform the following actions:

1. Set the `nm ipvalidate` keyword to `none` in the `localopts` file on the primary domain managers.
2. Change the `HOSTNAME` value of the server using the `TOPOLOGY` statement.
3. Restart the server with the new `HOSTNAME` value.
4. Renew the Symphony file.
5. If the renewal ends successfully, you can set the `ipvalidate` to `full` on the first level domain managers.

If this parameter is not specified, then the end-to-end server uses the `HOSTNAME` as specified in the TCP/IP Data `HOSTNAME` parameter. In this case the end-to-end server processes all the incoming requests regardless of the IP

address used (bind with `INADDR_ANY`). If this parameter is specified, then IBM Workload Scheduler uses only the address/alias\_name supplied (a specific bind to a single IP address is issued). Incoming requests to other IP addresses are not processed. When more than one IP address is defined to the system where the end-to-end server is active, be aware that: if `HOSTNAME` is specified and this connection fails, access to the server will require that the server be reconfigured to an alternative IP address before a connection with the domain manager can be established.



**Note:** If the `HOSTNAME` value represents a `DVIPA` address, then make sure that the user of the end-to-end server has `UPDATE` authorization to the `EZB.BINDDVIPARANGE` resource. Failure to do this might cause problems, such as the failure to re-link the agents after a `Symphony Renew`, `CP replan` Or `CP extend`.

### **LOCALPSW(YES|NO)**

Set this parameter to YES if the user ID and password, required to run a job on a Windows™ workstation, must be searched from a local file on the workstation, when missing from the Symphony file. That is the case when you do not specify the `USRREC` statement, so that the planning process does not include the user's credentials the Symphony file. If you set `LOCALPSW` to YES, you do not need to define the `USRREC` statements for the users defined locally. Ensure that the Windows™ workstation supports the local search. The default is NO.

### **LOGLINES(number of lines|100)**

Specifies the maximum number of lines that the job log retriever returns for a single job log. The default value is 100. In all cases, the job log retriever does not return more than half the number of records existing in the input queue. When the job log retriever does not return all the job log lines, in the retrieved job log output you can read the following line that specifies the number (nn) of job log lines not displayed between the first lines and the last line of the job log, that is:

```
*** nn lines have been discarded. Final part of Joblog ... *****
```

### **NOPTIMEDEPENDENCY(YES|NO)**

Acts on NOPed operations defined to run on fault-tolerant workstations or standard agents and applies to both centralized and non-centralized scripts.

Set this option to `YES` and the operations complete only after their time dependency has expired.

Set this option to `NO` and the operations complete as soon as all their predecessors have completed, even if their time dependency has not expired. This is the default setting.

### **PLANAUDITLEVEL(0|1)**

Enables or disables plan auditing for distributed agents. Valid values are 0 to disable plan auditing, and 1 to activate plan auditing. Auditing information is logged to a flat file in the `TWA_home/TWS/audit/plan` directory. Each IBM Workload Scheduler workstation maintains its own log. Only actions are logged in the auditing file, not the success or failure of any action.

If you change the value, you also need to restart the end-to-end server and renew the Symphony file.



**PORTNUMBER(port|31111)**

Defines the TCP/IP port number used by the server to communicate with the distributed agents. This value must be different from that specified in the SERVOPTS member. The default value is 31111. The accepted values are from 0 to 65535.

If you change the value, you also need to restart the end-to-end server and renew the Symphony file.

**POSTPONE(YES|NO)**

Defines the time and mode for stopping the end-to-end with fault tolerance capabilities network upon the issue of a re-plan, Symphony renew, or current plan extension command. Stopping the network is necessary to allow for the generation and distribution of a new Symphony file. This keyword provides a choice for gradually stopping the end-to-end distributed agents after the Symphony plan is generated, as opposed to stopping the entire end-to-end with fault tolerance capabilities network simultaneously when generation begins. Stopping the network progressively reduces the outage imposed on the distributed agents to the time strictly necessary to receive the new Symphony. This is recommended with large numbers of distributed agents and where fault-tolerance is largely used.

Progressive stop is not useful if the following are widely used:

- Centralized jobs
- Special resources and jobs depending on z/OS® jobs

because none of the jobs that depend on these objects can start until the link with the end-to-end server is reactivated.

Values can be:

**YES**

The end-to-end with fault tolerance capabilities network is stopped after the `symnew` file is created. The server stops only the primary domain manager using the `stop progressive` command and sends the new Symphony file to it. The `stop progressive` command stops scheduler production processes hierarchically: each domain manager stops the agents in its domain and stops itself while `stop progressive` continues to run on subordinate domains. Every agent is restarted as soon as it is issued its copy of the Symphony file.

To function, the `YES` value requires that

- The `ENABLESWITCHFT` parameter be set to `NO` (or not set at all).
- The domain managers have the `mm start tomserver` local option set to `yes`. See [Rules for customizing the CCLog properties file on page 101](#) for reference.

**NO**

The entire end-to-end with fault tolerance capabilities network is stopped when the synchronization between the server and the controller is started. In this case, a normal `stop`

command is sent concurrently to all the distributed agents. This suspends batch scheduling on all distributed agents until the Symphony file generation completes. Every agent is restarted as soon as it is issued its copy of the Symphony file.

This is the default.

**RECOVERYPROMPTBASE(*number of prompts*|1000)**

Specifies the maximum number of prompts that can be displayed to the operator after a job abends. The default value is 1000.

**RECOVERYPROMPTDELTA(*number of prompts*|1000)**

Specifies an additional number of prompts with respect to the value defined in RECOVERYPROMPTBASE to use when a job is rerun after abending and the limit specified in RECOVERYPROMPTBASE has been reached. The default value is 1000.

**SSLLEVEL(ON|OFF|ENABLED|FORCE)**

Defines the type of SSL authentication for the workstation. It must have one of the following values:

**ON**

The server uses SSL authentication only if another workstation requires it.

**OFF**

The server does not support SSL authentication for its connections. It is the default value.

**ENABLED**

The server uses SSL authentication only if another workstation requires it.

**FORCE**

The server uses SSL authentication for all of its connections. It refuses any incoming connection if it is not SSL.

If you change the value, you also need to restart the end-to-end server and renew the Symphony file.

**SSLPORT(*SSL port number*|31113)**

Defines the port used to listen for incoming SSL connections on the server. It substitutes the value of **nm SSL port** in the localopts file, activating SSL support on the server. If **SSLLEVEL** is specified and **SSLPORT** is missing, 31113 is used as the default value. If not even **SSLLEVEL** is specified, the default value of this parameter is 0 on the server, which indicates that no SSL authentication is required.

If you change the value, you also need to restart the end-to-end server and renew the Symphony file.

**TCPIPJOBNAME (*TCP/IP started-task name*|TCPIP)**

Specifies the TCP/IP started-task name used by the server. Set this keyword when you have multiple TCP/IP stacks or a TCP/IP started task with a name different from TCPIP. You can specify a name, from 1 to 8 alphanumeric or national characters, where the first character is alphabetic or national.

**TIMEZONE** (*timezone*|UTC)

Defines the local time zone in the z/OS® system where the controller runs, for example `TIMEZONE('Europe/Rome')`. This value must match the GMT offset specified in the CLOCKxx parmlib member, for example `E.01.00.00` for Rome Standard Time, `E.02.00.00` for Rome Daylight Saving Time.

If you do not specify this parameter, or the specified value is incorrect, the scheduler uses the UTC value as default.

**TPLGYMEM**(*member name*|**TPLGINFO**)

Specifies the PARMLIB member where the domain (DOMREC) and CPU (CPUREC) definitions specific to the end-to-end with fault tolerance capabilities environment are. The default value is TPLGINFO.

If you change the value, you also need to restart the end-to-end server and renew the Symphony file.

**TRCDAYS** (*days*|14)

Specifies the number of days the trace files and stdlist directory are kept before being deleted. Every day the USS code creates the new stdlist directory to contain the logs for the day; all log directories older than the number of days specified in TRCDAYS() are automatically deleted. The default value is 14. Specify 0 if you do not want the trace files to be deleted.

**USRMEM**(*member name*|**USRINFO**)

Specifies the PARMLIB member where the user definitions are. This keyword is optional (for example, you can omit it if you are working with UNIX™ CPUs). The default value is USRINFO.

If you change the value, you also need to restart the end-to-end server and renew the Symphony file.

**WRKDIR**(*directory name*)

Specifies the location of the files of a subsystem.

Each IBM Z Workload Scheduler subsystem using the end-to-end scheduling with fault tolerance capabilities must have its own WRKDIR.

## Specifying the name of the end-to-end server with the TPLGYSRV keyword

Add the statement:

```
OPCOPTS TPLGYSRV(server_name)
```

to activate the end-to-end scheduling with fault tolerance capabilities in the controller by starting the end-to-end enabler task. The **server\_name** you specify with this keyword is that of the server that handles the events to and from the distributed agents. Only one server can handle events to and from the distributed agents.

## Activating end-to-end with fault tolerance capabilities communication in the server with the TPLGYPRM keyword

Add the statement:

```
SERVOPTS PROTOCOL(E2E) TPLGYPRM(member_name|TPLGPARM)
```

to specify:

- That the communication protocol is E2E.
- The name of the member in the EQQPARM library where you define the [TOPOLOGY on page 51](#) statement.

This statement specifies initialization options for the end-to-end server.

## Activating the end-to-end scheduling with fault tolerance capabilities in the Daily Planning batch programs

Add the statement:

```
BATCHOPT TPLGYPRM(member_name|TPLGPARM)
```

to specify the name of the member in the EQQPARM library where you define the [TOPOLOGY on page 51](#) statement.

This statement enables the scheduler to create the Symphony file and to run jobs in the distributed environment.

## Defining standards for generating job names in the Symphony file

Add the statement:

```
JTOPTS TWSJOBNAME(EXTNAME|EXTNOCC|JOBNAME|OCCNAME)
```

to define the rules the scheduler is to follow for generating the job name in the Symphony file in USS. The default parameter is OCCNAME.

If you specify EXTNAME, EXTNOCC, or JOBNAME, the job name in the Symphony file is made up according to one of the following formats:

- `X_Num_JobInfo`, when the job is created.
- `X_Num_Ext_JobInfo`, when the job is first deleted and then re-created in the current plan.

where:

**X**

Is **J** for normal operations, **P** for jobs representing pending predecessors, or **R** for recovery jobs. For jobs representing pending predecessors, the job name is in all cases generated by using the OCCNAME criterion. This is because, in the case of pending predecessors, the current plan does not contain the required information (except the name of the occurrence) to build the Symphony name according to the other criteria.

**Num**

The operation number.

**Ext**

The hexadecimal value of a sequential number that is increased every time an operation is deleted and then re-created.

**JobInfo**

Depends on the chosen criterion:

**For EXTNAME**

*JobInfo* is filled with the first 32 characters of the extended job name associated to that job (if it exists) or with the 8-character job name (if the extended name does not exist). Note that the extended job name, in addition to being defined in the database, must also exist in the current plan.

**For EXTNOCC**

*JobInfo* is filled with the first 32 characters of the extended job name associated to that job (if it exists) or with the application name (if the extended name does not exist). Note that the extended job name, in addition to being defined in the database, must also exist in the current plan.

**For JOBNAME**

*JobInfo* is filled with the 8-character job name.

If you specify OCCNAME, the job name in the Symphony file is made up according to one of the following formats:

- *X\_Num\_Application Name*, when the job is created.
- *X\_Num\_Ext\_Application Name*, when the job is first deleted and then re-created in the current plan.

where:

**X**

Can be J for normal operations, P for jobs representing pending predecessors, and R for recovery jobs.

**Num**

The operation number.

**Ext**

A sequential decimal number that is increased every time an operation is deleted and then re-created.

**Application Name**

The name of the occurrence to which the operation belongs.

The criterion used to generate a job name will be maintained throughout the entire life of the job.

To choose the EXTNAME, EXTNOCC, or JOBNAME criterion, the EQQTWSOU data set must have a record length of 160 bytes. Before using any of the above keywords, you must migrate the EQQTWSOU data set. Sample EQQMTWSO is available to migrate this data set from 120 to 160 bytes.

The following limitations apply when using the EXTNAME and EXTNOCC criteria:

- The job name in the Symphony file can contain only alphanumeric characters, dashes, and underscores. All the other characters accepted for the extended job name are changed to dashes. Note that a similar limitation applies also for JOBNAME: when defining members of partitioned data sets (such as the script or the job libraries), national characters can be used, but they are changed to dashes in the Symphony file.
- The job name in the Symphony file must be in uppercase. All lowercase characters in the extended name are automatically changed to uppercase by IBM Z Workload Scheduler.



**Note:** Using the job name (or the extended name as part of the job name) in the Symphony file implies that it becomes a key for identifying the job. This also means that the extended name or jobname is used as a key for addressing all the events directed to the agents. For this reason, be aware of the following for operations included in the Symphony file:

- Editing the Extended Name is inhibited for operations created when TWSJOBNAME was set to EXTNAME or EXTNOCC.
- Editing the Jobname is inhibited for operations created when the TWSJOBNAME keyword was set to EXTNAME or JOBNAME.

## Step 7. Defining the fault-tolerant workstations

### About this task

In this step you define as fault-tolerant workstations in the IBM Z Workload Scheduler workstation description database all the agents (fault-tolerant or standard) for which you created `CPUREC` statements.

Follow these steps to create a definition for each standard or fault-tolerant agent for which you created a `CPUREC` statement:

1. Access the workstation panel by entering option 1.1 from the main ISPF menu. The MAINTAINING WORK STATION DESCRIPTIONS menu is displayed.
2. Select option 2 (`LIST`). The LIST OF WORKSTATION DESCRIPTIONS panel, shown next, is displayed.

Figure 7. EQQWMLSL - List of workstation descriptions

```

EQQWMLSL ----- LIST OF WORK STATION DESCRIPTIONS ----- ROW 1 TO 6
Command ==>>> SCROLL ==>>

Enter the CREATE command above to create a workstation description or enter
any of the following row commands:
B - Browse, D - Delete, M - Modify, C - Copy.

Row  Work station          T  R  Last update
cmd  name  description              user  date
'    CPU1  Main JES processor      C  A  XRAYNER  06/07/30
'    PAY1  payroll office          C  A  XRAYNER  06/07/30
'    PRT1  Printer pool            C  N  XRAYNER  06/07/30
'    SETP  Used to prepare JCL     G  N  XRAYNER  06/07/30
'    STC1  Processor for started   C  A  XRAYNER  06/07/30
'    WT01  Messages for NetView    G  A  XRAYNER  06/07/30
***** BOTTOM OF DATA *****

```

3. Enter the `CREATE` command. The CREATING GENERAL INFORMATION ABOUT A WORKSTATION panel, shown next, is displayed.

Figure 8. EQQWCGEP - Creating general information about a workstation

```

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  ==>
DESCRIPTION        ==>
WORK STATION TYPE  ==> -----
                                G General, C Computer, P Printer
                                R Remote Engine
REPORTING ATTR     ==>
                                A Automatic, S Manual start and completion
                                C Completion only, N Non reporting

PRINTOUT ROUTING   ==>
SERVER USAGE       ==>
DESTINATION        ==> -----
                                The ddname of daily plan printout data set
                                Parallel server usage, B, N, P, or C
                                Name of destination

Options: allowed Y or N
SPLITTABLE         ==> N      JOB SETUP           ==> N
STARTED TASK, STC ==> N      WTO              ==> N
AUTOMATION         ==> N      FAULT-TOLERANT AGENT ==>
WAIT               ==> N      Z-CENTRIC AGENT  ==>
VIRTUAL           ==> N      DYNAMIC          ==> N

REMOTE ENGINE TYPE ==>
Defaults:
TRANSPORT TIME    ==> 00.00   Time from previous work station HH.MM
DURATION          ==> 00.00.00 Duration for a normal operation HH.MM.SS

```

4. Type in these fields as follows:

**WORK STATION NAME**

The same name you used in [CPUREC on page 42](#).

**WORK STATION TYPE**

C

**REPORTING ATTR**

A

**FT Work station**

Y

**SERVER USAGE**

N

5. Specify the `PRINTOUT ROUTING` field and leave the `DESTINATION` field blank.

6. Set or leave all Options to N, and all Defaults to 0.

Note that the R, A, and M commands are meaningless with fault-tolerant workstations.

## Step 8. Verifying your installation

### About this task

When you have completed the steps described in the previous sections, check the message logs to verify your activation of the end-to-end scheduling with fault tolerance capabilities on IBM Z Workload Scheduler.

Perform the following steps:

1. First, start the controller and check that these messages appear in the controller EQQMLOG:

```

EQQZ005I OPC SUBTASK END TO END ENABLER IS BEING STARTED
EQQZ085I OPC SUBTASK END TO END SENDER IS BEING STARTED
EQQZ085I OPC SUBTASK END TO END RECEIVER IS BEING STARTED
EQQG001I SUBTASK END TO END ENABLER HAS STARTED
EQQG001I SUBTASK END TO END SENDER HAS STARTED
EQQG001I SUBTASK END TO END RECEIVER HAS STARTED
EQQW097I END-TO-END RECEIVER STARTED SYNCHRONIZATION WITH THE EVENT MANAGER
EQQW097I 0 EVENTS IN EQQTWSIN WILL BE REPROCESSED
EQQW098I END-TO-END RECEIVER FINISHED SYNCHRONIZATION WITH THE EVENT MANAGER
EQQ3120E END-TO-END TRANSLATOR SERVER PROCESS IS NOT AVAILABLE
EQQZ193I END-TO-END TRANSLATOR SERVER PROCESSS NOW IS AVAILABLE

```

The first time that the controller is started with the fault-tolerant end-to-end scheduling in use or after the end-to-end event data sets (EQQTWSIN and EQQTWSOU) have been reallocated, the end-to-end event data sets need to be formatted. The following messages will appear in the controller EQQMLOG before the messages about sender and receiver have started:

```

EQQW030I A DISK DATA SET WILL BE FORMATTED, DDNAME = EQQTWSIN
EQQW030I A DISK DATA SET WILL BE FORMATTED, DDNAME = EQQTWSOU
EQQW038I A DISK DATA SET HAS BEEN FORMATTED, DDNAME = EQQTWSIN
EQQW038I A DISK DATA SET HAS BEEN FORMATTED, DDNAME = EQQTWSOU

```

2. After the controller has started, start the end-to-end server and check that these messages appear in the end-to-end server EQQMLOG:

```

EQQPH00I SERVER TASK HAS STARTED
EQQPH33I THE END-TO-END PROCESSES HAVE BEEN STARTED
EQQZ024I Initializing wait parameters
EQQPT01I Program "/usr/lpp/TWS/TWS850/bin/translator" has been started,
pid is 67371783
EQQPT01I Program "/usr/lpp/TWS/TWS850/bin/netman" has been started,
pid is 67371919
EQQPT56W The /DD:EQQTWSIN queue has not been formatted yet
EQQPT22I Input Translator thread stopped until new Symphony will be available

```

These messages are correct the first time you start the server because no Symphony file has yet been created. When a Symphony file exists and is active, the two messages at the bottom are replaced by:

```

EQQPT20I Input Translator waiting for Batchman and Mailman are started
EQQPT21I Input Translator finished waiting for Batchman and Mailman

```

Also, the following messages are displayed at the first startup because the EQQTWSIN and EQQTWSOU data sets are both empty and no Symphony file has yet been created:

```

EQQPT56W The /DD:EQQTWSIN queue has not been formatted yet
EQQPT56W The /DD:EQQTWSOU queue has not been formatted yet

```

## Installing the IBM Workload Scheduler end

Before you begin the installation, review the Interoperability tables shown in the *Memo to Users* document, to ensure that your proposed environment is possible.



When you install the agents of an end-to-end with fault tolerance capabilities network, you should consider the following:

- The name of each agent can have a maximum of four characters, must start with a letter, and all letters must be uppercase.
- The master domain manager is the IBM Z Workload Scheduler controller and is named `OPCMaster`.
- The master domain is predefined and is named `MASTERDM`.

Apart from this, installation is not different from that of the agents of a distributed IBM Workload Scheduler network. However, similarly to installing a distributed IBM Workload Scheduler network, installing large end-to-end with fault tolerance capabilities networks requires some planning of network topology and agent naming conventions.

When you decide to install an end-to-end with fault tolerance capabilities network, it is likely that you either have to install the agents from scratch or that you want to use agents from an already existing IBM Workload Scheduler network.

In the latter case, you want to install an additional instance of IBM Workload Scheduler on computers where the agent is already running just to keep the end-to-end and the distributed environments separate. While you install an additional IBM Workload Scheduler instance, be sure to specify for the following parameters different values from the ones belonging to the existing instance:

- The IBM Workload Scheduler home directory
- The IBM Workload Scheduler user
- The netman port number

## Installation checklist

### About this task

The following is a list of recommended steps for installing the IBM Workload Scheduler agents of an end-to-end with fault tolerance capabilities network:

1. Plan the network topology. For details, see [Step 1. Planning the network topology on page 65](#).
2. Plan a naming convention for the fault-tolerant workstations. For details, see [Step 2. Planning a naming convention on page 67](#).
3. Install the agents and the domain managers. For details, see [Step 3. Installing the agents on page 67](#).
4. Define the domains and the workstations to IBM Z Workload Scheduler. For details, see [Step 4. Defining the topology to IBM Z Workload Scheduler on page 68](#).

Steps [1 on page 65](#) and [2 on page 65](#) are planning activities. They apply particularly to large network situations.

## Installation details

The following sections explain the planning and installation steps.

### Step 1. Planning the network topology

#### About this task

Before you start installing the agents, it is important to dedicate some time to planning the topology of your end-to-end with fault tolerance capabilities network. The time spent for planning will very likely save you time when you start to install, particularly if you will be adding agents to the network at later times. Having a clear idea of where to locate future agents will also guarantee that your network expands in a more systematic, orderly, and logical manner.

To choose a topology that most closely meets your practical requirements, you must consider the costs and benefits of organizing your agents in a single domain or in multiple domains.

You would be likely to choose a single domain if:

- You plan to connect a small number of agents.
- The size of the daily workload you plan to run on the agents is small.
- The rate of your network traffic is acceptable and enables one domain manager to handle the daily communication exchange with all the agents in its domain.
- The network is located within the same geographic location and the communication links with the agents are straightforward.
- The business scope of the workload that will be running on the agents is fairly homogeneous.

Under these conditions, you might also decide to connect standard and fault-tolerant agents directly to the end-to-end server without the interposition of a domain manager. In addition, you can set up only centralized scripts for the jobs that are to run on these agents, in order to achieve total mainframe control of the distributed workload.

You would be likely to choose multiple domains if:

- You plan to connect a huge number of agents and you plan to run a significant quantity of daily jobs. You therefore want to distribute the agents and the workload amongst several domain managers.
- The network will be scattered across different geographic locations and it would be advisable to have a domain manager in each location interface `OPCMaster` and the local agents.
- The workload will comprise jobs for different business functions, departments, and applications and you therefore want to reflect that in the logical organization of your network.
- You do not want to base the fault-tolerance of the entire network on a single domain manager (and its backup). With multiple domains, you would limit the possibility of a domain manager failure only to the agents that depend on it.
- You want to keep the jobs that share dependencies grouped in the same domain, despite geographical, location, and organizational differences for easier management.
- You want to place agents in different domains according to platform and operating system for easier management

A special case of a multiple domain configuration is when every fault-tolerant agent is a domain manager, that is, every FTA has `CPUFULLSTATUS(ON)`. Avoid this configuration, because it is inefficient, in that every message for every event on every FTA must be sent to all the `CPUFULLSTATUS` FTAs. If you use this configuration, you must increase the `REGION SIZE` to an appropriate value, which you obtain by tuning your network environment.

See also section [Choosing the agents on page 20](#) for additional information.

## Step 2. Planning a naming convention

### About this task

Outlining a naming methodology for the agents you plan to install now and in the future is a good strategy, especially because names can include a maximum of four characters. A well-defined and documented naming convention will help you to automatically name any agents you add in the future.

Because it is difficult to have meaningful names of four characters, you should take advantage of each character to provide information about the agent. For example, you could use the first letter to classify the agent as a fault-tolerant workstation from a regular IBM Z Workload Scheduler CPU, the second character to identify the domain, the platform, or the operating system, and the last two digits to identify the specific agent.

Another good idea is to take advantage of the workstation description field available in every IBM Z Workload Scheduler workstation definition. You can use this 32-character field to write more readable information for identifying agents.

## Step 3. Installing the agents

### About this task

The master domain manager of the distributed end is the IBM Z Workload Scheduler controller, referred to as `OPCMASTER`. You must specify `OPCMASTER` in the `Master CPU` field of every new agent you install.

For details about installing the agents, see *IBM Workload Scheduler: Planning and Installation*. If you have not installed IBM Workload Scheduler agents before, here are a few introductory notes about installing IBM Workload Scheduler:

- During installation, there is no difference between installing a fault-tolerant agent or a standard agent. You install a generic IBM Workload Scheduler agent. The difference is made in the `CPUTYPE` keyword of the `CPUREC` statement, where you specify if the agent will be standard or fault-tolerant (or extended). The difference between the two is that a fault-tolerant agent is autonomous in carrying on its workload after it receives the daily plan from its domain manager. A standard agent needs to be managed by its domain manager to run jobs. If the domain manager goes down, a standard agent cannot run jobs. Standard agents require less power and are normally considered for simpler workloads.
- Among the information you are asked to provide while installing all agents, remember:
  - The value of the `Master CPU` field must be `OPCMASTER`.
  - The workstation name in the `This CPU` field is reduced to four characters (the first must be a letter) for end-to-end agents, while it normally extends to 16 characters in IBM Workload Scheduler. If you want to keep the same name in the IBM Z Workload Scheduler definition, you should follow this rule at installation time.
- Domain managers are not determined by the installation program. Their role is specified when you configure the workstations in IBM Z Workload Scheduler.
- If you want to include extended agents in the end-to-end with fault tolerance capabilities network, remember that they are logical entities hosted physically by fault-tolerant or standard agents. For each extended agent that you want to install:

- Install one of the access methods available from IBM Workload Scheduler for Applications on one of your fault-tolerant or standard agents.
- Follow the four-character naming convention as you did for the hosting agent. The name must be different from its host, because it will be defined as a different agent in IBM Z Workload Scheduler.

## Step 4. Defining the topology to IBM Z Workload Scheduler

### About this task

In a normal IBM Workload Scheduler network, you define the sub-domains, the domain managers, and the agents in a database that resides in the master domain manager. In the end-to-end with fault tolerance capabilities environment, you make these definitions to IBM Z Workload Scheduler because the controller acts as the master.

If you have not done it before, now that you have planned the architecture of your distributed network and you have installed the agents, it is time to complete this task. On IBM Z Workload Scheduler, follow these steps:

1. Use the `CPUREC` statement to define each agent and the `DOMREC` statement to define every domain and its corresponding manager. These initialization statements are explained in [Defining the distributed agents and domains on page 42](#).

If you installed agents that run on Windows™, define the user IDs and passwords for the Windows™ users either in the `USRREC` statement or in a local file on the Windows™ workstation. If you define the users locally, you must also set `LOCALPSW(YES)` in the `TOPOLOGY on page 51` statement.

See [Defining user IDs and passwords for Windows users on page 49](#) for a complete description of this statement.

[Table 5: CPUREC parameters on page 68](#) summarizes the `CPUREC` keywords:

**Table 5. CPUREC parameters**

CPUREC parameters	Description
CPUNAME	Name of the IBM Workload Scheduler workstation.
CPUOS	Host CPU operating system related to the IBM Workload Scheduler workstation.
CPUNODE	Node name or the IP address of the CPU.
CPUTCPIP	TCP/IP port number of <code>netman</code> on the current CPU.
CPUDOMAIN	Name of the IBM Workload Scheduler domain of the CPU or <code>MASTERDM</code> for directly connected fault-tolerant agents.
CPUHOST	Name of the host CPU of the agent.
CPUACCESS	Name of the access method.
CPUTYPE	Agent type.
CPUAUTOLNK	Autolink between the agent and the domain manager.

CPUREC parameters	Description
CPUFULLSTAT	Link from the domain manager operates in <code>Full Status</code> mode.
CPURESDEP	Agent's production control process operates in <code>Resolve All Dependencies</code> mode.
CPUSERVER	A <code>mailman</code> server process on the end-to-end server or a domain manager that sends messages to the agent.
CPULIMIT	Number of jobs that can run at the same time in a CPU.
CPUTZ	Local time zone of the fault-tolerant workstation.
CPUUSER	Default user for the workstation.
SSLLEVEL	The workstation uses SSL authentication when it connects with its domain manager.
SSLPORT	Port used to listen for incoming SSL connections.
FIREWALL	The communication between a workstation and its domain manager must cross a firewall.

Some notes about `CPUREC` are:

- The value of `CPUNAME`, the four-character workstation name, is the name you entered in the `This CPU` field at installation time and that you will also enter in the next step when you define the fault-tolerant workstations in the IBM Z Workload Scheduler workstations database.
- The netman port number in `CPUTCPIP` is the number you entered in the `TCP/IP Port Number` field at installation time.
- The name in `CPUDOMAIN` is one of the following:
  - The name of one of the domains you define with the `DOMREC` statement
  - `MASTERDM` for directly connected fault-tolerant agents
- The value of `CPUTYPE` specifies the type of agent, fault-tolerant, standard, or extended, as specified by your installation. You determine which of the fault-tolerant agents are to be the domain managers by specifying their `CPUNAME` in the `DOMREC` statement.
- The value of `CPUACCESS` is one of the access methods you installed for the extended agent. Use it only if you defined `CPUTYPE` as `XAGENT`.
- The values of `CPUFULLSTAT` and `CPURESDEP` must be `ON` for the fault-tolerant agents you intend to be domain managers and backup domain managers. Leave the values as `OFF` for all other fault-tolerant agents to limit network traffic and processing overhead.

Table 6: [DOMREC parameters on page 69](#) summarizes the `DOMREC` keywords:

**Table 6. `DOMREC` parameters**

DOMREC parameters	Description
DOMAIN	Name of the domain that you are defining.
DOMMNGR	Name of the fault-tolerant agent that is to be the manager of this domain.

DOMREC parameters	Description
DOMPARENT	Name of the parent domain.

Some notes about DOMREC are:

- The fault-tolerant agents specified in `DOMMNGR` must have the `CPUFULLSTAT` and `CPURESDEP` switches turned to `ON`.
- In `DOMPARENT` write `MASTERDM`, if you are defining a primary domain. If you are defining a secondary or greater level domain, specify the name of the domain immediately above.

[Table 7: USRREC parameters on page 70](#) summarizes the `USRREC` keywords:

**Table 7. USRREC parameters**

Keywords	Description
USRCPU	The workstation on which the user can launch jobs.
USRNAM	The user name.
USRPWD	The user password.

2. Define all the agents you specified with the `CPUREC` statement as fault-tolerant workstations in the IBM Z Workload Scheduler workstations database. Use either the ISPF dialogs or the Dynamic Workload Console linked to the controller to accomplish this task. [Table 8: Settings for fault-tolerant workstations on page 70](#) summarizes the attributes required for defining these workstations:

**Table 8. Settings for fault-tolerant workstations**

Field	Value	Value Description
Workstation type	C	Computer
Reporting attribute	A	Automatic
Started task, STC	N	No
Destination	Blank	–
Server Usage	N	Neither

3. Activate the definitions by running the `replan`, `extend plan`, or `Symphony renew` programs in IBM Z Workload Scheduler. With either of these programs IBM Z Workload Scheduler creates the Symphony file and distributes it to the primary domain managers. These in turn distribute the file to their subordinate domain managers and agents. The process is repeated by all levels of domain managers until the plan is distributed to the entire network.

## Step 5. Verifying the installation

### About this task

The `replan`, `extend plan`, and `Symphony renew` programs also provide the verification information you need to know if the fault-tolerant workstations have become available and have linked status in the IBM Z Workload Scheduler plan.

When `replan`, `extend plan` or `Symphony renew` has finished running, you can check any of the message logs of:

- The controller to verify that the Symphony file was created successfully and that the fault-tolerant workstations are active and linked.
- The end-to-end server to verify that the Symphony file was created successfully and that the server was able to switch to the new file.
- The plan batch job to verify that the Symphony file was created successfully.

You can also verify that all the workstations are active and linked with either the ISPF dialogs or the Dynamic Workload Console.

# Chapter 3. Customizing

Customization tasks:

- Defining job scripts for fault-tolerant workstations in [Defining centralized and non-centralized scripts on page 72](#).
- Setting SSL security for the distributed agents in [Setting the security features on page 85](#).
- Tuning global and local options on the distributed agents in [Tuning global and local options in the distributed network on page 98](#).
- Selecting sets of end-to-end server messages for the logs in [Selecting end-to-end server messages on page 101](#).

## Defining centralized and non-centralized scripts

A job can use two kinds of scripts, centralized or non-centralized. A centralized script is a script that resides in IBM Z Workload Scheduler and that is downloaded to the fault-tolerant workstation every time the job is submitted.

A non-centralized script is a script that resides on the fault-tolerant workstation.

Centralized scripts are defined in both the controller job library (`EQQJBLIB` DD statement, also called `JOBLIB`) and in the controller script library (`EQQSCLIB` DD statement, also called `SCRPTLIB`). Non-centralized scripts are defined only in the `SCRPTLIB`.



**Note:** If you have jobs dispatched to Tivoli® Dynamic Workload Broker through standard agents, these jobs cannot use centralized scripts.

Because both centralized and non-centralized scripts have definition entries in the `SCRPTLIB`, avoid the risk of overlapping definitions by using different naming conventions to identify these types of scripts.

### Centralized scripts

For a job running on a fault-tolerant workstation to use a centralized script, you must define that script in a job definition member of the `JOBLIB` as you would for a normal job JCL.

In addition, for that same job, you can also use the following statements in a job definition member of the `SCRPTLIB`:

- `VARSUB`
- `JOBUSR`
- `INTRACTV`
- `RCCONDSUC`

Note that while the `VARSUB` statement of the `JOBLIB` is used when the plan is submitted, the `VARSUB` statement of the `SCRPTLIB` takes effect while the plan is running.

With centralized scripts you can perform variable substitution, automatic recovery, JCL editing, and job setup (as for standard non-fault-tolerant workstation jobs). You can also run the `ARC` command and use the job-submit exit (`EQQUX001`).



## Downloading a centralized script

To download a centralized script, DD statement `EQQTWSCS` must be present in the controller and server started tasks.

During the download operation, the `TWA_home/TWS/centralized` directory is created in the fault-tolerant workstation. The script is downloaded to this directory. If any error occurs during this operation, the controller retries the download every minute for a maximum of ten times. After this number of retries, if the script download still fails, an error with code `OSUF` is issued.

## Rules for creating a centralized script

To create a centralized script follow these rules:

- 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 distributed agent.
- Each line starts between column 1 and column 80.
- Backslash (`\`) at column 80 is the character of continuation.
- Blanks at the end of the line are automatically removed.

You can define a job to use a centralized script when you specify the automatic job options of an operation by setting the centralized script field. For example, if you are using the IBM Z Workload Scheduler panels, you must define the centralized script in the `EQQAMJBP` panel, that represents the automatic job options of an operation in the database.

## Non-centralized scripts

For a job running on a fault-tolerant workstation to use a non-centralized script, you must define the script by specifying the `JOBSCR` or the `JOB CMD` keywords in the `JOBREC` statement of a `SCRPTLIB` member.

You can also use variable substitution and recovery by specifying `VARSUB` and `RECOVERY` statements in a `SCRPTLIB` member.

The variable substitution provided for job definitions stored in a `SCRPTLIB` member of the `EQQSCLIB` data set is different from the variable substitution for job definitions stored in a `JOBLIB` member of the `EQQJBLIB` data set.

With non-centralized scripts you can perform script browse from the Modify Current Plan option, having the following configuration: the `JOBLIB` and the `SCRPTLIB` have to be concatenated in the `EQQJBLIB` ddname. An informational message is always issued when browsing a non-centralized script to remember that if a member with the same name exists also in the `JOBLIB`, then this last JCL is displayed.

## Configuring the SCRPTLIB

Jobs that run on fault-tolerant workstations can use solely a non-centralized script or a combination of centralized and non-centralized scripts (if they are to take advantage of the `SCRPTLIB` statements). These jobs must be associated to members of the `SCRPTLIB` (identified by the `EQQSCLIB` DD statement of the controller and the daily planning JCL). You also define some keywords such as the user name, the return code, and the variable substitution in the `SCRPTLIB`.

In relation with the variable substitution in the `SCRPTLIB`, you can also use IBM Z Workload Scheduler variables, with the exception of the Dynamic-Format variables which require the `SETFORM` directive. No job tailoring keyword or directive is allowed in the `SCRPTLIB`, except for the `VARSUB` keyword.

You must use the `JOBREC` statement in every `SCRIPTLIB` member to specify the script or command to run. The following statements are optional:

**VARSUB**

To use the IBM Z Workload Scheduler automatic substitution of variables when the Symphony file is created or when an operation on a fault-tolerant workstation is added to the current plan dynamically.

**RECOVERY**

To use the IBM Workload Scheduler recovery.

The syntax for centralized scripts is:

```
VARSUB
  TABLES(GLOBAL|tab1,tab2,..|APPL)
  PREFIX('char')
  BACKPREF('char')
  VARFAIL(YES|NO)
  TRUNCATE(YES|NO)
JOBREC
  JOBUSR ('username')
  INTRACTV(YES|NO)
  RCCONDSUC('success condition')
```

The syntax for non-centralized scripts is:

```
VARSUB
  TABLES(GLOBAL|tab1,tab2,..|APPL)
  PREFIX('char')
  BACKPREF('char')
  VARFAIL(YES|NO)
  TRUNCATE(YES|NO)
JOBREC
  JOBSCR|JOBCMD ('task')
  JOBUSR ('username')
  INTRACTV(YES|NO)
  RCCONDSUC('success condition')
RECOVERY
  OPTION(STOP|CONTINUE|RERUN)
  MESSAGE('message')
  JOBCMD|JOBSCR('task')
  JOBUSR ('username')
  JOBWS('wsname')
  INTRACTV(YES|NO)
  RCCONDSUC('success condition')
```

The following is an example of a non-centralized script:

```
VARSUB
  TABLES(APPL)
  PREFIX('&')
  BACKPREF('%')
  VARFAIL(NO)
  TRUNCATE(YES)
JOBREC
  JOBCMD ('ls -l /')
  JOBUSR ('tws')
```

```

INTRACTV(NO)
RCCONDSUC(' (RC=4) OR ((RC>=6) AND (RC<9))')
RECOVERY
OPTION(RERUN)
MESSAGE('MESSAGE')
JOB CMD ('ls /')
JOBUSR ('&USER')
JOBWS ('TWMD')
INTRACTV(YES)
RCCONDSUC(' (RC=3) OR ((RC>=5) AND (RC<10))')

```

If in the IBM Z Workload Scheduler database you define a job with a `SCRPTLIB` member that contains errors, the daily planning batch job sets the status of that job to `Failed` in the Symphony file. This change of status is not shown in the IBM Z Workload Scheduler interface. You can find the messages that explain the error in the log of the daily planning batch job.

If in the current plan you dynamically add a job whose associated `SCRPTLIB` member contains errors, the job is not added. You can find the messages that explain this failure in the log of the controller.

However, if you install PTFs `UK02507` and `UK02508`, you can use the end-to-end `JOBREC` utility that detects errors in your `SCRPTLIB` definitions before the current plan is generated. See [Determining errors in the SCRPTLIB configuration on page 84](#) for details.

## Specifying the statements

Each statement consists of a statement name, keywords, and keyword values, and follows TSO command syntax rules.

When you specify `SCRPTLIB` statements, follow these rules:

- Statement data must be in columns 1 through 72. Information in columns 73 through 80 is ignored.
- A blank serves as the delimiter between two keywords; if you supply more than one delimiter, the extra delimiters are ignored.
- Continuation characters and blanks are not used to define a statement that continues on the next line.
- Values for keywords are contained within parentheses. If a keyword can have multiple values, the list of values must be separated by valid delimiters. Delimiters are not allowed between a keyword and the left parenthesis of the specified value.
- Type `/*` to start a comment and `*/` to end a comment. A comment can span record images in the parameter member and can appear anywhere *except* in the middle of a keyword or a specified value.
- A statement continues until the next statement or until the end of records in the member.
- If the value of a keyword includes spaces, enclose the value within single or double quotation marks as follows:

```

JOB CMD ('ls -la')
JOBSCR ('C:/USERLIB/PROG/XME.EXE')
JOBSCR ("C:/USERLIB/PROG/XME.EXE")
JOBSCR ("C:/USERLIB/PROG/XME.EXE 'THIS IS THE PARAMETER LIST' ")
JOBSCR ('C:/USERLIB/PROG/XME.EXE 'THIS IS THE PARAMETER LIST' ')

```

## VARSUB

### Purpose

This statement defines the variable substitution options. It must always be the first statement in the `SCRIPTLIB`.

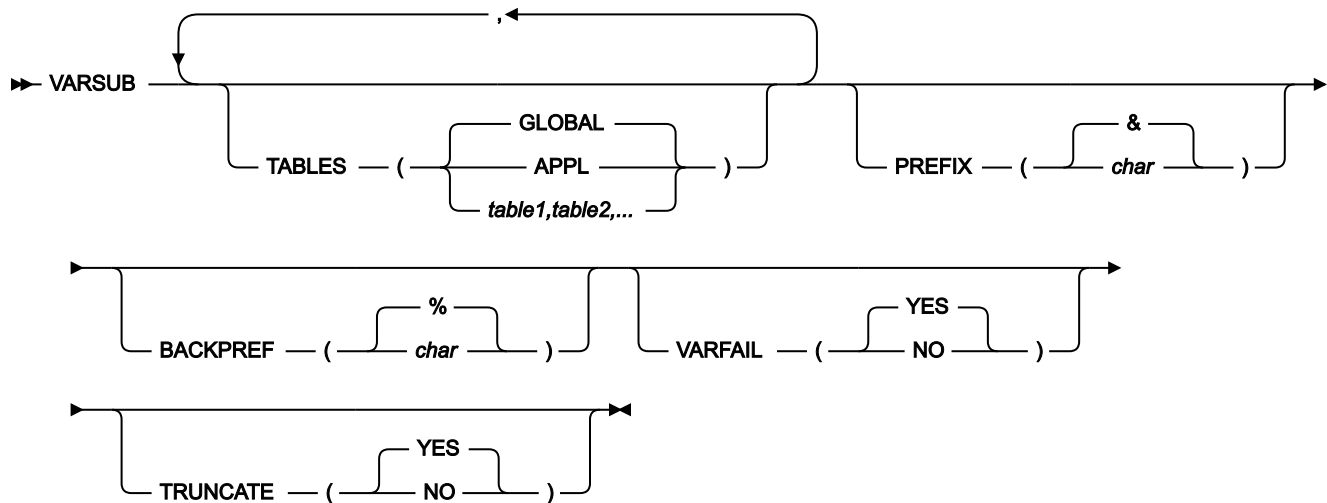
Variable substitution occurs when:

- The Daily Planning process creates the Symphony file.
- The job is added to the plan using the Modify Current Plan dialog.

A variable consists of up to eight alphanumeric characters. Any non-alphanumeric character, except blank, can be used as a symbol to indicate that the characters that follow represent a variable. You can define two kinds of symbols using the `PREFIX` or `BACKPREFIX` keywords in the `VARSUB` statement to define simple and compound variables.

Supplied Variables and User-defined Variables tables are supported. Refer to *IBM Z Workload Scheduler Managing the Workload* for details about associating variable tables with applications.

### Format



### Parameters

#### `TABLES(GLOBAL|APPL|table1,table2,...)`

Identifies the variable tables that must be searched, and the search order. **APPL** indicates the application variable table (see the `VARIABLE TABLE` field in the MCP panel, at Occurrence level). **GLOBAL** indicates the table defined in the `GTABLE` keyword of the `OPCOPTS` controller and `BATCHOPT` batch options.

#### `PREFIX(char&)`

A non-alphanumeric character that precedes a variable. It serves the same purpose as the ampersand (&) character used in variable substitution in z/OS® JCL.

**BACKPREF(char%)**

A non-alphanumeric character that delimits a variable to form simple and compound variables. It serves the same purpose as the percent (%) character used in variable substitution in z/OS® JCL.

**VARFAIL(NO|YES)**

Specifies whether IBM Z Workload Scheduler is to issue an error message when a variable substitution error occurs. If you specify NO, the variable string is left unchanged without any translation.

**TRUNCATE(YES|NO)**

Specifies if variables are to be truncated if they are longer than the allowed length. If you specify NO and the keywords are longer than the allowed length, an error message is issued. The allowed length is the length of the keyword for which you use the variable. For example if you specify a variable of five characters for the `JOBWS` keyword, the variable is truncated to the first four characters.

**Example****Examples**

The following example shows how to use a simple variable. When scheduling the operation, the scheduler automatically sets the simple `VUSER` variable to the value that is specified in the `E2EVARTAB` variable table that is defined in the Application Description file.

```
VARSUB
  TABLES(E2EVARTAB)
  PREFIX('&')
JOBREC
  JOBCMD('dir')
  JOBUSR('&VUSER')
```

In the following example of a compound variable, two substitution steps are performed automatically. The `%OWSID` string is processed first. `OWSID` is an operation-related supplied variable that is automatically set to the workstation ID for the current operation: assuming that `FTW1` is the workstation ID, the first substitution step produces `&FTW1SDIR.\my.cmd` string. If you define the variable `&FTW1SDIR` in the `E2EVARTAB` application description table and set it to the `c:\win32app\maestro` value, the final result is that `c:\win32app\maestro\my.cmd` will be scheduled.

```
VARSUB
  TABLES(E2EVARTAB)
  PREFIX('&')
  BACKPREF('%')
JOBREC
  JOBSCR('&%OWSID.SDIR.\my.cmd')
  JOBUSR('tws')
```

**Restrictions**

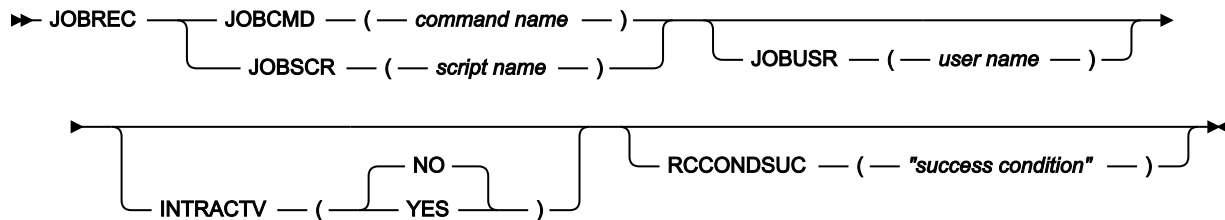
You can have variables that depend on other variables according to the data specified in the dependency value list. Dependent and independent variables must be defined inside the same table and a default value must be assigned to the independent variable.

## JOBREC

### Purpose

This statement defines the fault-tolerant workstation job properties. You must specify `JOBREC` for each member of the `SCRPTLIB`. For each job this statement specifies the script or the command to run and the user that must run the script or command.

### Format



### Parameters

#### **JOBSCR(*script name*)**

Specifies the name of the shell script or executable file to run for the job. The maximum length is 4095 characters. If the script includes more than one word, it must be enclosed within single or double quotation marks. Do not specify this keyword if the job uses a centralized script.

#### **JOBCMD(*command name*)**

Specifies the name of the shell command to run the job. The maximum length is 4095 characters. If the command includes more than one word, it must be enclosed within single or double quotation marks. Do not specify this keyword if the job uses a centralized script.

#### **JOBUSR(*user name*)**

Specifies the name of the user submitting the specified script or command. The maximum length is 47 characters. If you do not specify the user in the `JOBUSR` keyword, the user defined in the `CPUUSER` keyword of the `CPUREC` statement is used. The `CPUREC` statement is the one related to the workstation on which the specified script or command must run. If the user is not specified in the `CPUUSER` keyword, the `tws` user is used.

If the script is centralized, you can also use the job-submit exit (`EQQUX001`) to specify the user name. This user name overrides the value specified in the `JOBUSR` keyword. In turn, the value specified in the `JOBUSR` keyword overrides that specified in the `CPUUSER` keyword of the `CPUREC` statement. If no user name is specified, the `tws` user is used.

If you use this keyword to specify the name of the user who submits the specified script or command on a Windows™ fault-tolerant workstation, you must associate this user name to the Windows™ workstation in the `USRREC` initialization statement.

**INTRACTV(YES|NO)**

Specifies that a Windows™ job runs interactively on the Windows™ desktop. This keyword is used only for jobs running on Windows™ fault-tolerant workstations.

**RCCONDSUC("success condition")**

An expression which determines the return code (RC) required to consider a job as successful. If you do not specify this keyword, a return code of zero corresponds to a successful condition. A return code different from zero corresponds to the job abending.

The *success condition* maximum length is 256 characters and the total length of `JOB CMD` or `JOB SCR` plus the success condition must be 4086 characters. This is because the `TWSRCMAP` string is inserted between the success condition and the script or command name. For example the `dir` command together with the success condition `"rc<4"` is translated into `dir TWSRCMAP: RC<4`.

The *success condition* expression can contain a combination of comparison and Boolean expressions:

**Comparison expression**

Specifies the job return codes. The syntax is:

```
(RC operator operand)
```

**RC**

The RC keyword.

**operator**

Comparison operator. It can have the following values:

**Table 9. Comparison operators**

Example	Operator	Description
RC<a	<	Less than
RC<=a	<=	Less than or equal to
RC>a	>	Greater than
RC>=a	>=	Greater than or equal to
RC=a	=	Equal to
RC<>a	<>	Not equal to

**operand**

An integer between -2147483647 and 2147483647.

For example, you can define a successful job as a job that ends with a return code less than or equal to 3 as follows:

```
RCCONDSUC "(RC <= 3)"
```

**Boolean expression**

Specifies a logical combination of comparison expressions. The syntax is:

```
comparison_expression operator comparison_expression
```

**comparison\_expression**

The expression is evaluated from left to right. You can use parentheses to assign a priority to the expression evaluation.

**operator**

Logical operator. It can have the following values:

**Table 10. Logical operators**

Example	Operator	Result
expr_a and expr_b	And	TRUE if both expr_a and expr_b are TRUE.
expr_a or expr_b	Or	TRUE if either expr_a or expr_b is TRUE.
Not expr_a	Not	TRUE if expr_a is not TRUE.

For example, you can define a successful job as a job that ends with a return code less than or equal to 3 or with a return code not equal to 5, and less than 10 as follows:

```
RCCONDSUC "(RC<=3) OR ((RC<>5) AND (RC<10))"
```

**RECOVERY****Purpose**

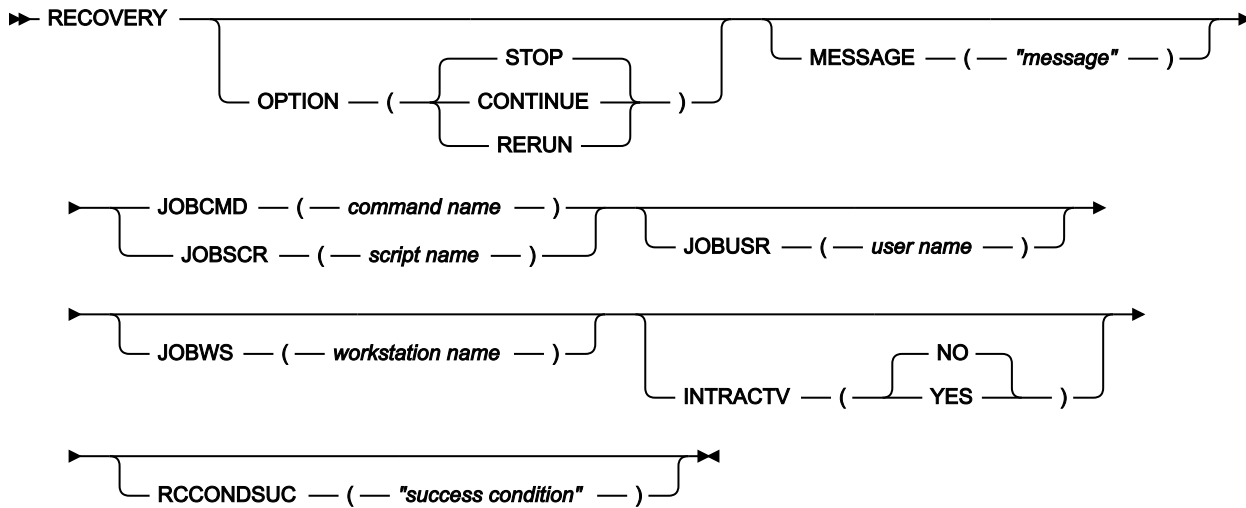
This statement defines the options to run IBM Workload Scheduler recovery for a job whose status is in error, but whose error code is not `FAIL`. To run the recovery you can specify one or both of the following recovery actions:

- A recovery job (`JOBCMD` or `JOBSCR` keywords)
- A recovery prompt (`MESSAGE` keyword)

The recovery actions must be followed by one of the recovery options (the `OPTION` keyword), stop, continue, or rerun. The default is stop with no recovery job and no recovery prompt. For more information about recovery in a distributed network, see *IBM Workload Scheduler: User's Guide and Reference*. The `RECOVERY` statement is ignored if it is used with a job that runs a centralized script.



## Format



## Parameters

### OPTION(STOP|CONTINUE|RERUN)

Specifies the option that IBM Z Workload Scheduler must use when a job abends. You can define a recovery option for every job. You can specify one of the following values:

#### STOP

Do not continue with the next job. The current job remains in error. You cannot specify this option if you use the `MESSAGE` recovery action.

#### CONTINUE

Continue with the next job. The current job status changes to complete in the z/OS® interface.

#### RERUN

Automatically rerun the job (once only). The job status changes to ready, and then to the status of the rerun. Before rerunning the job for a second time, an automatically generated recovery prompt is displayed.

### MESSAGE("message")

Specifies the text of a recovery prompt, enclosed in single or double quotation marks, to be displayed if the job abends. The text can contain up to 64 characters. If the text begins with a colon (:), the prompt is displayed, but no reply is required to continue processing. If the text begins with an exclamation mark (!), the prompt is not displayed, but a reply is required to proceed. You cannot use the recovery prompt if you specify the recovery `STOP` option without using a recovery job.

### JOB CMD(*command name*)

Specifies the name of the shell command to run if the job abends. The maximum length is 4095 characters. If the command includes more than one word, it must be enclosed within single or double quotation marks.

**JOBSCR(*script name*)**

Specifies the name of the shell script or executable file to be run if the job abends. The maximum length is 4095 characters. If the script includes more than one word, it must be enclosed within single or double quotation marks.

**JOBUSR(*user name*)**

Specifies the name of the user submitting the recovery job action. The maximum length is 47 characters. If you do not specify this keyword, the user defined in the `JOBUSR` keyword of the `JOBREC` statement is used. Otherwise the user defined in the `CPUSER` keyword of the `CPUREC` statement is used. The `CPUREC` statement is the one related to the workstation on which the recovery job must run. If the user is not specified in the `CPUSER` keyword, the `tw`s user is used.

If you use this keyword to specify the name of the user that runs the recovery on a Windows™ fault-tolerant workstation, you must associate this user name to the Windows™ workstation in the `USRREC` initialization statement

**JOBWS(*workstation name*)**

Specifies the name of the workstation on which the recovery job or command is submitted. The maximum length is four characters. The workstation must belong to the same domain as the workstation on which the main job runs. If you do not specify this keyword, the workstation name of the main job is used.

**INTRACTV(YES|NO)**

Specifies that the recovery job runs interactively on a Windows™ desktop. This keyword is used only for jobs running on Windows™ fault-tolerant workstations.

**RCCONDSUC("success condition")**

An expression which determines the return code (RC) required to consider a recovery job as successful. If you do not specify this keyword, a return code of zero corresponds to a successful condition. A return code different from zero corresponds to the job abending.

The *success condition* maximum length is 256 characters and the total length of the `JOB CMD` or `JOB SCR` plus the success condition must be 4086 characters. This is because the `TWSRCMAP` string is inserted between the success condition and the script or command name. For example, the `dir` command together with the success condition `"rc4"` is translated into `dir TWSRCMAP: RC4`.

The *success condition* expression can contain a combination of comparison and Boolean expressions:

**Comparison expression**

Specifies the recovery job return codes. The syntax is:

```
(RC operator operand)
```

**RC**

The RC keyword.

**operator**

Comparison operator. It can have the following values:

**Table 11. Comparison operators**

Example	Operator	Description
RC<a	<	Less than
RC<=a	<=	Less than or equal to
RC>a	>	Greater than
RC>=a	>=	Greater than or equal to
RC=a	=	Equal to
RC≠a	≠	Not equal to

**operand**

An integer between -2147483647 and 2147483647.

For example, you can define a successful recovery job as a job that ends with a return code less than or equal to 3 as follows:

```
RCCONDSUC "(RC = 3)"
```

**Boolean expression**

Specifies a logical combination of comparison expressions. The syntax is:

```
comparison_expression operator comparison_expression
```

**comparison\_expression**

The expression is evaluated from left to right. You can use parentheses to assign a priority to the expression evaluation.

**operator**

Logical operator. It can have the following values:

**Table 12. Logical operators**

Example	Operator	Result
expr_a and expr_b	And	TRUE if both expr_a and expr_b are TRUE.
expr_a or expr_b	Or	TRUE if either expr_a or expr_b is TRUE.
Not expr_a	Not	TRUE if expr_a is not TRUE.

For example, you can define a successful recovery job as a job that ends with a return code less than or equal to 3 or with a return code not equal to 5, and less than 10 as follows:

```
RCCONDSUC "(RC=3) OR ((RC5) AND (RC10))"
```

## Determining errors in the SCRPTLIB configuration

`SCRPTLIB` members can be checked for errors prior to generating the current plan. This gives you the possibility to highlight potential problems and to avoid them before they affect your business system. There are two ways to accomplish this task:

- Creating a trial plan from the ISPF dialogs
- Running the `EQQSLCHK` sample JCL produced from `EQQJOBS` when the end-to-end scheduling with fault tolerance capabilities is enabled (see [Table 4: Sample JCL for end-to-end scheduling with fault tolerance capabilities generated by the EQQJOBS8 dialog on page 32](#))



**Note:** This requires that PTFs `UK02507` and `UK02508` are installed.

Creating a trial plan before generating the current plan, exposes errors in the `SCRPTLIB` members. If errors are found, the same messages that are issued for the daily planning job log can help you make corrections and avoid problems when creating, extending, or replanning the current plan.

Trial plans do not support variable substitution in the `SCRPTLIB` members. Therefore, if you specified a recovery option on a fault-tolerant agent like the following:

```
RECOVERY OPTION(CONTINUE) JOBUSR(maestro) JOBWS(&OWSID)
```

the error message `EQQ3077E` is issued in the batch `EQQMLOG` and the trial plan fails.

The `EQQSLTOP` batch program contained in the `EQQSLCHK` sample JCL provides a syntactic check of all the script library members in the `EQQSCLIB`. For every member containing an error the parsing process issues a message describing the problem found. It also issues message `EQQ4033I` that identifies the script library where the failing member is defined.

In this way, even if different members with the same name are defined in more than one script library allocated in the `EQQSCLIB` ddname, members in error are located immediately. If all script libraries defined in the `EQQSCLIB` ddname are empty, warning message `EQQ4031W` is issued.

The `EQQSLCHK` sample runs in stand-alone mode, without the need to interact with the current plan database. You can use this sample to parse only a subset of scripts or all the members contained in the allocated script libraries.

You can customize the `SYSIN` statement of the `EQQSLTOP` program in one of the two following ways:

- To parse and check all the members of the script libraries allocated in the `EQQSCLIB` ddname:

```
SYSIN DD*
PARSE
```

- To parse and check only the stated members:

```

SYSIN DD*
PARSE MEMBER (membnme)
PARSE MEMBER (...)

```

where:

- *membnme* is the name of the script library member to check.
- Each member name can be at most eight characters long.
- Only one parse statement is allowed in every row.
- Blank rows are allowed in between. All the rows are processed until the end of the SYSIN ddname.

The following is an example of `EQQSLTOP`:

```

-----
File parse program
-----
PARSE EXEC PGM=EQQBATCH,PARM='EQQSLTOP',REGION=4M
STEPLIB DD DISP=SHR,DSN=OPCDEV.TEST.APFLIB
EQQLIB DD DISP=SHR,DSN=OPCDEV.TEST.MSGS
EQQPARM DD DISP=SHR,DSN=EID.EIDA.PARM(EIDA)
EQQSCLIB DD DISP=SHR,DSN=EID.EIDA.SCRPTLIB
SYSDUMP DD DISP=MOD,DSN=EID.EIDA.SYSDUMPB
EQQDUMP DD SYSOUT=*
EQQMLLOG DD SYSOUT=*
EQQDMSG DD SYSOUT=*
SYSIN DD *
PARSE MEMBER (membnme) (commented out)
PARSE MEMBER (...)

```

## Setting the security features

The security features of IBM Workload Scheduler on distributed networks include:

- Strong authentication and encryption (SSL)
- FIPS compliance over SSL connection
- Work across firewalls

These features are also true when scheduling end-to-end from IBM Z Workload Scheduler.

## Setting strong authentication and encryption

You can enable the IBM Workload Scheduler processes that run as USS processes in the IBM Z Workload Scheduler address space to establish SSL authentication between an IBM Z Workload Scheduler master and the underlying IBM Workload Scheduler workstations.

The authentication mechanism of IBM Workload Scheduler is based on the OpenSSL toolkit, while IBM Z Workload Scheduler uses the SSL services of z/OS®. This section describes how to configure IBM Z Workload Scheduler for SSL authentication when functioning as a master in fault-tolerant end-to-end scheduling. To find out how to enable SSL in the IBM Workload Scheduler domain managers, see *IBM Workload Scheduler Planning and Installation*.

For further details, see also *z/OS® Cryptographic Services Secure Socket Layer Programming*.

## Key SSL concepts

To authenticate a peer identity, the SSL protocol uses X.509 certificates called digital certificates. Digital certificates are electronic ID cards that are issued by trusted parties and enable a user to verify both the sender and the recipient of the certificate through the use of public-key cryptography.

Public-key cryptography uses two different cryptographic keys: a private key and a public key. Public-key cryptography is also known as asymmetric cryptography, because you can encrypt information with one key and decrypt it with the complement key from a given public-private key pair. Public-private key pairs are simply long strings of data that act as keys to a user encryption scheme. The user keeps the private key in a secure place (for example, encrypted on a computer hard drive) and provides the public key to anyone with whom the user wants to communicate. The private key is used to digitally sign all secure communications sent from the user while the public key is used by the recipient to verify the sender signature.

Public-key cryptography is built on trust: the recipient of a public key needs to have confidence that the key really belongs to the sender and not to an impostor. Digital certificates provide that confidence. For this reason, the IBM Workload Scheduler workstations that share an SSL session must have locally installed repositories for the X.509 certificates that will be exchanged during the SSL session establishment phase to authenticate the session.

A digital certificate is issued by a trusted authority, also called a certificate authority. A signed digital certificate contains:

- The owner distinguished name
- The owner public key
- The certificate authority (issuer) distinguished name
- The signature of the certificate authority for these fields

A certificate request that is sent to a certificate authority for signing contains:

- The owner (requester) distinguished name
- The owner public key
- The owner signature for these fields

A Certificate Authority (such as, for example, VeriSign or Thawte) is trusted by a client (or a server) application when their root certificate (that is the certificate that contains the Certification Authority signature) is listed in the client (server) trusted Certification Authority lists. The way an application creates its trusted Certification Authority list depends on the SSL implementation. Using OpenSSL, for example, the Trusted Certification Authority List is simply a file containing the concatenated Certificates of all the Certification Authorities that should be trusted. Using the z/OS® Cryptographic Services System SSL, the Trusted Certification Authority List is a proprietary database containing the certificates of the Trusted Certification Authority. The certificate authority verifies this signature with the public key in the digital certificate to ensure that the certificate request was not modified while being sent between the requester and the Certification Authority and that the requester is in possession of the private key that matches the public key in the certificate request.

The Certification Authority is also responsible for some level of identification verification. This can range from very little proof to absolute assurance of the owner identity. A particular kind of certificate is the self-signed digital certificate. It contains:

- The owner distinguished name
- The owner public key
- The owner signature for these fields

A root Certification Authority digital certificate is an example of a self-signed digital certificate. Users can also create their own self-signed digital certificates for testing purposes.

The following example describes in a simplified way how digital certificates are used in establishing an SSL session. In this scenario, Appl1 is a client process that opens an SSL connection with the server application Appl2:

1. Client Appl1 asks to open an SSL session with server Appl2
2. Appl2 starts the SSL handshake protocol. It encrypts the information using its private key and sends its certificate with the matching public key to Appl1
3. Appl1 receives the certificate from Appl2 and verifies that it is signed by a trusted certification authority. If it is Appl1 can optionally extract some information (such as the distinguished name) stored in the certificate and perform additional authentication checks on Appl2
4. At this point, the server process has been authenticated, and the client process starts its part of the authentication process; that is, Appl1 encrypts the information using its private key and sends the certificate with its public key to Appl2
5. Appl2 receives the certificate from Appl1 and verifies that it is signed by a trusted certification authority
6. If the certificate is signed by a trusted Certification Authority, Appl2 can optionally extract some information (such as the distinguished name) stored in the certificate and perform additional authentication checks on Appl1.

## SSL connection for communication across the network

IBM Workload Scheduler provides a secure, authenticated, and encrypted connection mechanism for communication across the network topology. This mechanism is based on the Secure Sockets Layer (SSL) protocol and uses the OpenSSL Toolkit, which is automatically installed with IBM Workload Scheduler.

The SSL protocol is based on a private and public key methodology. SSL provides the following authentication methods:

### **CA trusting only**

Two workstations trust each other if each receives from the other a certificate that is signed or is trusted. That is, if the CA certificate is in the list of trusted CAs on each workstation. With this authentication level, a workstation does not perform any additional checks on certificate content, such as the distinguished name. Any signed or trusted certificate can be used to establish an SSL session. For a definition of the caonly option, see [Configuring SSL local options on page 94](#).

### **Check if the distinguished name matches a defined string**

Two workstations trust each other if, after receiving a trusted or signed certificate, each performs a further check by extracting the distinguished name from the certificate and comparing it with a string that was defined in its local options file. For a definition of the string option, see [Configuring SSL local options on page 94](#).

### Check if the distinguished name matches the workstation name

Two workstations trust each other if, after receiving a signed or trusted certificate, each performs a further check by extracting the distinguished name from the certificate and comparing it with the name of the workstation that sent the certificate. For a definition of the `cpu` option, see [Configuring SSL local options on page 94](#).

To provide SSL security for a domain manager attached to z/OS® in a fault-tolerant end-to-end connection, configure the OS/390® Cryptographic Services System SSL in the IBM Workload Scheduler code that runs in the OS/390® USS UNIX™ shell in the IBM Z Workload Scheduler server address space.

When configuring SSL you can:

#### Use the same certificate for the entire network

If the workstations are configured with CA trusting only, they accept connections with any other workstation that sends a signed or trusted certificate. To enforce the authentication you define a name or a list of names that must match the contents of the certificate distinguished name (DN) field in the `localopts` file of each workstation.

#### Use a certificate for each domain

Install private keys and signed certificates for each domain in the network. Then, configure each workstation to accept a connection only with partners that have a particular string of the certificate DN field in the `localopts` file of each workstation.

#### Use a certificate for each workstation

Install a different key and a signed certificate on each workstation and add a Trusted CA list containing the CA that signed the certificate. Then, configure each workstation to accept a connection only with partners that have their workstation name specified in the `Symphony` file recorded in the DN field of the certificate.

## Setting up private keys and certificates

### About this task

To use SSL authentication on a workstation, you need to create and install the following:

- The private key and the corresponding certificate that identify the workstation in an SSL session.
- The list of certificate authorities that can be trusted by the workstation.

Use the **openssl** command line utility to:

- Create a file containing pseudo random generated bytes (TWS.rnd). This file is needed on some operating systems for SSL to function correctly.
- Create a private key.
- Save the password you used to create the key into a file.
- Create a Certificate Signing Request.
- Send this Certificate Signing Request (CSR) to a Certifying Authority (CA) for signing, or:



- Create your own Certificate Authority (CA)
- Create a self-signed CA Certificate (X.509 structure) with the RSA key of your own CA
- Use your own Certificate Authority (CA) to sign and create real certificates

These actions will produce the following files that you will install on the workstation(s):

- A private key file (for example, TWS.key). This file should be protected, so that it is not stolen to use the workstation's identity. You should save it in a directory that allows read access to the TWS user of the workstation, for example `TWA_home/TWS/ssl/TWS.key`.
- The corresponding certificate file (for example, TWS.crt). You should save it in a directory that allows read access to the TWS user of the workstation, such as `TWA_home/TWS/ssl/TWS.crt`.
- A file containing a pseudo-random generated sequence of bytes. You can save it in any directory that allows read access to the TWS user of the workstation, such as `TWA_home/TWS/ssl/TWS.rnd`.

In addition, you should create the following:

- A file containing the password used to encrypt the private key. You should save it in a directory that allows read access to the TWS user of the workstation, such as `TWA_home/TWS/ssl/TWS.sth`.
- The certificate chain file. It contains the concatenation of the PEM-encoded certificates of certification authorities which form the certificate chain of the workstation's certificate. This starts with the issuing CA certificate of the workstation's certificate and can range up to the root CA certificate. Such a file is simply the concatenation of the various PEM-encoded CA certificate files, usually in certificate chain order.
- The trusted CAs file. It contains the trusted CA certificates to use during authentication. The CAs in this file are also used to build the list of acceptable client CAs passed to the client when the server side of the connection requests a client certificate. This file is simply the concatenation of the various PEM-encoded CA certificate files, in order of preference.

## Creating private keys and certificates

### About this task

The following steps explain how to create one key and one certificate. You can decide whether to use one key and certificate pair for the entire network, one for each domain, or one for each workstation. The steps below assume that you will be creating a key and certificate pair for each workstation and thus the name of the output files created during the process has been generalized to *workstationname*.

On each workstation, perform the following steps to create a private key and a certificate:

1. Enter the following command from the SSL directory to initialize the pseudo random number generator, otherwise subsequent commands could not work.
  - On Windows™ operating systems:

```
$ openssl rand -out workstationname.rnd -rand ./openssl.exe 8192
```

- On UNIX™ and Linux™ operating systems :

```
$ openssl rand -out workstationname.rnd -rand ./openssl 8192
```

2. Enter the following command to create the private key (this example shows triple-DES encryption):

```
$ openssl genrsa -des3 -out workstationname.key 2048
```

Then, save the password that was requested to encrypt the key in a file named *workstationname.pwd*.



**Note:** Verify that file *workstationname.pwd* contains just the characters in the password. For instance, if you specified the word *maestro* as the password, your *workstationname.pwd* file should not contain any CR or LF characters at the end (it should be 7 bytes long).

3. Enter the following command to save your password, encoding it in base64 into the appropriate stash file:

```
$ openssl base64 -in workstationname.pwd -out workstationname.sth
```

You can then delete file *workstationname.pwd*.

4. Enter the following command to create a certificate signing request (CSR):

```
$ openssl req -new -key workstationname.key -out workstationname.csr
-config ./openssl.cnf
```

Some values-such as company name, personal name, and more- will be requested at screen. For future compatibility, you can specify the workstation name as the distinguished name.

5. Send the *workstationname.csr* file to your CA in order to get the matching certificate for this private key.

Using its private key (TWSca.key) and certificate (TWSca.crt), the CA will sign the CSR (*workstationname.csr*) and create a signed certificate (*workstationname.crt*) with the following command:

```
$ openssl x509 -req -CA TWSca.crt -CAkey TWSca.key -days 365
-in workstationname.csr -out workstationname.crt -CAcreateserial
```

6. Distribute to the workstation the new certificate *workstationname.crt* and the public CA certificate TWSca.crt.

Table 13: Files for local options on page 90 summarizes which of the files created during the process have to be set as values for the workstation's local options.

**Table 13. Files for local options**

Local option	File
SSL key	<i>workstationname.key</i>
SSL certificate	<i>workstationname.crt</i>
SSL key pwd	<i>workstationname.sth</i>
SSL ca certificate	TWSca.crt
SSL random seed	<i>workstationname.rnd</i>

## Configuring SSL attributes

### About this task

To update the workstation definition in the database, use the composer command line or the Dynamic Workload Console. For more detailed information, see the *IBM Workload Scheduler: User's Guide and Reference* or *Dynamic Workload Console User's Guide*.

Configure the following attributes:

#### **secureaddr**

Defines the port used to listen for incoming SSL connections. This value must match the one defined in the **nm SSL port** local option of the workstation. It must be different from the **nm port** local option that defines the port used for normal communications. If **securitylevel** is specified but this attribute is missing, 31113 is used as the default value.

#### **securitylevel**

Specifies the type of SSL authentication for the workstation. It must have one of the following values:

##### **enabled**

The workstation uses SSL authentication only if its domain manager workstation or another fault-tolerant agent below it in the domain hierarchy requires it.

##### **on**

The workstation uses SSL authentication when it connects with its domain manager. The domain manager uses SSL authentication when it connects to its parent domain manager. The fault-tolerant agent refuses any incoming connection from its domain manager if it is not an SSL connection.

##### **force**

The workstation uses SSL authentication for all of its connections and accepts connections from both parent and subordinate domain managers. It refuses any incoming connection if it is not an SSL connection.

If this attribute is omitted, the workstation is not configured for SSL connections. In this case, any value for **secureaddr** will be ignored. You should also set the **nm ssl port** local option to 0 to be sure that this port is not opened by netman. [Table 14: Type of communication depending on the securitylevel value on page 91](#) describes the type of communication used for each type of **securitylevel** setting.

**Table 14. Type of communication depending on the securitylevel value**

Fault-tolerant agent (domain manager)	Domain manager (parent domain manager)	Connection type
-	-	TCP/IP
Enabled	-	TCP/IP

**Table 14. Type of communication depending on the securitylevel value (continued)**

Fault-tolerant agent (domain manager)	Domain manager (parent domain manager)	Connection type
On	-	No connection
Force	-	No connection
-	On	TCP/IP
Enabled	On	TCP/IP
On	On	SSL
Force	On	SSL
-	Enabled	TCP/IP
Enabled	Enabled	TCP/IP
On	Enabled	SSL
Force	Enabled	SSL
-	Force	No connection
Enabled	Force	SSL
On	Force	SSL
Force	Force	SSL

The following example shows a workstation definition that includes the security attributes:

```

cpuname ENNETI3
os WNT
node apollo
tcpaddr 30112
secureaddr 32222
for maestro
autolink off
fullstatus on
securitylevel on
end

```

## Configuring the SSL connection protocol for the network

### About this task

To configure SSL for your network, perform the following steps:

1. Create an SSL directory under the `TWA_home` directory. By default, the path `TWA_home\TWS\ssl` is registered in the `localopts` file. If you create a directory with a name different from `ssl` in the `TWA_home` directory, then update the `localopts` file accordingly.
2. Copy `openssl.cnf` and `openssl.exe` to the SSL directory
3. Create as many private keys, certificates, and Trusted CA lists as you plan to use in your network.
4. For each workstation that will use SSL authentication:
  - Update its definition in the IBM Workload Scheduler database with the SSL attributes.
  - Add the SSL local options in the `localopts` file.

Although you are not required to follow a particular sequence, these tasks must all be completed to activate SSL support.

In IBM Workload Scheduler, SSL support is available for the fault-tolerant agents only (including the master and the domain managers), but not for the extended agents. If you want to use SSL authentication for a workstation that runs an extended agent, you must specify this parameter in the definition of the host workstation of the extended agent.

## Enabling SSL authentication on the IBM Z Workload Scheduler server

To enable SSL authentication for your end-to-end with fault-tolerance capabilities network, you must perform the following actions:

- Create as many private keys, certificates, and trusted certification authority chains as you plan to use in your network. See [Creating private keys, certificates, and a Trusted Certification Authority Chain on page 93](#).
- Configure the IBM Z Workload Scheduler master (`OPCMaster`) by:
  - Updating the SSL attributes of the IBM Z Workload Scheduler master using the `TOPOLOGY` statement. See [Configuring the IBM Z Workload Scheduler SSL attributes on page 94](#).
  - Adding the SSL local options in the `localopts` file. See [Configuring SSL local options on page 94](#).
- Configure the SSL attributes of the IBM Workload Scheduler workstations using the `CPUREC` on [page 42](#) statement. See [Configuring the SSL attributes for the IBM Workload Scheduler workstations on page 95](#).

Although you are not required to follow a particular sequence, these tasks must all be completed to enable SSL authentication.

## Creating private keys, certificates, and a Trusted Certification Authority Chain

### About this task

Use the **GskKyman** command line utility of *z/OS® Cryptographic Services System SSL* to accomplish the following tasks:

1. Create the keystore database on the IBM Z Workload Scheduler master. This database contains the private key and trusted certificates. Because this keystore database contains a private key, it should be protected. It can be saved under any directory that allows read access to the `tws` user such as `WRKDIR/ssl/TWS.kdb`, where `WRKDIR` is the directory specified in the `WRKDIR` keyword in the `TOPOLOGY` statement
2. Store the Keystore database password in a file. This password will be encrypted and saved in a file that should be stored in any directory that allows read access to the `tws` user, such as `WRKDIR/ssl/TWS.sth`

After these steps, the administrator can start to:

- Create public-private key pairs
- Create certificate requests
- Store signed certificates in the database (the *WRKDIR/Security* directory)
- Create self-signed certificates
- Add or remove Certification Authority from the trust list

For details, see *z/OS® Cryptographic Services Secure Socket Layer Programming, SC24-5901*.

## Configuring the IBM Z Workload Scheduler SSL attributes

### About this task

To configure IBM Z Workload Scheduler to establish SSL connections with the IBM Workload Scheduler domain managers, you must update the `SSLLEVEL` and `SSLPORT` definitions for the IBM Z Workload Scheduler master (`OPCMaster`). You must define these options using the [TOPOLOGY on page 51](#) statement. For example:

```
TOPOLOGY
...
...
SSLLEVEL(ENABLED)
SSLPORT(31113)
...
```

## Configuring SSL local options

### About this task

To set the SSL local options you must edit the *WRKDIR/localopts* file of the IBM Z Workload Scheduler master by removing the # sign in column 1 and changing the value of the corresponding SSL option. This file is also present on the IBM Workload Scheduler workstations and must be customized as follows:

#### SSL key store

The filename of the GSK database containing keys and certificates. The default value is *WRKDIR/ssl/TWS.kdb*.

The GSK database replaces the following information specified in the distributed localopts file:

- SSL Certification Authority certificate
- SSL certificate
- SSL certificate chain
- SSL random seed
- SSL key

#### SSL key store pwd

The name of the file containing the key password. The default is *WRKDIR/ssl/TWS.sth*. It replaces the **SSL key pwd** option of the distributed localopts file.

**SSL auth mode**

The kind of checks that IBM Z Workload Scheduler performs to verify the certificate validity. You can specify one of the following values:

**caonly**

IBM Z Workload Scheduler checks the certificate validity by verifying that a recognized Certification Authority has issued the peer certificate. Information contained in the certificate is not checked. If you do not specify the SSL auth mode keyword or you define a non-permitted value, the caonly value is used.

**string**

IBM Z Workload Scheduler checks the certificate validity as described in the caonly option. It also verifies that the Common Name (CN) of the Certificate Subject matches the string specified in the **SSL auth string** option.

**cpu**

IBM Z Workload Scheduler checks the certificate validity as described in the caonly option. It also verifies that the Common Name (CN) of the Certificate Subject matches the name of the CPU that requested the service.

**SSL auth string**

A string (1 to 64 characters in length) used to verify the certificate validity when you specify string as the SSL auth mode value. If the SSL auth string option is required and it is not specified, *tws* is used as the default value.



**Note:** The following parameters are ignored:

- **nm port** and **nm ssl port** because they are replaced by **SSLPORT** in the [TOPOLOGY on page 51](#) statement.
- **ssl encryption cipher** because it is replaced by the ciphers that the workstation operating system supports during an SSL connection.

## Configuring the SSL attributes for the IBM Workload Scheduler workstations

**About this task**

To configure IBM Workload Scheduler workstations to establish SSL connections with the IBM Z Workload Scheduler master (**OPCMaster**), you must update the **SSLLEVEL** and **SSLPORT** definitions for the IBM Workload Scheduler workstations. You must define these options using the **CPUREC** statement. Remember that the **SSLPORT** indicated in **CPUREC** (not the one in **TOPOLOGY**) for the workstation must match exactly the port number indicated in the local configuration on the workstation. For more details, see [CPUREC on page 42](#).

For detailed information about the additional SSL customization activities needed on the workstations, see also *IBM Workload Scheduler: Planning and Installation*.

## Configuring TLS to connect with the IBM Z Workload Scheduler server

### About this task

To configure the TLS connection between the IBM Z Workload Scheduler server and fault-tolerant agent perform the following steps.



**Note:** In the following procedure, TLS v1.2 is used as an example. If you want to set the TLS V1.1 protocol, replace the values set for TLS v1.2 with TLS v1.1.

1. Specify the following statements in the server started task:

```
PARM='ENVAR("_CEE_ENVFILE:DD=STDENV")'
```

Insert this statement at the top of the started-task JCL. It is used to export the environment variable to the Language Environment.

#### //STDENV DD card

Add this DD card to the server started-task JCL to point to a PDS member (for example, a member of the PARMLIB) where you specify the values for the environment variable that you need. For example,

```
//STDENV DD DISP=SHR,DSN=TWS.SUBSYSN.PARM(ENVVAR)
```

In the PDS member (`ENVVAR` in the previous example), define the following values:

- GSK\_PROTOCOL\_SSLV2=OFF
- GSK\_PROTOCOL\_SSLV3=OFF
- GSK\_PROTOCOL\_TL SV1=OFF
- GSK\_PROTOCOL\_TL SV1\_1=OFF
- GSK\_PROTOCOL\_TL SV1\_2=ON

2. Edit the `localopts` file of the fault-tolerant agent to set the following parameters:

```
SSL Encryption Cipher = TLSv1.2
ssl tls12 cipher = HIGH
```

3. To make the changes effective, stop and restart the fault-tolerant agent and renew the Symphony file on the Z controller.

## Enabling FIPS compliance over IBM Z Workload Scheduler 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 Z Workload Scheduler uses the System SSL configuration. To run IBM Z Workload Scheduler in "FIPS mode", you must enable FIPS compliance over System SSL connections.

To enable SSL authentication for your end-to-end with fault-tolerance capabilities network, you must 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 [TOPOLOGY on page 51](#) statement.
3. On the distributed agent, ensure that:
  - SSL is configured, as described in the section about using FIPS certificates in the *IBM Workload Scheduler: Administration Guide*.
  - FIPS-compliance is enabled, as described in the section about setting localopts parameters for FIPS in the *IBM Workload Scheduler: Administration Guide*.



**Note:** IBM Z Workload Scheduler relies upon System SSL to automatically enable the SSL V2, SSL V3, TLSV1, TLSV1\_1, or TLSV1\_2 protocol. Alternatively, you can export the related environment variable (GSK\_PROTOCOL\_SSLV2, GSK\_PROTOCOL\_SSLV3, GSK\_PROTOCOL\_TLSV1, GSK\_PROTOCOL\_TLSV1\_1, or GSK\_PROTOCOL\_TLSV1\_2 respectively) by specifying the following statements in the server started task:

```
PARM='ENVAR("_CEE_ENVFILE:DD=STDENV")'
```

For details, see [Configuring TLS to connect with the IBM Z Workload Scheduler server on page 96](#)

```
//STDENV DD card
```

For details, see [Configuring TLS to connect with the IBM Z Workload Scheduler server on page 96](#)

Consider that with APAR OA46489, only the TLSV1.0 protocol is enabled by default. The SSL V2 and SSL V3 protocols are no longer enabled by default.

If you enable FIPS, the STDENV DD card settings are ignored.

## Setting for work across firewalls

Prior to IBM Z Workload Scheduler version 8.2, running the commands to start or stop a workstation or to get the standard list, required opening a direct TCP/IP connection between the originator and the destination nodes. In a firewall environment, this forces users to break the firewall to open a direct communication path between the master and each fault-tolerant agent in the network.

The `FIREWALL` configurable attribute of the `CPUREC` statement can be configured to send commands following the domain hierarchy, instead of making the master or the domain manager open a direct connection.

In the design phase of an IBM Workload Scheduler network, the administrator must know where the firewalls are positioned in the network, which fault-tolerant agents and which domain managers belong to a particular firewall, and what are the entry points into the firewalls. When this has been clearly understood, the administrator should define the `FIREWALL` option for all

workstations whose link with the corresponding domain manager is across a firewall. This keyword corresponds to *behind firewall* in the workstation's definition in the IBM Workload Scheduler database.

When the `FIREWALL` option is set to YES for a workstation by using the `CPUREC` statement, it means that a firewall exists between that particular workstation and its domain manager, and that the link between the domain manager and the workstation (which can itself be another domain manager) is the only allowed link between the respective domains. Also, for all the workstations having this option set to YES, the commands to start (*start wkstation*) or stop (*stop wkstation*) the workstation or to get the standard list (*showjobs*) and the download of centralized scripts, follow the domain hierarchy instead of opening a direct connection between the master (or domain manager) and the workstation. This makes a significant improvement in security and performance. The default value for `FIREWALL` is NO, meaning that there is no firewall boundary between the workstation and its domain manager.

To specify that an extended agent is behind a firewall, set the `FIREWALL` keyword for the host workstation. The host workstation is the IBM Workload Scheduler workstation with which the extended agent communicates and where its access method resides.

## Tuning global and local options in the distributed network

As explained in [Step 4. Creating and customizing the work directory on page 35](#), when you run the EQQPCS05 sample, global and local options files are created in the end-to-end work directory on USS. Additionally, each distributed agent uses also its own `globalopts` and `localopts` files. The following tuning tips apply to some of the options defined locally on each agent.

### Global options

If you plan to use the IBM Workload Scheduler time zone feature, ensure that

```
Timezone enable=yes
```

is true on every domain manager and agent of the end-to-end with fault-tolerance capabilities configuration.


### Local options

Setting the following local options on every distributed agent with the values shown helps improve scheduling performance.

**Table 15. Recommended localopts settings on distributed agents**

Local option	Value	Notes
bm look	9	This value must be lower than <code>bm read</code> .
mm cache enable	yes	
mm cache mailbox	yes	

**Table 15. Recommended localopts settings on distributed agents (continued)**

Local option	Value	Notes
mm cache size	<ul style="list-style-type: none"> <li>• 512 bytes for large workloads (250000 jobs)</li> <li>• 32 bytes for small and medium workloads (10000-50000 jobs)</li> </ul>	Keep the value low if the computer hosting the agent runs on low memory. Unnecessarily large values would reduce the available memory that could be allocated to other processes or applications.
mm start tomserver	yes	<p>If you set the <code>TOPOLOGY POSTPONE</code> keyword to <code>yes</code> so that the end-to-end with fault-tolerance capabilities network is stopped gradually while the Symphony file is being distributed, you must also set this option to <code>yes</code> on the domain managers.</p> <p>The option starts an additional mailman server named <code>server@</code> on the domain manager with the task to send its communication load upstream to its parent domain via the <code>tomaster</code> pobox.</p> <p> <b>Note:</b> When <code>server@</code> gets no response from another workstation, it does not unlink from it, but tries to link every 60 seconds (the <code>server@</code> retry link value is always set to 60 seconds).</p>
wr enable compression	yes	Compress Symphony/Sinfonia if it is 4 MB or larger.
sync level	low	

## Tuning network timeouts

Network issues that make the connection to specific agents too slow can impact the performance of overall scheduling. IBM Workload Scheduler provides network timeout settings that you can tune to avoid problems with specific agents that affect the entire system.

When you run the EQQPCS05 sample, global and local options files are created in the end-to-end work directory on USS. For more information, see [Step 4. Creating and customizing the work directory on page 35](#). You can add the following optional

parameters to the localopts file to modify the default timeouts used by IBM Workload Scheduler, tuning the values to find the best settings for your environment.

Lower timeout values might cause the connection to fail too often due to slow systems or network that prevents scheduling on slower nodes. Higher values might cause the slowness of a single system that compromises the performance of the entire scheduling environment.

Setting the following local options on every distributed agent with the suggested values can help improve scheduling performance.

**Table 16. Network timeout settings in localopts**

Local option	Default value	Description
tcp connect timeout	15	This is the timeout in seconds used while opening a connection to an agent or domain manager (TCP/IP connect call). The actual time spent connecting to each agent can be a multiple of this value, since IBM Workload Scheduler can make multiple connection attempts for the same agent depending on the TCP/IP configuration.
tcp timeout	60	<p>This is the time in seconds that the caller (mailman or translator) waits for the invoked service on the agent to return a result.</p> <p>The default value, if not specified, is 60. The default value in the LOCALOPTS file is 300. This is the suggested value unless specific problems are found.</p> <p>This is not used during some long operations, for example, when mailman is downloading the symphony on an agent or domain manager.</p> <p>If these parameters are not present in the LOCALOPTS, they can be added manually as in this example:</p> <pre data-bbox="950 1581 1458 1644">TCP TIMEOUT          = 300 TCP CONNECT TIMEOUT = 15</pre>

If these parameters are not present in the localopts, they can be added manually as in this example:

```
tcp timeout          = 60
tcp connect timeout = 15
```

Restart the end-to-end server to assure that all the processes work with the new settings.

## Selecting end-to-end server messages

The output devices for the end-to-end server messages are the following:

- The server `MLOG`; this file records the EQQ-prefixed messages issued by IBM Z Workload Scheduler and most of the EQQPT-prefixed messages issuing from the end-to-end with fault-tolerance capabilities environment.
- The `stdlist` directory in the `workdir` (`TWSMERGE.log`, `E2EMERGE.log`, and `NETMAN.log`); these files record all the EQQPT-prefixed messages issuing from the end-to-end with fault-tolerance capabilities environment and all the messages issuing from the IBM Workload Scheduler processes (prefixed by AWS).

You can edit the `TWSCCLog.properties` file located in the end-to-end work directory in UNIX™ Systems Services to specify that some or all of these messages are re-directed to either or both the `MLOG` and the system output console (`SYSLOG`). To use this functionality, you must apply the fix for APAR PK11341 on IBM Z Workload Scheduler.

To specify the messages, you can modify the following `TWSCCLog.properties` keywords:

### **console.msgIdFilter.msgIds**

Where you list the messages that are to be sent to the `syslog` upon issue.

If you leave blank, it implies no messages are to be sent to the `syslog`.

### **tws.console.msgIdFilter.msgIds**

Where you list the messages that must be prevented from being sent to the `syslog` upon issue. This keyword can be useful when you use the wildcard character in `console.msgIdFilter.msgIds`.

If you leave blank, it implies no messages are to be prevented from being sent to the `syslog`.

### **ServerMlog.msgIdFilter.msgIds**

Where you list the messages that are to be sent to the end-to-end server `mlog` upon issue.

If you leave blank, it implies no messages are to be sent to the `mlog`.

### **tws.ServerMlog.msgIdFilter.msgIds**

Where you list the messages that are not to be sent to the end-to-end server `mlog` upon issue. This keyword can be useful when you use the wildcard character in `ServerMlog.msgIdFilter.msgIds`.

If you leave blank, it implies no messages are to be prevented from being sent to the `mlog`.

The changes made to `TWSCCLog.properties` become active when the server task is restarted.

## Rules for customizing the CCLog properties file

Follow these rules when you customize any of the four keywords introduced above:

- Field names are case sensitive.
- Asterisks (\*) can be used as wildcard characters to specify groups of messages. For example, `EQQPT*I` specifies all the EQQPT information messages. Alternatively, you can specify messages individually by writing their complete IDs (for example, `EQQPT20I EQQPT21I AWSEDW075I`).

You can use the asterisk wildcard alone. For example, the statement:

```
console.msgIdFilter.msgIds=*
```

causes all the EQQPT and AWS messages to be sent to the `syslog`.

- When selecting a subset of AWS messages based on severity, use the following formats:

```
AWS information messages: specify AWS*I AWS*I*I
AWS warning messages    : specify AWS*W AWS*W*W
AWS error messages      : specify AWS*E AWS*E*E
```

- With the IBM-037 codepage, you can insert the carriage return character, backslash (\), hexadecimal value `x'E0'`, between message IDs to make long message lines more readable. Leave a blank between the last character in the string and the backslash. For example:

```
console.msgIdFilter.msgIds= AWSEDW059E AWSEDW063E EQQPT20I \           AWSEDW075I
EQQPT21I EQQPT23I
```

- Messages referenced in the text of other messages cannot be traced separately. These messages can be traced by filtering on the main message. For example, message `AWSDCJ202W` contained in the following message:

```
AWSBVC001E Error Batchman abended, Status: AWSDCJ202W Terminated by SIGKIL
```

can be traced only by filtering on message `AWSBVC001E`.

## Sample TWSCCLog.properties customization

The following sample shows the portion of a CCLog properties file that was customized to enable:

- Messages `EQQPT20I` and `AWSEDW075I` and all `AWS` error messages, with the exclusion of message `AWSBVC014E`, to be issued in the `syslog`.
- All `AWS` information and warning messages and all `EQQPT` information messages, with the exclusion of messages `EQQPT64I`, `EQQPT65I`, `EQQPT66I`, and `EQQPT69I`, to be issued in the end-to-end server `mlog`.

```
-----
TWS filters properties (SYSLOG)
-----
```

For the E2E environment CUSTOMIZABLE fields:

```
console.msgIdFilter.msgIds: send messages specified to the SYSLOG
tws.console.msgIdFilter.msgIds: avoid sending specified messages to
                               the SYSLOG
-----
```

```
console.msgIdFilter.className=ccg_msgidfilter
console.msgIdFilter.msgIds.mode=PASSTHRU
console.msgIdFilter.msgIds=AWS*E AWS*E*E EQQPT20I AWSEDW075I
console.msgIdFilter.listenerNames=tws.console.msgIdFilter
```

```
tws.console.msgIdFilter.className=ccg_msgidfilter
tws.console.msgIdFilter.mode=BLOCK
```

```

twS.console.msgIdFilter.msgIds=AWSBCW014E
twS.console.msgIdFilter.listenerNames=consoleHandler
-----
TWS filters properties (Server MLOG)
-----
For the E2E environment CUSTOMIZABLE fields:
ServerMlog.msgIdFilter.msgIds: send messages specified to the SERVER MLOG
twS.ServerMlog.msgIdFilter.msgIds :avoid sending specified messages to the
                                SERVER MLOG
-----
ServerMlog.msgIdFilter.className=ccg_msgidfilter
ServerMlog.msgIdFilter.mode=PASSTHRU
ServerMlog.msgIdFilter.msgIds=AWS*W*W AWS*W AWS*I*I AWS*I EQQPT*I
ServerMlog.msgIdFilter.listenerNames=twS.ServerMlog.msgIdFilter

twS.ServerMlog.msgIdFilter.className=ccg_msgidfilter
twS.ServerMlog.msgIdFilter.mode=BLOCK
twS.ServerMlog.msgIdFilter.msgIds=EQQPT64I EQQPT65I EQQPT66I EQQPT69I
twS.ServerMlog.msgIdFilter.listenerNames=twSHnd.ServerMlog
-----

```

# Chapter 4. Migrating

This chapter explains the procedures for:

- [Migrating from a tracker agent to a distributed agent on page 104](#)
- [Migrating from a distributed to an end-to-end with fault tolerance capabilities environment on page 106](#)
- [Migrating from centralized to non-centralized jobs on page 108](#)
- [Changing the topology of the network on page 108](#)

## Migrating from a tracker agent to a distributed agent

### About this task

Follow these steps to migrate a computer from a tracker agent to a distributed agent for end-to-end scheduling with fault tolerance capabilities:

1. Install an IBM Workload Scheduler distributed agent on the computer hosting the tracker agent that you want to migrate.
2. In IBM Z Workload Scheduler define the topology of the fault-tolerant workstations.

Each workstation should be defined twice, once for the tracker agent and once for the distributed agent. In this way, you can run a distributed agent and a tracker agent on the same computer. In this way you can gradually migrate jobs from tracker agents to distributed agents.

3. For migrating the job scripts, you have two alternatives:
  - You can transfer the scripts from the IBM Z Workload Scheduler `JOBLIB` locally to the agents. You do this by using the migration tool described in [Using the tracker jobs migration tool on page 105](#) and `FTP` (file transfer program). If the scripts contain variables, you should replace them with script parameters. If the `JOBLIB` contains only a command or the invocation for a local script, then you can write this directly in the `SCRIPTLIB`.
  - As an alternative to transferring the scripts, you can define the jobs as using a centralized script (you can do this when you specify the job options of an operation). With this solution the script still resides in the `JOBLIB`. Keep in mind, however, that this alternative is not recommended because it implies a loss in fault tolerance and the additional work required to download the script to the agent every time the job is submitted.
4. Create a new script library member for each script to specify the job definition: script or command and user ID.

You do not need to do this if you are using centralized scripts. You can create a job definition member in the `SCRIPTLIB`, but it must have the same name as the member in the `JOBLIB` for the following:

- To specify the user and the interactive parameters for centralized script
  - For return code mapping
5. For each user running jobs on Windows™ agents, define either a `USRREC` statement to provide the Windows™ user and password or define the user and password locally on the Windows™ workstation. If you define the users locally, you



must also set LOCALPSW(YES) in the [TOPOLOGY on page 51](#) statement. You can use both these methods to define different sets of user IDs and related passwords.

6. Modify the workstation name for the operation in the APPLICATION DESCRIPTION panel. Remember to change the `JOBNAME` if the member in the script library has a name different from the member of the `JOBLIB`.

## Using the tracker jobs migration tool

### About this task

This tool helps to create a data set containing all the jobs that need to be transferred (via `FTP`) to a fault-tolerant workstation while migrating from a tracker agent. It helps to determine which jobs should use a centralized script and which should not.

To migrate the old tracker agents to the distributed agents using the migration tool, perform the following steps:

1. To run the migration tool, choose option 5 of the `EQQWSSP` panel in ISPF (fastpath 1.1.5).
2. Change the value of the centralized script flag, based on the results of the tool report.
3. Run Migration Tool as many times as you want (for example you can run it until no warning messages are reported).
4. Transfer the scripts from the `JOBDIS` data set to distributed agents.
5. In the `SCRIPTLIB` create a member for every job of `JOBDIS` and, optionally, for the jobs in `JOBCEM`.
6. Define fault-tolerant workstations.
7. Run Mass Update to change the operations running on a tracker agent workstation to the operations running on a fault-tolerant workstation.

For the workstation you specified in the `ISPF` dialog, the tool produces a report containing a cross-reference between the jobs and the operations in the Application Description database. The report contains information about what has been changed and suggestions about what should remain centralized for a workstation.

Before submitting the job, modify it by adding all the `JOBLIBS` for that workstation, including the `JOBLIB` processed by the job-library-read exit (`EQQUX002`). To have a permanent change, modify the sample migration job skeleton `EQQWMIGZ`.

Check the tool report for the following warnings:

- Operations that are associated with a job library member that uses variables and that have the centralized script option set to `No`. Change the setting to `Yes`.

```
JOBNAME  APPL ID  VALID TO  OpTYPE_OpNUMBER
-----  -
VARJOB   VARAPPL  71/12/31  DIST_001
WARNING: Member VARJOB contains directives (/*%OPC) or variables (& or % or ?).
         Modify the member manually or change the operation type to centralized.
```

- Scripts that do not have variables and are associated with operations that have the centralized script option set to `Yes`. This situation decreases performance significantly. Change the setting to `No`.

```
JOBNAME  APPL ID  VALID TO  OpTYPE_OpNUMBER
-----  -
NOVARJOB NOVARAPPL 71/12/31  CENT_001
WARNING: You should change operation(s) to NON centralized type.
```

Depending on which centralized option you chose for the operations, the tool produces the following files:

**JOBLIB**

Contains all detected jobs for a specific workstation

**JOBCEN**

Contains all jobs that have centralized scripts for a specific workstation

**JOBDIS**

Contains all jobs that do not have centralized scripts for a specific workstation. These jobs need to be transferred to the fault-tolerant workstation.

Before you migrate the tracker agent to a distributed agent, you should use this tool to obtain these files, and then to transfer `JOBDIS` to the new fault-tolerant workstation.

## Migrating backwards

### About this task

Perform the following steps for every distributed agent before you begin a backward migration of the controller to OPC version 2.3 or earlier:

1. Install the tracker agent on the computer.
2. Define a new destination in the `ROUTOPTS` initialization statement of the controller and restart the controller.
3. Make a duplicate of the workstation definition of the computer. Define the new workstation as Computer Automatic instead of Fault Tolerant and specify the destination you defined in step 2 on page 106. In this way, the same computer can be run as a fault-tolerant workstation and as a tracker agent, making migration smoother.
4. For non-centralized scripts, copy the scripts from the fault-tolerant workstation repository to the `JOBLIB`. As an alternative, copy the script to a local directory that can be accessed by the tracker agent and create a `JOBLIB` member to run the script. You can do this by using FTP.
5. Implement the `EQQUX001` sample to run jobs with the correct user ID.
6. Modify the workstation name for the operation in the APPLICATION DESCRIPTION panel. Remember to change the `JOBNAME` if the member in the `JOBLIB` has a name different from the member of the script library.

## Migrating from a distributed to an end-to-end with fault tolerance capabilities environment

Migrating from a totally distributed IBM Workload Scheduler network to an end-to-end with fault tolerance capabilities environment implies transferring the responsibility for database and plan management from the IBM Workload Scheduler master domain manager to the IBM Z Workload Scheduler engine.

The migration process is to change the IBM Workload Scheduler master domain manager to the first-level domain manager and then connect it to the IBM Z Workload Scheduler engine. The IBM Z Workload Scheduler engine becomes the new master domain manager. The result is an end-to-end with fault tolerance capabilities network managed by IBM Z Workload Scheduler.

The migration effort impacts the following:

- Agents
- Job scripts
- Database objects

## Migrating the agents

You must decide at this stage whether you want to completely convert your distributed network into end-to-end with fault tolerance capabilities, or to continue to run it alongside the end-to-end environment. If you decide to replace the distributed network with the fault-tolerant end-to-end environment, on the affected workstations you can leave the current IBM Workload Scheduler engine installation. You just have to edit the following files on each agent:

- The `globalopts` file to replace the name of the master domain manager with `OPCMaster` in the `master` option.
- The `localopts` file to replace the name of the agent with a four-character name (that starts with a letter) in the `thiscpu` option.

If you decide to run the two environments, you must install another instance of IBM Workload Scheduler on each of the affected workstations, making sure that:

- You use four-character names that start with a letter for the agents to comply with the IBM Z Workload Scheduler naming rule for workstations.
- You enter a different TCP/IP port number from the one used by the present instance.
- You enter `OPCMaster` as the name of the master domain manager.

The following step is to define the agents and the topology to IBM Z Workload Scheduler, using the `CPUREC`, `DOMREC`, `USRREC`, and `TOPOLOGY` statements. For details about how to specify these statements, see [Step 6. Customizing the PARMLIB members on page 40](#).

## Migrating the job scripts

The most straightforward way of migrating a job script of a IBM Workload Scheduler agent to end-to-end with fault tolerance capabilities is to define it as a centralized script in IBM Z Workload Scheduler. This means that the script continues to reside locally in the agent and that it is also defined in the `EQQSCLIB` DD statement, commonly known as `SCRPTLIB`.

To define the job script in the `SCRPTLIB`, you must specify at least the `JOBSCR` or `JOBCMD` keywords in a `JOBREC` statement. These keywords specify the name of the script or of the command that runs the job.

Other optional keywords you can use in the `JOBREC` statement are:

- `VARSUB`
- `JOBUSR`
- `INTRACTV`
- `RCCONDSUC`

As an alternative, you can move the script to IBM Z Workload Scheduler. This means defining it in the `EQQJBLIB` DD statement, usually known as `JOBLIB`, the common repository for all the scheduler's JCLs and centralized scripts. The limitation of this choice is the loss of fault-tolerance on the part of the agent if communication problems occur, because the script must be downloaded from the host every time before running.

See [Defining centralized and non-centralized scripts on page 72](#) for details on defining end-to-end agent job scripts.

## Migrating the database objects

IBM Workload Scheduler database objects that are to be converted, that is job streams, resources, and calendars, probably cannot be converted directly to IBM Z Workload Scheduler. In this case, you must change these objects into IBM Z Workload Scheduler format and create the corresponding objects in the respective IBM Z Workload Scheduler databases.

In particular, for these object definitions you have to design alternative ways of handling for IBM Z Workload Scheduler:

- Job stream run-cycles for job streams and use of calendars in IBM Workload Scheduler.
- Use of local (workstation-specific) resources in IBM Workload Scheduler (local resources converted to global resources by the IBM Z Workload Scheduler master).
- Jobs defined with "repeat range" (for example, run every 10 minutes in job streams).
- Job streams defined with dependencies on job stream level.
- Jobs defined with IBM Workload Scheduler recovery actions.

## Migrating from centralized to non-centralized jobs

You can use the Job Migration tool and `FTP` whenever you decide to transfer the scripts from the IBM Z Workload Scheduler `JOBLIB` locally to the agents. For details, see [Using the tracker jobs migration tool on page 105](#).

## Changing the topology of the network

### About this task

You can change the topology of your end-to-end network, for example, to migrate from a network where fault-tolerant agents are connected to master domain managers, which in turn are connected to the IBM Z Workload Scheduler connector, to a network where fault-tolerant agents and standard agents are directly connected to the IBM Z Workload Scheduler connector.

To change your topology, perform the following procedure:



**Note:** This procedure does not cause any loss of events related to jobs currently scheduled in the fault-tolerant agents' current plan, because the jobs will be normally submitted after you complete the steps.

1. Ensure that all the fault-tolerant agents in your network are connected and active.
2. Stop the job submission by selecting option 9 (SERVICE FUNCTIONS) from the IBM Z Workload Scheduler main menu.
3. Ensure that the jobs defined on the fault-tolerant agent workstations have one of the following statuses:

- Complete
- Error
- Arriving, with extended status Job submission is deactivated
- Ready, with extended status Job submission is deactivated

This step is optional, but it is required to make sure that none of the events related to the job submission is lost.

4. Change your network topology by modifying the CPUREC statement as required. For example, suppose that your current topology includes master domain managers and fault-tolerant agents as follows:

```

DOMREC  DOMAIN(DM1DOM)
        DOMMNGR(DM1)
        DOMPARENT(MASTERDM)
CPUREC  CPUNAME(DM1)
        CPUOS(UNIX)
        CPUNODE('XXX.YYY.ZZZ')
        CPUTCPIP(31111)
        CPUDOMAIN(DM1DOM)
        CPUTYPE(FTA)
        CPUAUTOLNK(ON)
        CPUFULLSTAT(ON)
        CPULIMIT(10)
        CPUTZ(SST)
        CPUUSER(domuser)
CPUREC  CPUNAME(FTA1)
        CPUOS(UNIX)
        CPUNODE('YYY.ZZZ.XXX')
        CPUTCPIP(31111)
        CPUDOMAIN(MASTERDM)
        CPUTYPE(FTA)
        CPUAUTOLNK(ON)
        CPUFULLSTAT(OFF)
        CPULIMIT(10)
        CPUSERVER(A)
        CPUTZ(SST)
        CPUUSER(ftauser)

```

To modify it so that the fault-tolerant agents are directly connected to the end-to-end server, change the CPUREC statement as follows:

```

CPUREC  CPUNAME(DM1)
        CPUOS(UNIX)
        CPUNODE('XXX.YYY.ZZZ')
        CPUTCPIP(31111)
        CPUDOMAIN(MASTERDM)
        CPUTYPE(FTA)
        CPUAUTOLNK(ON)
        CPUFULLSTAT(OFF)
        CPULIMIT(10)
        CPUTZ(SST)
        CPUUSER(domuser)
CPUREC  CPUNAME(FTA1)
        CPUOS(UNIX)
        CPUNODE('YYY.ZZZ.XXX')
        CPUTCPIP(31111)
        CPUDOMAIN(DM1DOM)
        CPUTYPE(FTA)
        CPUAUTOLNK(ON)

```

```
CPUFULLSTAT(OFF)
CPULIMIT(10)
CPUSERVER(A)
CPUTZ(SST)
CPUUSER(ftauser)
```



**Note:** To connect a fault-tolerant agent to the end-to-end server, the CPUTYPE must be set to `FTA`. To connect a standard agent to the end-to-end server, the CPUTYPE must be set to `SA`.

5. Renew the Symphony file and then verify that the agents are connected and active.
6. Reactivate the job submission by selecting option 9 (SERVICE FUNCTIONS) from the IBM Z Workload Scheduler main menu.

# Chapter 5. Managing end-to-end scheduling

This chapter describes the following topics:

- Notes on defining fault-tolerant workstation jobs.
- Notes on how to control the distributed plan with the ISPF dialogs, the Dynamic Workload Console, and the IBM Workload Scheduler conman command line.
- A summary of the process that creates the new current plan and the Symphony file.
- How to increase the size of an IBM Workload Scheduler event file in USS.
- The mechanism for using backup domain managers.

## Defining fault-tolerant workstation jobs

The jobs scheduled to run in an end-to-end environment must be added to the `SCRIPTLIB`, and optionally in the `JOBLIB`, of IBM Z Workload Scheduler.

In the `OPERATIONS` ISPF dialog of IBM Z Workload Scheduler, the end-to-end jobs are defined like all other operations, but the choice of a fault-tolerant workstation denotes their kind. The following example shows the definition of a job named `SAPJOB`. This is the IBM Z Workload Scheduler job that is to drive the run of an R/3 job (named `BAPRINT46B` as shown in the next example). It shows as an extended agent job because the associated workstation is an extended agent workstation named `R3XA`.

Figure 9. Defining a job for end-to-end scheduling

```
----- OPERATIONS -----Row 1 to 1 of 1
Command ==>                               Scroll ==> PAGE

Enter/Change data in the rows, and/or enter any of the following
row commands:
I(nn) - Insert, R(nn),RR(nn) - Repeat, D(nn),DD - Delete
S - Select operation details, J - Edit JCL
Enter the TEXT command above to include operation text in this list, or,
enter the GRAPH command to view the list graphically.

Application           : APLL1           FTW appl

Row Oper      Duration Job name  Internal predecessors      Morepreds
cmd ws   no.  HH.MM.SS
''' R3XA 001  00.00.01  SAPJOB_  -----  0 0
***** Bottom of data *****
```

For each job, a member must be created in the `SCRIPTLIB` of IBM Z Workload Scheduler with details about the job in a `JOBREC` statement. A `SAPJOB` member was created for the job of the previous example. It contains a `JOBREC` statement like this:

```
JOBREC
  JOBCMD('/-job BAPRINT46B -user MAESTRO -i 14160001 -c C')
  JOBUSR(twsila)
```

The string in `JOBCMD` is read and interpreted by the access method prior to running the job. The job of this example, `BAPRINT46B`, was previously defined on R/3 and assigned with an ID of 14160001, that was manually transcribed in `JOBCMD`.

The following example is for a PeopleSoft job. The entire string that follows the `JOBCMD` keyword must be enclosed in double quotation marks rather than single quotation marks, because single quotation marks are already used in the string.

```
JOBREC
JOBBCMD("/-process XRFWIN -type 'SQR Report' -runcontrol TWS")
JOBUSR(PsBuild)
```

The following example is for a Tivoli® Dynamic Workload Broker job. When you define a job that is to run in a Tivoli® Dynamic Workload Broker system, the argument of the `JOBBCMD` keyword is simply the name of the Tivoli® Dynamic Workload Broker job (properly defined in the Tivoli® Dynamic Workload Broker Web Console or in the Job Brokering Definition Console - see the *Tivoli® Dynamic Workload Broker User's Guide* for more information).

```
JOBREC
JOBBCMD("dynamicjob23")
JOBUSR(tdwbuild)
```

## Defining file dependencies to check for file changes

You can use the **filewatch** utility to check for file system changes of 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.

Only a Tivoli® Workload Scheduler agent version 8.5 or later can run **filewatch**. Do not run it as stand-alone utility.

### 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.

A detailed description follows.

### 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:

#### **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/mozart/filewatchdb.log_extension` database, otherwise `TWA_home/TWS/mozart/filewatchdb` is updated.

For details about the log database, see [Maintaining the filewatch log database on page 114](#).

#### **-returncode**

The exit return code, if the file is not found 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/mozart` 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**

**[-d[atabase] log\_extension]**

**[-r[eturncode] rc]**

The arguments are not positional. A detailed description follows.

## 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/mozart/filewatchdb.log_extension` database, otherwise `TWA_home/TWS/mozart/filewatchdb` is accessed.

## Usage Notes

To run the **filewatch** utility, you can use SCRPTLIB members (non-centralized scripts) or JOBLIB members (centralized scripts). In both these cases, consider using variable substitution, that occurs when:

- The controller or the daily planning generate the Symphony file after processing a SCRPTLIB member. For details, see [Configuring the SCRPTLIB on page 73](#).
- The controller submits the job associated to a JOBLIB member. For details about the scheduler job tailoring, see *IBM Z Workload Scheduler: Managing the Workload*.
- The agent submits the job. For details about configuring the job environment in a distributed network, see *IBM Workload Scheduler: User's Guide and Reference*.

## Examples

- Using SCRPTLIB members to run non-centralized scripts:
  - In the following example, **filewatch** checks for any file stored in the `C:\bin\my test\` path, every 15 seconds indefinitely:

```
JOBREC
JOBSCR('/tw/bin/filewatch -condition wcr
      -f "/tmp/bin/my test"
      -dea 0
      -i 15')
```

- In the following example, **filewatch** performs the first and unique loop after 30 seconds:

```
JOBREC
JOBSCR('/tw/bin/filewatch -c wmr
      -f /tmp/bin/file01
      -dea 30
      -i 45')
```

- In the following example, **filewatch** performs the last loop after 680 seconds:

```
JOBREC
JOBSCR('/tw/bin/filewatch -c wmc
      -f /tmp/bin/file01')
```

```
-dea 680
-i 45')
```

- The following example shows how you can use the `OXJOBNAM` supplied variable. This variable is automatically set to the extended job name field, defined at operation level. It allows associating the same SCRPTLIB member to any job to be used for file watching, provided that the corresponding extended job name is set to the name of the file that has to be monitored.

The scheduler will substitute `OXJOBNAM` when updating the Symphony file and the agent will substitute `HOME` when submitting the job. The `BACKPREF('$')` parameter specifies a value different from the default `%`, to leave this character as identifier for the variable to be substituted by the agent.

```
VARSUB
  TABLES(TABLE01)
  PREFIX('&')
  BACKPREF('$')
  VARFAIL(YES)
  TRUNCATE(YES)
JOBREC
  JOBSCR('%HOME%\bin\filewatch -c wmc
          -dea 60
          -i 30
          -r 125
          -fi D:\temp\&OXJOBNAM')
```

- The following example shows a JOBLIB member for a centralized script. Regarding the `OXJOBNAM` supplied variable, same considerations apply as in the previous example.

```
//*%OPC SCAN
SET FILE= -fi &OXJOBNAM.
SET PARM= -c wmc -dea 60 -i 30 -r 125
//*%OPC BEGIN ACTION=NOSCAN
echo %HOME%\bin\filewatch %PARM% %FILE%
%HOME%\bin\filewatch %PARM% %FILE%
//*%OPC END ACTION=NOSCAN
```

## Creating and activating the current plan

Daily planning is the process of producing and maintaining the current plan, which is the detailed schedule of the work that IBM Z Workload Scheduler will carry out on your system.

When scheduling jobs in the IBM Workload Scheduler environment, current plan processing also includes the automatic generation of the Symphony file. The Symphony file contains the necessary details for jobs scheduled to run on distributed agents.

The Symphony file, as a part of the current plan, uses data from the following IBM Z Workload Scheduler data sets:

- The copy of the current plan used to produce the Symphony file (SCP).
- The script library.
- The [TOPOLOGY on page 51](#) initialization statement.

## Creating and activating the new current plan

### About this task

The creation of a new current plan is performed by the daily planning batch jobs. The two possible options are *extend* and *replan* the current plan. You also use the extend function when creating a current plan for the first time. Both the extend and replan functions cause the creation of a new current plan in the new current-plan (NCP) data set.

The steps below describe in detail how the new current plan is created and then brought into normal processing, including the use of the Symphony file:

1. A daily planning batch job starts, signals the IBM Z Workload Scheduler subsystem using an ENQ, and then waits.
2. IBM Z Workload Scheduler detects the ENQ from the batch job and locks the current plan, thereby preventing more updates. All events related to job streams and jobs in IBM Workload Scheduler are queued for later processing.
3. The active current plan is updated with the latest information from in-storage control blocks representing workstations and active operations. The extension file (CX), which is held in a data space, is refreshed to DASD.
4. The inactive current plan is erased.
5. The active current plan is copied to the inactive current plan. They are now identical.
6. The inactive current plan becomes the active current plan, and the previously active becomes the inactive.
7. The job-tracking logs are switched.
8. The current plan is unlocked, and normal processing continues. The current plan is updated, processing the queued events: these updates are logged into the currently active job-tracking log. The Symphony file is updated consequently on the domain manager (waiting dependencies are eventually resolved).
9. The data from the inactive job-tracking log is appended to the job-tracking archive log.
10. IBM Z Workload Scheduler signals to the batch job that backup processing is complete.
11. The batch job starts executing again. The inactive current plan is used (together with the long-term plan, application description, resource database, and workstation for a current plan extend) as input, and a new current plan is created in the NCP and new current plan extension (NCX) data sets. While the batch job is building the new current plan, IBM Z Workload Scheduler continues normal processing except that a current plan backup is not permitted because the batch job is using the inactive current plan data set.
12. In the new current plan, the daily planning job flags the jobs and job streams that will be added to the Symphony file.
13. The contents of the job-tracking archive data set are copied to the data set in the daily planning job referenced by the `EQQTROUT` ddname.
14. When the new current plan is ready, the checkpoint data set is updated to show if the new current plan was successfully created. The normal mode manager (NMM) subtask is notified that the daily planning process has completed.
15. The subsystem examines the checkpoint data set to see if a new current plan was successfully created.

If it was not, IBM Z Workload Scheduler continues its normal processing as if a daily planning process had not run.

If the new plan was created, or when it is created, a command is sent to stop all the IBM Workload Scheduler CPUs and the local end-to-end processing. No more events for IBM Workload Scheduler are processed. These events are processed later in the new plan using the usual processing for IBM Workload Scheduler. Processing continues with the next step.

16. The current plan is locked, preventing it from being updated.
17. The new current plan is copied to the active current plan, and the NCX is copied to the current plan extension (CX).
18. The job-tracking archive log is emptied.
19. The active job-tracking log now contains a record of the updates to the current plan that were made while the daily plan job was running. These are read, and the current plan is updated accordingly.
20. In-storage control blocks representing workstations and active operations are rebuilt from the active current plan, and a data space is created from the current-plan-extension data set.
21. A current plan backup is performed. The inactive current plan is erased.
22. The active current plan is copied to the inactive one, and `EQQSCPDS` is produced. This VSAM is a copy of the current plan and will be used by data processing to add job information to the Symphony file.
23. The next available job-tracking log becomes active.
24. The current plan is unlocked and normal processing continues. All changes to jobs in the Symphony file are sent to the distributed agents, when the Symphony file is available.
25. The data from the now inactive job-tracking log is copied into the job-tracking archive log.
26. Starting from `EQQSCPDS` that contains the updates for the job-tracking file, the job and job stream information is added to the new Symphony file. If the Symphony file was not created because of errors during the building phase, you can run the Symphony `renew` or `replan` functions to create it, after correcting the errors.
27. IBM Z Workload Scheduler sends the Symphony file to IBM Workload Scheduler.
28. Before distributing the new Symphony file, all the events of the old Symphony file that are still outstanding are applied to the new Symphony file.
29. In IBM Workload Scheduler, the new Symphony file is distributed.

## Renewing the Symphony file

In some situations the Symphony file in IBM Workload Scheduler is not created. For example, there might be a recovery from a disaster situation.

In this example, the new current plan is created, but the Symphony file is not. In this situation, you can keep the current plan up-to-date by selecting the `SYMPHONY RENEW` option from the Producing OPC Daily Plans dialog shown below.

Figure 10. EQQDPLNP - Producing daily plans

```

EQQDPLNP ----- PRODUCING OPC DAILY PLANS -----
Option ==>

Select one of the following :

1 REPLAN          - Replan current planning period
2 EXTEND          - Extend the current planning period
3 TRIAL           - Produce a trial plan
4 PRINT CURRENT   - Print statistics for current planning period
5 SYMPHONY RENEW  - Create Symphony file starting from Current Plan

```

Here are some other cases where you renew the Symphony file to recover from error situations:

- There are configuration errors such as: the specified directories do not exist, the file system is full, or the workstation is not correctly defined.
- The workstation definitions are incorrect.
- An incorrect Windows™ user name or password was specified.

- Job `EQQPCS05` is run to create the work directory (`WRKDIR`) or the Symphony file was deleted (`rm` command) or renamed (`mv` command) from `WRKDIR`.
- TCP/IP goes down after the `Symnew` file is created but before it is switched. When TCP/IP is restarted, the end-to-end server is not able to switch the Symphony file.

There are also regular operation situations when you need to renew the Symphony file, such as:

- You change the TCP/IP port number on a workstation.
- You modify the `CPUREC` or `DOMREC` initialization statements.



**Note:** Each time you add a new `CPUREC` statement, you must define the new agent in the workstation (WS) database and submit a `CP EXTEND` or a `CP REPLAN` job, but not a `Symphony Renew`.

- You add or change a `USRREC` statement (including `USRCPU`, `USRNAM`, `USRPSW`).

## Recovering from a missing or corrupted Symphony file

### About this task

You cannot start an end-to-end server on which a Symphony file is missing, unless the `Translator.chk` file is also missing. To prevent this problem from occurring, rename the Symphony file (`mv` command) instead of deleting it (`rm` command). In this way, if you need to restart the end-to-end server before a new Symphony file is created, you can restore the old Symphony file by renaming it back to its original name.

If you try to start an end-to-end server on which a Symphony file is missing and the `Translator.chk` file is present, the following messages are stored in the `EQQMLOG`:

```
EQQPT67E A Dump was taken for Problem Determination purpose
EQQPT35E Unable to Open Symphony and/or events files,
Reason: AWSBIO019E File "/WRKDIR/Symphony could not be opened"
The error was: "EDC5129I No such file or directory"
EQQPT40I Output Translator thread is shutting down
```

Also, the following message is stored in the `JESMSGLOG` of the end-to-end server:

```
DUMPID=001 REQUESTED BY JOB (jobname)
DUMP TITLE=DUMP TAKEN BY MODULE TWSPlanManagerClass.C AT PLACEHOLDER NUMBER 01
```

If you try to submit a CP batch job (`REPLAN`, `EXTEND`, or `SYMPHONY RENEW`) to create a new Symphony file, the batch job fails and the following messages are issued:

```
EQQ3091E FAILED SYNCHRONIZATION WITH THE END-TO-END DISTRIBUTED ENVIRONMENT
EQQ3088E THE SYMPHONY FILE HAS NOT BEEN CREATED
```

Depending on whether the Symphony file was renamed or deleted, perform the following recovery procedure:

### Recovering from a Symphony file that was renamed



1. Stop the end-to-end server.
2. Rename the Symphony file back to its original name.
3. Restart the end-to-end server.
4. Run the CP batch job, if you want to update the Symphony file.

### Recovering from a Symphony file that was deleted

1. Check for any records that were not processed in the EQQTWSIN and EQQTWSOU files by performing the following steps:
  - a. From the IBM Z Workload Scheduler main menu, deactivate the job submission on fault-tolerant agents.
  - b. From SEQQSAMP, run the EQQCHKEV utility.

Look at the resulting output. If you see the messages `ALL DATASET EVENTS HAVE BEEN PROCESSED` related to EQQTWSIN and EQQTWSOU, all the records were processed:

```

EQQ3161I DDNAME          = EQQTWSIN
EQQ3161I RECORDS NUMBER = nnnnnnnn
EQQ3161I STATUS IS      = ALL DATA SET EVENTS HAVE BEEN PROCESSED
EQQ3161I DDNAME          = EQQTWSOU
EQQ3161I RECORDS NUMBER = nnnnnnnn
EQQ3161I STATUS IS      = ALL DATASET EVENTS HAVE BEEN PROCESSED

```

2. If all the records were processed, perform the following recovery procedure:
  - a. Stop the end-to-end server.
  - b. Rename or delete the `Translator.chk` file.
  - c. Restart the end-to-end server.
  - d. If required, from the IBM Z Workload Scheduler main menu activate the job submission on fault-tolerant agents.
  - e. Run a Symphony renew to create the Symphony file.



**Note:** By following this procedure, messages about discarding some events might be stored in the controller EQQMLOG. These events, if any, are those left in the USS `*.msg` files.

3. If there were some records not processed, perform the following recovery procedure:
  - a. Stop the controller and end-to-end server tasks.
  - b. Delete the `WRKDIR` by issuing the command `rm -r /WRKDIR`
  - c. Run the EQQPCS05 file to create a new `WRKDIR`.
  - d. Delete or rename and reallocate the EQQTWSIN and EQQTWSOU files, by using the JCL taken from sample job EQQPCS06.
  - e. Delete and re-create the EQQSCPDS file, by using the sample taken from job EQQPCS01.
  - f. Start the controller and end-to-end server started tasks (EQQTWSIN and EQQTWSOU files will be formatted by the controller started task).

- g. If required, from the IBM Z Workload Scheduler main menu, activate the job submission on fault-tolerant agents.
- h. Run a CP REPLAN to create a new Symphony file

## Managing Windows™ users and passwords locally

In an end-to-end configuration, use the **users** utility to create and manage Windows™ user IDs and passwords locally. With this command, you define on a Windows™ fault-tolerant agent the credentials of the users that are associated with job definitions by the JOBUSR parameter.

When a job is started, the fault-tolerant agent looks for the ID and password of the user who submitted it in the Symphony file. If they are not found, and LOCALPSW is set to YES in the [TOPOLOGY on page 51](#) statement, the agent looks for the user credentials in the workstation local database.

If the user ID is not found in the Symphony file and local database, the job does not start.



**Note:** For performance reasons, the users that you define locally must be also stored in the user table. Therefore, if you define or modify users' credentials, stop and restart the agent or run a Symphony renew.

## Authorization

To run the **users** utility, you must:

- Have the Impersonate a client after authentication and Log on as a batch job rights.
- Have display access to the database of the locally defined users.
- Depending on the command option you want to use, be authorized with the following access types defined in the security file:

**Table 17. Authorizations needed to run the users utility**

Command option	Object type	Access type
-u   -v	None	None
<i>username</i>	USEROBJ	DISPLAY
-s	USEROBJ	ADD (if username does not exist) MODIFY (if username exists)
-d	USEROBJ	DELETE
-e	FILE	DISPLAY
	USEROBJ	DISPLAY (to extract user definition with password marked with asterisks) MODIFY (to extract user definition with encrypted password)

Command option	Object type	Access type
-r	FILE	MODIFY
	USEROBJ	ADD (to add a new user definition) MODIFY (to replace an existing user definition)
-b	FILE	BUILD

## Syntax

**users** *username*

**users** **{[-V | -v | -U | -u] | -b[uild]}**

**users** **{-r[eplace] | -e[xtract]}** *filename*

**users** **-d[ele]te** *username*

**users** **-s[et]** *username userpwd*

## Arguments

Argument	Description
<i>username</i>	The name of the user to be created, replaced, or removed. If you specify only this argument, it returns information about whether the <i>username</i> is defined in the local database.
<i>userpwd</i>	The password of the user to be added.
<i>filename</i>	The name of the file with the user definitions to be either extracted or replaced.
<b>-v</b>	Displays the command version and exits.
<b>-u</b>	Displays command usage information and exits.
<b>-build</b>	Creates a local user database, if it does not exist. If the database exists, rebuilds it removing the records that are deleted.
<b>-extract</b>	Extracts all the user definitions from the local database and stores them into <i>filename</i> . You can later reimport them with the <b>-r</b> option.
<b>-replace</b>	Adds new user definitions stored in <i>filename</i> to the local database, or replace the existing ones.
<b>-delete</b>	Deletes the user with <i>username</i> from the local database.
<b>-set</b>	Adds the <i>username</i> and <i>userpwd</i> , if they do not exist or modify the user credentials if the <i>username</i> already exists.

## Usage Notes

By entering **users** without any argument, you are prompted for user names and passwords.

## Examples

To create a new user named `Bob` with password `My123pwd`, run the following command:

```
users -s Bob My123pwd
```

To delete a user named `Jack`, run the following command:

```
users -d Jack
```

To extract the definition of all users defined in the local database into a file named `allusers.def`, run the following command:

```
users -e allusers.def
```

## Fault-tolerant workstations and replanning

When creating a new Symphony, you can configure IBM Z Workload Scheduler to automatically send the Symphony file to IBM Workload Scheduler after you modify the current plan.

To send the new Symphony file automatically, set the `CPUAUTOLNK` parameter of the `CPUREC` initialization statement to ON.

Otherwise, you can control manually when the Symphony file is delivered to the distributed agent, set the `CPUAUTOLNK` parameter to OFF. You can improve scheduling performance in these ways:

- Reduce network congestion by controlling when the file is delivered
- Keep jobs from being scheduled on a workstation

For the syntax and description of `CPUAUTOLNK`, see [CPUREC on page 42](#).

When replanning, the distributed agent can receive and run the Symphony file and report job status back to IBM Z Workload Scheduler only if the workstation is active and linked. From the Modifying Work Stations in the Current Plan dialog, you can view and modify the status of the fault-tolerant agent. Select option 5.5 from the main menu to display the dialog:

Figure 11. EQQMWSLL - Modifying work stations in the current plan

```

EQQMWSLL----- MODIFYING WORK STATIONS IN THE CURRENT PLAN -----
Enter the row command S to select a work station for modification, or
I to browse system information for the destination.

```

Row	Work station	L	S	T	R	Completed	Active	Remaining
cmd	name	text				oper	oper	oper
						dur.	dur.	dur.
'	CPUA	Workstation 1	A	C	A	0	0.00	0
'	CPUV	Workstation 2	A	C	A	0	0.00	0
S	CPU2	Workstation 3	L	A	C	A	0	0.00
'	PRT1	Printer pool		P	A	0	0.00	0
'	SETP	Used to Prepare JCL		G	S	0	0.00	0
'	STC1	Started task		O	C	A	1	0.00
'	VST			A	C	S	0	0.00
'	WT01	Messages		A	G	C	0	0.00

\*\*\*\*\* Bottom of data \*\*\*\*\*

In the S column, the possible values for status are the following:

**A**

Active. The workstation is running and scheduling its own plan.

**O**

Offline. The workstation is not running and scheduling its own plan.

In the L column, the possible values are the following:

**L**

Linked. Communication is established between IBM Z Workload Scheduler and the workstation.

**U**

Unlinked. There is not an established connection to send and exchange events.

Job status includes information such as job started, job run, and job completed. When the new plan is produced, the distributed agents are stopped, and the scheduling on the agents is interrupted. The scheduling is resumed only when the new Symphony file is delivered.

When you send the Symphony file (`CPUAUTOLNK=OFF`) manually, you must manually link the fault-tolerant agent. Perform the following steps:

1. In the Modifying Work Stations in the Current Plan dialog ([Figure 11: EQQMWSLL - Modifying work stations in the current plan on page 124](#)), use the S row command to select the fault-tolerant agent.
2. In the Modifying a Work Station in the Current Plan dialog that is shown next, enter the following command:

Figure 12. EQQMWSSTP - Modifying the status of a fault-tolerant workstation in the current plan

```

EQQMWSSTP ----- MODIFYING A WORK STATION IN THE CURRENT PLAN -----
Command ==>>

Select one of the following command:
L LINK           To send command link work station
U UNLINK        To send command unlink work station
S START         To send command start work station
P STOP         To send command stop work station

Work station      : FMAX           COMPUTER AUTOMATIC FTW
Type              : Computer      FT Work Station
Status           : Active
Status link      : Linked

```

**L**

To link the workstation

3. The Modifying Work Station Status in the Current Plan dialog displays the change.

## Controlling the distributed plan

In addition to using the standard IBM Z Workload Scheduler user interfaces (the ISPF dialogs and the Dynamic Workload Console) to control the plan, you can control the part of the plan running on the distributed network (the Symphony file) directly on an IBM Workload Scheduler workstation. You can do this on the following interfaces:

- The conman program of IBM Workload Scheduler.
- The Dynamic Workload Console for the distributed environment.



**Note:** When the plan is part of end-to-end scheduling, some actions are inhibited. The following two sections list the subset of actions available for end-to-end scheduling.

## Conventions used for the end-to-end scheduling plan

The Symphony file used for end-to-end scheduling is created during the IBM Z Workload Scheduler current plan creation process. It contains all the jobs on distributed systems and all the z/OS® jobs that are predecessors of at least one job on the distributed system. It also contains topology information on the distributed network with all the workstation and domain definitions, including the master domain manager of the distributed network, that is, the IBM Z Workload Scheduler host.

The following list describes the conventions set for the Symphony file used for end-to-end scheduling:

- Workstations:
  - The name used for the master of the distributed network is `OPCMASTER`.
  - The name of the other workstations is the same name used for the distributed agents.
- Schedules (job streams) and jobs:
  - All the schedules are defined on `OPCMASTER`.
  - The jobs are defined on the workstation where they have to run.
  - Symphony also contains a dummy schedule named `GLOBAL` that holds a job named `SPECIAL_RESOURCES`. This job is used to represent dependencies on special resources.
  - The name of a schedule is a hexadecimal number reporting the occurrence token. This is a unique identifier for an occurrence in the current plan. Use the occurrence token value to find an occurrence in the current plan from the ISPF dialogs or from the Dynamic Workload Console.
  - The name of a job is made up according to the specifications described in [Defining standards for generating job names in the Symphony file on page 60](#).

The following information applies to job names related to `TWSJOBNAME`:

- The new rules for the job name in the Symphony file depend on the setting of the `TWSJOBNAME` parameter
- All the characters that are not allowed in the Symphony file are changed to dashes (-)
- The resulting string is converted to uppercase.

## Controlling the plan with conman

Use conman from the command line interface to start and stop processing, alter and display the production plan, and control workstation linking in a network.

The following table lists the conman commands available:

Table 18. Conman commands for controlling the plan

Command	Description	Type <sup>1</sup>
<b>altpass</b>	Alters a user object definition password.	M,F
<b>continue</b>	Ignores the next error.	M,F,A
<b>display</b>	Displays job files.	M,F
<b>exit</b>	Terminates conman.	M,F,A
<b>fence</b>	Sets the job fence.	M,F,A <sup>4</sup>
<b>help<sup>3</sup></b>	Displays command information.	M,F,A
<b>kill</b>	Stops a running job.	M,F
<b>limit cpu</b>	Changes a workstation limit.	M,F,A <sup>2</sup>
<b>limit sched</b>	Changes the job limit for a job stream.	M,F,A <sup>2,5</sup>
<b>link</b>	Opens workstation links.	M,F,A
<b>listsym</b>	Displays a list of Symphony log files.	M,F
<b>recall</b>	Displays prompts waiting for a response.	M,F
<b>redo</b>	Edits the previous command.	M,F,A
<b>reply</b>	Replies to a job prompt.	M,F,A
<b>setsym</b>	Selects a Symphony log file.	M,F
<b>showcpus</b>	Displays workstation and link information.	M,F,A
<b>showdomain</b>	Displays domain information.	M,F,A
<b>showfiles</b>	Displays information about files.	M,F
<b>showjobs</b>	Displays information about jobs.	M,F
<b>showprompts</b>	Displays information about prompts.	M,F
<b>showresources</b>	Displays information about resources.	M,F
<b>showschedules</b>	Displays information about job streams.	M,F
<b>shutdown</b>	Stops all the IBM Workload Scheduler production processes.	M,F,A
<b>start</b>	Starts IBM Workload Scheduler production processes.	M,F,A <sup>3</sup>
<b>status</b>	Displays the IBM Workload Scheduler production status.	M,F,A
<b>stop</b>	Stops IBM Workload Scheduler production processes.	M,F,A <sup>3</sup>
<b>switchmgr</b>	Switches the domain manager.	M,F
<b>sys-command</b>	Sends a command to the system.	M,F,A

**Table 18. Conman commands for controlling the plan (continued)**

Command	Description	Type <sup>1</sup>
<b>tellop</b>	Sends a message to the console.	M,F,A
<b>unlink</b>	Closes workstation links.	M,F,A
<b>version</b>	Displays the conman program banner.	M,F,A

**Note:**

1. Workstation types: domain managers (M), fault-tolerant agents (F), standard agents (A).
2. You can change the limit only of jobs launched on a standard agent workstation.
3. You cannot run the command on `OPCMaster` (the master domain manager).
4. Not available for jobs running on standard agents managed directly from `OPCMaster`.
5. Not available for jobs running on standard or fault-tolerant agents managed directly from `OPCMaster`.

For details about using conman and its commands, see the *IBM Workload Scheduler: User's Guide and Reference*.

## Controlling the plan with the Dynamic Workload Console

As an alternative to using conman, you can control and monitor the plan from a Dynamic Workload Console connected to an IBM Workload Scheduler workstation.

You can run queries on jobs and job streams to see their properties and their status and you can take advantage of a timeline view to graphically display their time restrictions.

You can perform the following actions for the objects in the plan:

### Job streams

You can:

- View the properties.
- View successors and predecessors.

### Jobs

You can:

- View the properties.
- View successors and predecessors.
- Browse the job log.
- Kill.



## Workstations

You can:

- View the properties.
- Change the job limit or the job fence.
- Start or stop.
- Link or unlink.

## Domains

You can:

- Switch the domain manager.
- Start or stop the workstations of the domain.
- Link or unlink the workstations of the domain.

In addition, you can select a different plan than the current plan.

## Killing jobs on standard agents

When a standard agent is connected to a domain manager, it is easy to kill jobs using the `conman kill` command from the domain manager. When the standard agent is connected directly to the end-to-end server (`OPCMASTER`), which does not run `conman`, you have other options to kill jobs or recovery jobs:

- Use the Dynamic Workload Console, with either the Z connector or distributed connector.
- Use the IBM Z Workload Scheduler interfaces: ISPF, program interface (PIF), batch command interface tool (BCIT), or control language (OCL).

This action can be taken only on `STARTED` jobs or recovery jobs that are in the `EXECUTING` status, so that their `JOBID` is known. The `JOBID` is required to identify the operation that is to be killed.

You can kill the same job, while it runs, as many times as required.

## Killing from ISPF

You can use the `kill` command in the MODIFYING WORKSTATIONS IN THE CURRENT PLAN panel (EQQMOPRL) shown next:

Figure 13. EQQMOPRL - Modifying operations in the current plan

```

EQQMOPRL ----- MODIFYING OPERATIONS IN THE CURRENT PLAN -- ROW 1 TO 11 OF 11
Command ==>                               Scroll ==> PAGE

Enter the GRAPH command above to view list graphically or
enter any of the following row commands:
J - Edit JCL                               M - Modify           B - Browse details
DEL - Delete Occurrence                    MH - Man. HOLD      MR - Man. RELEASE oper
O - Browse operator instructions           NP - NOP oper       UN - UN-NOP oper
EX - EXECUTE operation                     D - Delete Oper     RG - Remove from group
L - Browse joblog                          K - Kill            RI - Recovery Info
RC - Restart and CleanUp                   FJR - Fast path JR
TCJ - Target Critical Job                  FSR - Fast path SR
BND - Reset bind information                FSC - Fast path SC

Row  Application id  Operat   Jobname  Input Arrival  Duration  Opt Depen S Op
cmd                               ws  no.    Date    Time  HH.MM.SS  ST Su Pr  HN
'''  APPLCP1          CPU1 001  VPJOB1  06/10/09 1111  00.00.01  YN  0 0 U NN
'''  APPLCP1          CPU1 001  VPJOB1  06/10/09 1111  00.00.01  YN  0 0 C NN
'''  APPLSA01CEN      SA01 015  VPCEN1  06/10/09 1029  00.00.01  YN  0 0 C NN
***** BOTTOM OF DATA *****
    
```

The same command can be used in the OPERATION RECOVERY INFO panel (EQQREINP - accessible by using the RI row command in panel EQQMOPRL) to kill recovery jobs. The following is an example of the EQQREINP panel for a recovery job running on a standard agent connected to `OPCMaster`:

Figure 14. EQQREINP - Operation recovery info

```

EQQREINP ----- OPERATION RECOVERY INFO -----
Option ==>

Enter any of the following row commands:
L - Browse joblog                          K - Kill recovery job
PY - Prompt reply Yes                      PN - Prompt reply No

Recovered job info
Application      : APPLSA01                06/10/10 1328
Operation       : SA01 1                  UNX01485

Recovery option  : Rerun
Recovery Action  : Prompt / Job

Recovery job info
Recovery jobid   :                          Duration:
Work Station Name : SA01                  Status  : Waiting
Recovery job start/end :                      /

Recovery prompt info
Prompt Id : 77                               Status  : Asked
Message  : continue with recovery?
    
```

## Killing with the PIF

Use the

```
MODIFY CPOP OPCMD=KJ
```

request to kill a job, and the

```
MODIFY CPOP OPCMD=KR
```

request to kill a recovery job.

## Killing with the BCIT

Use the

```
ACTION=MODIFY,RESOURCE=CPOP,OPCMD=KJ
```

instruction to kill a job, and the

```
ACTION=MODIFY,RESOURCE=CPOP,OPCMD=KR
```

request to kill a recovery job.

## Killing with the OCL

Use the

```
KILLJOB
```

instruction to kill a job, and the

```
KILLREC
```

request to kill a recovery job.

## Sizing event files on USS

You can define the maximum size of the IBM Workload Scheduler event files by running the `evtsize` command from the `BINDIR/bin` directory on USS. Use this command to increase the size of an existing event file after receiving the message:

```
End of file on events files
```

In an IBM Workload Scheduler network, it might happen that an event file reaches its maximum size. If this happens during the distribution of the Symphony file, or just after a `Symphony renew`, a current plan extension or replan, the maximum size value changes automatically so that the distribution of Symphony can be completed without errors. At the end of this phase, you can use `evtsize` to reset the maximum size of the file manually.

Only a user with USS uid equal to 0, or the RACF® user associated with the server started task, can run this command. Be sure to stop the IBM Workload Scheduler engine before running this command.

**evtsize -v | -u**

**evtsize filename size**

**evtsize -c filename size**

**evtsize -show filename**

where:

**-c**

Compresses the *filename* event file and changes the maximum size value to a specified *size*. Use this option if the `read` pointer is greater than the `write` pointer, which denotes a file wrap condition. Otherwise, use:

```
evtsize filename size
```

to change the maximum size.

**-u**

Displays command usage information and exits.

**-v**

Displays the command version and exits.

**filename**

The name of the event file. Specify one of the following:

- Courier.msg
- Intercom.msg
- Mailbox.msg
- NetReq.msg
- pobox/workstation.msg

**size**

The maximum size of the event file in bytes. When first built by the scheduler, the maximum size is set to 10 MB.

**-show WRKDIR/filename**

Queries the current queue length of the specified file.

The following example sets the maximum size of the **pobox** file for workstation FTW1 to 15 MB:

```
evtsize WRKDIR\pobox\FTW1.msg 15000000
```

The following example sets the maximum size of the **pobox** file for workstation FTW1 to 15 MB and compacts the file:

```
evtsize -c WRKDIR\pobox\FTW1.msg 15000000
```

The following example:

```
evtsize -show WRKDIR\Intercom.msg
```

returns the following output:

```
TWS for Windows NT/EVTSIZE 8.2 (1.2.2.2)
Licensed Materials Property of IBM
5698-WKB
(C) Copyright IBM Corp 1998,2001
US Government User Restricted Rights
Use, duplication or disclosure restricted by GSA ADP
Schedule Contract with IBM Corp.
AWS11140703 Queue size current 880, maximum 10000000 bytes
(read 48, write 928)
```

where:

**880**

Is the size of the current queue of the Intercom.msg file

**1000000**

Is the maximum size of the Intercom.msg file

**read 48**

Is the pointer position to read records

**write 928**

Is the pointer position to write records

## Enabling time zones

In an end-to-end network, use the `CPUTZ` keyword to set the local time zone of a workstation when you define the workstation with the `CPUREC` statement.

Ensure that the time zone specified in `CPUTZ` matches the time zone of the operating system of the workstation. If the time zones do not match, a message coded as `AWSBHT128I` is displayed in the log file of the workstation.

In the end-to-end environment, the time zone feature is always enabled and you do not need to set it in the `globalopts` file of the end-to-end server. But you have to set it in the `globalopts` of every domain manager and agent where you want to use it. Specify `Timezone enable=yes`.

If you do not specify the `CPUTZ` keyword, the default value is UTC (universal coordinated time).

## Switching to a backup domain manager

Each domain has a domain manager and, optionally, one or more backup domain managers. A backup domain manager must be in the same domain as the domain manager it is backing up. The backup domain managers must be fault-tolerant agents running the same product version of the domain manager they are supposed to replace, and must have the Resolve Dependencies and Full Status options enabled in their workstation definitions.

If a domain manager fails during the production day, you can use the Dynamic Workload Console, the `switchmgr` command in the `conman` command line, or the IBM Z Workload Scheduler `WSSTAT` command to switch to a backup domain manager. A `switch manager` action can be run by anyone with start and stop access to the domain manager and backup domain manager workstations.

A `switch manager` operation stops the backup manager, then restarts it as the new domain manager, and converts the old domain manager to a fault-tolerant agent.

In the end-to-end environment the original identities of the domain managers are re-established in accordance with the definitions of the Symphony file anytime the Symphony file is renewed following a plan extension or a replan. So, unless you change these definitions, any switches decay with the coming into effect of the next Symphony.

## Collecting job metrics

You can run the following SQL queries on the Workload Scheduler database 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.

### 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 document provides information about copyright, trademarks, terms and conditions for product documentation.

© Copyright IBM Corporation 1993, 2016 / © Copyright HCL Technologies Limited 2016, 2024

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.

Portions of this code are derived from IBM Corp. Sample Programs.

© Copyright IBM Corp. 2016

## Trademarks

IBM, the IBM logo, and [ibm.com](http://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](http://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

## A

- accessibility viii
- agents, choosing 20
- APARs
  - PI54546 viii, 64
  - PI81106 97
  - PK00336 55
  - PK00789 47, 50, 50
  - PK01415 36
  - PK34310 35
  - PK40356 40, 47, 50, 51, 68, 104, 122
  - PK45255 64
  - PK71399 51, 59
  - PK80427 84
  - PK82310 40
  - PK87389 66
  - PM02690 108, 120, 120
  - PQ82126 59
  - PQ84104 51
  - PQ84233 51
  - PQ87710 37
  - PQ89715 34
  - PQ90090 46

## B

- BACKPREF, keyword of VARSUB 77
- backup domain manager 15
  - switching 133
- backwards migrating 106
- batch-jobs
  - generating skeleton JCL 32
- BATCHOPT initialization statement
  - TPLGYPRM keyword 60
- BINDIR, keyword of TOPOLOGY 53
- bm look 98

## C

- centralized jobs, migrating to non-centralized 108
- centralized script
  - configuring the SCRPTLIB 74
  - creating 73
  - defining 72
  - definition 72
  - downloading 73
- certificates 88
- choosing topology 20
- CLI
  - conman 133
  - switchmgr 133
- Cloud & Smarter Infrastructure technical training ix
- CODEPAGE, keyword of TOPOLOGY 53
- command
  - EVTSIZE
    - definition 131
- commands
  - conman 126
- configurations
  - examples
    - server 23
  - multiple domains 16
  - single domain 16
- conman 126
- console.msgldFilter.msglds, keyword of TWSCCLog.properties 101
- CPUACCESS, keyword of CPUREC 44
- CPUAUTOLNK parameter 124

- CPUAUTOLNK, keyword of CPUREC 45
- CPUDOMAIN, keyword of CPUREC 44
- CPUFULLSTAT, keyword of CPUREC 45
- CPUHOST, keyword of CPUREC 44
- CPULIMIT, keyword of CPUREC 46
- CPUNODE, keyword of CPUREC 44
- CPUOS, keyword of CPUREC 44
- CPUREC initialization statement 124
  - CPUACCESS keyword 44
  - CPUAUTOLNK keyword 45
  - CPUDOMAIN keyword 44
  - CPUFULLSTAT keyword 45
  - CPUHOST keyword 44
  - CPULIMIT keyword 46
  - CPUNODE keyword 44
  - CPUOS keyword 44
  - CPURESDEP keyword 45
  - CPUSERVER keyword 46
  - CPUTCPIP keyword 44
  - CPUTYPE keyword 44
  - CPUTZ keyword 46
  - CPUUSER keyword 47
    - description 42
  - FIREWALL keyword 48
  - SSLLEVEL keyword 48
  - SSLPORT keyword 48
- CPURESDEP, keyword of CPUREC 45
- CPUSERVER, keyword of CPUREC 46
- CPUTCPIP, keyword of CPUREC 44
- CPUTYPE, keyword of CPUREC 44
- CPUTZ , keyword of CPUREC 46
- CPUUSER , keyword of CPUREC 47
- creating general information about a workstation panel 62
- current plan
  - catch up 118
  - NCP takeover 118
  - new current plan 118
  - turnover 118

## D

- daily planning
  - new current plan process 118
- database
  - collecting job metrics 134
- database objects, migrating 108
- DB2
  - collecting job metrics 134
- DB2 for zOS
  - collecting job metrics 134
- dialogs
  - EQQMWSLL 125
- direct connection
  - maximum number 19
- distributed agent
  - installing 16
  - working with 15
- domain
  - possible configurations 16
- domain manager 15
- DOMAIN, keyword of DOMREC 49
- DOMMNGR, keyword of DOMREC 49
- DOMPARENT, keyword of DOMREC 49
- DOMREC initialization statement
  - definition 49
  - DOMAIN keyword 49
  - DOMMNGR keyword 49
  - DOMPARENT keyword 49

- DVIPA address 56
- Dynamic Workload Console
  - accessibility viii
- Dynamic Workload Console (TDWC)
  - terminology mapping 28

## E

- education ix
- ENABLEFIPS, keyword of TOPOLOGY 55
- ENABLELISTSECCHECK, keyword of TOPOLOGY 55
- ENABLESWITCHFT, keyword of TOPOLOGY 55
- end-to-end
  - acting on
    - IBM Z Workload Scheduler 15
  - actions 21
  - centralized script data set (EQQTWSCS) 33
  - functional enhancements compared with IBM Workload Scheduler 26
  - functional enhancements compared with IBM Z Workload Scheduler 26
  - IBM Workload Scheduler
    - functionality not supported 27
  - input events data set (EQQTWSIN) 34
    - integrating with IBM Workload Scheduler 22
  - output devices for server messages 101
  - output events data set (EQQTWSOU) 34
  - scheduling architecture 16
  - scheduling possibilities 14
  - script library (EQQSCLIB) 33
- end-to-end scheduling
  - defining distributed agent jobs 111
- END-TO-END WITH FAULT TOLERANCE 31
- EQQJOBS installation aid
  - generating batch-job skeletons 32
- EQQMOPRL panel 129
- EQQMWSLL dialog 125
- EQQREINP panel 129
- EQQSCLIB (parameter library)
  - creating the statements 75
- EQQTWSCS (end-to-end centralized script dataset) 33
- EQQTWSIN (end-to-end input events data set ) 34
- EQQTWSOU (end-to-end output events data set) 34
- EQQWCGEP panel 62
- EQQWMLSL panel 62
- EVTSIZE command
  - definition 131
- extended agent 15
- extended agent job defining with
  - end-to-end scheduling 111
- EZB.BINDDVIPARANGE 56

## F

- fault-tolerant agent
  - direct connection 19
- fault-tolerant agent (FTA) 15
- FIREWALL, keyword of CPUREC 48
- FTA (fault-tolerant agent) 15

FTW 44

## G

generating batch-job skeletons 32  
GID 36  
global options to use the time zone feature 98  
GRANTLOGONASBATCH, keyword of  
TOPOLOGY 55

## H

HOSTNAME, keyword of TOPOLOGY 55

## I

IBM Workload Scheduler  
, terminology mapping  
28  
IBM Z Workload Scheduler  
, terminology mapping  
28  
improving scheduling performance 98  
initialization statements  
  BATCHOPT  
    TPLGYPRM keyword 60  
  CPUREC  
    CPUACCESS keyword 44  
    CPUAUTOLNK keyword 45  
    CPUDOMAIN keyword 44  
    CPUFULLSTAT keyword 45  
    CPUHOST keyword 44  
    CPULIMIT keyword 46  
    CPUNODE keyword 44  
    CPUOS keyword 44  
    CPURESDEP keyword 45  
    CPUSERVER keyword 46  
    CPUTCPIP keyword 44  
    CPUTYPE keyword 44, 44  
    CPUTZ keyword 46  
    CPUUSER keyword 47  
    definition 42  
    FIREWALL keyword 48  
    SSLLEVEL keyword 48  
    SSLPORT keyword 48  
  DOMREC 49  
    DOMAIN keyword 49  
    DOMMNGR keyword 49  
    DOMPARENT keyword 49  
  OPCOPTS  
    TPLGYSRV keyword 59  
  SERVOPTS  
    TPLGYPRM keyword 59  
  TOPOLOGY  
    BINDIR keyword 53  
    CODEPAGE keyword 53  
    definition 51  
    ENABLEFIPS keyword 55  
    ENABLELISTSECHECK keyword 55  
    ENABLESWITCHFT keyword 55  
    GRANTLOGONASBATCH keyword 55  
    HOSTNAME keyword 55  
    LOCALPSW keyword 56  
    LOGLINES keyword 56  
    NOPTIMEDEPENDENCY keyword 56  
    PLANAUDITLEVEL keyword 56  
    PORTNUMBER keyword 57  
    POSTPONE keyword 57  
    RECOVERYPROMPTBASE keyword 58  
    RECOVERYPROMPTDELTA keyword 58  
    SSLLEVEL keyword 58  
    SSLPORT keyword 58  
    TCPIPJOBNAME keyword 58  
    TIMEZONE keyword 59  
    TPLGYMEM keyword 59

TRCDAYS keyword 59  
USRMEM keyword 59  
WRKDIR keyword 59  
USRREC  
  definition 50  
  USRCPU keyword 50  
  USRNAM keyword 50  
  USRPSW keyword 50

initialization statements  
  CPUREC  
    CPUTYPE keyword 45, 45  
installation directory 31  
installing  
  EQQJOBS installation aid  
    generating batch-job skeletons 32  
  IBM Workload Scheduler  
  end  
    considerations 64  
    procedure 65  
    verification information 70  
  IBM Z Workload Scheduler  
  end  
    prerequisites 30  
    procedure 30  
  message logs, checking 63  
  PARMLIB members, customizing 40  
  started-task operations, implementing  
  support for 39  
INTRACTV, keyword of JOBREC 79  
INTRACTV, keyword of RECOVERY 82

## J

job metrics  
  collecting 134  
  SQL queries 134  
job scripts, migrating 107  
JOB CMD, keyword of JOBREC 78  
JOB CMD, keyword of RECOVERY 81  
JOBREC statement of SCRPTLIB  
  definition 78  
  INTRACTV keyword 79  
  JOB CMD keyword 78  
  JOBSCR keyword 78  
  JOBUSR keyword 78  
  RCCONDSUC keyword 79  
JOBSCR, keyword of JOBREC 78  
JOBSCR, keyword of RECOVERY 82  
JOBUSR, keyword of JOBREC 78  
JOBUSR, keyword of RECOVERY 82  
JOBWS, keyword of RECOVERY 82

## L

List of Work Station Descriptions panel 62  
local options  
  network timeout 99  
local options to improve scheduling  
performance 98  
localopts 36  
LOCALPSW, keyword of TOPOLOGY 56  
LOGLINES, keyword of TOPOLOGY 56

## M

mailman server 19  
master domain manager 16  
MESSAGE, keyword of RECOVERY 81  
migrating  
  backwards 106  
  centralized to non-centralized jobs 108  
  database objects 108  
  distributed to end-to-end environment 106  
  from a topology to another 108  
  job scripts 107

tracker agent to distributed agent 104  
tracker jobs migration tool 105  
mm cache enable 98  
mm cache mailbox 98  
mm cache size 99  
mm start tomserver 21, 99  
modifying  
  fault-tolerant workstation 125  
Modifying Operations in the Current Plan  
panel 129  
mozart/globalopts 36

## N

netconf 36  
network shutdown 57  
network timeout 99  
network timeout, setting 99  
new current plan  
  creating 118  
non-centralized script  
  configuring the SCRPTLIB 74  
  defining 73  
  definition 72  
NOPTIMEDEPENDENCY, keyword of  
TOPOLOGY 56

## O

OPCOPTS initialization statement  
  TPLGYSRV keyword 59  
Operation Recovery Info panel 129  
OPTION, keyword of RECOVERY 81  
Oracle  
  collecting job metrics 134

## P

panels  
  EQQWCGEP 62  
  EQQWMLSL 62  
parameter library (EQQSCLIB)  
  creating the statements 75  
  PARMLIB members, customizing 40  
  PLANAUDITLEVEL, keyword of TOPOLOGY 56  
  PORTNUMBER, keyword of TOPOLOGY 57  
  POSTPONE, keyword of TOPOLOGY 21, 57  
  PREFIX, keyword of VARSUB 76  
  private keys 88

## R

RACF 36  
RCCONDSUC, keyword of JOBREC 79  
RCCONDSUC, keyword of RECOVERY 82  
RECOVERY statement of SCRPTLIB  
  definition 80  
  INTRACTV keyword 82  
  JOB CMD keyword 81  
  JOBSCR keyword 82  
  JOBUSR keyword 82  
  JOBWS keyword 82  
  MESSAGE keyword 81  
  OPTION keyword 81  
  RCCONDSUC keyword 82  
RECOVERYPROMPTBASE, keyword of  
TOPOLOGY 58  
RECOVERYPROMPTDELTA, keyword of  
TOPOLOGY 58  
refresh CP group 31  
replacing  
  centralized scripts with non-centralized 104  
  IBM Workload Scheduler  
  networks with end-to-end environments  
  104  
  trackers with fault-tolerant agents 104

## S

- SAGENT 45
- scheduling in end-to-end environment
  - functional enhancements compared with IBM Workload Scheduler 26
  - functional enhancements compared with IBM Z Workload Scheduler 26
  - IBM Workload Scheduler
    - functionality not supported 27
- scheduling performance, improving 98
- SCP 117
- SCRPTLIB statements
  - creating the SCRPTLIB statements 75
  - JOBREC
    - definition 78
    - INTRACTV keyword 79
    - JOBCMD keyword 78
    - JOBSCR keyword 78
    - JOBUSR keyword 78
    - RCCONDSUC keyword 79
  - RECOVERY
    - definition 80
    - INTRACTV keyword 82
    - JOBCMD keyword 81
    - JOBSCR keyword 82
    - JOBUSR keyword 82
    - JOBWS keyword 82
    - MESSAGE keyword 81
    - OPTION keyword 81
    - RCCONDSUC keyword 82
  - syntax rules 75
  - VARSUB
    - BACKPREF keyword 77
    - definition 76
    - PREFIX keyword 76
    - TABLES keyword 76
    - TRUNCATE keyword 77
    - VARFAIL keyword 77
- security level
  - enabled 91
  - force 91
  - on 91
- server@ 99
- ServerMlog,msgldFilter.msglds, keyword of
- TWSCCLog.properties 101
- SERVOPTS initialization statement
  - TPLGYPRM keyword 59
- SQL queries for job metrics 134
  - DB2 134
  - DB2 for zOS 134
  - Oracle 134
- SSL attributes
  - configuring 91
- SSL communication
  - enabled 91
  - force 91
  - on 91
- SSL support
  - configuring 92
- SSLLEVEL, keyword of CPUREC 48
- SSLLEVEL, keyword of TOPOLOGY 58
- SSLPORT, keyword of CPUREC 48
- SSLPORT, keyword of TOPOLOGY 58
- standard agent 15
  - direct connection 19
- started-task operations, implementing support for 39

- switching
  - backup domain manager 133
- switchmgr 133
- Symphony file 126
  - description 22
  - missing 120
  - not supported 27
  - recovering 120
  - restoring 120
- sync level 99
- syntax diagrams, how to read ix
- syntax rules for SCRPTLIB statements 75

## T

- TABLES, keyword of VARSUB 76
- tcp connect timeout 100
- tcp timeout 100
- TCPIPJOBNAME, keyword of TOPOLOGY 58
- TDWC (Tivoli Dynamic Workload Console) 128
- technical training ix
- time zone feature, setting 98
- TIMEZONE, keyword of TOPOLOGY 59
- Tivoli Dynamic Workload Broker 45, 72, 112
- Tivoli Dynamic Workload Console (TDWC) 128
  - sample for end-to-end scheduling 32
- tomaster 99
- topology
  - best practices 20
  - modifying 108
- TOPOLOGY initialization statement
  - BINDIR keyword 53
  - CODEPAGE keyword 53
  - definition 51
  - ENABLEFIPS keyword 55
  - ENABLELISTSECHECK keyword 55
  - ENABLESWITCHFT keyword 55
  - GRANTLOGONASBATCH keyword 55
  - HOSTNAME keyword 55
  - LOCALPSW keyword 56
  - LOGLINES keyword 56
  - NOPTIMEDEPENDENCY keyword 56
  - PLANAUDITLEVEL keyword 56
  - PORTNUMBER keyword 57
  - POSTPONE keyword 57
  - RECOVERYPROMPTBASE keyword 58
  - RECOVERYPROMPTDELTA keyword 58
  - SSLLEVEL keyword 58
  - SSLPORT keyword 58
  - TCPIPJOBNAME keyword 58
  - TIMEZONE keyword 59
  - TPLGYMEM keyword 59
  - TRCDAYS keyword 59
  - USRMEM keyword 59
  - WRKDIR keyword 59
- TPLGYMEM, keyword of TOPOLOGY 59
- TPLGYPRM, keyword of BATCHOPT 60
- TPLGYPRM, keyword of SERVOPTS 59
- TPLGYSRV, keyword of OPCOPTS 59
- tracker jobs migration tool 105
- training
  - technical ix
- Translator.chk file 120
- TRCDAYS, keyword of TOPOLOGY 59
- TRUNCATE, keyword of VARSUB 77
- tuning, network timeout 99
- tws.console.msgldFilter.msglds, keyword of
- TWSCCLog.properties 101
- tws.ServerMlog,msgldFilter.msglds, keyword of
- TWSCCLog.properties 101
- TWSCCLog.properties 36
- console.msgldFilter.msglds, keyword 101

- rules for customizing keywords 101
- sample customization 102
- ServerMlog,msgldFilter.msglds, keyword 101
- tws.console.msgldFilter.msglds, keyword 101
- tws.ServerMlog,msgldFilter.msglds, keyword 101

## U

- UID 36
- user for OPC address space 31
- USRCPU, keyword of USRREC 50
- USRMEM, keyword of TOPOLOGY 59
- USRNAM, keyword of USRREC 50
- USRPSW, keyword of USRREC 50
- USRREC initialization statement
  - definition 50
  - USRCPU keyword 50
  - USRNAM keyword 50
  - USRPSW keyword 50

## V

- VARFAIL, keyword of VARSUB 77
- VARSUB statement of SCRPTLIB
  - BACKPREF keyword 77
  - definition 76
  - PREFIX keyword 76
  - TABLES keyword 76
  - TRUNCATE keyword 77
  - VARFAIL keyword 77

## W

- work directory 31
- workstation
  - modifying fault-tolerant 125
- workstation classes
  - not supported 27
- wr enable compression 99
- WRKDIR, keyword of TOPOLOGY 59

## X

- XAGENT 45