

Nusantara Software Industry

zJOS/Puspa



**Workloads
Scheduling
User Guide**



Table of Content

CHAPTER 1	INTRODUCTION	1
1.1.	Workloads Scheduling.....	1
1.2.	Automatic Workloads Scheduling	1
1.3.	Pipelining Workloads Operation	3
1.4.	Triggering Decision Method.....	5
1.5.	Automatic Scheduling with Puspa	7
CHAPTER 2	GETTING STARTED	8
2.1.	Preparing zJOS Address Spaces.....	8
2.2.	Starting and Stopping zJOS	8
2.3.	Preparing Puspa.....	8
2.4.	Starting and Stopping Puspa.....	10
2.5.	Preparing National Holiday Table.....	14
CHAPTER 3	WORKING WITH SCHEDULER TABLES	15
3.1.	Scheduled-Workload Table	17
3.2.	Scheduled-Workload Entry	22
3.2.1	Workload Identifier	24
3.2.2	Schedule Timeframe.....	26
3.2.3	Special Schedule Calendar	28
3.2.4	Crossed-Date (Midnight) Handling.....	31
3.2.5	Exception Handling	32
3.3.	Triggering Events Table	36
3.3.1	Managing Triggering Events Table.....	36
3.3.2	Triggering Logic Mechanism	40
3.3.3	Complex Triggering Logic Mechanism	42
3.4.	Triggering Events Entry	43
3.4.1	Triggering Event Identifier.....	44
CHAPTER 4	CONTROLLING PUSPA	47
4.1.	Status Information	47
4.1.1	Products Status Information	48

4.1.2	Statistics Information	51
4.2.	Load Schedule Tables	54
4.3.	Navigate Schedule Flow	55
4.3.1	Halting Schedule	55
4.3.2	Resuming Halted Schedule.....	57
4.3.3	Restarting Halted Schedule	57
4.4.	Control Panel Commands	58
4.4.1	LQ - Shows ENQued Processes	58
4.4.2	CAL - Shows Current Month Calendar.....	59
4.4.3	HOL - Shows Holiday Calendar.....	60
CHAPTER 5	MANAGING IN-MEMORY SCHEDULE TABLES	61
5.1.	DIV Capacity and Utilization.....	62
5.2.	Tables Statistics	63
5.3.	Managing Tables	64
5.3.1	Obtaining Helps.....	67
5.3.2	Selecting and Updating a Schedule Table	67
5.3.3	In-memory Table Internal Structure	69
5.3.4	Copying a Table	69
5.3.5	Reorganizing a Table.....	71
5.3.6	Deleting a Table	73
5.4.	Managing Table Entry.....	74
5.4.1	Bypass a Workload.....	75
5.4.2	Hold a Workload	76
5.4.3	Delete a Workload.....	77
5.4.4	Restore a Workload	78
5.5.	Saving Updates	79
5.6.	Reviewing the Updates.....	79
5.7.	Fixing Inconsistent Table Structure	80
CHAPTER 6	OPERATE SCHEDULING SYSTEM	82
6.1.	Preparing Schedule Flow.....	82
6.1.1	Initial Flow Workload	82
6.1.2	Cleaning Up Schedule Table.....	83
6.2.	Starting Schedule	83
6.3.	Monitoring Schedule Activities	85
6.3.1	Scheduler Logs	86
6.3.2	Detail Workload Information	88
6.3.3	Detail Triggering Information	89
6.3.4	Successors List Information	91



6.4.	Halt and Restart Schedule.....	92
6.5.	Force a Workload to Run.....	93
6.5.1	Rerun a Workload	94
6.5.2	Unconditionally Run a Workload.....	94
6.6.	Scheduling Report.....	95
6.6.1	Setting up Report.....	95
6.6.2	Producing Report.....	96
6.6.3	Re-downloading Report.....	98
 CHAPTER 7 INTEGRATED SCHEDULING.....		99
7.1.	Integrated zJOS Network.....	99
7.1.1	Hardware Requirements	99
7.1.2	Software Requirements	100
7.2.	Puspa for Integrated zJOS Network	101
7.2.1	Preparing zJOS Server.....	102
7.2.2	Preparing zJOS Agent for z/OS.....	105
7.3.	Puspa Agent for z/OS.....	107
7.3.1	Starting and Stopping zJOS Agent	107
7.3.2	Connecting and Disconnecting Agent	110
7.3.3	Controlling zJOS Agent	112
7.3.4	Remote Command.....	113
7.3.5	Remote Job Submission	115
7.4.	Planning The Integrated Scheduling	115
7.4.1	Puspa Concept versus GDPS.....	118
7.4.2	Modernize Conventional DR with Puspa	119
 CHAPTER 8 OPERATE INTEGRATED SCHEDULING SYSTEM.....		123
8.1.	Reviewing Agent-Server Connection.....	123
8.1.1	Reviewing Server Site	124
8.1.2	Reviewing Agent Site.....	125
8.2.	Monitoring Integrated Schedule	126
8.2.1	Monitoring Syslog	126
8.2.2	Monitoring Schedule Log.....	128
 CHAPTER 9 COMMANDS AND MESSAGES REFERENCE		131
9.1.	Puspa Commands Facilities.....	131
9.1.1	Entering Command via zJOS Subsystem	131
9.1.2	Entering Command via MODIFY	131
9.1.3	Entering Command via zJOS Control Panel	132
9.2.	Puspa Commands Reference.....	132
9.2.1	HALT request.....	132
9.2.2	HOLD request	133
9.2.3	INIT request	133



zJOS-XDI Technology

9.2.4	LOAD request	133
9.2.5	REFRESH request	134
9.2.6	RELOAD request	134
9.2.7	RESTART request	135
9.2.8	RESUME request	135
9.2.9	START request	135
9.2.10	STOP request	136
9.3.	zJOS System Commands Facilities	136
9.4.	zJOS System Commands Reference	137
9.4.1	ASCB request	137
9.4.2	HELP request	137
9.4.3	LIST request	137
9.4.4	RCMD request	137
9.4.5	RJOB request	138
9.4.6	SHUTDOWN request	138
9.4.7	START request	138
9.4.8	WTO or MSG request	139
9.5.	zJOS Agent Commands Facilities	139
9.6.	Agent Commands Reference	139
9.6.1	CONNECT request	140
9.6.2	DISCONNECT request	140
9.6.3	DROP request	140
9.6.4	GET request	140
9.6.5	HELP request	141
9.6.6	LIST request	141
9.6.7	START request	141
9.6.8	STOP request	141
9.7.	Puspa Messages	142
CHAPTER 10	ADVANCED TRICKS	143
10.1.	Using Sekar Standard EMS Features	143
10.2.	Using zJOS Rexx Functions	144
INDEX	ERROR! BOOKMARK NOT DEFINED.	



Chapter 1 Introduction

This topic introduces you to the concept of automatic workloads scheduling and how zJOS/Puspa® do it for you.

1.1. Workloads Scheduling

Although computer system is an automation symbol, it doesn't mean everything in the computer can go automatically. This only means automatic computing by mean of program, instead of manual computing. The way a program (in z/OS system called job, task or workload) runs, however, needs to be started manually by user or operator. It actually does not really matter for interactive computing such as online transactions (e.g. CICS) or queries (e.g. MQ, DB2), since they are started once a day, a week or even a month for long running process. For a number of short running workloads, which are usually batch jobs, manual start sometime become really matter.

In most of computing shops, z/OS system acts as mainframe or super server or at least database server or data center, which consists of huge amount of various complex databases. To manage huge amount of databases, sometime hundreds batch jobs are needed. Normally a batch job depends on other jobs, hence can only be started after one or more certain jobs has completed. Even, dependencies inter jobs sometime must be done conditionally. For example, job C needs only be started when job A finish with condition code 0 and job B finish with condition code 8. Otherwise, start job D. In such cases, to start jobs is a serious matter. Operation team can only work with job schedule manuals which called "run-book". That is the way mainframe computers were operated for tens years prior to 80's.

1.2. Automatic Workloads Scheduling

Since early 80's, many software industries start think about automatic scheduling chance. Automatic scheduling is a software product which acts as operators to start each job follows "soft run-book" which came from the original hardcopy run-book. Operator's efforts then totally reduced. They only need to load soft run-book and start first jobs. Once soft run-book is loaded and first jobs are started, subsequent jobs will be started automatically follow the jobs operation flows as stated on the soft run book. Operators team then have plenty of time to rest until certain intervention request occurs.



Automatic scheduler works much better than operator team. It doesn't need time to detect job status. Modern scheduler such as Puspa can even detect each job step status. Hence, triggering mechanism can be established in job step level, instead of job level as in manual operation. For example; there are 3 workloads, JOB0, JOB1 and JOB2 as illustrated below:

```
//JOB0
//STEP01 ...
//    produce file A1
//STEP02
//    produce file A2

//JOB1
//STEP11
//    read file A1
//    produce file B1
//STEP12
//    produce file B2
//STEP13
//    produce file B3

//JOB2
//STEP21
//    read file B1
//    produce file C1
//STEP22
//    produce file C2
//STEP23
//    produce file C3
```

In manual operation, operators usually do the following rule:

1. Start JOB0
2. Start JOB1 when JOB0 ended
3. Start JOB2 when JOB1 ended.

Job JOB1 depends on job JOB0 because job JOB1 needs file A1 which is produced by job JOB0. Job JOB2 needs file B1 which is yielded by job JOB1, hence job JOB2 depends on job JOB1. File A1 is actually produced by job step STEP01 of job JOB0. Hence to run job JOB1 doesn't need to wait until job JOB0 completely done. Job JOB1 can actually be started once file A1 produced and closed, which means, job step STEP01 of job JOB0 done. Job JOB2 can also be started once file B1 produced and closed, which means, job step STEP11 of job JOB1 done. Such mechanism is very impossible to be done manually. Although operators can browse each job using SDSF to see job step status, very impossible for them to take action exactly at the time end-of-step (EOS) or end-of-job (EOJ) event occurs. Because, information on SDSF session is snapshot



and updated only when you press enter key. Besides, each TSO user can only have 2 SDSF sessions, hence only 2 jobs can be viewed from one TSO user.

Using automatic scheduler, you can setup your soft run book to start job JOB1 at the time job step STEP01 of job JOB0 ended, and start job JOB2 at the time job step STEP11 of job JOB1 ended. Hence, STEP11 of job JOB1 is executed while STEP02 of job JOB0 is running. STEP21 of job JOB2 is begun at the same time as STEP12 of job JOB1. Therefore, all the above 3 jobs run faster with automatic scheduler. Imagine if you have a huge number of jobs to run, how significant you can reduce total turn-around time.

1.3. Pipelining Workloads Operation

Job step level triggering mechanism as explained in 1.2 is also called pipelining or semi parallel jobs operation. In most batch production environment, each job has inter-dependencies with other workloads as described in figure 1.1. More number of workloads gets more complex inter-dependencies each other. Using pipelining operation method, automatic scheduler can significantly reduce total turn-around time needed by all workloads to get complete.

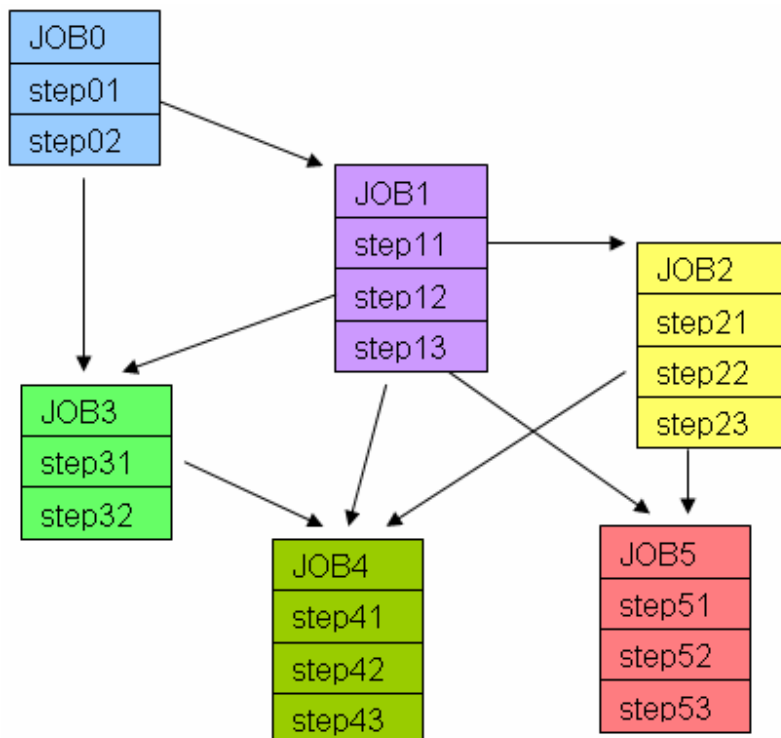


Figure 1.1: Inter-jobs dependencies.



In manual operation, since triggering mechanism is done in EOJ-level, assuming no think time delay, total turn-around time is sum of all job time reduced by sum of job time of jobs which can be executed concurrently, as illustrated in figure 1.2.

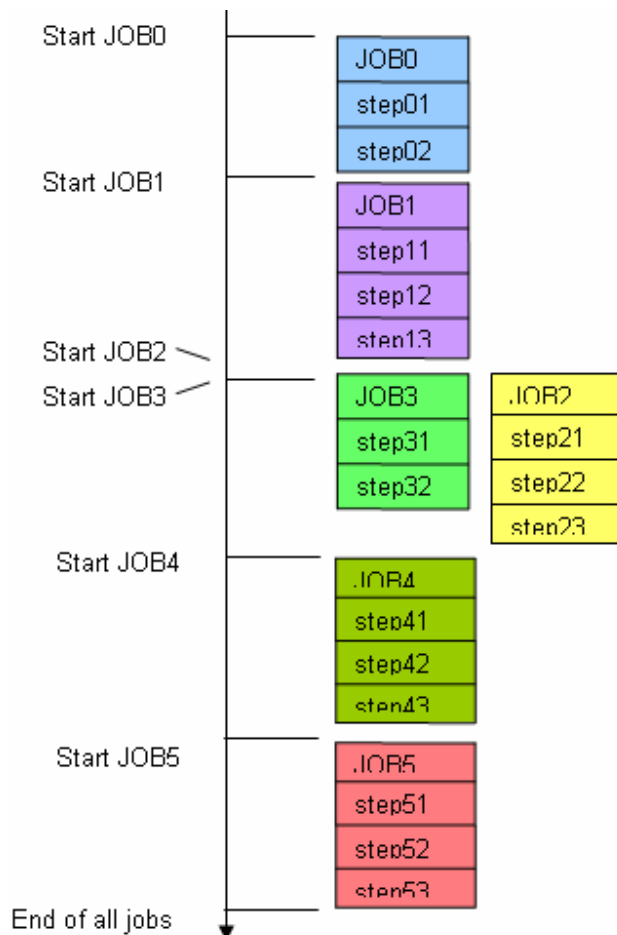


Figure 1.2: Turn-around time in EOJ level triggering.

The automatic scheduler capable to establish pipelining operation using EOS-level triggering mechanism. As explained on the above example, pipelining can reduce total turn-around time needed by all jobs to complete. Figure 1.3 shows illustration how total turn-around time needed in manual operation (figure 1.2) is reduced. To complete 5 jobs in this example, total job time more or less sum of 3 job time.

Total length of turn-around time depends on the major characteristic of all jobs. For certain application which their major inter-jobs dependencies in EOJ-level, automatic scheduler will not able to establish EOS-level triggering. Total job time will still the same as manual operation. Total turn-around time, however, will still be reduced since automatic scheduler doesn't have think time delay to watch message and type START command on console or SUBMIT command on TSO terminal. Besides, automatic scheduler could not be an ignorant like a human



when do the work. It always works consistently as exactly what you have defined in the soft run book, unless system or its program got problem.

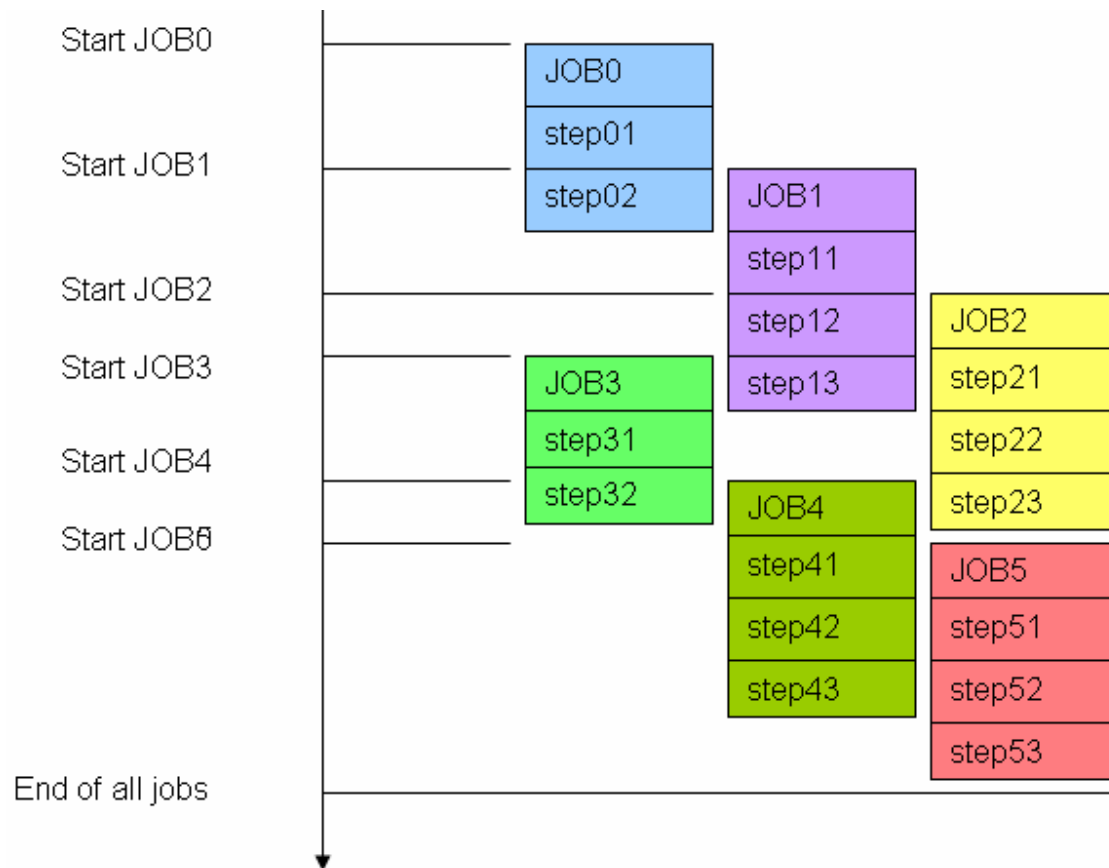


Figure 1.3: Turn-around time in EOS level triggering.

1.4. Triggering Decision Method

As a modern workload scheduling solution, Puspa offers a very sophisticated inter-workload triggering decision method, especially for batch-job and STC type of workload which has dependency with multiple predecessors. Unlike most of conventional job schedulers which only use a series of logical AND comparisons among predecessors to get a final triggering decision, zJOS/Puspa, however, allows you to group predecessors. Within each group, Puspa uses a series of logical OR comparisons among predecessors to get a group level triggering decision. Inter-group, Puspa uses a series of logical AND comparisons among groups to get a final triggering decision.

Combined with pipelining workload operation mechanism, this method becomes a very flexible method for users to draw and design the workload schedule flow

exactly meet their own idea. Figure 1.4 illustrates an example case of such method application.

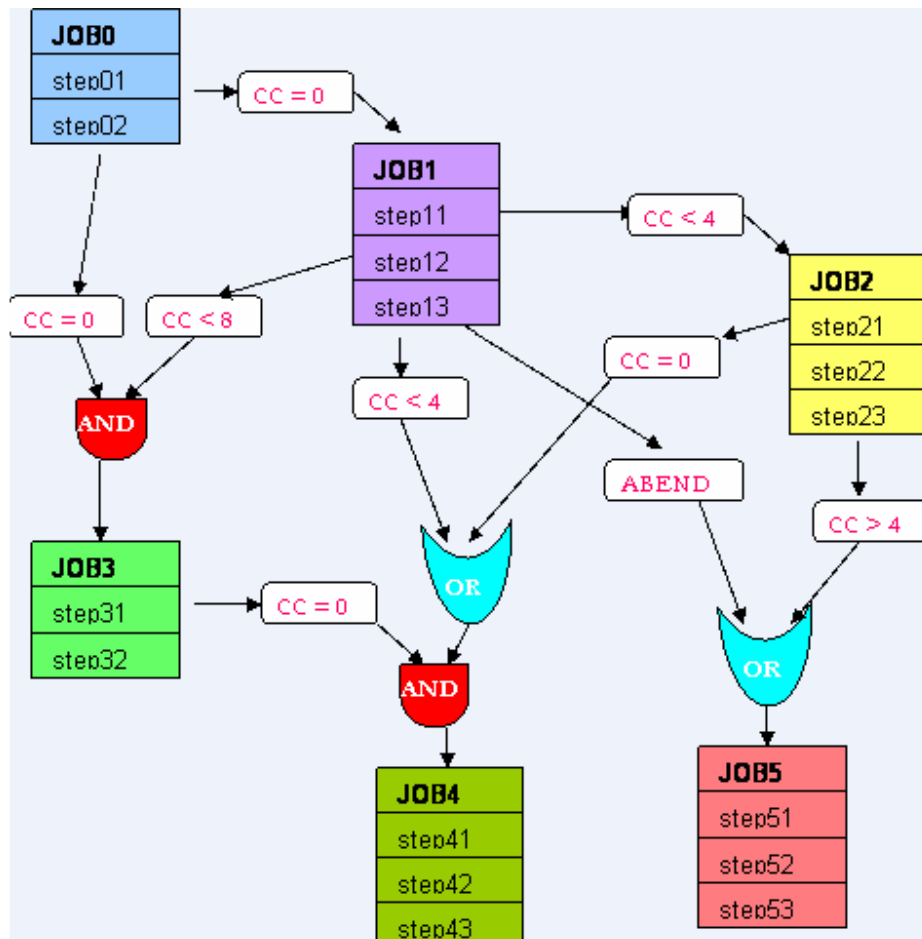


Figure 1.4: Combined multi-stages triggering decision in workload pipelining.

Puspa is capable to handle up to 100 groups. Each group can accommodate any number of workloads, no limit. Hence total number of predecessors of each scheduled workload is almost unlimited. The only limit is memory capacity.

This is exactly a revolution in workload scheduling technology. The original idea is to encourage users to be more creative and understand that scheduling flow is a fundamental of an application design as well as accompanying recovery flows. There is no standard for both types of flows. Hence in the future when they develop application, both types of flows are included in the design.

1.5. Automatic Scheduling with Puspa

Puspa or zJOS/Puspa® is one of modern automatic workload scheduler for z/OS or OS/390 system. It works based on job start, EOS and EOJ events, as well as commands and messages occurrence. Since it intercepts JES2 and resource manager directly, Puspa does not need SMF and JES2 exits. This is to avoid dependencies with users' modification area. As a modern workload scheduler, Puspa supports the following features:

- Multi-triggers schedule control with unlimited number of triggers.
- Pipelining operation with EOS-level triggering mechanism.
- Conditional EOS/EOJ triggering based on complex large-range Boolean condition code checking.
- Multi-schedule flow with single control.
- Timeframe control for each schedule entry.
- Integrated scheduling system on networked multi z/OS or OS/390 hosts.
- Embedded console commands on each schedule entry.

Multi triggers, pipeline and complex condition code checking are very common features on most of modern scheduler products. The significant feature of Puspa is its capability to establish an integrated scheduling system among several z/OS or OS/390 hosts on TCP/IP network, regardless sysplex is implemented. Puspa doesn't need sysplex. A z/OS or OS/390 host can join as a member in integrated scheduling system as long as it connected on the same TCP/IP network and has zJOS agent installed and active. Such configuration is then called as ***integrated zJOS network***.

The integrated scheduling is a scheduling system implemented in integrated zJOS network. It means a job on each z/OS or OS/390 host can be triggered by other jobs on any other z/OS or OS/390 host on the same network. Puspa runs on one of z/OS or OS/390 hosts on the network which is assigned as scheduling server.

Other significant feature is timeframe control. Combined with multi schedule flow capability, Puspa does not need schedule-id to select and activate a schedule flow. When you start schedule (issue SCD START), Puspa will automatically activate the correct schedules flows based on timeframe.



Chapter 2 **Getting Started**

zJOS/Puspa® is a modern workloads scheduling solution software product which is bundled together with zJOS/Sekar® (EMS manager) and XDI/AutoXfer® (report/spool distribution) in a single package called zJOS-XDI. All are running in a single MVS address space, named XDI, which is zJOS main address space.

Regardless AutoXfer is used in your environment, zJOS main address space is always accompanied by XDILGR address space, which actually is AutoXfer logger.

2.1. Preparing zJOS Address Spaces

Please refer to zJOS/Sekar® User Guide, point 2.1 chapter 2.

2.2. Starting and Stopping zJOS

Please refer to zJOS/Sekar® User Guide, point 2.2 chapter 2.

2.3. Preparing Puspa

By default, Puspa parameter sample is provided in zJOS-XDI product installation package, in XDISCD00 member. You should not tailor directly to any zJOS PARMLIB member, including XDISCDxx, which can destroy their sensitive binary information. You should use XDI ISPF panel instead. Issue “XDI” on any ISPF session, zJOS control panel then appears on your terminal screen as shown in figure 2.1 below.

Before you start zJOS for the first time, you have to make sure that VSAM LDS is already defined for Puspa. When zJOS is started, VSAM LDS is then formatted to accommodate all Puspa data areas and schedule tables. LDS is then used as DIV backup which is mapped in 2 ways. Internally within zJOS address space, DIV is mapped in extended private area. Externally, outside the address space, the DIV is mapped as dataspace and shared among zJOS external services and subsystem.

Once LDS was formatted, Puspa then load schedule table XDISCDxx pointed to by XDISYSxx onto allocated space on DIV and make it effective for use. Loaded schedule table will remain on DIV permanently, until you ask Puspa to reload it

(issue RELOAD command). zJOS control panel provides facility to update Puspa schedule table on DIV directly. Hence, you actually don't need to modify and reload XDISCDxx member.

For subsequent zJOS startup, Puspa checks whether addressed XDISYSxx is already on DIV. If so, no loading is performed, rather, just make it effective for use. Else, load it onto DIV and make it effective for use.

Before you initialize Puspa, you have to prepare its parameters correctly as you expect, which is schedule table. Schedule table is a "soft run-book", which consists of list of scheduled jobs and several triggering jobs. Each scheduled job accompanied by groups of triggering jobs, except for first job of each schedule flow. You can have more than one schedule table. But, only one table can active at a time. Each schedule table logically must consist at least one schedule flow.

Action		Help		zJOS-XDI Control Panel					Row 1 to 17 of 17	
Command	Product	State	Table	Suf	works	Usage	Day			
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000005	LCNSD/YR	1087			
	Puspa (SCD)	<UP> ACTIVE	loaded	00	000045	LCNSD/YR	1087			
	Autoxfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	1087			
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none			
Date	Time	Log								
08.275	00:35:07	Statistics:								
08.275	00:35:07	Config: SSN=XDI Load=LPA COM=0AE4E000 WSA=00B21F90								
08.275	00:35:07	Subtasks: Major=009 EVX=000 SVR=000 SCD=001 Abn=000								
08.275	00:35:07	Network agents: total=0000 active=0000 local=N/A								
08.275	00:35:07	Network traffic: Snd=00000000 Rcv=00000000 que=00000								
08.275	00:35:07	JES I/F: Up=Y PIT=Y Conn=Y Idr=Y FR(5=N,12=Y,22=N)								
08.275	00:35:07	Queues: ARQs=00000 SQBs=00002 EOTs=00001 RMG=00000								
08.275	00:35:07	State: NORMAL Parm: SYS=00 EMS=00 SCD=00 DEST=00								
08.275	00:35:07	SCD: Lib=0 O=EVXMS M=EVXMS Pos=RMTJOB-RVIEW EnQ=FREE								
08.275	00:35:07	SCD Free-pool: SCT=002732 TRG=0059709 EOT=0059941								
08.275	00:35:07	SCD Used-pool: SCT=000268 TRG=0000291 EOT=0000059								
08.275	00:35:07	SCD Curr-pool: SCT=000034 TRG=0000046 EOT=0000059								
08.275	00:35:07	Parmlib=SYS3.ZJOS213.PARMLIB								
08.275	00:35:07	LDS=SYS3.PUSPA.BNI.LDS,Vol=ZNIST1								
08.275	00:35:07	Your (IDMUSER) authorities: Oper=Y Setting=Y Update=Y								
08.275	00:35:07	Genlevel=20070903 CpuId=005E70/2096 System=BNIPROD /ZOS1								
***** Bottom of data *****										

Figure 2.1: zJOS primary control panel

Either to customize Puspa given sample of schedule table or build your own from scratch, you have to enter to XDI/ISPF interface from TSO. Login to TSO userid you have assigned as XDI administrator logonid (one that you used to install zJOS-XDI). Type "XDI" command on any ISPF panel or session, then zJOS primary control panel occurs as shown in figure 2.1. Then click action-bar, zJOS action-bar menu will appear in small window as shown in figure 2.2. To reach the parameters, select option 1 of action bar menu.



```

Action Help
- 1. Administering zJOS
  2. Viewing Scheduler Activities
  3. Managing In-memory Scheduler Tables
  4. Viewing AutoXfer Logs
  * 5. Starting AutoXfer Logger
  6. Stopping AutoXfer Logger
  * 7. Starting zJOS Subsystem
  8. Shutting-down zJOS Subsystem
  9. Exit

11.278 04:25:21 Statistics:
11.278 04:25:21 Config: SSN=zJOS Load=LPA COM=1DA6D040 WSA=00C6BF90
11.278 04:25:21 Tasks: Maj=010 EVX=01 Net=00 SCD=025 Abn=000 Prm=015
11.278 04:25:21 Network agents: total=0002 active=0000 local=N/A
11.278 04:25:21 Network traffic: Snd=00000000 Rcv=00000000 Que=00000

```

Figure 2.2: zJOS primary action bar menu

2.4. Starting and Stopping Puspa

Starting Puspa means start automatic scheduling process. Once start request is issued, all workloads names registered in the schedule table are then executed based on defined condition. Scheduling process continues until all registered workloads were done or an unresolved condition is encountered, or requested to stop. When all registered workloads were completed, Puspa then set its state to passive until the table is refreshed. When unresolved condition is encountered, Puspa will still in active state until the condition is resolved or table is refreshed.

When schedule table you attempt to use is already on DIV, you may straight requesting Puspa to start scheduling process. Such condition is indicated as "READY" in state column as shown in figure 2.3. Issue START request in Puspa command slot on control panel, then press enter-key.

Action Help		zJOS-XDI Control Panel						Row 15 to 27 of	
Command	Product	State	Table	suf	works	usage	Day		
	Sekar (EMS)	<UP> ACT(SS1)	loaded	00	000005	LICENSED	none		
	Puspa (SCD)	<UP> READY	loaded	00	000000	**DEMO**	..?!		
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!		
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none		

Figure 2.3: Puspa state when schedule table is loaded

When schedule table you attempt to use is already on DIV, you may not straight to start Puspa. Rather, you must initialize Puspa with a schedule table prior to start it. Issue INIT request in Puspa command slot on control panel as shown in figure 2.4, then press enter-key, or issue INIT command on console as follow:

```
.SCD INIT
```

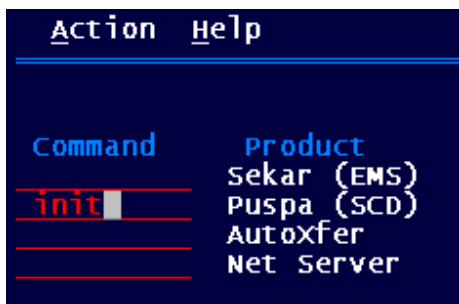


Figure 2.4: Initializing Puspa via control panel

```
.SCD INIT
DERCMD201I zJOS is ready to accept command.
DERSIP203I Request for zJOS/Puspa (scheduler) accepted
DERSIP320I zJOS/Puspa(R), z/OS job scheduler is being initialized.
DERSIP326I Scheduling table XDISCD00 is being loaded.
DERSIP217I Scheduled commands are being initialized.
DERSIP303I Sched-table of 00000024 records is being read.
DERSIP303I Trigg-table of 00000047 records is being read.
DERSIP220I Scheduled commands initialization complete.
DERSIP327I Scheduling table XDISCD00 is now loaded.
DERSIP323I Scheduling agent is being initialized.
DERSIP324I Scheduling agent is now active.
DERSIP322I zJOS/Puspa(R), z/OS job scheduler is now active.
```

Figure 2.5: Initializing Puspa using console command

Figure 2.5 shows console log of Puspa respond messages when you issue INIT command. It shows you, which table is loaded. Once INIT complete, Puspa is then ready for work, but it is not working yet. Message 322 tells that the status of Puspa is active. It doesn't mean Puspa is working, instead just ready. On state column of zJOS control panel is displayed as READY.

You can load more than one table. Each will be held on DIV permanently until you replace it by reloading the same table.

To start Puspa work, you can either issue START request in Puspa command slot on zJOS control panel as shown in figure 2.6 or issue START command on console as follow:

```
.SCD START
```

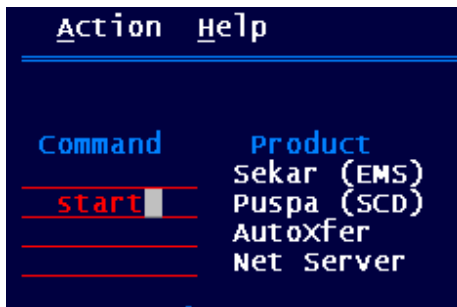



Figure 2.6: Starting Puspa

```
XDI SCD START
DERCMD201I zJOS is ready to accept command.
DERSIP203I Request for zJOS/Puspa (scheduler) accepted
DERSIP156I Scheduled job library allocation complete
DERSIP095I The first job is scheduled.
DERSIP154I Scheduling is now in progress.
IEC223I 20,IFG0200V,XDI,XDI,JCLLIB,0AA5,NITDAT,SYS5.XDIV212.SAMPJOBS
IEC223I 20,IFG0200V,XDI,XDI,SYS00001
DERSCS318I Job JTEST00 is triggered at 10:51:17 on Sun 2007/04/22.
$HASP100 JTEST00 ON INTRDR FROM STC05513 XDI
IRR010I USERID XDI IS ASSIGNED TO THIS JOB.
IEC223I 20,IFG0200V,XDI,XDI,JCLLIB,0AA5,NITDAT,SYS5.XDIV212.SAMPJOBS
$HASP100 JCOBA01 ON INTRDR FROM STC05513 XDI
IEC223I 20,IFG0200V,XDI,XDI,SYS00001
DERSCS318I Job JCOBA01 is triggered at 10:51:18 on Sun 2007/04/22.
IRR010I USERID XDI IS ASSIGNED TO THIS JOB.
ICH70001I XDI LAST ACCESS AT 22:06:50 ON SATURDAY, APRIL 21, 2007
$HASP373 JTEST00 STARTED - INIT 1 - CLASS A - SYS SYS1
IEF403I JTEST00 - STARTED - TIME=10.51.19
ICH70001I XDI LAST ACCESS AT 10:51:19 ON SUNDAY, APRIL 22, 2007
$HASP373 JCOBA01 STARTED - INIT 2 - CLASS A - SYS SYS1
IEF403I JCOBA01 - STARTED - TIME=10.51.20
DERSIP203I Request for zJOS/Puspa (scheduler) accepted
DERSCS316I Job(JCOBA01) step(JC01S01) pstep(**none**) was EOS
CC=S000-U004
DERSCS317I Job(JCOBA01) step(JC01S01) pstep(**none**) is triggering
job(JCOBA02).
DEREVX890I 01 EOS JCOBA01 (JOB05542) S(JC01S01 /**none**) S000-U004 HSCT
on MFOC
DERSCD098I J(JCOBA02) trig j(JCOBA01 /**none**.JC01S01)-> OK 00 -> OK
DERSCD598A J(JCOBA02) trig j(JCOBA01 /**none**.JC01S01)-> OK All-> OK
DERRMG799I EOS R=0025/JCOBA01 Tr=Y St=(**noname*/JC01S01) S#001
CC=S000-U004 I=N Sys=Local
IEC223I 20,IFG0200V,XDI,XDI,JCLLIB,0AA5,NITDAT,SYS5.XDIV212.SAMPJOBS
$HASP100 JCOBA02 ON INTRDR FROM STC05513 XDI
IEC223I 20,IFG0200V,XDI,XDI,SYS00001
```

Figure 2.7: Starting Puspa activity

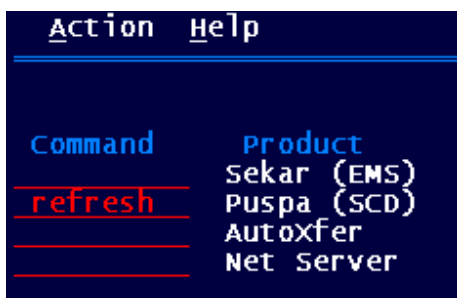
Puspa then starts schedule processing based on which loaded schedule table is selected. All schedule entries are searched to find all available schedule flows and schedule them as shown on console log in figure 2.7.



Availability of each schedule table is check against its timeframe, which might be influenced by your own specific operation calendar. Hence you don't need to be bothered to select which schedule to start today. Console log in figure 2.7 shows 2 jobs are started (submitted), job JTEST00 and JCOBA01. Each job represents a schedule flow, and as the first job of flow. Following these 2 jobs, Puspa will automatically start subsequent jobs according to the logic of each flow you have defined in schedule table. Jobs in one schedule flow can trigger other jobs in other flows.

When the last job of the longest flow completes, then one schedule cycle was completed. Puspa then enters to passive state and waits for next instruction. You can not reissue START command to restart schedule flows until you either issue REFRESH or INIT. Unless you need to reload schedule table, you can just issue REFRESH request in Puspa command slot on zJOS control panel as shown in figure 2.8 or issue REFRESH command on console to cleanup all recorded status in schedule table for next schedule cycle, as follow:

.SCD REFRESH



Action	Help
Command	Product
<u>refresh</u>	Sekar (EMS)
	Puspa (SCD)
	AutoXfer
	Net Server

Figure 2.8: Refreshing schedule table

Although you can manually start the same jobs before, within or after schedule cycle processing, Puspa will not record them. Puspa traces, detects and records only the jobs it was started. Nevertheless, manually started jobs might affect the execution logic of schedule flow when they cause conflict on results. For example, job A which is registered in a schedule flow, causing file A to be read and then deleted. If you start job A manually during schedule progress before job A is started by Puspa, then file A gone. All scheduled jobs which need file A of course affected. Hence, logic flow of schedule process then affected as well.

Puspa will automatically shutdown when zJOS address space is brought down. It can also be brought down without bringing zJOS address space down. To stop or shutdown Puspa, issue STOP request in Puspa command slot on zJOS control panel as in figure 2.9 or use the following STOP command on console:

.SCD STOP

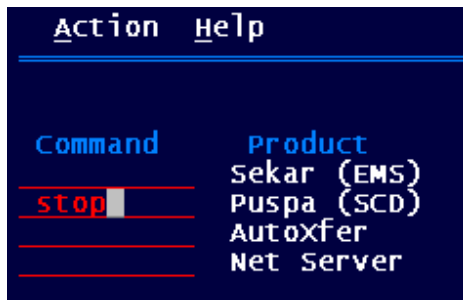


Figure 2.9: Stopping Puspa

Although actually nothing is sensitive, unless required for maintenance or other emergency situation, you don't need to shutdown Puspa. Once STOP command is issued, Puspa is then brought down regardless its current state. Hence, you have to make sure Puspa is really in idle state (no work) before you issue STOP.

In an integrated scheduling environment, where Puspa handles schedule of jobs which distributed on among integrated zJOS network, shutting down Puspa is not recommended. It causes all connected agents purge schedule table they have already received from Puspa and inactivate their job status listeners. It becomes inefficient time when you restart Puspa very soon later. You have to wait until all agents have received schedule table and confirmed ready.

Action Help		zJOS-XDI Control Panel						
		Row 323 to 335 of						
Command	Product	State	Table	suf	works	usage	Day	
	Sekar (EMS)	<UP> ACT(SSI)	loaded	00	000002	LICENSED	none	
	Puspa (SCD)	<UP> PASSIVE	loaded	00	000009	**DEMO**	..?!	
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!	
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none	

Figure 2.10: Puspa is in passive state

2.5. Preparing National Holiday Table

One of important timeframe factor is holiday calendar. For each schedule flow, you have to decide whether it available in the holiday. Further about preparing holiday table can be found in point 2.5, chapter 2 of zJOS/Sekar User Guide.

Chapter 3 Working with Scheduler Tables

Scheduler tables which also called scheduler parameters, consist of a paired of scheduled-workload table and several triggering-predecessor or trigger-event tables. To work with scheduler parameters, you have to login to administrator logonid, TSO userid of which you were using to install zJOS-XDI package. Issue XDI command in ISPF command line field on any panel of any session, then primary zJOS control panel appears as shown in figure 2.1 (chapter 2).

zJOS provides 2 ways to work with Puspa parameters, source parameters which deal with PDS member, and in-memory parameters which deal with dataspace which is backed up by DIV. When your Puspa is newly installed, which no table is ready in dataspace, you are recommended to prepare source table, instead of work directly with empty dataspace. Select option 1 on zJOS action-bar menu as shown in figure 2.2, then hit enter-key to obtain parameter panel as shown in figure 3.2. Prior to appearance of zJOS parameters panel, a window is popped up as shown in figure 3.1 asking which parameter suffix you are going to use and in which library base system parameters are retrieved. To proceed, you have to complete this window first. Parameter suffix must be filled with 2-digit xx to points to zJOS system parameter XDISYSxx. Although any 2 EBCDIC characters are allowed, the 2-digit suffix should be numeric characters.

PARMLIB dataset must be filled with name of partition dataset (PDS or PDSE) which is being used or planned to be used as zJOS parameter library. If dataset is being used by zJOS, by means concatenated as PARMLIB DD in XDI procedure, all parameters you are going to manage can be activated soon. Else, all parameters are just candidate for use later. You must concatenate the dataset in PARMLIB DD of XDI procedure first.

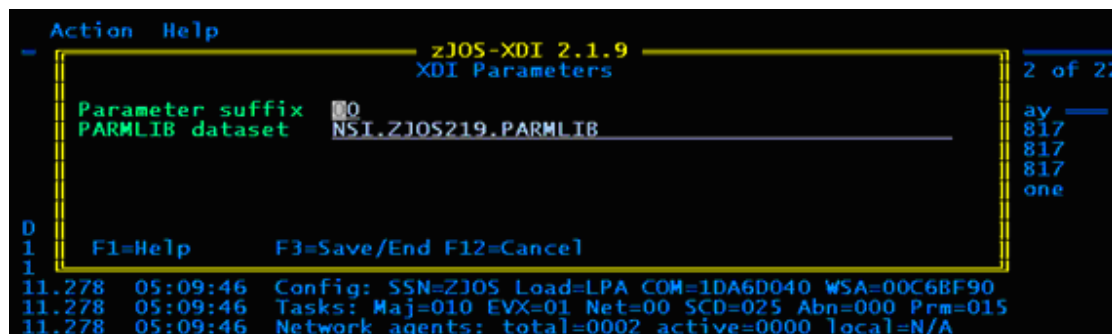


Figure 3.1: Asking parameter suffix and zJOS PARMLIB name



Upon completion of this question, window disappears and zJOS-XDI parameters panel is then reached as shown in figure 3.2. You may need to fill or update all necessary base parameters. This panel is larger than screen size, so you need to scroll it up to find next parameters.

On the top of this panel is a menu to which product you want to go. To reach the Puspa parameters panel, select option 2 (jobs scheduler) and press enter-key. At initial time, before you do it, you have to complete Puspa product key and automation suffix in this panel first. Ask your XDI support personnel to provide the product key.

```

Action Options Help
-----
zJOS-XDI Parameters

Working with ..... 1. System Automation
                   2. Jobs Scheduler
                   3. Reports Distribution

----- Scroll up/down for detail parameters below -----
AutoXfer prodkey ..... 1E5C89-3D7B5C-6E89E4  unlicensed users only
Sekar(R) prodkey ..... EC0E0C-F89AC5-5C9D2D  unlicensed users only
Puspa(R) prodkey ..... E89CDD-5CAD2D-CC1B2D  unlicensed users only
Automation suffix ..... 00  for Sekar and Puspa
                               AutoXfer users only
Spool input class .... PRQ  1-to-8 alphameric
Xmitter job class .... E  valid JES INIT class
Xmitter output class ... I  valid sysout class
Class for requeue .... O  valid sysout class
Target file format ... W  W=Windows or M=MVS
Forced to use VBA? ... Y  Y or N
Alt. report name ..... D  D=DDname or J=Jobname
Max output age ..... 1  1-99 days retention
Trace mode ..... N  Y=On or N=Off
Disp. after process ... K  K=Keep or D=Delete
Dest. table suffix ... 00  xx of XDITABxx
TempDS primary space ... 350  number of cylinders
TempDS 2ndary space ... 75  number of cylinders
Target PC drive ..... C  1 digit A-Z
Convert MCC to ASA? ... Y  Y or N
STEPLIB dataset ..... NIT.Z10S213.Z19LNK
NETRC dataset ..... 00  xx of NETRCxx
                               SSRF users only
Splitting parameter ... Y  Y if SSRF is installed
SSRF library ..... NIT.Z10S213.SSRFLIB
                               CA-Xcom users only
Flat file HLQ ..... XDI  (not used anymore)
Flatfile disp. .... K  K=Keep or D=Delete
Hold option ..... P  P=Process or I=Ignore
Xmitter job parameter
Command ==>
F1=Help F3=Save/End F7=Up F8=Down F12=Cancel
  
```

Figure 3.2: zJOS-XDI parameter

As mentioned above, a schedule table consists of one scheduled-job table and several trigger-event tables. Physically, however, all are placed in a single PDS member, XDISCDxx, which is assigned as zJOS-XDI parameters library. Once the member is loaded, one scheduled-job table and several triggering-job tables then generated in dataspace memory, as shown in figure 3.3.

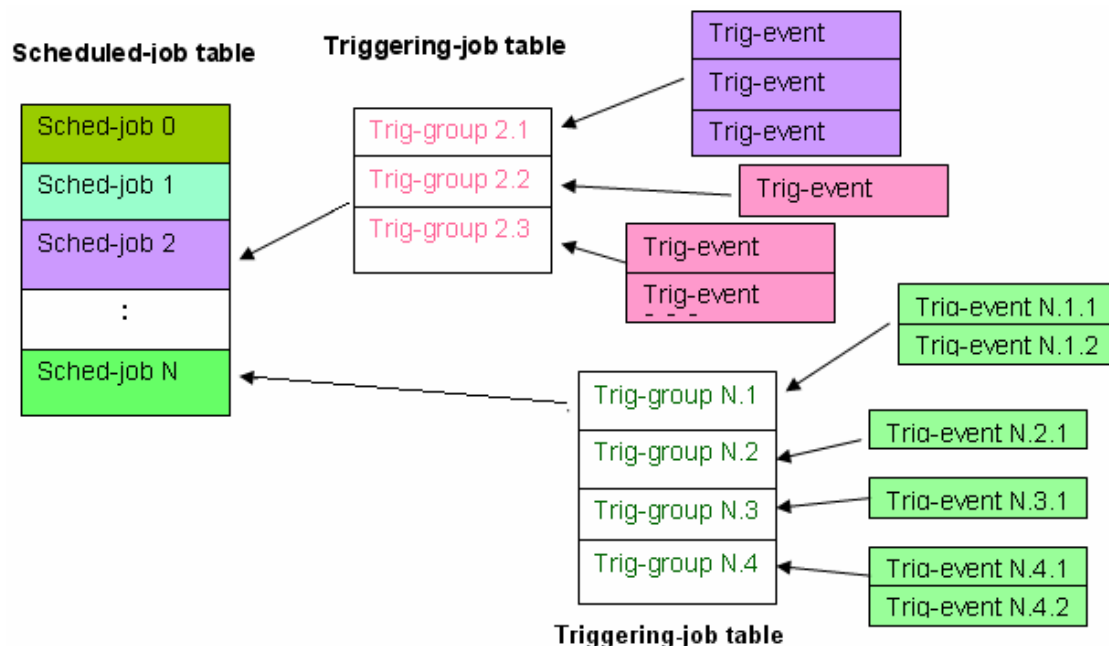


Figure 3.3: Scheduled-workload and trigger-event tables

3.1. Scheduled-Workload Table

Scheduled-job or scheduled-workload table as shown in figure 3.4 is Puspa primary schedule table. Initially, before you have in-memory table, you must build source-level table which is PARMLIB member. To reach this table, select option 2 on parameters panel in figure 3.2. Although, detail of each panel in source-level table handling is simpler, basic panel structure involved for source-level table handling is almost similar with one in in-memory table management. Beside, once converted to in-memory table, you would never back work with source-level table. Therefore, to avoid redundancy, each panel described here is actually for in-memory tables.

Managing scheduled-job table

Scheduled-job table panel (figure 3.4) provides facilities to manage scheduled-job table as described in figure 3.5, e.g.: Function keys, action bar menu and S column prefix command. Function keys consist of 4 keys:

- F1 – obtain help panel or window
- F3 – save and close the table
- F7 – scroll up screen
- F8 – scroll down screen
- F12 – abort all changes and close the table.

Action Help

scheduled-workload Table

Row 1 to 17 of 34

S	workload	System	Date	Time	Day	SMTWTFSS	Msg
JTEST00	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JTEST01	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JTEST02	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JTEST03	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JTEST04	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JTEST05A	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JTEST06	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JTEST07A	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JCOBA01	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	1	
JCOBA02	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	
JCOBA03	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	
JCOBA04	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	
JCOBA05	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	
JCOBA06X	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	
TSORXCLT	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	
TSORXCL1	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	
TSORXCL2	JOB	BNIPROD	Start	00:00:00	List	111111	
			End	24:00:00	Holiday	0	

Command ==> Scroll ==> CSR

Figure 3.4: Scheduled-job table panel

Action-bar of this panel provides facility to add a new entry. Exit choice in action bar menu is to save and close the table, the same as F3. Other facilities can be done in S column. S column is input column for 1-digit prefix command to manage the table. Valid prefix commands in this column are:

- **S** – Select particular workload entry in detail as shown in figure 3.6.
 - This gives you chance to update the detail of selected event entry
- **T** – Show associated triggering predecessors table as in figure 3.15.
 - This gives you chance to manage triggering predecessors table of selected scheduled-workload.
- **F** – Forecast successors list which is triggered by selected workload.
 - This gives you chance to check whether the workload you have defined is as what you desire.
 - This facility is applicable only in in-memory table handling.
- **D** – Delete particular entry from the table.
- **B** – Bypass workload from scheduling process.
 - During scheduling process, bypassed workload will always be treated as complied trigger for its successors.



- This facility is applicable only in in-memory table handling.
- **U** – Undo (restore) previous delete or bypass request.
 - This facility is applicable only in in-memory table handling.
- **A** – Add a new event entry to the table.
 - Selected entry is ignored, then obtains detail event entry panel with all field blanks and ask you to fill up.
 - This can also be done from action bar selection menu.

Heading of this panel shows current table suffix and main job/JCL library dataset name. If you concatenate job/JCL libraries in JCLLIB DD card of XDI procedure, this library name is ignored.

The screenshot shows a panel titled "Scheduled-jobs Table" with "Row 1 to 16 of 24". It displays a table with columns: S...Jobname, System, Date, Time, Day, SMTWTFSS, and Msg. The table lists jobs JTEST00 through JTEST07A, each with start/end dates and times, and a holiday status. Annotations include:

- A callout pointing to the "Action" menu bar: "Use action bar menu to add a new entry".
- A callout pointing to the table rows: "Type **S** to select entry for viewing or update, Type **T** to show its action table, Type **D** to delete it".
- A callout pointing to the bottom of the table: "Scroll **Up** and **Down** to have your preferred view".

At the bottom, it shows "Command ==>" and function keys: F1=Help, F3=Save/End, F7=Up, F8=Down, F12=Cancel. A "Scroll ==> CSR" option is also visible.

Figure 3.5: Managing scheduled-job table

As shown in figure 3.5, table suffix and job/JCL library dataset name appear on panel. These might not be schedule table and job/JCL library which currently is being used by Puspa, instead schedule table library currently you are managing, according to selected zJOS-XDI base parameter suffix before entering zJOS-XDI base parameter panel as in figure 3.2.

Shown in the panel, Job/JCL library is only a single dataset. You can only specify a single library on zJOS-XDI base parameter (XDISYSxx). To have multiple JCL libraries, use JCLLIB DD concatenation in the XDI procedure instead. When XDI connection to JES SSI is failed, however, Puspa will only used a single Job/JCL library specified in base parameter.



Workload column

Lists scheduled workload names and types. For batch-job (JES initiated job) and STC type of workload, the jobname must follow standard MVS job naming rule. For command type of workload, the jobname is just a unique name you assign to be used by Puspa (internally) to identify the workload.

For remote workload, at the moment Puspa supports batch-job type workload only. Job/JCL library must be on targeted system and concatenated in JCLLIB DD of XDA procedure, which is the zJOS Agent. Existence of remote job can not be checked by Puspa. Later in the operation, Puspa just send the jobname to zJOS Agent

System column

This column show name of system on which the workload to be scheduled. For local system, zJOS-XDI obtains the actual system name based on specified SYSNAME= parameter in your current IEASYSxx member in system parameter library (normally SYS1.PARMLIB). For remote system, the name is host name of the system in the TCP/IP network. For the z/OS host, refer to the value of HOSTNAME parameter specified in TCP/IP data.

Remote workloads can only be managed by Puspa only if zJOS-XDI agent is active on that remote host. Workload events (EOS and EOJ) will be detected by agent and reported to Puspa, for further actions, including triggering successor workloads. When condition is complied to trigger remote workload, Puspa then send request to agent in the targeted host to execute the workload.

Date and time columns

These are major timeframe columns which consist of start- and end-date, and start- and end-time to filter whether workload is valid to be processed. Checking is done for each particular filter and only if one specified.

Asterisk (*) in start and end dates indicate no date range specified. In detail entry panel, when selected, such dates appear as blank.

Day list and holiday column

These are minor timeframe to do second filter. Day list is a list of valid week day from Sunday to Saturday. Holiday means national holiday you have registered in the holiday calendar table. To schedule the workload, calendar must comply that current day is a checked valid day, means the day is selected day, including if a holiday. By default, all 7 week days are selected if not holiday. Means, the workload will be scheduled everyday except on holiday, unless holiday option is checked.



Minor timeframe filtering is overridden by special schedule calendar option if one is selected. The special schedule calendar doesn't appear in this panel. Rather, in workload detail panel as shown in figure 3.6 we discuss later in this chapter.

Saving and aborting changes

When you finish work with the schedule table and you want to save all changes you have made, press F3. Update progress appears in small window, then panel close and back to previous panel. XDISCDxx member in XDI parameter library is then physically changed.

When you want to abort all changes you have made, press F12 instead. The abortion alert then appears in small window. Panel then close and back to previous panel. XDISCDxx member in XDI parameter library is then remains unchanged. Take a note that in source-level table, abortion in this level is total cancellation. The XDI/ISPF interface will not remember what you have done so far. All changes you have done will be losing. If you want to abort some changes you have made but not all, you must do abort particularly while you were in event detail level.

Be careful when you delete a scheduled-workload entry. Delete command doesn't have detail level panel. Once "D" was invoked, selected entry is then instantly deleted from the table.

In in-memory table, however, total abortion (F12) is not supported. As you are working directly with memory, once a change is made, memory content is changed. The only chance to abort the changes is at particular workload level by pressing F12 in either workload detail panel or trigger (predecessor) detail panel. Therefore, to prevent your table from unexpected changes, most of table handling panels need confirmation to proceed. For entry deletion, Puspa does not really purge the entry, rather just mark it as deleted. You may restore deleted entry easily, just by typing 'U' prefix command.

3.2. Scheduled-Workload Entry

```

Action Help
-----
Scheduled-workload Entry Detail

workload name JTEST04 member name JTEST04 on system BNIPROD
Date start      end      (YYYY MM DD)
Time start - 00 00 00 end - 24 00 00 (hh mm ss)

workload type 1 1. Batch-job Spool Destination
                2. STC
                3. Command

CMD verb/text

valid days / Sunday options - calendar is verified once at the time More: +
           / Monday      - schedule cycle is started
           / Tuesday     - Use DDNAME as report name if no XWTR
           / Wednesday   - Execute commands at schedule time
           / Thursday    - Change REGION nolimit (0M)
           / Friday      - Change TIME nolimit (1440)
           / Saturday    - Use individual sysprint
           / Holiday

only on or Except on
- IDAO - IDAO - Initial day after off
- IWWD - IWWD - Initial week working day
- IMWD - IMWD - Initial month working day
- FWWD - FWWD - Final week working day
- FMWD - FMWD - Final month working day
- EMON - EMON - End of month

when exception do Exception when CC
1 1. Normal schedule 6 1. (EQ) Equal CC 000
  2. Issue command
  3. Refresh scheduler
  4. Halt scheduler
  5. Restart scheduler
  6. Hold the workload
  7. Cancel and hold workload
  7. (NE) No equal
  3. (LE) Less/equal
  4. (LT) Less than
  5. (GE) More/equal
  6. (GT) More than
  7. (AB) Abend

Command ==>
    
```

Figure 3.6: Scheduled-workload entry detail panel (scrollable)

Figure 3.6 shows scheduled-workload entry panel. It displays content of selected workload entry definition. This panel is obtained by typing S on selected entry and then hit enter-key. You can just review or update each field of this panel. To update, just overtype information fields you want to change, then hit enter-key. See figure 3.7 for brief description of each field.

Picture in figure 3.6 shows workload entry panel for in-memory table. For source-level table, such panel looks much simpler. Hence, initial table which is built in source-level need to be reedited once it loaded into DIV as an in-memory table. It will be no really matter since initial table normally is only few entries just to initialize the DIV.

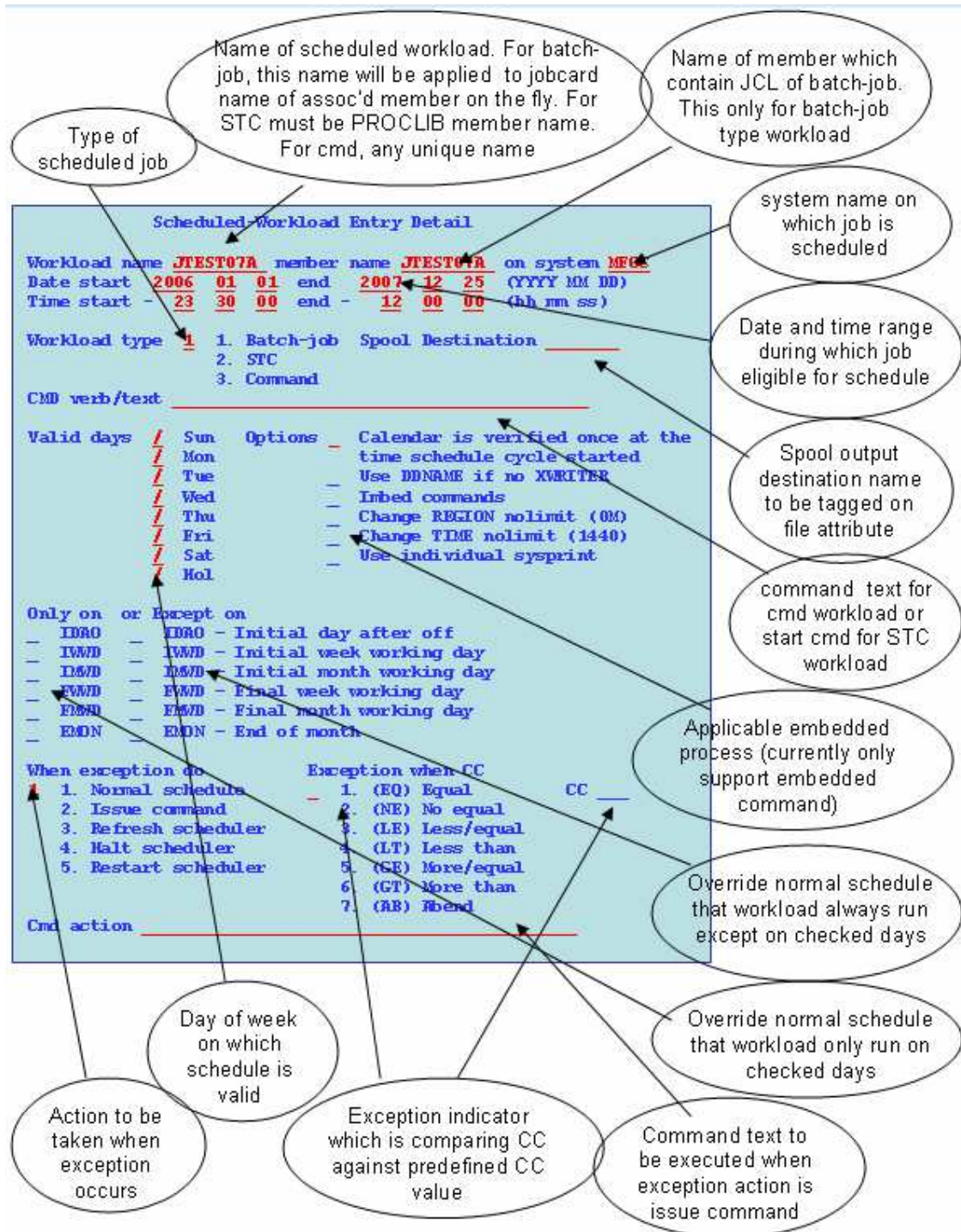


Figure 3.7: Managing scheduled-workload entry detail



3.2.1 Workload Identifier

Within Puspa scheduling system, workload is identified by name of job or more popular as jobname, name of system on which workload is to run, and type of workload. Other field included in this group is member name.

Type of workload

At the moment, Puspa only support 3 types of workload:

Batch-job (JOB)

is also known as JES initiated job or even more popular as just job. When you mention JOB, people automatically interpret as batch-job. Batch-job is one of z/OS standard workload. This workload runs on MVS initiator address space and fully managed by the job entry subsystem (either JES2 or JES3). Since number of initiators is fixed, hence, multiprogramming level of batch-jobs is also fixed.

To start a batch-job, the job deck of cards is stacked on card-reader, then release them. Currently we rather use virtual devices. Job deck of virtual cards, which is a fixed 80-byte record file of sequential dataset or PDS or PDSE member, is stacked onto internal-reader (INTRDR). Name of deck or dataset or member containing job JCL cards does not important. Name of batch-job is generated by JES based on name of JOB card found in job stream received by JES.

Started-task (STC)

is also known as system task. STC is also one of z/OS standard workload. In the system, such workload runs on its own address space. By default STCs are managed by primary subsystem which is JES. However, in some cases, STC can runs under non-primary subsystem, such as MASTER and others.

To start an STC, issue START <jobname> command. System then search all its current PROCLIB datasets to find a member with name match with jobname specified on START command text, which should containing task procedure cards. Matched member is then dumped onto virtual reader (STC internal reader) for execution. Name of STC is generated by system based on name of PROC card found in procedure stream.

Command



Although command is also z/OS standard workload, it's not very common to be involved in workload scheduling. Nevertheless, Puspa support this type of workload to be scheduled. However, the status information can not be explored, since usually command is just a subtask in either master or console address space. Command workload actually doesn't have name. In Puspa scheduling environment, you can name it with any unique word just to identify it.

Workload name

Name of workload, which is to be scheduled. Combined with workload type and system name must become a unique identifier within the whole schedule table. Though, workload name itself is recommended to be unique. The mean of job or workload name depends on its workload type.

For batch-job, jobname is a name specified on the JOB card of job JCL. Since Puspa deal straightly with the job JCL libraries, you are allowed to specify job name differently from member name. This gives you flexibility when you need to schedule more than once within a single schedule flow. You can easily define, for example, JOBX1, JOBX2, JOBX3 all with the same member, JOBX. Puspa changes the original jobname to specified jobname while dumping it onto internal reader. This is applicable only for local batch-job. For remote job, member name must always the same as jobname.

For STC workload, jobname is a name specified on the PROC card of job JCL. Puspa relies on START command to get STC up, instead of deal with dataset in which STC procedure is contained. Since the way START searches STC for execution is by searching name of PROCLIB members, therefore, STC jobname should the same as name of member containing current valid system procedure libraries. Hence, for STC, member name is always the same as workload name. Otherwise, Puspa ignores it.

For command workload, name or jobname is just a name to be used internally by Puspa to identify the workload. No member is to be addressed. When you specify member name, Puspa ignores it. No job status information is collected by Puspa, rather, just time of its occurrence.

System name

Name of the system on which workload is going to run. See explanation on the previous par (3.1).



Member name

Name of member of job JCL library (JCLLIB DD) dataset in which the job JCL is contained. Member name is applicable only for local batch-job type workload. For other type of workload, member name is ignored.

For local batch-job, member name can be different with its jobname. This is a chance for you to schedule a batch-job more than once during schedule cycle. Name of job in JOB card will be replaced by jobname specified in this schedule entry definition.

3.2.2 Schedule Timeframe

Timeframe is combined of major and minor time filter during which workload is considered eligible for scheduling. Minor time filter is overridden by special schedule calendar if one selected. When no triggering event is associated and timeframe is specified, then the workload will only be triggered by TOD event on every valid day.

Start-date and end-date

Start date and end date are major time filters during which workload is eligible for further scheduling evaluation. No default for start- and end-date. If not specified or set to zeros, no date filtering is done in this level, rather, straightly in clock time level filter. In schedule table panel appear as '****/**/****' and in schedule entry detail as blanks.

Date-range filtering can also specified partially, for example, '****/09/**'. This means, Puspa only evaluates month. Be careful to specify partial date range. Lets say, if you specify date range '****/09/**' to '****/**/****', you got the same effect as '****/09/**' to '****/12/**'. Puspa grants the schedule in month 09, 10, 11 and 12. If you expect to be granted only in every September, you must specify '****/09/**' to '****/09/**'.

Other example, to get granted on every 1st to 10th monthly, you must specify date range '****/**/01' to '****/**/10'. To get granted on every 1st to 10th monthly in every 1st semester, you need to specify date range '****/01/01' to '****/06/10'. Be careful, for partial date-range, specified value in start date must not higher than end date. Wrongly specified date-range will cause entry is flagged as error entry and will never be used to evaluate date-range filtering. It will affect to your whole scheduling system if this entry is to be used to trigger other entries.

Unless you very sure how schedule flow is expected to process, you are strongly recommended not to specify date-range filtering, except for workloads which are assigned as initial jobs for the schedule flow. In most of scheduling systems,



date-range decision is made only during initial process. Subsequent workloads normally just follow their predecessors.

Unless unspecified, start date must not greater than end date. Wrongly specified date range causes workload will never be scheduled.

Start-time and end-time

Start time and end time are second major time filters during which scheduled workload is eligible for further evaluation. Default start time is 00:00:00 and end time is 24:00:00. Leave them both default means, no filtering is done in this level, rather, straightly to minor time filters. Hence, any time non-TOD trigger occurs during valid date range, will be considered as valid schedule and filtering is then continued to minor timeframe. Workload will only be scheduled within this clock range timeframe.

Clock time filtering is very sensitive filtering for scheduling. Once a workload got late, it will affect all its successors will also late. Meanwhile, if its successors have clock time filtering, some might get expire. Hence, unexpected result may yield for the whole schedule flow.

If triggers come earlier than start time, workload is then placed in wait state to delay schedule until start-time is reached. Such situation sometime is expected in complex scheduling mechanism, where workload is planned to be triggered by its predecessors but allowed to start at certain time. However, if such situation is not expected, it even wastes the time and could potentially affecting the total turn-around time of the whole schedule flow.

Unlike date range, in clock range, start time may greater than end time. Crossing midnight is assumed for such cases. You may specify start time 20:00:00 and end time 01:00:00. This means, the workload will only be scheduled between 20:00:00 of the day and 01:00:00 on next day,

Unless you very sure how schedule flow is expected to process, you are strongly recommended not to specify time-range filtering, except for workloads which are assigned as initial jobs for the schedule flow. In most of scheduling systems, time-range decision is made only during initial process. Subsequent workloads normally just follow their predecessors.

Valid week-day list and holiday

Beside first and second major timeframe, Puspa offers other chances to validate schedule based on day-of-week and holiday. Day-list offers which day-of-week during major timeframe/timespec, specified workload is valid to be scheduled. To choose the day-of-week, mark check box in front of selected day name with



slash. By default, all days are selected. This means, workload will be scheduled everyday during date and time range.

Holiday here means national holiday. It offers chances whether you want to treat schedule differently on the holiday. By default, holiday is selected. This means, national holiday will be ignored by schedule filtering process. If you unmark holiday check box, workload will not be scheduled on the holiday, although that day is selected in both day-list and date-range.

National holiday is national specific calendar. Hence it can not be provided by zJOS-XDI. Before you can use holiday, you must build your national holiday table based on your national calendar. Otherwise, holiday filtering will always be ignored. See par 2.5 in chapter 2 of zJOS/Sekar® EMS User Guide manual for detail about preparing holiday table.

Unchecked holiday overrides week day list. This means, when holiday option is unchecked, holiday filtering will always be done regardless current week day is valid. For instance, if today is Wednesday and is checked, and holiday is unchecked, workload will not be scheduled unless today is not holiday.

Unchecked week day overrides holiday. This means, when a certain week day is unchecked, holiday filtering will not be done on that day. For instance, if today is Wednesday and is unchecked, workload will not be scheduled today regardless holiday.

3.2.3 Special Schedule Calendar

Many places have special culture on operation calendar. zJOS/Puspa is trying to adopt your culture by providing various special schedule calendar listed on workload detail panel as shown in figure 3.8. Calendar contains the following special days:

- IDAO – initial working day after off
- IWWD – initial week working day
- IMWD – initial month working day
- EMON – end of month
- FWWD – final week working day
- FMWD – final month working day

Only on	or	Except on
<input type="checkbox"/> IDAO	<input type="checkbox"/>	IDAO - Initial day after off
<input type="checkbox"/> IWWD	<input type="checkbox"/>	IWWD - Initial week working day
<input type="checkbox"/> IMWD	<input type="checkbox"/>	IMWD - Initial month working day
<input type="checkbox"/> FWWD	<input type="checkbox"/>	FWWD - Final week working day
<input type="checkbox"/> FMWD	<input type="checkbox"/>	FMWD - Final month working day
<input type="checkbox"/> EMON	<input type="checkbox"/>	EMON - End of month

Figure 3.8: Special schedule calendar.

This calendar features are optional, but when you select them, this calendar will override week day list and holiday filtering. For example, when IDAO is checked, the workload will only be scheduled on every initial working day after off day (Saturday, Sunday or holiday), regardless what you check on valid week day list and holiday options.

These calendars are accompanied by their exclusion which is also considered as special calendar. This means, schedule will be done normally except on checked special day. When that day is reached, workload will not be scheduled regardless what you checked on valid week day list and holiday options. For example, when FWWD exclusion option is checked, workload will be scheduled normally every day depend what you check on valid week day list and holiday options. When the final week working day is reached, however, workload will not be scheduled.

Special calendar and its exclusion are mutually exclusive. If any of special schedule calendars column are checked, you may not check anything in the exclusion column. The panel will prevent you to do so. Hence you must a tricky way when you need to combine special schedule calendar and its exclusion for a workload. For instance, for workload that only runs on IDAO except when fall on the same day with IMWD, you can not just select IDAO and exclusion of IMWD.

To solve such case, you need a dummy workload to be scheduled everyday except on IMWD. Then, schedule the real workload only on IDAO and place it as successor of dummy workload. Hence, when IDAO is reached, the real workload is triggered by dummy workload except when that day is also IMWD.

If you prefer not to use dummy workload, you need to insert a rexx program step on top of the workload (job or STC) to evaluate whether that day is IMWD using zJOS provided function zjcal(). If zjcal('IMWD') return 'Y', produce a certain return code to abort subsequent steps by using COND= keyword on second step. Then schedule the workload only on IDAO.

If you prefer not to user COND= keyword which need effort to modify each step, you may issue CANCEL command within inserted rexx program. The following code example shows how simple your rexx should be.

```

If zjcal('IMWD') = 'Y' then,
  X = zjcmd('CANCEL jobname')

```



Implementing complex scheduling mechanism might need you to be smarter and more creative, instead of just specifying entry definition. Sometime you need to combine dummy workload and inserted rexx program steps, or even rexx program in separate jobs. zJOS provides some useful rexx functions to help you code very simple rexx programs to do a big matter on scheduling mechanism. You can issue console message (zjwto() function) for triggering as well as WTOR (zjwto() function) or even “wait for TOD” (zjwait() function) to delay the workload. Be aware, unless zJOS/Sekar is used to perform automatic reply, using WTOR could break automation concept as it need human intervention to reply. If you don't license Sekar, to keep everything automatic, you got to code another rexx program to capture WTOR message by using zjset() function which need to be brought up prior to WTOR message occurrence.

IDAO – initial working day after off

IDAO is the first working day either following week-end (Saturday and Sunday) or holiday. Commonly, Monday is an IDAO, except holiday. If Monday is a holiday, Tuesday is then becomes IDAO. If Wednesday is also a holiday, then Thursday is considered as another IDAO. After holiday must be an IDAO except for Friday which is followed by week-end.

IWWD – initial week working day

IWWD is the first working day of week. Unless holiday, Monday will always be an IWWD. Unlike IDAO which can be happened more than once a week, IWWD always happens once a week. The first IDAO in a week is an IWWD. If there is a second IDAO, it will not be considered as an IWWD.

IMWD – initial month working day

IMWD is the first working day of month. If the first day of month falls on Sunday, then IMWD falls on the third day (Monday) unless that day is holiday. If IMWD falls on Monday, IDAO and IWWD must also be fall on the same day.

EMON – end of month

EMON is the last day of month. For example, in October 2008, EMON fall on 31st day. EMON is the exact day which varies in each month and especially in February depends on whether the year is leap year.

FWWD – final week working day

FWWD is the last working day of week. Commonly, Friday is a FWWD unless holiday. When Friday is a holiday, then FWWD is shifted on Thursday. Like IWWD, FWWD also once a week.

FMWD – final month working day

FMWD is the last working day of month. You should not confuse with end-of-month day. Unlike EMON, FMWD will not happen on the off day (Saturday, Sunday or holiday). In some places, official month-end day is FMWD, instead of EMON.

3.2.4 Crossed-Date (Midnight) Handling

Normally batch operation is done in the night which possibly crosses between 2 days at 00:00:00. Workloads which run before 00:00:00 normally do not really matter since operation calendar still in effect as current calendar. Workloads, however, which run after 00:00:00 mean, run on the next day although still in the same operation calendar. In such case, you must carefully consider schedule characteristic of each workload. Some workloads may need to be consistently scheduled with operation calendar. For example, if current operation calendar is end-of-month (EMON), once 00:00:00 clock is reached, the day is no longer end-of-month. Nevertheless, current operation calendar is still for EMON. Workloads that are scheduled only for EMON must run although the actual day is no longer EMON. More over, EMON could be immediately followed by FMWD if the day next to EMON is working day. Hence, FWWD workloads could unexpectedly run if schedule is improperly setup.

To avoid unexpected schedule problem, you must consider using option to force zJOS/Puspa to override current calendar with operation calendar. This option is placed in scheduled-workload entry detail panel as shown in figure 3.6 and for clearer, is zoomed as shown in figure 3.9 below. In the panel stated “*calendar is verified once at the time schedule cycle is started*”.

The effect of this option is not general. Rather, it only effective for workload with this option checked. When this option is selected for a workload, zJOS/Puspa verifies calendar once at the time schedule cycle is started (.SCD START command is issued) and retains it as operation calendar for that workload.

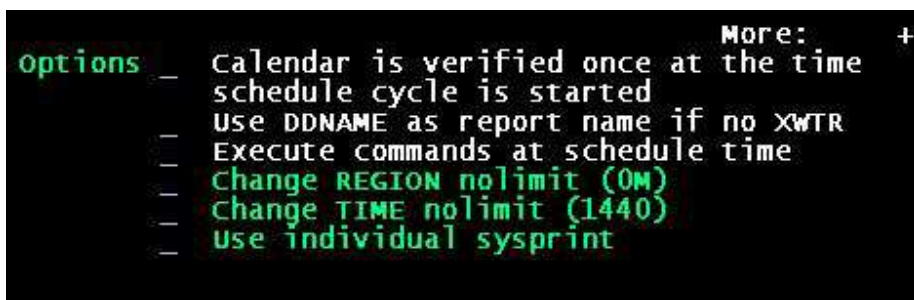


Figure 3.9: Option to force operation calendar effective during schedule cycle.



Workloads which are scheduled with holiday option unchecked or with selected week day list, or with special schedule calendars such as IDAO, IWWD, IMWD etc. or their exclusion, normally and logically need to be consistent with operation calendar. Hence, this option must be checked on. Avoiding this option could causes unexpected overlapping schedule.

3.2.5 Exception Handling

The last portion of scheduled-workload entry definition is exception handling. This offers chances you to ask Puspa performs certain action when an exception is encountered while a workload is being processed.

The perfect schedule should have complete exception flow for recovery on each scheduled workload. But, to prepare them, you need very long time to learn the behavior of each workload. To speed up scheduler setup, you need to prioritize complete exception flow only for certain workloads which already known unstable on day-to-day operation. The rest, which are supposed more stable workloads, might only need simple exception handling using this facility.

Exception handling facility works based on system and user condition codes. Any value of system and/or user condition codes can be used to decide whether workload got exception.

Exception definition

The first thing you should do, is defining exception itself. What condition you consider as an exception? For example, JOB01 is considered as exception when it is ended with CC > 16, as shown in figure 3.10. Note that CC is coded in hexadecimal notation.

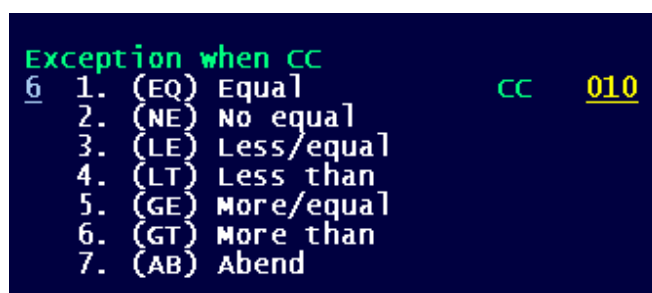


Figure 3.10: Considered as exception when CC > 16

CC for Abend exception

When you select 7 (AB) for exception definition, workload will be considered exception when abnormally ended. Without specifying CC value or fill it with 000 means that any abend code is considered as an exception, as shown in figure



3.11. However, if you specify a certain value in CC field, then only abend with that CC is considered as an exception, as shown in figure 3.12.

Note that abend code (system CC) is greater than user CC. Therefore, when you define exception with GE or GT operator, abend code will also comply the rule.

```
Exception when CC
7 1. (EQ) Equal          CC  000
   2. (NE) No equal
   3. (LE) Less/equal
   4. (LT) Less than
   5. (GE) More/equal
   6. (GT) More than
   7. (AB) Abend
```

Figure 3.11: Any ABEND is considered as exception

Figure 3.12 shows that you defined abend with system code 0C4 is an exception. Abend other than 0C4 will not trigger exception action.

```
Exception when CC
7 1. (EQ) Equal          CC  0c4
   2. (NE) No equal
   3. (LE) Less/equal
   4. (LT) Less than
   5. (GE) More/equal
   6. (GT) More than
   7. (AB) Abend
```

Figure 3.12: Only ABEND 0C4 is considered as exception

Simple action against exception

Once exception was defined, next is defining which action got to be taken when exception is encountered. Puspa offers you to choose one of 7 provided simple actions as shown in figure 3.13.

1. Normal schedule
2. Issue command
3. Refresh scheduler
4. Halt scheduler
5. Restart scheduler
6. Hold the workload
7. Cancel and hold the workload

Normal schedule (option 1) means no action. This is the default. Regardless exception definition you have prepared, no evaluation is performed. Schedule continues in normal way. If you have prepared complete schedule flows for a workload, you might not need simple action here, by leave its default as no action.

Be aware, handling exception here may override schedule flow for a workload. For example, workload JOB1 is set to trigger JOB2 when CC = 0 and trigger JOB3 when CC not 0, JOB3 will never be triggered if you define exception 2 (not equal) for CC = 0 with simple action hold (option 6) or hold and cancel (option 7). Because, when JOB1 ended with CC not 0, JOB1 is then placed in hold status and Puspa ask you to fix it. Workload in hold status will not trigger its successors until it is fixed and rescheduled and ended without exception.

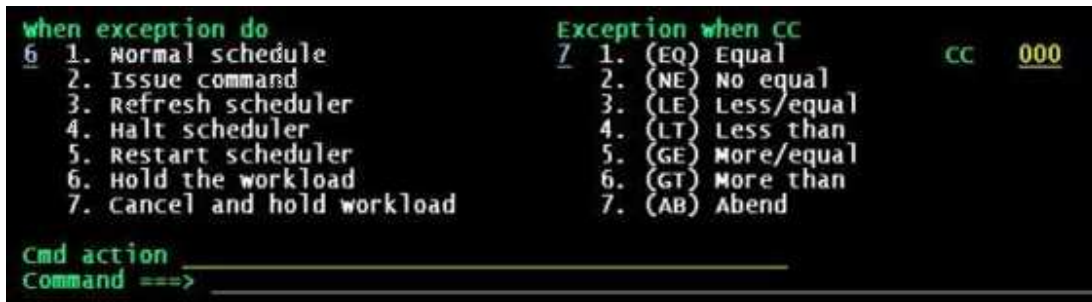


Figure 3.13: List of selectable simple actions against exception

Issue command (option 2) means specified text in *cmd action* field is issued as a command when exception is encountered. Figure 3.14 gives example that when workload got abend 0C1, is considered exception. Action is to send a message "JTEST03 abend 0C1" to userid IBMUSER.

This action option does not affect schedule flow as the workload is not placed in "error state". Schedule flow will still affected by returned system and/or user CC. If no successor is matched with returned CC, schedule flow of this workload then stuck without notification. Hence, command should something that alert user regarding this exception. Once recognized, user/programmer can fix it, then you can reschedule it by using provided rerun facility.

Note that such exception definition does not prevent its successors from being scheduled as long as returned CC is complying to trigger them. Use option 6 or 7 if you prefer to hold its schedule flow until it got fixed.

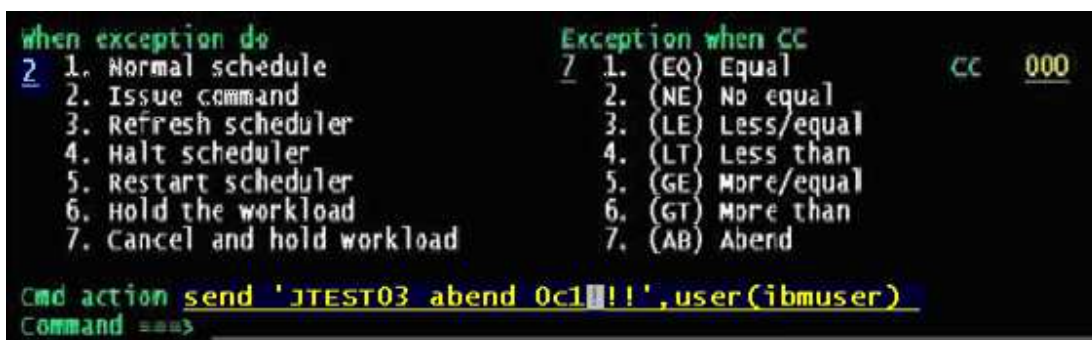


Figure 3.14: Command text must be specified when issue command action is selected

Refresh scheduler (option 3) means current active schedule table is refreshed when exception is encountered. This means, current scheduling process is terminated and cleanup all collected workloads status information, hence Puspa back to ready state. This is a very serious action. Unless you really expect such situation, you should never select this action.

Halt scheduler (option 4) means current scheduling process is halted when exception is encountered. This means, current scheduling process is suspended until you ask Puspa to resume the suspended workload or restart from a certain workload which has been finished. When it happens, operator console is notified that scheduling is halted.

Restart scheduler (option 5) means current active schedule table is refreshed and immediately start next schedule cycle when exception is encountered. This is more serious than refresh action (option 3). Unless you really expect such situation, you should never select this action.

Hold the workload (option 6) means to place this workload in “error state” and issue message to notify user to fix it. Error state doesn’t mean that workload is immediately terminated. Workload rather, continues on its own logic to finish. Puspa however, is no longer listening its progress status. . During in error state, schedule flow of this workload is suspended until its fixed version is rescheduled. When the workload is fixed, you can manually start it either in your own way or using “rerun” facility by typing ‘X’ in scheduler log panel. Once the workload is started, schedule flow of this workload is then resumed.

This option (6) is the most popular simple exception action, especially for newly inserted workload in the schedule table. Before you really familiar with behavior of the workload, this action is the best way to learn. Later when you its behavior is clearly known, you can setup alternate schedule flow for recovery.

Cancel and hold the workload (option 7) means cancel the workload and place it in “error state” and issue message to notify user to fix it like option 6. During in error state, schedule flow of this workload is suspended until its fixed version is rescheduled. When the workload is fixed, you can manually start it either in your own way or using “rerun” facility by typing ‘X’ in scheduler log panel. Once the workload is started, schedule flow of this workload is then resumed.

This option (7) is the second most popular simple exception action, especially for newly inserted workload in the schedule table. If you don’t need this workload to finish, use this option instead of option 6. Normally this option is useful for multi-step jobs or STCs to prevent them continue to finish all steps when got exception in early step.



3.3. Triggering Events Table

Triggering events table is a list of event definitions of each predecessor to trigger scheduling process. Normally, triggering events are events generated by some workloads to trigger their successor workloads. Figure 3.15 shows appearance of triggering event table for a workload. Panel only displays selected entries associated to selected workload. In this figure, panel only displays all trigger events associated to workload JTEST14 on system MFOC.

Triggering Events Table Row 39 from

Trigger group-id	Triggered job	JTEST14	on system	MFOC
Grp S	Trigger	System	Step name	Condition
00	JTEST03	J MFOC	Job:	check: EQ Spec CC: 000
00	JMSG001	M MFOC	Proc:	Text: IST0001A
01	JCMD001	C MFOC	Job:	check: EQ Spec CC: 000
02	JSTC001	S MFOC	Proc:	Text: V NET,ACT,ID
02	JRMT003	J MFOC	Job:	check: EQ Spec CC: 000
03	JCOBA03	J MFOC	Proc:	Text: EQ Spec CC: 000
03	JRMT005A	J MFOC	Job:	check: EQ Spec CC: 000
04	JUNNAH	J MFOC	Proc:	Text: EQ Spec CC: 000
04	JABC001	J MFOC	Job:	check: EQ Spec CC: 000
05	JUNNAH2	J MFOC	Proc:	Text: EQ Spec CC: 000
05	J55555	S MFOC	Job:	check: EQ Spec CC: 000
***** Bottom of data *****				

Figure 3.15: Triggering-event table panel

3.3.1 Managing Triggering Events Table

Figure 3.16 below shows brief description of each field of triggering event in predecessor list panel. Information displayed on panel of triggering event table describes triggering logic mechanism applied for a scheduled workload. Grp column describes triggers grouping indicated by 2-digit group id. Predecessors with the same group id are grouped to obtain a single group triggering decision. Condition column describes how each particular trigger event is generated.

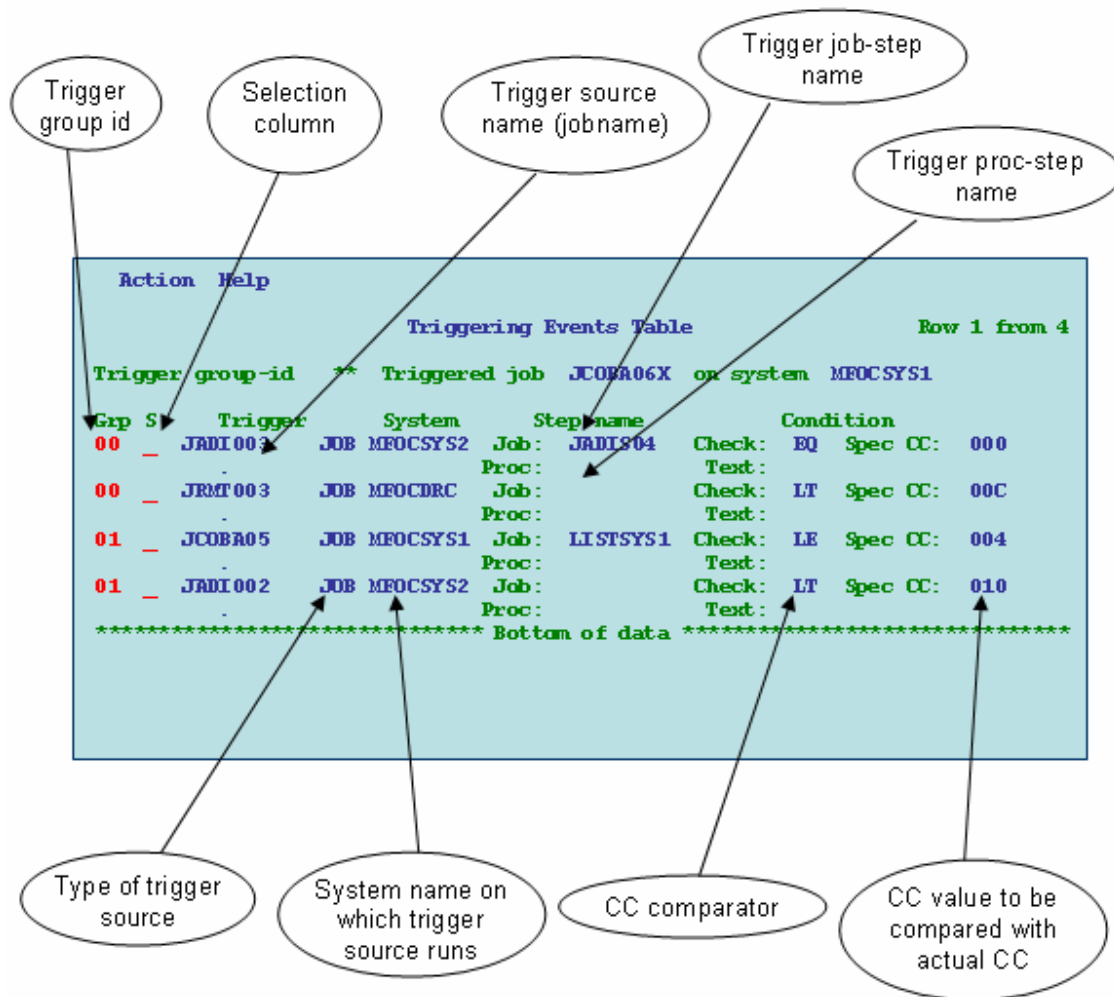


Figure 3.16: Managing triggering-event table

As all triggering events entries for all defined scheduled workload are contained in a single table, table is actually much larger than its appearance on the panel. However, panel only displays a portion of entries, which associated to selected workload. Besides, this makes you easier to manage and understand the logic of triggering mechanism applied to each workload.

The panel is scrollable. If the table is larger than screen size, you have to scroll it up and down to reach the position of entry you want to review. As standard ISPF convention, use F7 and F8 keys to navigate scrolling.

Selection (S) column

Second column of the table region on the panel is a selection column. This is the column of input fields into which you can enter action character to manage each entry of table. Valid action characters are:

3. Working with Scheduler Tables



- S – Select particular triggering event entry in detail as in figure 3.21.
 - This gives you chance to update the detail of selected event entry
- D – Delete particular entry from the table.
- A – Add a new event entry to the table.
 - Selected entry is ignored, then obtains detail event entry panel with all field blanks and ask you to fill up.
 - This can also be done from action-bar selection menu.

Action-bar

Action-bar menu consists of 2 selectable actions as shown in figure 3.17. Choice 1 is to add or insert a new entry. This function is similar as if you enter action character A in selection. However, when the table is empty, for example is newly created table, the only way to add the first entry is via this action-bar choice.

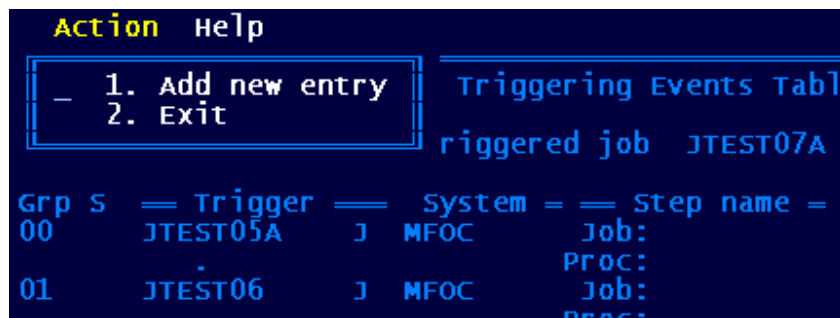


Figure 3.17: Action-bar menu of triggering-event table panel

Choice 2 is to exit from triggering event table panel and return to previous panel. This is similar as of you hit F3 key or issue END command. Changes on the table are saved prior to exit.

Function keys

Function keys consist of 5 keys available to navigate the panel:

- F1 – obtain help panel or window
- F3 – save and close the table
- F7 – scroll up screen
- F8 – scroll down screen
- F12 – abort all changes and close the table.

Grp column

This column lists triggering event group-id. To build triggering logic, triggers are grouped and each group identified by 2-digit numeric 00 to 99. Group-id is decided by you when entering a new trigger event source definition. Schedule for a workload is granted only if all associated triggering-event groups comply the

triggering condition. Compliance of each group is ANDed each other to produce final compliance. The final compliance is TRUE only if each group is TRUE.

Trigger column

Column labeled as trigger lists triggering source names and types. Triggering source name is name of source of event, which is used to trigger the scheduling process. Trigger source is normally other scheduled workload in the same schedule flow, which is a predecessor workload. However, it can also come from outside scheduler, which is an external workload.

Triggering source type is type of source. It can be batch-job (JOB), system-task (STC), command (CMD), message (MSG), dataset access (DSA) or dataset release (DSR). MSG, DSA and DSR source types always external workload, since they can not be scheduled as workload.

System column

Lists name of the system on which trigger event source is running, or trigger event is generated. See explanation on the par 3.1.

Stepname column

For trigger source type of batch-job and STC, Puspa supports 2 types of event for triggering, end-of-step (EOS) and end-of-job (EOJ) events. The use of EOS event in scheduling mechanism is the best way to achieve optimum scheduling path. It is known as step-level scheduling mechanism for pipelining operation.

EOS event is identified by job and step names. Whereas, EOJ event is identified by job name only. Hence, if step name is specified, means EOS event is used for triggering. Otherwise, EOJ event is used. There are 2 types of step names, job-step name for batch-job, and procedure step name for STC. Batch-job which calls procedure can have both step names.

Condition column

Compliance of each particular triggering source is evaluated based on condition yielded by either EOJ or EOS event. Condition evaluation is done by comparing returned condition-code (CC) and specified CC in trigger source definition. When comparison yields logical-TRUE, then triggering event is stated as complied.

Example

One of trigger source entry in figure 3.16 is job JADI003 step JADIS04 in group 00. It compliant only if jobstep JADIS04 of job JADI003 is ended with CC = 000.

3.3.2 Triggering Logic Mechanism

As a modern scheduling system solution, zJOS/Puspa provides almost unlimited complex triggering logic mechanism. Decision whether a workload is scheduled can be based on complex mathematical combination among condition-code (CC) resulted by all associated triggering event sources. Technically, trigger sources are grouped into a necessary number of groups.

Compliance of a trigger group is decided based on logical OR relationship among all individual triggering entries within this group. Hence, a group is considered complied only when at least one of triggering entries within the group is complied.

Final compliance is decided based on logical AND relationship among all groups. In other word, whole triggering is complied only when all triggering groups are complied. Such method is suitable to build either simple or complex triggering logic mechanism. Simple triggering mechanism can be established in either 2 way:

- ✓ OR-based condition evaluation -- no grouping
- ✓ AND-based condition evaluation

OR-based Condition Evaluation

OR-based condition is group triggering decision which is granted if at list one of particular trigger event comply triggering decision.

Action Help		Triggering Events Table				Row 2 from	
Trigger group-id		Triggered job		JTEST02		on system	
Grp S		Trigger		System		Step name	
00	JTEST01	J	MFOC	Job:	STPSATU	check:	LT Spec CC: 008
				Proc:		Text:	
00	JTEST00	J	MFOC	Job:	DUA	check:	EQ Spec CC: 000
				Proc:		Text:	

Figure 3.18: OR-based (simple) triggering mechanism

If you desire Puspa to grant a workload for scheduling based on at least one of triggering event entries, then you can easily just ungroup the entries. This means put all entries into a single group. Puspa then evaluate until one triggering event entry is complied. Once complied entry is encountered, Puspa then straightly grant the workload without waiting all other unfinished entries, unless schedule is constrained by TOD (e.g. when it happens earlier then specified start-time). Figure 3.18 shows example of OR-based triggering mechanism. Workload job JTEST02 is granted for scheduling if either workload job JTEST01 jobstep STPSATU comply CC < 8 or job JTEST00 jobstep DUA comply CC = 0.

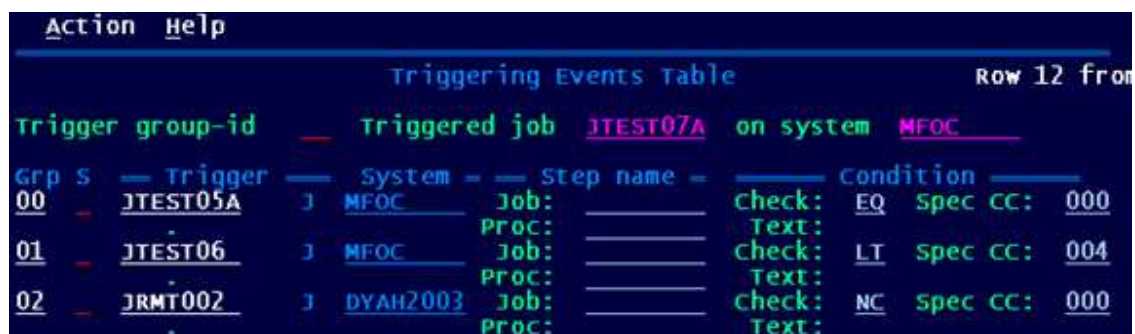


OR-based triggering mechanism can involve almost unlimited triggering entries. You can logically define 1 up to 32,767 triggering event entries for a single group. Hence, it sounds the only limit is storage capacity allocated for Puspa schedule table in DIV/dataspace.

AND-based Condition Evaluation

Other simple triggering mechanism is AND-based condition evaluation. If you desire Puspa to grant a workload for scheduling only if all defined triggering event entries are complied, you can easily just put each individual trigger entry into a separate group. This means each group only contains one entry. Puspa then evaluates all entries, and workload is granted only when all entries are complied.

Figure 3.19 shows an example of an AND-based triggering mechanism. This is also a simple triggering. Workload JTEST07A is granted for scheduling only if workload JTEST05A comply CC = 0 and JTEST06 comply CC < 4 and JRMT002 just ends. Note that CC for EOJ event is maximum CC.



Trigger group-id		Triggered job		on system		MFOC	
Grp S	Trigger	System	Step name	Condition			
00	JTEST05A	J	MFOC	Job:	Check:	EQ	Spec CC: 000
			Proc:		Text:		
01	JTEST06	J	MFOC	Job:	Check:	LT	Spec CC: 004
			Proc:		Text:		
02	JRMT002	J	DYAH2003	Job:	Check:	NC	Spec CC: 000
			Proc:		Text:		

Figure 3.19: AND-based (simple) triggering mechanism

Since group-id is 2-digit numeric from 00 up to 99, you can only define maximum 100 groups for a single scheduled-workload. Although 100 is a small number compared to 32,767, this number is more than enough to build multiple groups of trigger event table in most scheduling environment.

However, incase you need more than 100 triggering event entries; you can smartly simulate some dummy workloads as necessary. For example, JOB1 needs 150 triggering event entries (AND-based). You can then provide dummy workload JOB1A and JOB1B which each contains IEFBR14. Then you define first 100 triggering event entries for JOB1A, and the rest (50) for JOB1B. Finally, you define workload JOB1A and JOB1B as triggering entries for workload JOB1.

3.3.3 Complex Triggering Logic Mechanism

Action Help		Triggering Events Table					Row 1 fr
Trigger	group-id	**	Triggered job	JCOBA06X	on system	MFOCSYS1	
Grp S	Trigger		System	Step name	Condition		
00	JADI003	J	MFOCSYS2	Job: JADIS04	check: EQ	Spec CC: 000	
				Proc:	Text:		
00	JRMT003	J	MFOCDRC	Job:	check: LT	Spec CC: 00C	
				Proc:	Text:		
01	JCOBA05	J	MFOCSYS1	Job: LISTSYS1	check: LE	Spec CC: 004	
				Proc:	Text:		
01	JADI002	J	MFOCSYS2	Job:	check: LT	Spec CC: 010	
				Proc:	Text:		
***** Bottom of data *****							

Figure 3.20: Complex triggering mechanism

Triggering event table as shown in figure 3.20 describes complete triggering logic mechanism applied for a scheduled workload. Figure shows triggering logic for workload JCOBA06X on system MFOCSYS1. Table consists of 2 groups of triggering event sources:

- Group 00
 1. EOS event of job JADI003 step JADIS04 on system MFOCSYS2 → complied when CC = 000
 2. EOJ event of job JRMT003 on system MFOCDRC → complied when CC < 00C
 1. Compliance of group 00 = compliance of entry 1 OR 2
In math stated as: $(CC_{00,1} = 0) \text{ OR } (CC_{00,2} < 12)$
- Group 01
 1. EOS event of job JCOBA05 step LISTSYS1 on system MFOCSYS1 → complied when CC ≤ 004
 2. EOJ event of job JADI002 on system MFOCSYS2 → complied when CC < 010
 2. Compliance of group 01 = compliance of entry 1 OR 2
In math stated as: $(CC_{01,1} \leq 4) \text{ OR } (CC_{01,2} < 16)$

Final compliance = compliance of group 00 AND 01

Hence in mathematical notation, workload JCOBA06X on system MFOCSYS1 is scheduled only when:

$$((CC_{00,1} = 0) \text{ OR } (CC_{00,2} < 12)) \text{ AND } ((CC_{01,1} \leq 4) \text{ OR } (CC_{01,2} < 16)) = 1$$

Complex triggering mechanism can accommodate large amount of triggering entries. Based on standard zJOS programming logic, Puspa offers up to 100 groups, where each group can accommodate 32,767 entries. Hence total entries for a single workload are up to 3,276,700. This is a really huge number for such

purpose. However, incase you need more than 3,276,700 (although it sound impossible), you can smartly simulate some dummy workloads as predecessors as explained in previous par.

3.4. Triggering Events Entry

When you enter S in selection column of triggering events table as in figure 3.15 and press enter-key, selected entry is then displayed in new panel as shown in figure 3.21.

```

Triggering Event Detail

scheduled job name ....: JTTEST03  on system  BNIPROD

Trigger group-id ..... 1?

Trigger event/job name  JTTEST02  on system  BNIPROD
Trigger event/job type  1  1. (JOB) Batch-job
                           2. (STC) System-task
                           3. (MSG) Message
                           4. (CMD) Command
                           5. (DSA) Dataset access
                           6. (DSR) Dataset release

For batch=job or STC event type only -----
Job/STC step name ..... DUA003  Proc step
Match when actual CC    1  1. (EQ) Equal to          this
                           2. (NE) Not equal to       CC  004
                           3. (LE) Less/equal
                           4. (LT) Less than
                           5. (GE) More/equal
                           6. (GT) More than
                           7. (AB) Abend
                           8. (NC) Not checked

For MSG/CMD/DSA/DSR event type only -----
MSG/CMD text or DSN ...
For dataset access/release _ check jobname
  
```

Figure 3.21: Triggering event entry

The same panel is also displayed when you add/insert a new entry, except with empty fields.



Heading of the panel shows targeted scheduled workload name and system on which it is going to be scheduled. This to avoid you lost the track when updating the entry.

3.4.1 Triggering Event Identifier

Second part of triggering event entry panel is identifier of the entry, as shown in figure 3.22. It consist of group-id, event source name, name of system on which event source is running, type of event source and step names for event source type of batch-job and STC only. Description of each field already explained in par 3.3.



Figure 3.22: Triggering event entry identifier

Trigger Group-id

Two-digit numeric from 00 up to 99 specifies into which group this event entry is assigned. To alter it, you can just overwrite it.

Trigger event name

Name of triggering-event source, which depend on the type of source. For batch-job (JOB) and STC status events, this name must be its valid jobname. Event will only be captured from JOB or STC with name match to the specified name here.

For dataset access (DSA) or dataset release (DSR) events, name can be name of JOB or STC of which accessing or freeing specified dataset. By using such true name, DSA or DSR event originated from specified dataset name will only be captured when accessing or releasing workload name is matched with the specified name here. You must check on the *check jobname* checkbox option to notify Puspa that specified name is true name.

For DSA and DSR events, however, you can also specify any non-true name just to identify the event. By checking off the *check jobname* checkbox option, Puspa

will capture all DSA or DSR events from specified dataset name regardless the name of accessing or releasing workload.

For message (MSG) and command (CMD) events, this name is just a word to be used by Puspa internally to identify the event.

System name

Name of system on which triggering event source is running. See explanation on the par 3.1. To alter it, you can just overtype it.

Trigger event type

Puspa support 6 types of triggering event source; which are:

1. (JOB) Batch-job
2. (STC) System-task
3. (MSG) Message
4. (CMD) Command
5. (DSA) Dataset access
6. (DSR) Dataset release.

Batch-job, STC and command events can either originated from other scheduled workload, by means the predecessor, or from external workload which is outside scheduling system. Whereas message source must be external workload, since Puspa does not support workload type of message.

DSA and DSR are also external trigger event sources which are actually just system events instead of workload. DSA event occurs when dataset is accessed (allocated or opened). DSR event occurs when dataset is released (de-allocated or closed and freed). In certain cases, DSA and/or DSR can be used to simplify a schedule flow. For inter-workload dependencies that are caused by interleaved access to a dataset, the use of DSA and DSR may more efficient and simpler than jobstep based flow.

Job/STC step names

Step names applicable only for batch-job (JOB) and STC status event source types. The use of step names indicates the scheduling uses EOS event, instead of EOJ. This is a smart way to establish pipelining workload operation as explained in chapter 1 par 1.3 which can potentially shorten batch window time. There are 2 types of step names; job-step name and procedure-step name.

Job-step name is a name of EXEC card within a job stream which directly related to JOB card. Jobstep name is applicable only for batch-job and is the real step in batch processing flow.



Procedure-step or proc-step name is a name of EXEC card within a procedure stream which related directly to PROC card. Proc-step name is applicable for either STC or batch-job. Step in STC always identified by proc-step name and is the real step in task processing flow.

Step in batch-job can only be identified by jobstep or combined of both jobstep and proc-step. A batch-job which does not call procedure always uses jobstep only to identify its processing step. A batch-job which calls a procedure, uses combined of both jobstep and proc-step to identify its processing step. You must aware such understanding prior to implement EOS-level pipelining workload scheduling.

MSG/CMD text or DSN

This field is applicable for MSG, CMD, DSA or DSR event types only. For MSG or CMD type of event, you must specify a string here which to be used by Puspa to capture message or command. The string can be full or part of message or command text. Comparison is done exactly based on length and content of string you specified here. Hence, don't put quotes unless the target is also quoted. No wild card is allowed. Although the target is longer then specified string, you do not need to worry as long as specified string represent a unique portion of the targeted message or command text. Trailing blank in this field will be ignored. Be aware that comparison is always in uppcase mode from the first character position.

For DSA and DSR event, this string must be targeted dataset name. Hence you must follow dataset naming rule. Specified dataset name here must be a full name of dataset, instead of portion and no quotes are allowed. No PDS member name is allowed. Maximum length of dataset name is 44. For VSAM dataset, use cluster name.

Chapter 4 Controlling Puspa

As a modern automatic scheduling system solution, Puspa has powerful facilities to manage system workload operation. Most of interaction activities which usually performed by operators can be handled by Puspa. Combined with automation system or event management (e.g. Sekar), automatic spool/report distribution (e.g. AutoXfer) and tape management system coupled with robotic feature, you would have fully automated system which drastically reduces human (operators) interventions. Though, it does not mean your system can totally operate itself. It will still need human interventions, at least to control the automation program (e.g. Puspa)

Controlling Puspa is quite simple. You only need to interact with zJOS address space via console or via TSO/ISPF interface. Later when you already familiar with zJOS-XDI, you might automate some control interactions using Sekar.

4.1. Status Information

When you issue ".STATUS" on console or just press enter on zJOS control panel in XDI session in TSO, zJOS operation status information is then displayed. On console, status information appears as follow:

```
Component- Stat- -Agent-- Tbl  Works -Usage-- #dayX
Sekar (EMS)  UP    ACT(SS) IN    00005 LICENSED none
Puspa (SCD)  DOWN INACTIVE OUT   00000 **DEMO**  ..?!
AutoXfer     UP    ACTIVE  IN    00000 **DEMO**  ..?!
Net-Server   DOWN INACTIVE N/A   00000 standard none
zJOS XDI statistics:
Config: SSN=XDI Load=LPA COM=0802A3A0 WSA=00C42F90
Subtasks: Major=009 EVX=000 SVR=000 SCD=000 Abn=000
Network agents: total=0000 active=0000 local=N/A
Network traffic: Snd=00000000 Rcv=00000000 Que=00000
JES I/F: Up=Y PIT=Y Conn=Y Irdr=Y FR(5=N,12=Y,22=N)
Queues: ARQs=00001 SQBs=00000 EOTs=00000 RMG=00000
State: NORMAL Parm: SYS=00 EMS=00 SCD=00 DEST=00
SCD: Lib=O O=EVXMS M=EVXMS Pos=EVALUATE-JMR EnQ=FREE
```

On TSO/ISPF XDI session, zJOS status information appears as in figure 2.1. Both are similar except for the following additional rows showing statistics of DIV utilization for Puspa (scheduler) which only applicable in zJOS control panel on TSO/ISPF XDI session:



```
SCD Free-pool: SCT=009588 TRG=0199479 EOT=0500000
SCD Used-pool: SCT=000412 TRG=0000521 EOT=0000000
SCD Curr-pool: SCT=000024 TRG=0000047 EOT=0000000
```

Status information consists of 2 major information areas:

1. Product status information
2. Statistic information

4.1.1 Products Status Information

Products status information is information regarding each specific zJOS-XDI product or component. There are 4 components bundled in zJOS-XDI package and run together in XDI address space:

- **Sekar** – event management system (zJOS product)
- **Puspa** – automatic scheduling system (zJOS product)
- **AutoXfer** – automatic spool distribution (XDI product)
- **Net-server** – socket server program (zJOS standard feature)

Products status information is shown as a simple table which is the first part of status information, as shown below:

Action Help		zJOS-XDI Control Panel						Row 1 to 13 of	
Command	Product	State	Table	suf	works	usage	Day		
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000005	LICENSED	none		
	Puspa (SCD)	<UP> READY	loaded	00	000000	**DEMO**	..?!		
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!		
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none		

Figure 4.1: Product status information

State column

The above information describes whole status of zJOS-XDI products. Status of Puspa is indicated by **reversed red** color. State column describe whether the product UP or DOWN. UP indicates the product is active, and DOWN indicates the product is inactive. You must activate the product if you want it work for you. Normally Puspa is automatically up and ready when you start zJOS address space. Incase Puspa not automatically up, issue **.SCD INIT** command or INIT request on control panel (figure 4.2) to make it up and ready.



Action Help	
Command	Product
init	Sekar (EMS)
	Puspa (SCD)
	AutoXfer
	Net Server

Figure 4.2: Initializing Puspa to bring it ready

Agent column

Agent column in control panel (figure 4.3) is shown as second sub-column under state column. It shows current internal agent status. When Puspa up, its internal agent may READY, ACTIVE or PASSIVE. When PUSPA down, internal agent is INACTIVE.

Action Help			
ZJOS-XDI Control Panel			
Command	Product	State	Table
	Sekar (EMS)	<UP> ACT (SSI)	loaded
	Puspa (SCD)	<UP> READY	loaded
	AutoXfer	DOWN INACTIVE	unloaded
	Net Server	DOWN INACTIVE	N/A

Figure 4.3: Internal agent status (2nd State column)

READY means Puspa is well prepared to perform scheduling activities. Tables are loaded, job status listeners are enabled its connection to JES is established. Any time you enter START, Puspa will start all scheduling process.

ACTIVE means that Puspa has been starting scheduling activities. Workloads scheduling is being progress. This state is triggered by START request. Puspa remains in active state until whole schedule cycle complete. However, it can be changed immediately back to ready state by issuing REFRESH request, and then current scheduling progress is terminated. Hence, you should do it, unless really want to abort current schedule process.

PASSIVE means that scheduling progress has been halted, either completely done or got serious problem. While in passive state, scheduler will not be able to back to active state until you refresh it. Once Puspa is refreshed, it then changes to ready state, and you can request it to start.



Table column

This column shows status of parameters table, which is indicated as “loaded” (in console: IN) or “unloaded” (in console: OUT). Loaded (IN) indicates schedule table is already loaded, and unloaded (OUT) indicates that table is not loaded yet or unsuccessfully loaded.

For Puspa, if status is UP, table must be IN. If you find Puspa status UP with table OUT, issue `.SCD RELOAD` command on console or RELOAD request on control panel (figure 4.4) and recheck the status. If table remains OUT, you must recheck Puspa parameters using XDI ISPF interface as explained in chapter 3. Make sure you have already prepared Sekar parameters. If all event table and all associated action tables have already been prepared, please recheck to make sure you address the 2-digit suffix currently assigned for Sekar table correctly in XDI system parameter. If table and its suffix are correct and you still got the same problem, try recycling Puspa. Issue `.SCD STOP`, then issue `.SCD INIT`. Please call XDI support personnel immediately if you find the problem persist.

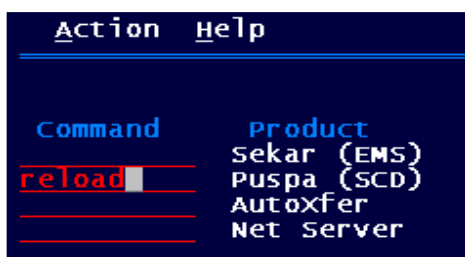


Figure 4.4: Request Puspa to reload schedule table

Work column

This column shows cumulative number of works on the current day since 0:00:00 clock. For Puspa, number of works represent number of workloads has been scheduled. It does not matter if Puspa is permanently or yearly licensed. It only impacts during demo period, where Puspa available for you only 30 works per day. When this work limit is reached, although shown remain up and active, Puspa will ignore all subsequent workload until next day.

Usage column

This column describes whether the product is in demo period or already licensed. ****DEMO**** indicates the product is in demo period and the key is expired and will be limited for 30 works per day. Ask XDI support personnel for renewal.

LCNSD/YR indicates the product is yearly licensed. In this kind of usage, you have to ask XDI representative personnel to renew the product key once a year.



LICENSED indicates the product is permanently licensed. In this kind of usage, you don't need product key anymore. Product will always available for you unless you change the hardware or system identifier. Hence, if you license XDI product permanently, you have to notify XDI representative personnel when you change your hardware or reconfigure you system.

Day column

On console appears as #dayX, is number of day's product key to expire. This is an important notice for yearly licensed usage only. If product is permanently licensed, this column is shown as 'none'. Non permanent licensed users should pay attention to this information. You will be warned when #dayX is less than 30 days.

4.1.2 Statistics Information

Statistics information is recorded statistics data regarding activities of each important zJOS-XDI task and/or routine in XDI address space.

```
Config: SSN=XDI Load=LPA COM=0802A3A0 WSA=00C42F90
Subtasks: Major=009 EVX=000 SVR=000 SCD=000 Abn=000
Network agents: total=0000 active=0000 local=N/A
Network traffic: Snd=00000000 Rcv=00000000 Que=00000
JES I/F: Up=Y PIT=Y Conn=Y IrdR=Y FR(5=N,12=Y,22=N)
Queues: ARQs=00001 SQBs=00000 EOTs=00000 RMG=00000
State: NORMAL Parm: SYS=00 EMS=00 SCD=00 DEST=00
SCD: Lib=O O=EVXMS M=EVXMS Pos=EVALUATE-JMR EnQ=FREE
```

Configuration line

This is actually not a statistic, instead, just show current internal configuration of zJOS-XDI in XDI address space. Shown in this line, subsystem name (SSI) for zJOS-XDI as assigned by SSN keyword in XDI procedure or in START command when XDI was started. On the above example shown SSI=XDI, which is the default.

COM and WSA show address of current communication and working storage area control blocks. These are shown here for debugging purposes only.

Subtasks line

These are statistics which describes current active zJOS subtasks within zJOS address space. Major=nnn shows number of major subtask which are currently



active. Normally zJOS is supported by 10 major subtasks when run on a single system, or 11 major subtasks when run on networked systems.

EVX=nnn shows number of active event executor minor subtask which is belong to Sekar (EMS). It can be tens or even hundreds depend on current workload. But, since most of EVX minor subtask is typically once work task, it up in very short time, hence you will find EVX looks likely always 0.

SVR=nnn shows number of active server's worker. When automation was setup for networked systems, zJOS server must up to handle connection with all agents from all connected hosts. Server is typically a concurrent socket server, which must able to interact with more than one agent at the same time. To do so, worker subtask is assigned for each connection. Hence, nnn here represent number of currently connected agents. Server is a major subtask of which main function is port listener. Whereas, server's worker is a minor subtask.

SCD=nnn shows number of active scheduler minor subtasks which is belong to Puspa. It can be tens or even hundreds depend on current workload. But, since most of SCD minor subtask is typically once work task, it up in very short time. Because there is only one SCD minor subtask which is assigned to be up along with scheduling activities, hence you will find SCD looks likely always 1 when Puspa is working.

Abn=nnn shows cumulative number of abended subtasks since zJOS address space was started. Each zJOS-XDI task and/or routine is accompanied with ESTAE type recovery handler. Hence, you should not worry with this indicator. It just for XDI supports personnel to inform R & D site for future enhancement.

Net-agent line

This is statistics of network connection, which represent number of generated network connection control blocks (NETCCB). When Sekar EMS table is loaded, and when Puspa schedule table is loaded, number of non-local system names is recorded. When zJOS-XDI Server is activated, it then generates NETCCB, one for each non-local system. Total number of generated NETCCBs is shown as total=nnnn. Active=nnnn shows number of NETCCBs currently being used for agent connection. Hence, active=nnnn represent number of currently connected agents. It must also the same as shown in SVR=nnn in subtask line.

Network traffic line

This is statistics of server-agent interaction. Each transaction accompanied by network access control block (NACCB) to hold send/receive control status and data being sent or received. R=nnnnnnnn (on control panel: Rcv=nnnnnnnn) shows number NACCB received by server. S=nnnnnnnn (on control panel



Snd=nnnnnnnn) shows number of NACCB sent to agents. Q=nnnn (on control panel: Que=nnnn) shows number of NACCB which still in queue for service.

Queues line

This is actually workload statistics. In normal situation, all of these queues are zeroes, which means all workloads are processed instantly. When automation workloads are too high, for example too many events are being processed (for Sekar), and/or too many jobs are being scheduled (for Puspa), then might some workloads must be queued.

Such situation can also happen when system too busy. For example, in peak time when the system is overloaded, all tasks are slowing down, including XDI address space. Hence, it impacts zJOS work slower, which is causing some automation workloads must be queued.

ARQ=nnnn shows number of action request queue blocks, which represents number of currently queued action requests. This represents Sekar performance.

SQB=nnnn shows number of scheduler queue block, which represents number of currently queued schedule requests. This represents Puspa performance.

EOT=nnnn shows number of end-of-task event blocks, which represents number of currently queued job status events. This represents Puspa performance.

RMG=nnnn is for XDI internal R & D only.

SCD trace: O=EVX M=EVX Pos=ASID-STACK EnQ=FREE

JES I/F line

This line shows JES (currently only support JES2) interface indicators, which describes readiness of Puspa-JES interoperability. Up=(Y or N) indicates JES state. PIT=(Y or N) indicates whether initiators are captured by Puspa. Conn=(Y or N) indicates whether JES-Puspa connection is established. Ildr=(Y or N) indicates whether JES internal reader is being allocated by Puspa. The rest are for XDI internal R & D use only. All are managed by Puspa internal agent. When Up=N, then all other indicators will be N.

Puspa is ready only when all indicators are Y. Otherwise, you must check whether JES is not up. Bring JES up if so. If Up=Y (JES up) and any one of other indicators is N, try stop Puspa and reinitialize it again. If problem persist, please call XDI support personnel.



SCD trace line

This is scheduler trace information, which is for XDI internal R & D only. When you report problem with scheduler (Puspa) to XDI supports personnel, this trace information should be reported as well.

4.2. Load Schedule Tables

Since DIV has been implemented for schedule tables, the way to manage tables was changed. INIT, LOAD and RELOAD requests got different effect to Puspa. In the past, INIT was to load currently addressed table and get Puspa ready for work. LOAD was to just bring selected table onto memory and RELOAD was just to reload currently addressed table onto memory. All were actually moving table from PDS member on disk onto ECSA. The use of ECSA might affect the whole system performance when it does not well prepared, since schedule table can be very large.

As a modern scheduling system, Puspa should not put itself in risky situation, even, should accommodate new mainframe technology as maximum as possible. Puspa DIV was designed to avoid the use of ECSA and PDS. The use of ECSA, which is a system high sensitive area, is replaced by shared dataspace. For large schedule tables, this feature drastically reduces ECSA usage. Incase an unexpected crash is happened, theoretically it won't affect to the system.

Other positive side effect is reducing the use of PDS to keep schedule tables. As PDS is easy to access, keeping sensitive data in PDS is not very good. Since Puspa dataspace is DIV, which is backed up by VSAM LDS, you can strict people viewing it. Besides, schedule tables in DIV are already in memory format and the way I/O is performed uses paging I/O mechanism which is very efficient.

Since DIV has been implemented, you are directed to migrate your tables to DIV. Once a PDS member is loaded onto DIV, you will never able to reload it, unless you delete the loaded table on DIV first. For example, once XDISCD00 is loaded onto DIV as "in-memory" table SCD 00, it won't be reloaded when you issue INIT, LOAD or RELOAD request. Rather, Puspa just switch to address SCD 00 as a current table. The only ways to reload XDISCD00 is by deleting SCD 00 first, or rename it to; let say XDISCD01 and SCD 01 does not exist, then load it. The second load will be kept on DIV as table SCD 01.

Further managing Puspa tables is discussed in chapter 5.

4.3. Navigate Schedule Flow

Once a schedule is started, process is then kept go on until the last workloads are completely performed. Normally, when schedule flow has been perfectly prepared, you do not need to do anything. Once the flow is completed, Puspa is then entering passive state, is called a schedule cycle complete. Next cycle, you just refresh the table by issuing REFRESH request, then start it again by issuing START request.

In certain situation, however, you may need to halt scheduling progress. This may happen when some workloads were changed or new workloads were added in respond of business changes. For example, when a certain workload got wrong data, because operator forgot to restore newly updated databases from development system. You then decided to halt Puspa and give time to operators to restore the databases. Afterward, you then let Puspa either to continue from halted position (resume) or certain position you desire (restart).

4.3.1 Halting Schedule

To halt schedule flow, issue the following command:

```
.SCD HALT
```

On zJOS control panel, you can issue HALT request as shown in figure 4.5. You can also use short form request H.

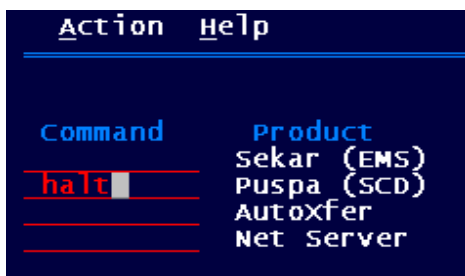


Figure 4.5: Request Puspa to halt schedule flow

In respond to HALT request, you will see 2 messages on the console as shown in figure 4.6. Message DERSCS346I occurs immediately following HALT request issuance. Puspa then takes some amount of time to find a workload which is currently being scheduled or going to be scheduled. Once the workload is found, message DERSCS347I occurs and schedule is then halted.



```
*11.32.26 S[C09591 *DERSCS346I scheduler is requested to be halted.
00 *11.33.26 STC09591 *DERSCS347I scheduler is halted while executing JADI001
```

Figure 4.6: Console messages in respond to HALT request

zJOS-XDI Control Panel							
Row 243 to 255 of							
Command	Product	State	Table	Suf	works	usage	Day
	Sekar (EMS)	<UP> ACT(SSI)	loaded	00	000005	LICENSED	none
	Puspa (SCD)	<UP> HALTED	loaded	00	000021	**DEMO**	..?!
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!
	Net Server	<UP> ACTIVE	N/A	**	000119	standard	none

Figure 4.7: Puspa is being halted

On zJOS control panel, status of Puspa is shown as HALTED as in figure 4.7. To halt schedule flow at a certain workload, issue the following command:

```
.SCD HALT JOB=jobname
```

On control panel, you can issue HALT request followed by jobname of workload you desire to halt as shown in figure 4.8. Since the command slot length is only 10 characters which can not fit the argument, you must use short form H.

Action Help	
Command	Product
h jobabcd	Sekar (EMS)
	Puspa (SCD)
	AutoXfer
	Net Server

Figure 4.8: Request Puspa to halt schedule flow at certain job

The workload you address by jobname (for example JOBABCD) must be exist in the schedule table. Otherwise, Puspa responds you as shown in figure 4.9.

```
Date      Time      Log
07.272    00:04:52  DERISC349w scheduler has no job JOBABCD
```

Figure 4.9: Response when requested workload does not exist

Halted schedule is a condition that Puspa is not doing scheduling. Workload at which the schedule process was started to halt is called halted workload. Halted workload is not actually halted. Rather, its status is not monitored by Puspa, as well as all its successors. All triggering mechanisms are disabled. No workload is scheduled afterward. All Puspa major functions are entering idle state. Such situation is then called as halted schedule.

4.3.2 Resuming Halted Schedule

Resuming halted schedule means let scheduler continue working. The starting point is always at halted workload. All triggering mechanisms are enabled for all its successors. Then schedule operation is completely resumed.

To resume the halted schedule flow, issue the following command:

```
.SCD RESUME
```

On zJOS control panel, you can issue RESUME request as shown in figure 4.10. You can also use short form request RES.

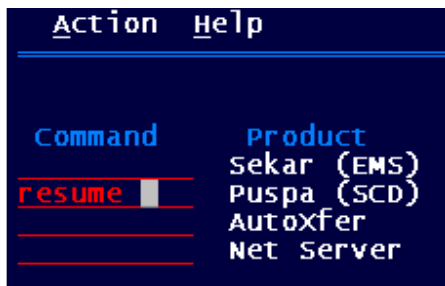


Figure 4.10: Request Puspa to resume halted schedule flow

4.3.3 Restarting Halted Schedule

Unlike resuming halted schedule, restarting halted schedule however, reactivate schedule not at the position it was halted. If you desire to reactivate schedule starting at a certain job or workload, then you must choose RESTART instead of RESUME. The workload you address as the starting point must be a “finished” workload. This means, the workload must be already terminated while schedule was halted. Unfinished workload or workload which is not scheduled yet, can not be used as starting point to restart halted schedule.

To restart the halted schedule at a certain workload, issue command:

```
.SCD RESTART JOB=jobname
```

On zJOS control panel, you can issue RESTART request followed by jobname of workload you desire to halt as shown in figure 4.11. Since the slot length is only 10 characters which can not fit the argument, you must use short form R.



Action	Help
zJOS-XDI Control Panel	
Command	Product State
r jobxyz	Sekar (EMS) <UP> ACT(SS1)
	Puspa (SCD) <UP> HALTED
	AutoXfer DOWN INACTIVE
	Net Server <UP> ACTIVE

Figure 4.11: Request Puspa to restart halted schedule flow at certain job

The workload you address by jobname (for example JOBXYZ) must be exist in the schedule table. Otherwise, Puspa responds you as shown in figure 4.12.

Action	Help						
zJOS-XDI Control Panel							Row 14 to 14 of
Command	Product	State	Table	Suf	Works	Usage	Day
*RESTART	Sekar (EMS)	<UP> ACT(SS1)	loaded	00	000005	LICENSED	none
	Puspa (SCD)	<UP> HALTED	loaded	00	000021	**DEMO**	..?!
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!
	Net Server	<UP> ACTIVE	N/A	**	000119	standard	none
Date Time Log							
07.271 14:23:32 DERISC349w scheduler has no job JOBXYZ							
***** bottom of data *****							

Figure 4.12: Response when requested workload does not exist

4.4. Control Panel Commands

4.4.1 LQ - Shows ENQued Processes

Schedule tables are loaded in dataspace which is backed up by VSAM LDS by using DIV technology. As scheduling process is internally done by several tasks asynchronously, to guarantee the content of the tables safe, each access to the table is tightly controlled. All read-only access can be done together at the same time and controlled as non-exclusive sharing mode. Write access, however, can only be done serially, which is controlled as exclusive sharing mode.

Some other shareable resources are also handled in the same way. Resources which applicable only within zJOS address space are controlled locally. The others which are shared as system level between zJOS address space, internal agents and subsystem routines are controlled at system wide level.



Use LQ command to check whether one or more processes are enqueued for a certain resource. Figure 4.13 shows the response when LQ command is invoked. As it was carefully designed, in normal condition, there should no task placed in queue for significant time interval. When a queue is significantly held, major and minor names of resource are displayed.

```

Action Help
zJOS-XDI Control Panel Row 46 to 46 of 46

Command Product State Table Suf works Usage Day
Sekar (EMS) <UP> ACT(SS) loaded 00 000005 LCNSD/YR 1087
Puspa (SCD) <UP> ACTIVE loaded 00 000045 LCNSD/YR 1087
AutoXfer DOWN INACTIVE unloaded 00 000000 LCNSD/YR 1087
Net Server DOWN INACTIVE N/A 00 000000 standard none

Date Time Log
08.275 01:04:23 No held ENqueue/lock was detected.
***** Bottom of data *****

```

Figure 4.13: Sample of LQ command response.

Held queues for schedule table may indicate that some pointer fields are broken. To avoid such case, make sure each schedule table is well organized.

4.4.2 CAL - Shows Current Month Calendar

```

Action Help
zJOS-XDI Control Panel Row 18 to 25 of 25

Command Product State Table Suf works Usage Day
Sekar (EMS) <UP> ACT(SS) loaded 00 000005 LCNSD/YR 1087
Puspa (SCD) <UP> ACTIVE loaded 00 000045 LCNSD/YR 1087
AutoXfer DOWN INACTIVE unloaded 00 000000 LCNSD/YR 1087
Net Server DOWN INACTIVE N/A 00 000000 standard none

Date Time Log
08.275 01:02:19 Current Month Calendar:
08.275 01:02:19 Today is initial month working day.
08.275 01:02:19 The nearest holiday this month is next 01 days.
08.275 01:02:19 --> Thu 2008/oct/02 loro oktober
08.275 01:02:19 The final holiday this month is next 18 days.
08.275 01:02:19 --> Sun 2008/oct/19 Hari HUT zJOS Architect
08.275 01:02:19 Final week work day is next 02 days, on Fri 2008/oct/03
08.275 01:02:19 Final month work day is next 30 days, on Fri 2008/oct/31
***** Bottom of data *****

```

Figure 4.14: Sample of CAL command response.

In most cases, calendar is a key factor in automatic scheduling. Puspa uses calendar information as triggering decision factors, such as week day, holiday and special calendars (IDAO, IWWD etc.). Once holiday table is loaded, it then is combined with computer TOD calendar to produce day-to-day zJOS system calendar which is referred as operation calendar. You can check current month calendar by using CAL command in control panel as shown in figure 4.14.



4.4.3 HOL - Shows Holiday Calendar

HOL command is to display currently loaded holiday calendar in control panel. Use this command make sure that holiday table you have prepared in zJOS parameter library is correctly loaded.

Action

Help

zJOS-XDI Control Panel

Row 18 to 29 of 29

Command	Product	State	Table	Suf	Works	usage	Day
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LCNSD/YR	1067
	Puspa (SCD)	<UP> READY	loaded	00	000000	LCNSD/YR	1067
	Autoxfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	1067
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none

Date

Time

Log

08.295

03:07:04

Holiday Calendar:

08.295

03:07:04

2008/04/28 Puspa Birthday

08.295

03:07:04

2008/05/16 sekar Birthday

08.295

03:07:04

2008/05/20 National Education Day

08.295

03:07:04

2008/06/01 Pancasila Birthday

08.295

03:07:04

2008/08/17 Independent Day

08.295

03:07:04

2008/08/18 Pramuka Day

08.295

03:07:04

2008/09/30 September Movement Day (G30s)

08.295

03:07:04

2008/10/05 National Military Day

08.295

03:07:04

2008/10/19 zJOS Architectting Day

08.295

03:07:04

2008/11/10 National Warrior Day

08.295

03:07:04

2008/12/20 Mother Day

Bottom of data

Figure 4.15: Sample of HOL command response.



Chapter 5 Managing In-memory Schedule Tables

As mentioned in par 4.2 of chapter 4, Puspa uses DIV to load, keep and manage schedule tables. As DIV is actualize by paging facility as a memory, Puspa tables on DIV have characteristics as if they loaded on memory. To reach this facility, you can either via choice 3 of action-bar menu as shown in figure 5.1 or by entering PARM request as shown in figure 5.2, then stroke enter-key. A list of in-memory schedule tables is then obtained as shown in figure 5.3.

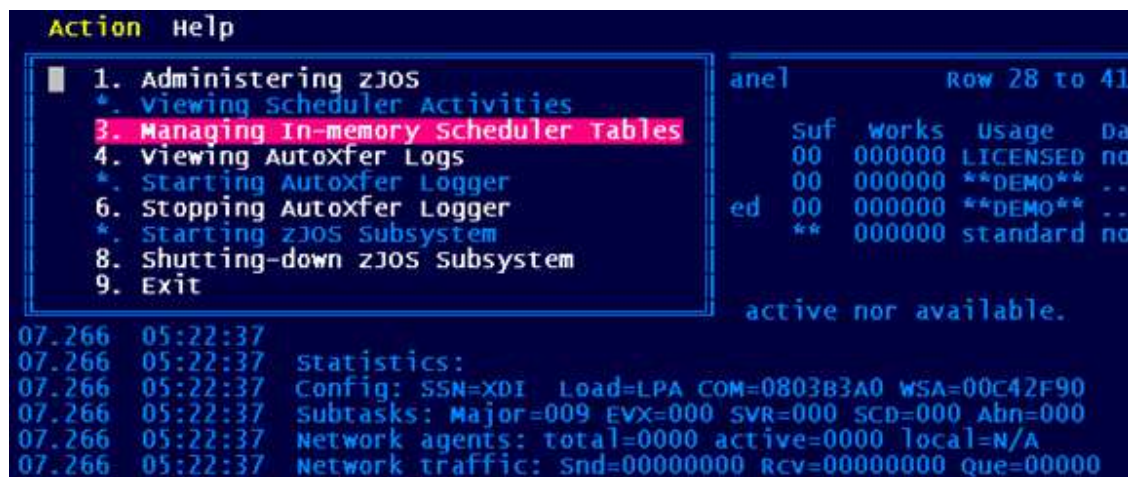


Figure 5.1: Entering in-memory schedule tables management via action-bar choice

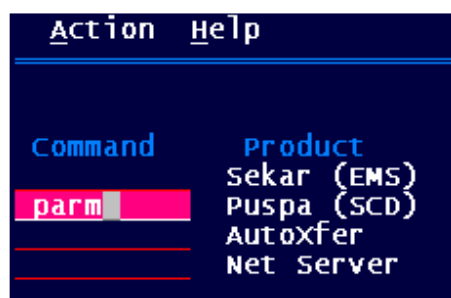


Figure 5.2: Entering in-memory schedule tables management via command request

Panel of scheduler table list (figure 5.3) was designed as shadowed window to primary control panel, where product status still appears as background, instead of heading.



Action Help		Scheduler Table List						
		Row 1 to 25 of 100						
Command	Product	State	Table	Suf	Works	Usage	Day	
PARM	Sekar (EMS)	<UP> ACT(SSI)	loaded	00	000000	LICENSED	none	
	Puspa (SCD)	<UP> READY	loaded	00	000000	**DEMO**	..?!	
	Autoxfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!	
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none	
Capacity: SCT=		10000	TRG=	200000	EOT=	500000	EVB=	100 ACT= 1000
FreePool: SCT=		9588	TRG=	199479	EOT=	500000	EVB=	100 ACT= 1000
		Statistics			Last Modified			
S	Id	Status	#wklds	#trigs	#bufs	Date	Time	Note
00		CURRENT	24	47	0	2007/09/10	19:20:40	
01			20	40	0	2007/08/26	03:55:58	
02			0	0	0			
03			20	40	0	2007/08/27	20:39:17	
04			0	0	0			
05			28	59	0	2007/08/28	14:49:38	
06			0	0	0			
07			0	0	0			
08			0	0	0			
09			0	0	0			
10			24	47	0	2007/08/28	14:57:04	
11			0	0	0			
12			0	0	0			
13			0	0	0			
14			0	0	0			
15			0	0	0			
16			0	0	0			
17			0	0	0			
18			0	0	0			
19			0	0	0			
20			0	0	0			
21			0	0	0			
22			0	0	0			
23			0	0	0			
24			0	0	0			
Command ==>		scroll ==> CSR						

Figure 5.3: List of in-memory schedule tables

5.1. DIV Capacity and Utilization

Major parts of panel are DIV global information and list of tables with statistics. DIV global information consists of DIV capacity and current utilization as shown in figure 5.4. This to inform you whether Puspa has already well tuned. The best situation is when utilization of SCT, TRG, EVB and ACT are averagely 75%.

Capacity: SCT=	10000	TRG=	200000	EOT=	500000	EVB=	100	ACT=	1000
FreePool: SCT=	9460	TRG=	199320	EOT=	500000	EVB=	100	ACT=	996

Figure 5.4: DIV capacity and current utilization



Higher than 75% is good only for settled system. For system which dynamically growth, above 75% could be a critical situation. Meanwhile, utilization too low indicates that the DIV is too large. It will affect in zJOS performance than requirement.

Puspa control blocks

Capacity record shows you the capacity of Puspa. It's depend on how you setup Puspa. There are 5 main Puspa control blocks to do an automatic workload scheduling system. SCT represents a scheduled workload. Each SCT holds one workload definition and operation status information. It very easy to estimate how many SCT you need.

TRG represents a trigger event source. Each TRG holds trigger event source definition and operation status information. Number of triggers for each workload is very relative, depend on how triggering mechanism you desire. Hence you need to study very seriously to have the optimum number of TRGs. It must not less than you need. But, if it too large, will affect system performance.

EOT represents a job step. It is actually not one-to-one correlated to the number of steps. EOT is actually a zJOS control block to hold end-of-task event status information. But it is not placed on hold status, unless captured event is an end-of-jobstep (EOS) event which comes from any source that associated with Puspa (scheduler).

EVB represent a non-job event. It actually belongs to Sekar which is an event management system (EMS), instead of Puspa.

ACT represents an action. It is actually belong to Sekar to hold action definition and status information against an event. Since Puspa also has some action definition, for example, against exception condition of a certain workload, so Puspa could be need some ACTs. Puspa also use ACT to hold message and/or command text of message/command type workload or trigger event source.

By default, zJOS is distributed with 10,000 SCTs, 200,000 TRGs, 500,000 EOTs, 100 EVBs and 1,000 ACTs. Total around 100 MBs space and expected suitable to accommodate 10,000 scheduled workloads. You can customize this capacity specification to accommodate your specific requirement.

5.2. Tables Statistics

Next part of the panel is schedule tables statistics. It lists all tables, from 00 up to 99. Each table is represented in one row, which consist of ID, status, number of loaded workloads (#wklds) definitions, number of triggers (#trigs) definitions,



number of several attached buffers (#buffs), and last modified time stamp. Blank row indicates that associated table is empty.

S	Id	Status	Statistics			Last Modified		Note
			#wklds	#trigs	#buffs	Date	Time	
00		CURRENT	24	47	0	2007/09/10	19:20:40	
01			20	40	0	2007/08/26	03:55:58	
02			0	0	0			
03			20	40	0	2007/08/27	20:39:17	
04			0	0	0			
05			28	59	0	2007/09/23	05:31:17	
06			0	0	0			
07			28	59	0	2007/09/23	05:30:08	
08			0	0	0			
09			0	0	0			
10			24	47	0	2007/08/28	14:57:04	

Figure 5.5: Statistics of in-memory schedule tables

Although each SCT represents a workload definition, #wklds appears on the list does not represent exact number of allocated SCTs. Instead, only number of SCTs which are really contained workload definition. Puspa allocates spare at least 10% of number of SCTs to anticipate future update.

The above rule is also applied for TRGs allocation. #trig is number of TRGs which are really contained triggering-event source definition. Puspa allocates spare at least 10% of number of TRGs to anticipate future update.

However, when you load new table form XDISCDxx member, Puspa just load it as it is. Puspa allocates number of SCTs and TRGs exactly as much as number of workloads and trigger-event definition in XDISCDxx member. Hence, you can not add nor insert new definitions. To have spare SCTs and TRGs, you should either reorganize the table or copy it onto new table. Copy of the table will have at least 10% SCTs and TRGs spare.

5.3. Managing Tables

Table which status as CURRENT means currently addressed by Puspa for scheduling. When you start scheduling, Puspa will select CURRENT table for work. Status of CURRENT will remain until you change 2-digit suffix on primary control panel.

Figure 5.6 shows brief description of each field and/or column of table list panel. Description can also be found on help panel by pressing F1 key with cursor anywhere except in selection column, which is the only F1-key sensitive area. Help panel as shown in figure 5.9 explains each field and/or column more detail.

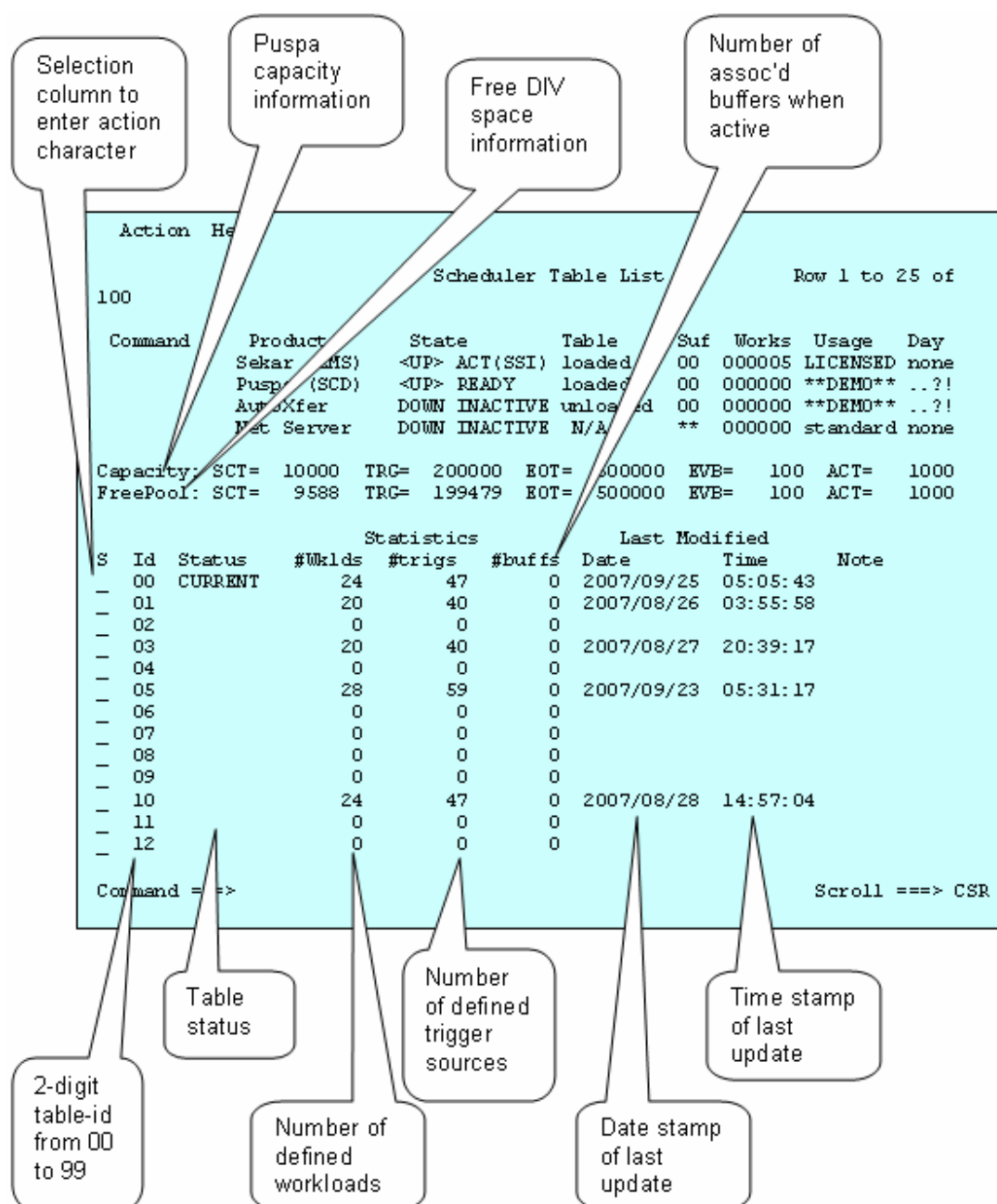


Figure 5.6: Managing tables

Selection column

Lists input fields on which you may enter action character. Puspa provides 3 action characters for you to manage your in-memory schedule tables. When you place cursor on this column and press F1-key, a message then appears as in figure 5.7 hint you what to do as follow:



S	Id	Status	Statistics			Last Modified		Note
			#wkl ds	#trigs	#buffs	Date	Time	
	00	CURRENT	24	47	0	2007/09/25	05:05:43	
Enter S=Show, R=Reorganize D=Delete selected schedule table.								
	04		0	0	0			
	05		28	59	0	2007/09/23	05:31:17	
	06		0	0	0			

Figure 5.7: Message explaining selection column

- S → to display a selected table, as shown in figure 5.11.
 - This gives you chance to do further with a table.
- R → to reorganize a selected table.
 - When a table has been intensively updated, its inter-link structure can be less optimal for operation. SCT and TRG spare may shortage. In such situation, you need to reorganize the table.
 - When a table is reorganized, some SCT and TRG spare may added as necessary.
- D → to delete a selected table.

Be aware once a table is deleted, there is no way to restore it back. You have to shutdown zJOS and restore its VSAM LDS to get the deleted table back. Confirmation menu will be displayed following your selection progress to remind you what you are doing. See figure 5.8 for the appearance of confirmation menu.

Action Help		Scheduler Table List							Row 1 to 25 of	
Command	Product	State	Table	Suf	Works	Usage	Day			
	SEKAR (EMS)	<UP> ACT(SST)	LOADED	00	000005	LCNSD/YR	1087			
	PUSPA (SCD)	<UP> ACTIVE	LOADED	00	000045	LCNSD/YR	1087			
	AUTOXFER	DOWN INACTIVE	UNLOADED	00	000000	LCNSD/YR	1087			
	NET SERVER	DOWN INACTIVE	N/A	**	000000	STANDARD	NONE			
Capacity: SCT=		3000	TRG=	60000	EOT=	60000	EVB=	100	ACT= 1000	
FreePool: SCT=								100	ACT= 1000	
zJOS-XDI 2.1.3										
Table Selection										
S	Id	Status	Schedule 03							ed
	00	ACTIVE	Select							me Note
	01		1. Show detail							:33:09
	02		2. Reorganize							
/	03		3. Delete table							:31:21
	04		cmd ==>							
	05									
	06									
	07		0	0	0					
	08		0	0	0					
	09		0	0	0					
	10		0	0	0					

Figure 5.8: Confirmation menu for table selection.



5.3.1 Obtaining Helps

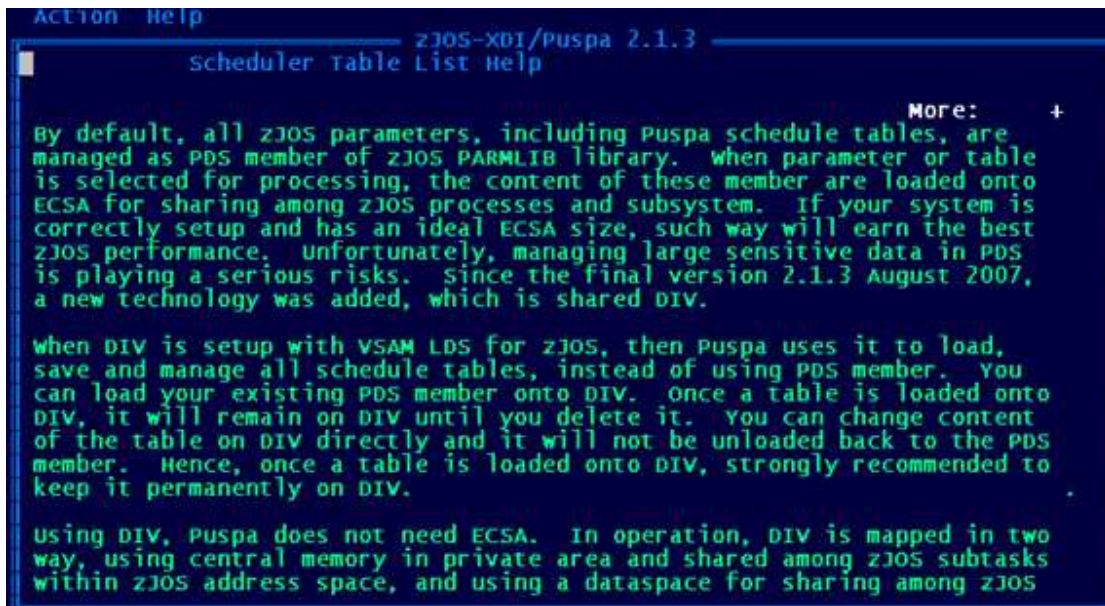


Figure 5.9: Help panel for table list

Help panel is provided to explain in more detail each field and/or column on the table list panel. To obtain help, you can either use help-bar or by hitting F1-key. Figure 5.9 shows appearance of help window.

5.3.2 Selecting and Updating a Schedule Table

		Statistics			Last Modified	
S	Id	Status	#wklDs	#trigs	#buffs	Date Time
—	00	CURRENT	24	47	0	2007/09/25 05:05:43
—	01		20	40	0	2007/08/26 03:55:58
—	02		0	0	0	
—	03		20	40	0	2007/08/27 20:39:17
—	04		0	0	0	
S	05		28	59	0	2007/09/23 05:31:17
—	06		0	0	0	
—	07		0	0	0	
—	08		0	0	0	
—	09		0	0	0	
—	10		24	47	0	2007/08/28 14:57:04
—	11		0	0	0	

Figure 5.10: Selecting a table

By typing action character S on selection column (figure 5.10) and then hit enter-key, next panel is then obtained, which displaying the selected table as shown in figure 5.11. Most of management tasks that you can do with source table as discussed in chapter 3, are applicable for in-memory table here. See chapter 3

for some more review. The only different is, there are some additional functions which applicable only for in-memory table:

- Copy table → duplicate content of table to other (empty) table
- Bypass → flags a workload to be bypassed from schedule process
- Hold → flags a workload to be held during schedule process
- Delete → flags a workload as deleted

The other different, which is very principle, that the whole database is floated on the memory for performance reason. So you are directly working with memory, instead of dataset and all updates will immediately effective. Saving to dataset (VSAM LDS) is just to make them permanent.

Action Help		Scheduled-workload Table						Row 1 to 17 of 28	
Schedule ID	Copy to ID	System	Date	Time	Day	SMTWTFSS	Msg		
5 - workload -									
JTEST00	J	MFOC	Start	2006/01/01	00:00:00	List	1111111		
			End	2007/12/25	23:10:00	Holiday	1		
JTEST01	J	MFOC	Start	2006/01/01	00:05:00	List	1111111		
			End	2007/12/25	23:15:00	Holiday	1		
JTEST02	J	MFOC	Start	****/01/01	00:10:00	List	1111111		
			End	****/08/17	23:30:00	Holiday	1		
JTEST03	J	MFOC	Start	2006/01/01	00:30:00	List	1111111		
			End	2007/12/25	23:00:00	Holiday	1		
JTEST04	J	MFOC	Start	2006/01/01	00:45:00	List	1111111		
			End	2007/12/25	23:50:00	Holiday	1		
JTEST05A	J	MFOC	Start	2006/**/01	00:50:00	List	1111111		
			End	2007/**/25	23:59:00	Holiday	1		
JTEST06	J	MFOC	Start	2006/01/01	00:30:00	List	1111111		
			End	2007/12/25	24:00:00	Holiday	1		
JTEST07A	J	MFOC	Start	2006/01/01	00:30:00	List	1111111		
			End	2007/12/25	24:00:00	Holiday	1		
JCOBA01	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JCOBA02	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JCOBA03	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JCOBA04	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JCOBA05	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JCOBA06X	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
TSORXCLT	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
TSORXCL1	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
TSORXCL2	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
Command ==> █								Scroll ==> CSR	

Figure 5.11: Panel displays in-memory schedule table

The changes you have made here won't affect the original source table. For example, if you are working with schedule table 01, then when you finish, all updates are not affecting your original XDISCD01. There is no way to restore back from in-memory table to XDISCDnn member. Hence, you are strongly recommended to keep daily backup of your DIV LDS.

5.3.3 In-memory Table Internal Structure

In-memory table is built from 2 types of control block, SCT and TRG. Each SCT holds a workload information, and each TRG holds a triggering event information for used by all related zJOS subtasks, tasks and routines in either address space or subsystem scope. Newly loaded table does not have spare SCT and TRG for new entries. Hence, you can not add entry until you reorganize the table.

Besides the status information, some fields in SCT and TRG contain pointers to build logic structure of schedule table by interconnecting among SCTs, among TRGs and between SCT and TRG. The way scheduling and triggering processes are performed is absolutely based on these pointers. Therefore, these pointers must never be broken.

5.3.4 Copying a Table

Action Help		Scheduled-workload Table						Row 1 to 1
S	= workload =	System =	Date	Time	Day	SMTWTFSS		
-	JTEST00	J	MFOC	Start	2006/01/01	00:00:00	List	1111111
-				End	2007/12/25	23:10:00	Holiday	1
-	JTEST01	J	MFOC	Start	2006/01/01	00:05:00	List	1111111
-				End	2007/12/25	23:15:00	Holiday	1
-	JTEST02	J	MFOC	Start	****/01/01	00:10:00	List	1111111
-				End	****/08/17	23:30:00	Holiday	1
-	JTEST03	J	MFOC	Start	2006/01/01	00:30:00	List	1111111
-				End	2007/12/25	23:00:00	Holiday	1

Figure 5.12: Slot field to address table id targeted for copy

Copy table facility is only provided for in-memory table. This is to duplicate the content of a table to other table. Targeted table must be an empty table. To do this, first you must select and open the table you are going to copy. Type S on selection column of table list panel, then hit enter-key. When table is obtained, you will see on the heading an empty slot labeled "Copy to ID" as shown in figure 5.12 (highlighted). Fill it with targeted table id for duplication. For example, as shown in figure 5.13, table 07 is targeted to contain the duplicate of table 05. Note that table 07 must be an empty table. Otherwise, copy request is rejected. Then, you should not hit enter-key, rather, click the action-bar menu and select choice 2 (copy to specified target), as shown in figure 5.14. Once you hit enter-key, then duplication is proceed. When copy is completed, a message explaining number of duplicated SCTs and TRGs appears on the bottom of the panel as shown in figure 5.15.



Action Help		Scheduled-workload Table						Row 1 to 1
Schedule ID 05 Copy to ID 07								
S	= workload =	System =		Date	= Time =	Day =	SMTWTFS	
JTEST00	J	MFOC	Start	2006/01/01	00:00:00	List	1111111	
			End	2007/12/25	23:10:00	Holiday	1	
JTEST01	J	MFOC	Start	2006/01/01	00:05:00	List	1111111	
			End	2007/12/25	23:15:00	Holiday	1	
JTEST02	J	MFOC	Start	****/01/01	00:10:00	List	1111111	
			End	****/08/17	23:30:00	Holiday	1	
JTEST03	J	MFOC	Start	2006/01/01	00:30:00	List	1111111	
			End	2007/12/25	23:00:00	Holiday	1	

Figure 5.13: Table 07 is targeted to contain the duplicate of table 05

Action Help		d-workload Table						Row 1 to 17
1. Add new entry 2. Copy to specified target 3. Exit								
JTEST00	J	MFOC	Start	2006/01/01	00:00:00	List	1111111	
			End	2007/12/25	23:10:00	Holiday	1	
JTEST01	J	MFOC	Start	2006/01/01	00:05:00	List	1111111	
			End	2007/12/25	23:15:00	Holiday	1	
JTEST02	J	MFOC	Start	****/01/01	00:10:00	List	1111111	
			End	****/08/17	23:30:00	Holiday	1	

Figure 5.14: Select action-bar choice 2 to proceed copy to targeted table

TSORXCL1	J	MFOC	Start	****/01/01	00:00:00	List	1111111	
			End	****/01/01	24:00:00	Holiday	1	
TSORX								1
00028 SCTs and 000059 TRGs were saved to schedule ID 07								
TSORX								1
Command ==> Scroll ==> CSR								

Figure 5.15: Message appears when copy process is completed

When you return to table list panel, you will see table 07 is not empty any more, as shown in figure 5.16. It was just updated with the content of table 05.

When a table is copied, the copy is automatically reorganized as well as resized. Some necessary spares of SCTs and TRGs are added.



Action Help		scheduler Table List							Row 1 to 25 of 100	
Command	Product	State	Table	Suf	Works	Usage	Day			
PARM	sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LICENSED	none			
	Puspa (SCD)	<UP> READY	loaded	00	000000	**DEMO**	..?!			
	Autoxfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!			
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none			
Capacity: SCT= 10000		TRG= 200000	EOT= 500000	EVB= 100	ACT= 1000					
FreePool: SCT= 9588		TRG= 199479	EOT= 500000	EVB= 100	ACT= 1000					
		Statistics			Last Modified					
S	Id	Status	#wklds	#trigs	#buffs	Date	Time	Note		
-	00	CURRENT	24	47	0	2007/09/10	19:20:40			
-	01		20	40	0	2007/08/26	03:55:58			
-	02		0	0	0					
-	03		20	40	0	2007/08/27	20:39:17			
-	04		0	0	0					
-	05		28	59	0	2007/08/28	14:49:38			
-	06		0	0	0					
-	07		28	59	0	2007/09/23	05:31:17	updated		
-	08		0	0	0					
-	09		0	0	0					
-	10		24	47	0	2007/08/28	14:57:04			
-	11		0	0	0					

Figure 5.16: Newly contained table 07 appears on the list

Internally, copy processing is not just duplicating the content of the table. Rather, it checks and correct all pointers contained in SCTs and TRGs to correct broken structure. More over, SCTs and TRGs are sorted based on their interconnection pointers to improve (tune-up) the whole table logic structure.

5.3.5 Reorganizing a Table

S	Id	Status	Statistics			Last Modified	
			#wklds	#trigs	#buffs	Date	Time
-	00	CURRENT	24	47	0	2007/09/10	19:20:40
-	01		20	40	0	2007/08/26	03:55:58
-	02		0	0	0		
-	03		20	40	0	2007/08/27	20:39:17
-	04		0	0	0		
-	05		28	59	0	2007/09/23	05:31:17
-	06		0	0	0		
r	07		28	59	0	2007/09/23	05:30:08
-	08		0	0	0		
-	09		0	0	0		
-	10		24	47	0	2007/08/28	14:57:04
-	11		0	0	0		
-	12		0	0	0		

Figure 5.17: Attempt to reorganize table 07

When you insert new entries in either workloads or trigger events table, newly added workload entries need several pointer interlink with the existing linked-list of workload, newly added trigger event entries need several pointer interlink with



the exiting linked-list of similar trigger events and with workload origin as well as triggered workload. When you update trigger event source name, or change step names of JOB or STC type trigger event source, updated entries probably need to change their pointer interlink. Such cases will require table to be reorganized. zJOS user interface will automatically reorganize the table once you finish. You may, however, reorganize the table manually when you think necessary.

To reorganize a table manually, type R on selected table as shown in figure 5.17 and then hit enter-key. This actually, internally is copying selected table content to itself. When complete, a message occurs explaining number of reorganized SCTs and TRGs appears on the bottom of the panel as shown in figure 5.18.

		Statistics			Last Modified			
S	Id	Status	#wklds	#trigs	#bufs	Date	Time	Note
—	00	CURRENT	24	47	0	2007/09/10	19:20:40	
—	01		20	40	0	2007/08/26	03:55:58	
—	02		0	0	0			
—	03		20	40	0	2007/08/27	20:39:17	
—	04		0	0	0			
—	05		28	59	0	2007/09/23	05:31:17	
—	06		0	0	0			
R	07		28	59	0	2007/09/23	05:30:08	
—	08		0	0	0			
—	09		0	0	0			
—	10		24	47	0	2007/08/28	14:57:04	
—	11		0	0	0			
—	12		0	0	0			
—	13		0	0	0			
—	14		0	0	0			
—	15		0	0	0			
—	16		0	0	0			
—	17		0	0	0			
—	18		0	0	0			
—	19		0	0	0			
—	20		0	0	0			
—	21		0	0	0			
—	22		0	0	0			
—	23							
—	24							
—	25							

00028 SCTs and 000059 TRGs were saved to schedule ID 07

Command ==>

scroll

Figure 5.18: Completion message when table 07 was successfully reorganized

When you press enter-key, message disappears and you would see the time stamps of reorganized table were updated, as shown in figure 5.19.



S	Id	Status	Statistics			Last Modified		Note
			#wklDs	#trigs	#buffs	Date	Time	
	00	CURRENT	24	47	0	2007/09/10	19:20:40	
-	01		20	40	0	2007/08/26	03:55:58	
-	02		0	0	0			
-	03		20	40	0	2007/08/27	20:39:17	
-	04		0	0	0			
-	05		28	59	0	2007/09/23	05:31:17	
-	06		0	0	0			
-	07		28	59	0	2007/09/25	04:43:49	
-	08		0	0	0			
-	09		0	0	0			
-	10		24	47	0	2007/08/28	14:57:04	
-	11		0	0	0			
-	12		0	0	0			

Figure 5.19: Time stamps of table 07 were changed upon completion of reorganization

5.3.6 Deleting a Table

Delete a table does not mean to delete the table from memory, instead, empty the content of the table, and return all allocated SCTs and TRGs back to DIV free pool. Table id will remains permanently in memory.

S	Id	Status	Statistics			Last Modified		Note
			#wklDs	#trigs	#buffs	Date	Time	
	00	CURRENT	24	47	0	2007/09/10	19:20:40	
-	01		20	40	0	2007/08/26	03:55:58	
-	02		0	0	0			
-	03		20	40	0	2007/08/27	20:39:17	
-	04		0	0	0			
-	05		28	59	0	2007/09/23	05:31:17	
-	06		0	0	0			
d	07		28	59	0	2007/09/25	04:43:49	
-	08		0	0	0			
-	09		0	0	0			
-	10		24	47	0	2007/08/28	14:57:04	
-	11		0	0	0			

Figure 5.20: Attempt to delete a table

S	Id	Status	Statistics			Last Modified		Note
			#wklDs	#trigs	#buffs	Date	Time	
	00	CURRENT	24	47	0	2007/09/10	19:20:40	
-	01		20	40	0	2007/08/26	03:55:58	
-	02		0	0	0			
-	03		20	40	0	2007/08/27	20:39:17	
-	04		0	0	0			
-	05		28	59	0	2007/09/23	05:31:17	
-	06		0	0	0			
*	07		28	59	0			updated
-	08		0	0	0			
-	09		0	0	0			
-	10		24	47	0	2007/08/28	14:57:04	

Figure 5.21: Firstly appears as updated

S	Id	Status	#wlds	#trigs	#buffs	Date	Last Modified Time
00		CURRENT	24	47	0	2007/09/10	19:20:40
01			20	40	0	2007/08/26	03:55:58
02			0	0	0		
03			20	40	0	2007/08/27	20:39:17
04			0	0	0		
05			28	59	0	2007/09/23	05:31:17
06			0	0	0		
07			0	0	0		
08			0	0	0		
09			0	0	0		
10			24	47	0	2007/08/28	14:57:04
11			0	0	0		

Figure 5.22: Finally, deleted table becomes empty table

To delete a table, type D on selected table as shown in figure 5.20, then press enter-key. Firstly will appear as updated, as shown in figure 5.21. Then when you hit enter-key again, you would see that the deleted table becomes an empty table, as shown in figure 5.22.

5.4. Managing Table Entry

As explained in 5.3.2, most of source table management functions as discussed in chapter 3 are applicable for in-memory table. In addition, bypass and hold functions are applicable only for in-memory table. Delete function for in-memory table is performed differently. Hence all available action characters to manage an in-memory table are:

- A → Add/insert a new entry (see chapter 3)
- S → Select an entry for further management (see chapter 3)
- T → Display list of associated trigger-event definition (see chapter 3)
- F → Forecast successors list
- B → Flag an entry as a bypassed workload
- H → Flag an entry as held workload
- D → Flag an entry as deleted
- U → Undo flag (restore original flag)
- R → Restore deleted entry (you can also use U)

To obtain help message as shown in figure 5.23, position the cursor on selection column and hit F1-key.



```
Action Help
Scheduled-workload Table Row 2 to 18

Schedule ID 05 Copy to ID
S = workload = System = Date = Time = Day = SMTWTFS =
JTEST01 J MFOC Start 2006/01/01 00:05:00 List 1111111

Specify 1-digit selection code, S=Select/show detail, I=Insert/add,
B=Bypass, H=Hold, D=Delete, T=Show trigger list, U=Undo (unbypass or
unhold), or R=Restore (undelete).

JTEST04 J MFOC Start 2006/01/01 00:45:00 List 1111111
```

Figure 5.23: Help message shows available selection code (action character)

5.4.1 Bypass a Workload

Bypass a scheduled workload means flag a request to Puspa to bypass selected workload from being scheduled. Schedule is running normal until this workload is triggered (all trigger events have accomplished). The workload then, however, is not scheduled. Rather, just assume that the workload was done. Since no CC is produced, if bypassed workload is used as a trigger to its successor, it always assumed as complied for triggering. It always supports triggering decision and gives positive affect.

```
Action Help
Scheduled-workload Table Row 20 to 28 of 28

Schedule ID 05 Copy to ID
S = workload = System = Date = Time = Day = SMTWTFS = Msg
JRMT002 J DYAH2003 Start ****/**/**** 00:00:00 List 1111111
JRMT002 J DYAH2003 End ****/**/**** 24:00:00 Holiday 1
JRMT003 J DYAH2003 Start ****/**/**** 00:00:00 List 1111111
JRMT003 J DYAH2003 End ****/**/**** 24:00:00 Holiday 1
JADI001 J ADI2007 Start ****/**/**** 00:00:00 List 1111111
JADI001 J ADI2007 End ****/**/**** 24:00:00 Holiday 1
b JADI002 J ADI2007 Start ****/**/**** 00:00:00 List 1111111
JADI002 J ADI2007 End ****/**/**** 24:00:00 Holiday 1
JADI003 J ADI2007 Start ****/**/**** 00:00:00 List 1111111
JADI003 J ADI2007 End ****/**/**** 24:00:00 Holiday 1
```

Figure 5.24: Issuing bypass request for a workload

To bypass a workload, type B on selected entry as shown in figure 5.24, then hit enter-key. Then, bypass-flag is turned on and it is remarked "bypass" on msg column as shown in figure 5.25. Once a workload is flagged bypass, it remains bypassed until you restore its original flag. To restore bypass-flag, type U (undo) on bypassed workload entry, then hit enter-key.

Action Help		scheduled-workload Table						Row 20 to 28 of 28	
Schedule ID	Copy to ID		Date	Time	Day	SMTWTFS	Msg		
JRMT002	J	DYAH2003	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JRMT003	J	DYAH2003	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JADI001	J	ADI2007	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JADI002	J	ADI2007	Start	****/**/****	00:00:00	List	1111111	bypass	
			End	****/**/****	24:00:00	Holiday	1		
JADI003	J	ADI2007	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		

Figure 5.25: Bypass request for a workload was confirmed

5.4.2 Hold a Workload

Hold in this case is different with hold schedule process. Hold in this case means turn hold-flag on, so workload is placed on held status. Whereas, hold schedule means to hold the process of scheduler. When jobname is specified, schedule process will be held while specified jobname complete.

When a workload is on hold (hold-flag on), Puspa ignores it from scheduling. But, if it also used as a trigger for its successors, it is not bypassed. Hence, it still involves in triggering decision and gives negative affect (opposite to bypass-flag).

Action Help		scheduled-workload Table						Row 18 to 28	
Schedule ID	Copy to ID		Date	Time	Day	SMTWTFS	Msg		
TSORXSRV	J	MFOC	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JRMT001	J	DYAH2003	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
h JRMT002	J	DYAH2003	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		
JRMT003	J	DYAH2003	Start	****/**/****	00:00:00	List	1111111		
			End	****/**/****	24:00:00	Holiday	1		

Figure 5.26: Attempt to hold a workload

To hold a workload, type H on selected entry as shown in figure 5.26, then hit enter-key. Then, hold-flag is turned on and it is remarked "on hold" on msg column as shown in figure 5.27. Once a workload is flagged hold, it remains on hold until you restore its original flag. To restore hold-flag, type U (undo) on on-hold workload entry, then hit enter-key.



Action Help		Scheduled-workload Table							Row 18 to 28 of 28	
Schedule ID	05	Copy to ID								
S	= workload =	System =		Date	= Time =	Day =	SMTWTFSS	= Msg		
-	TSORXSRV	J	MFOC	Start	****/**/**	00:00:00	List	1111111		
				End	****/**/**	24:00:00	Holiday	1		
-	JRMT001	J	DYAH2003	Start	****/**/**	00:00:00	List	1111111		
				End	****/**/**	24:00:00	Holiday	1		
*	JRMT002	J	DYAH2003	Start	0000/00/00	00:00:00	List	1111111	held	
				End	0000/00/00	24:00:00	Holiday	1		
-	JRMT003	J	DYAH2003	Start	****/**/**	00:00:00	List	1111111		
				End	****/**/**	24:00:00	Holiday	1		

Figure 5.27: Hold request for a workload was confirmed

5.4.3 Delete a Workload

Delete function for in-memory table has the same effect for scheduling process as delete function for source table. The only different, for in-memory table, entry is not physically deleted. Rather, it just flagged as deleted. Hence, you can easily restore it back in the future.

To delete a workload, type D on selected entry as shown in figure 5.28, then hit enter-key. Then, delete-flag is turned on and it is remarked "delete" on msg column as shown in figure 5.29. Once a workload is flagged delete, it remains unavailable until you restore its original flag. To restore delete-flag, type U (undo) or R (restore) on deleted workload entry, then hit enter-key.

Action Help		Scheduled-workload Table							Row 19 to 2	
Schedule ID	05	Copy to ID								
S	= workload =	System =		Date	= Time =	Day =	SMTWTFSS			
-	JRMT001	J	DYAH2003	Start	****/**/**	00:00:00	List	1111111		
				End	****/**/**	24:00:00	Holiday	1		
-	JRMT002	J	DYAH2003	Start	****/**/**	00:00:00	List	1111111		
				End	****/**/**	24:00:00	Holiday	1		
-	JRMT003	J	DYAH2003	Start	****/**/**	00:00:00	List	1111111		
				End	****/**/**	24:00:00	Holiday	1		
d	JADI001	J	ADI2007	Start	****/**/**	00:00:00	List	1111111		
				End	****/**/**	24:00:00	Holiday	1		

Figure 5.28: Attempt to delete a workload



Action	Help	scheduled-workload Table							Row 19 to 28 of 28
Schedule ID 05	Copy to ID								
S = workload =	System =	Date	Time	Day =	SMTWTFSS	Msg			
JRMT001	J	DYAH2003	Start	****/**/	00:00:00	List 1111111			
			End	****/**/	24:00:00	Holiday 1			
JRMT002	J	DYAH2003	Start	****/**/	00:00:00	List 1111111	on hol		
			End	****/**/	24:00:00	Holiday 1			
JRMT003	J	DYAH2003	Start	****/**/	00:00:00	List 1111111			
			End	****/**/	24:00:00	Holiday 1			
* JADI001	J	ADI2007	Start	0000/00/00	00:00:00	List 1111111	delete		
			End	0000/00/00	24:00:00	Holiday 1			

Figure 5.29: Delete request for a workload was confirmed

Deleted workload will be always assumed no longer exist. Anything related to the deleted workload also assumed gone. You can't even select it for review.

5.4.4 Restore a Workload

Action	Help	scheduled-workload Table							Row 22 to 28 of 28
Schedule ID 05	Copy to ID								
S = workload =	System =	Date	Time	Day =	SMTWTFSS	Msg			
JADI001	J	ADI2007	Start	****/**/	00:00:00	List 1111111			
			End	****/**/	24:00:00	Holiday 1			
U JADI002	J	ADI2007	Start	****/**/	00:00:00	List 1111111	delete		
			End	****/**/	24:00:00	Holiday 1			

Figure 5.30: Attempt to restore (undo) deleted entry

Workload which is already flagged as either bypassed, on hold or deleted, can easily be restored to its original status. Just type U on flagged workload as shown in figure 5.30 and hit enter-key, then restore is confirmed as shown in figure 5.31. Its flag is then turned off. Once flagged workload is restored, all associated information is also restored.

Action	Help	scheduled-workload Table							Row 22 to 28 of 28
Schedule ID 05	Copy to ID								
S = workload =	System =	Date	Time	Day =	SMTWTFSS	Msg			
JADI001	J	ADI2007	Start	****/**/	00:00:00	List 1111111			
			End	****/**/	24:00:00	Holiday 1			
JADI002	J	ADI2007	Start	0000/00/00	00:00:00	List 1111111	restor		
			End	0000/00/00	24:00:00	Holiday 1			

Figure 5.31: Undo request for a workload was confirmed



5.5. Saving Updates

Unlike when you work with source table, updates in in-memory table immediately effective to all Puspa processes (within and outside zJOS address space), since that memory is shared among them as a dataspace. Updates will immediately be saved into VSAM LDS using DIV function, once you end the table session by hitting F3 key. Figure 5.32 shows console message indicating that DIV saving is complete.

```
04.45.06 STC09437 DERSIS202I Request for zJOS base accepted.  
04.45.30 STC09437 DERSIP818I Saving dataspace to DIV was successfull.
```

Figure 5.32: Console message confirming DIV saving

If updates you have done do not change any pointers, e.g. update timeframe, week day list, MSG or CMD text, saving is immediately performed once you exit from table session. If update, however, involve pointers changes, e.g. change name of workload or trigger event, or insert new entries (workloads or triggers), table is then reorganized prior to saving.

5.6. Reviewing the Updates

As table logic structure is key factor in most of scheduling and triggering process, you should not start schedule without reviewing its table first. The first review is before you exit from table session. Each time you update an entry, do necessary rechecking the detail of the workload definition, list of associated triggers, and the detail of each trigger event definition. Make sure each entry is correct before exit and save. Make necessary correction if you find anything inconsistent with your plan. Once you exit and save, reselect the table for second review.

The first step of second review is exactly the same as the first review; check the detail of workload, list of triggers and the detail of each trigger event. Next step of second review is checking successors list of each workload, by using F prefix command. Figure 5.33 shows an example of successors list in respond to F prefix command. See whether successors list appeared here is consistent with your plan. If consistent, then, your update is successfully reorganized. To make sure the whole table is consistent (strongly recommended), you should review successors list of every workload.

```

Action Help
-----
                                scheduled-workload Table                                Row 1 to 17 of 34
Schedule id 03  Copy to ID
S - workload - System -
JTEST00 JOB BNIPROD Start ****/**/**** 00:00:00 List 1111111
                                End ****/**/**** 24:00:00 Holiday 1
F JTEST01 JOB BNIPROD Start ****/**/**** 00:00:00 List 1111111
                                End ****/**/**** 24:00:00 Holiday 1
JTEST02 JOB BNIPROD Start ****/**/**** 00:00:00 List 1111111
-----
                                zJOS-XDI/Puspa 2.1.3
JTEST0 Successors List Row 1 to 5 of 5
JTEST0 wkld JTEST01 JOB on system BNIPROD
JTEST0 ----- Successor ----- Dependency -----
JTEST0 wkld Type System Trigger Condition
JTEST0 JTEST02 JOB BNIPROD Step=STPTIGA CC EQ 00C
JTEST0 JTEST03 JOB BNIPROD Step=STPAT CC EQ 000
JTEST0 JTEST04 JOB BNIPROD End of job Any CC
JTEST0 JIDAO JOB BNIPROD Step=STPDUA CC EQ 00C
JTEST0 JIMWD JOB BNIPROD Step=STPDUA CC EQ 008
***** Bottom of data *****
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0
JTEST0

```

Figure 5.33: Successors list forecast of selected workload

Why forecasting successors is not mentioned in the first review? Successors list forecast, although can also be done in the first review, the result, however, not very accurate for conclusion. If your update involves pointers changes, F prefix command may yield wrong successors information before table is reorganized. Hence, do not make any correction based on successors list forecast, unless you sure that the table is well organized.

5.7. Fixing Inconsistent Table Structure

When you find inconsistencies in your table, the first step is reorganizing it by typing R prefix command in tables list panel. Then display the table and check again consistency of each workload by reviewing its predecessors list (T prefix command) and successors list (F prefix command). Unless the table is seriously corrupted (more than one pointer lost), it should consistent.



If the table still inconsistent after reorganized, then you need to copy it and delete the original one. Its copy normally should consistent, as copy process involve sort and reorganize.

If the table copy still inconsistent, then you must restore the VSAM LDS from the latest backup. That is why you must back it up every day.



Chapter 6 **Operate Scheduling System**

All schedules you have prepared will remain inactive until you issue START request as shown in figure 6.2. Before you start a schedule, you have to make sure 2 things:

- Each schedule flow has been correctly prepared.
- Schedule table must be clean

6.1. Preparing Schedule Flow

Preparing schedule flow means prepare the logic of each triggering mechanism of each particular workload against its predecessors and successors to obtain global operable flow. Wrongly specified triggering mechanism might cause logic flow stuck in the middle and schedule will never finish. The most important things you must carefully prepare are:

- Initial flow workloads
- Guarantee always have alternate flows

6.1.1 Initial Flow Workload

Workload which is triggered automatically when schedule is started called initial flow workload. Such workload must not have predecessors. In other word, must not have triggering-event sources associated with it. Otherwise, it will not eligible as initial flow workload.

Other thing you should care is timeframe. Timeframe can potentially affect the initial flow workload. For example, if it is started earlier than specified start-time, it will be placed in wait state until start-time is reached, hence you feel as if it is delayed. In most scheduling system, timeframe of initial flow workload is the only timeframe for the whole schedule flow. All its successors should not have its own individual timeframe unless it really needed for very specific reasons.

A schedule table may contain more than one schedule flows. Each flow may have more than one initial flow workloads. Each flow may join to other flow in a certain position, or may split into several sub-flows in certain position. There is no identifier for a flow.

6.1.2 Cleaning Up Schedule Table

Before started, selected schedule table must be clean. When you just select a schedule table, Puspa automatically refresh it and make it ready for use. Shown in figure 6.1 is an example when you just selected schedule table 05. Puspa then make it a current table and refresh it for you.

```
- 02.00.16 STC09558 DERSIP816I Scd 05 with 00028 SCTs and 000059 TRGs is
- now in effect.
*02.00.16 STC09558 *DERSIP144I schedule was refreshed. Next cycle ready
```

Figure 6.1: Console message responding a selection to a schedule table

However, when you reuse selected table, Puspa won't refresh it. You have to do it yourself by issuing REFRESH request. Otherwise, table will not eligible for next start.

Normally, cleaning up schedule table is performed once for each schedule cycle. For example, if your schedule is daily and started at every 22:00:00, regardless what time schedule is completed, you should not refresh the table until next day at 22.00.00. Because, once you refresh the table, all status information is no longer available.

6.2. Starting Schedule

To start the schedule, firstly, select schedule table by filling up schedule id in Suf column on control panel. Then, issue START request as shown in figure 6.2. Request is then sent to Puspa in zJOS address space as shown in logged response in figure 6.2. In zJOS address space, Puspa search all initial-flow workloads and schedule them.

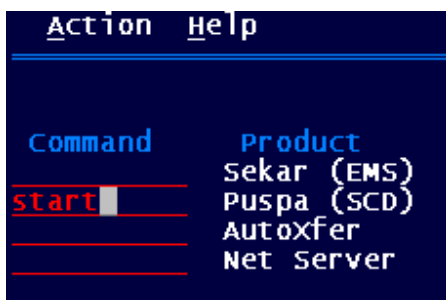


Figure 6.2: Issuing START request to start a scheduling process



Action	Help							
		zJOS-XDI Control Panel				Row 499 to 499 of		
Command	Product	State	Table	Suf	works	Usage	Day	
*START	Sekar (EMS)	<UP> ACT(SSl)	loaded	00	000002	LICENSED	none	
	Puspa (SCD)	<UP> READY	loaded	00	000009	**DEMO**	..?!	
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!	
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none	
Date Time Log								
07.268 05:06:03 START request to Puspa (SCD) was sent to zJOS.								
***** Bottom of data *****								

Figure 6.3: START request is sent to Puspa in zJOS address space

At the same time, Puspa changes its state from READY to ACTIVE state. When you hit enter-key on zJOS control panel once again, Puspa state is then shown as active as in figure 6.4.

Action	Help	zJOS-XDI Control Panel						Row 513 to 526 of
Command	Product	State	Table	Suf	Works	Usage	Day	
	Sekar (EMS)	<UP> ACT(SSI)	loaded	00	000003	LICENSED	none	
	Puspa (SCD)	<UP> ACTIVE	loaded	00	000016	**DEMO**	..?!	
	Autoxfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!	
	Net Server	DOWN INACTIVE	N/A	**	000000	standard	none	

Figure 6.4: Scheduling process is active

When you go to table list panel, current table is also shown active when schedule is active, as shown in figure 6.5. Although you can still open the table, you will be warn when open the active table. You should not make any changes in the active table unless you have urgently reason and rely the risk you might take.

Capacity: SCT=		10000	TRG=	200000	EOT=	500000	EVB=	100	ACT=	1000
FreePool: SCT=		9588	TRG=	199479	EOT=	499982	EVB=	100	ACT=	996
		Statistics				Last Modified				
S	Id	Status	#wklds	#trigs	#bufs	Date	Time	Note		
00		ACTIVE	24	47	0	2007/09/25	05:05:43			
01			20	40	0	2007/08/26	03:55:58			
02			0	0	0					
03			20	40	0	2007/08/27	20:39:17			
04			0	0	0					
05			28	59	0	2007/09/23	05:31:17			
06			0	0	0					
07			0	0	0					
08			0	0	0					
09			0	0	0					
10			24	47	0	2007/08/28	14:57:04			
11			0	0	0					

Figure 6.5: Current table is active when scheduling process is active

6.3. Monitoring Schedule Activities

zJOS primary control panel provides a facility to monitor schedule activities and progress. To reach this facility, you can either select action-bar choice 2 as in figure 6.6, or just issuing LIST request as in figure 6.7.

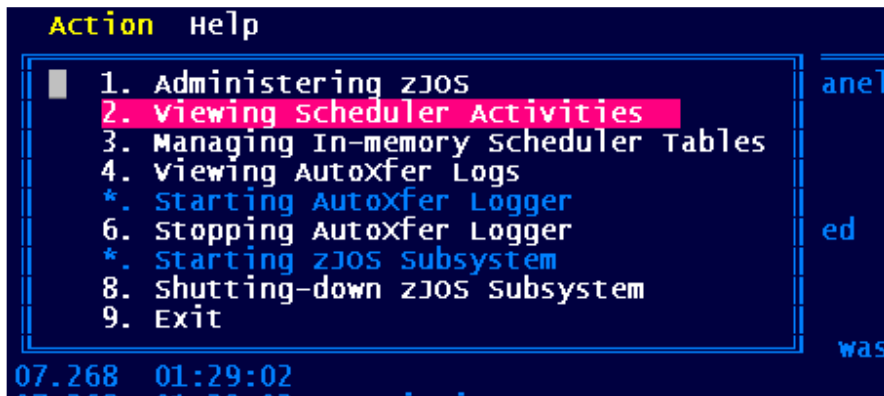


Figure 6.6: Attempt to view schedule activities by selecting action-bar choice

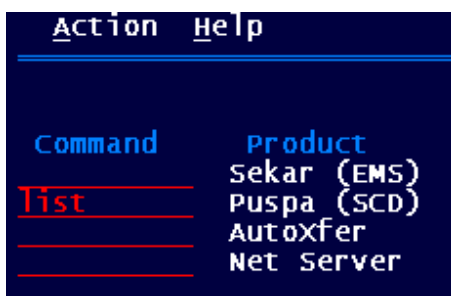


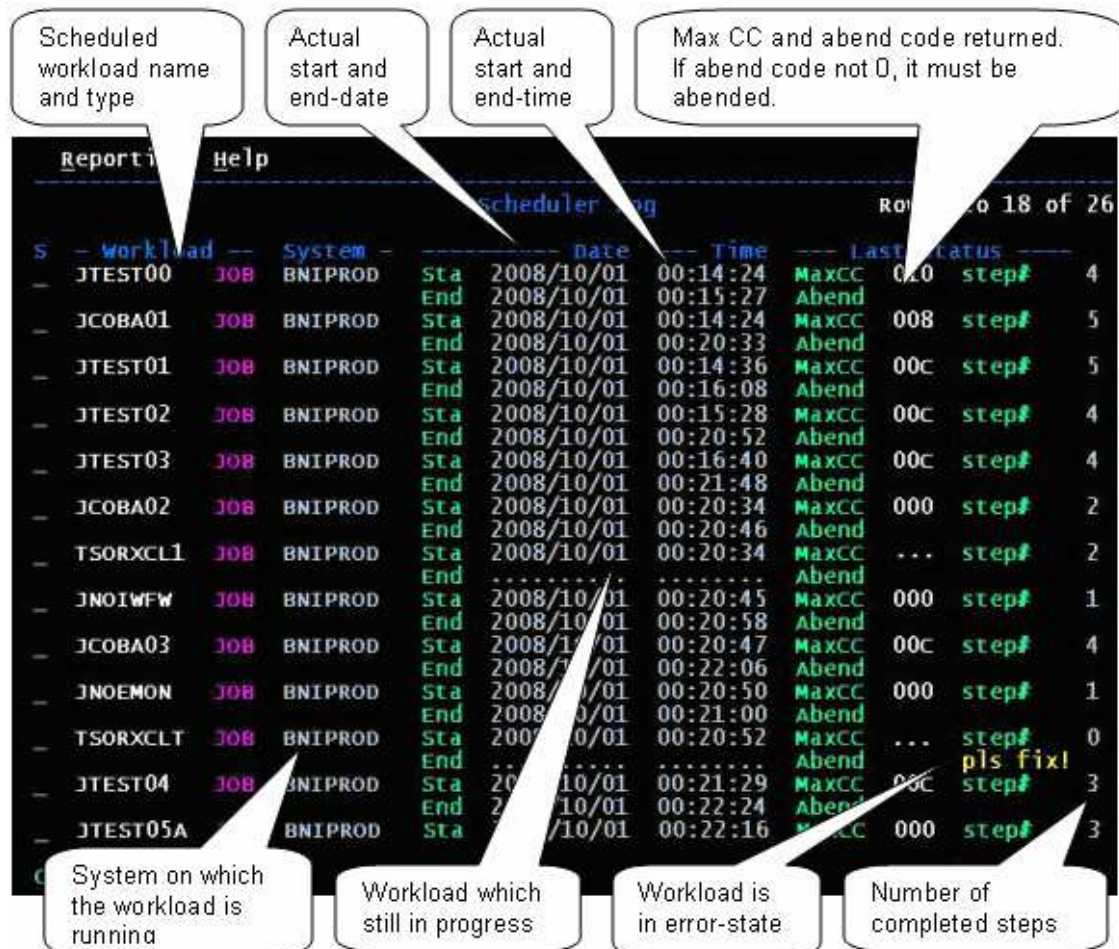
Figure 6.7: Attempt to view schedule activities by issuing LIST request

scheduled-jobs Log							
Row 1 to 6 of 6							
S	Jobname	=	System	=	Date	Time	Condition = Misc. Info =
-	JTEST00	MFOC	Start	2007/09/25	01:13:07	MaxCC	... #steps 2
-			End	Abend	... #trigs 0
-	JCOBA01	MFOC	Start	2007/09/25	01:13:07	MaxCC	... #steps 2
-			End	Abend	... #trigs 0
-	JTEST08	MFOC	Start	2007/09/25	01:13:07	MaxCC	000 #steps 0
-			End	2007/09/25	01:13:08	Abend	000 #trigs 0
-	JSTC5555	MFOC	Start	2007/09/25	01:13:07	MaxCC	... #steps 0
-			End	Abend	... #trigs 0
-	JCOBA02	MFOC	Start	2007/09/25	01:13:18	MaxCC	... #steps 1
-			End	Abend	... #trigs 1
-	JTEST01	MFOC	Start	2007/09/25	01:13:22	MaxCC	... #steps 0
-			End	Abend	... #trigs 1
***** Bottom of data *****							

Figure 6.8: Schedule activities and progress logs at initial time

6.3.1 Scheduler Logs

Schedule activities and progress are logged in provided space in schedule table. Summary of the logs are displayed on scrollable panel. Initially only some initial flow workloads appear as shown in figure 6.8. Later when scheduling process is moving forward, more logs appear as shown in figure 6.9. This figure also shows detail of each field briefly.



The screenshot shows a 'Scheduler Log' window with a table of log entries. Callouts provide details for various fields:

- Scheduled workload name and type:** Points to the 'Workload' column.
- Actual start and end-date:** Points to the 'Date' column.
- Actual start and end-time:** Points to the 'Time' column.
- Max CC and abend code returned. If abend code not 0, it must be abended.** Points to the 'Last' column.
- System on which the workload is running** (points to 'System' column).
- Workload which still in progress** (points to 'Status' column).
- Workload is in error-state** (points to 'Status' column).
- Number of completed steps** (points to 'step#' column).

S	Workload	System	Sta	Date	Time	Last	Status	step#	
-	JTEST00	JOB	BNIPROD	2008/10/01	00:14:24	MaxCC	020	step#	4
-	JCOBA01	JOB	BNIPROD	2008/10/01	00:14:24	MaxCC	008	step#	5
-	JTEST01	JOB	BNIPROD	2008/10/01	00:20:33	Abend	00c	step#	5
-	JTEST02	JOB	BNIPROD	2008/10/01	00:14:36	MaxCC	00c	step#	4
-	JTEST03	JOB	BNIPROD	2008/10/01	00:16:08	Abend	00c	step#	4
-	JTEST03	JOB	BNIPROD	2008/10/01	00:15:28	MaxCC	00c	step#	4
-	JCOBA02	JOB	BNIPROD	2008/10/01	00:20:52	Abend	00c	step#	2
-	TSORXCL1	JOB	BNIPROD	2008/10/01	00:16:40	MaxCC	...	step#	2
-	JNOIWF	JOB	BNIPROD	2008/10/01	00:20:34	Abend	000	step#	1
-	JCOBA03	JOB	BNIPROD	2008/10/01	00:20:46	MaxCC	000	step#	4
-	JNOEMON	JOB	BNIPROD	2008/10/01	00:20:47	MaxCC	00c	step#	1
-	TSORXCLT	JOB	BNIPROD	2008/10/01	00:22:06	Abend	000	step#	0
-	JTEST04	JOB	BNIPROD	2008/10/01	00:20:50	MaxCC	...	step#	3
-	JTEST05A	JOB	BNIPROD	2008/10/01	00:21:00	Abend	00c	step#	3

Figure 6.9: Schedule activities and progress logs

Workload name, type and system

Workload name is jobname for JOB, STC and probably DSA and DSR type of workload, or just a name for other workload type. System is name of system on which workload is or was running. Combined of 3 is an identifier of scheduled workload.



Date and time

Date and time are actual date- and time-range during which the workload was executed. Dotted lines on end-date and end-time indicate the workload is still in progress. End-date and end-time appear only when workload was completed.

Condition

This information describes summary of the final condition when workload was done. MaxCC is the highest CC ever resulted by the whole workload. Abend is abend code returned when workload was abnormally ended. If abend is not 0, maxCC must be 0. If maxCC is not 0, abend must be 0. Both condition values appear only when workload was completed. Otherwise, they appear as dotted lines. Condition information is applicable only for batch-job and STC workloads.

Last status column

This column contains information of number of completed steps and remark. Number of completed steps applicable is only for JOB and STC type of workload. Other workload types always one step.

Remark is short message describing current status of workload. Workload which is still in progress or was completed with system code of zero always no remark. Applicable remarks for this column are:

- Abend** – JOB or STC type of workload was terminated with nonzero system code.
- Wait TOD** – Workload is waiting for schedule start time. When start time is reached, it will automatically be scheduled and remark then disappears.
- Pls fix!** – JOB or STC type of workload is placed in error state and waiting for fixing. It can be because of JCL error or experiencing defined exception condition with hold or cancel and hold action. Once it got fixed, you may rerun it to continue schedule flow and remark disappears.
- No unit** – JOB or STC type of workload was canceled because of no device unit satisfied. Once it got fixed or device unit is provided, you may rerun it to continue its schedule flow and remark then disappears.
- No JCL** – JOB type of workload is not found in the specified library. Once the library member representing this workload is provided, it will automatically be scheduled and remark then disappears.
- Avoided** – Workload is not scheduled because of applied special calendar filtering.
- Error** – Workload is in undefined error state.



Selection column

The most left column of the panel is a selection column. Action characters are provided for further monitoring and managing schedule operation. These are:

- S → see detail workload information
- T → see triggering information
- F → display list of successors.
- H → halt a workload from being scheduled
- R → restart schedule from selected workload
- X → rerun finished/unfinished workload
- Z → unconditionally run the workload

6.3.2 Detail Workload Information

When you enter S on selection column of scheduler logs, the detail of selected workload information is then displayed on window as shown in figure.6.10. Detail workload information is actually information as shown in logs summary plus some other information, such as jobid, ASID, day-of-week and list of completed steps.

To exit and return to previous panel, hit F3 or F12 key.

```

S JTEST01 JOB BNIPROD Sta 2008/10/01 00:14:36 MaxCC 00C step# 5
End 2008/10/01 00:16:08 Abend
zJOS-XDI/Puspa 2.1.3

Action
Scheduled-job Log Detail Row 5 from 59

Workload JOB JTEST01 Jobid JOB00953 Asid 0022 on system BNIPROD
Status: #Step 5 #Trig 1 MaxCC 00C SCC was completed
Run on: wed 2008/10/01 at 00:14:36 to wed 2008/10/01 at 00:16:08
Text/DSN:

No. Jobstep Procstep UCC SCC End-time Elapsed
1 STPSATU 004 000 00:14:43.93 000007.41
2 STPDUA 008 000 00:15:02.41 000018.48
3 STPTIGA 00C 000 00:15:07.66 000005.25
4 STREMPAT 000 000 00:15:17.17 000009.51
5 STPLIME 000 000 00:16:08.08 000050.91
***** Bottom of data *****

Command ==> scroll ==> CSR
JNOFMWD JOB BNIPROD Sta 2008/10/01 00:21:49 MaxCC 000 step# 1
End 2008/10/01 00:21:55 Abend
    
```

Figure 6.10: Detail of a scheduled workload

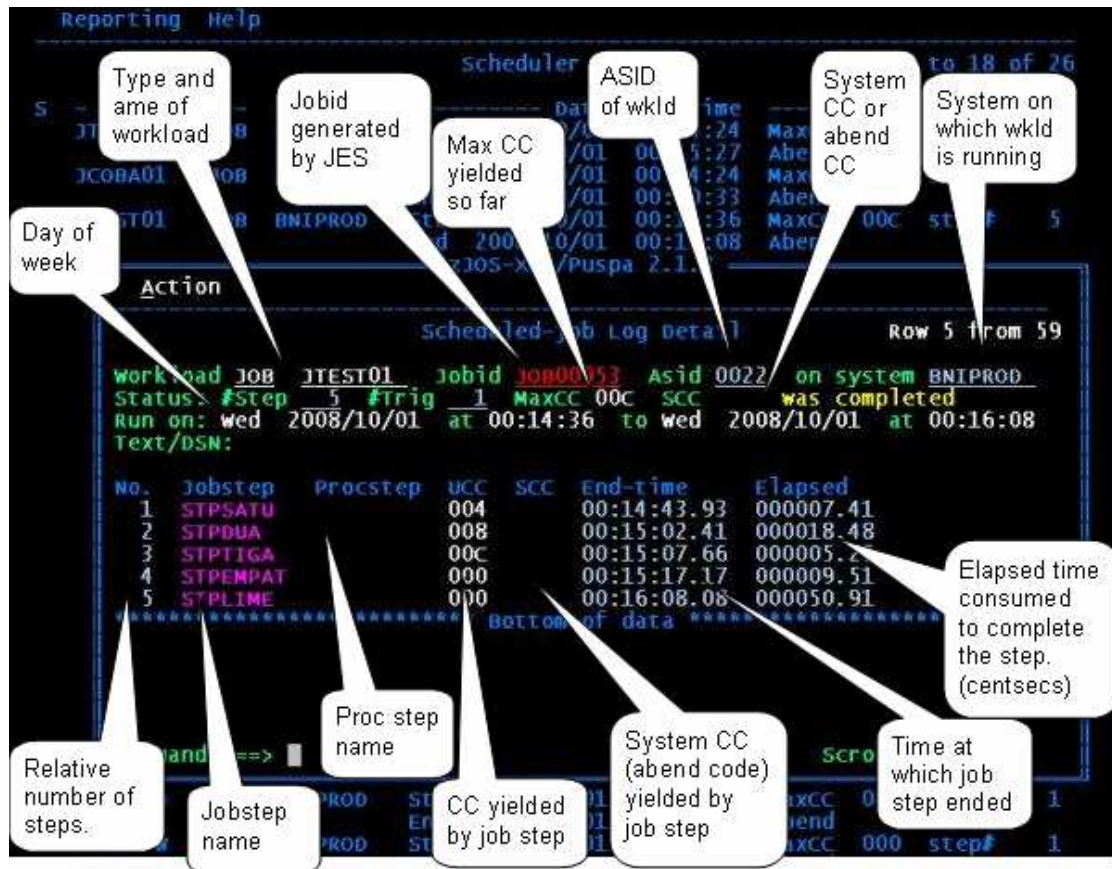


Figure 6.11: Brief description of a scheduled workload detail window

Figure 6.11 shows brief description of each field. This window is just informing you the detail status of an entry. Nothing else you can do with this panel. When list of job step status larger than window depth, you can scroll it up and down by hitting F7 and F8 keys. Scrolling feature follows ISPF standard convention, which can be per page or based on cursor position.

Involvement of steps list information in workload detail information is applicable only for batch-job (JOB) and STC type of workloads. If other type of scheduled workload is selected, such panel appears without step information.

6.3.3 Detail Triggering Information

When you enter T on selection column of the scheduler logs, a list of triggering information associated to selected workload is then displayed on a popped up window as shown in figure.6.12. Heading of window shows the same information as appear on the detail workload window.



Brief description of each field is shown in figure 6.13. This window is not for update. Rather, just informing you a list of triggering event status so far. Nothing else you can do with this panel. When list of triggers larger than window depth, you can scroll it up and down by hitting F7 and F8 keys. Scrolling feature follows ISPF standard convention, which can be per page or based on cursor position.

To exit and return to previous panel, hit F3 or F12 key.

```

reporting help
-----
Scheduler Log
Row 1 to 18 of 26

S - workload -- System - ----- Date --- Time --- Last status -----
JTEST00 JOB BNIPROD Sta 2008/10/01 00:14:24 MaxCC 010 step# 4
End 2008/10/01 00:15:27 Abend
JCOBA01 JOB BNIPROD Sta 2008/10/01 00:14:24 MaxCC 008 step# 5
End 2008/10/01 00:20:33 Abend
JTEST01 JOB BNIPROD Sta 2008/10/01 00:14:36 MaxCC 00C step# 5
End 2008/10/01 00:16:08 Abend
-----
ZJOS-XDI/Puspa 2.1.3
Triggering Jobs Log
Row 3 from 38

Workload JOB JTEST03 Jobid JOB00957 Asid 0020 on system BNIPROD
Status: #Step 4 #Trig 4 MaxCC 00C SCC was completed
Run on: wed 2008/10/01 at 00:16:40 to wed 2008/10/01 at 00:21:48
Text/DSN:

TrigJob System Step ProcStep Grp SCC UCC Op RCC Flags
JRMT001 TLC2007 00 n/a n/a NC 000 ignored
JTEST01 BNIPROD STPEMPAT 00 000 EQ 000 10..0..0
JTEST00 BNIPROD 01 010 NC 000 10..1..0
JTEST02 BNIPROD DUA003 02 004 EQ 004 10..0..0
***** Bottom of data *****

Command ==> 
scroll ==> CSR

JNOFMWD JOB BNIPROD Sta 2008/10/01 00:21:49 MaxCC 000 step# 1
End 2008/10/01 00:21:55 Abend
JNOFMWD JOB BNIPROD Sta 2008/10/01 00:22:11 MaxCC 000 step# 1

```

Figure 6.12: List of triggers for a scheduled workload

Detail triggering event information panel is applicable only for workload which is scheduled by complied triggering events. Workload which is scheduled by time only or unconditionally selected to run, doesn't have triggering event information. If T prefix command is issued for such workload, panel is appeared without detail triggering event information listed. The only information you can get is the status information in the panel header.

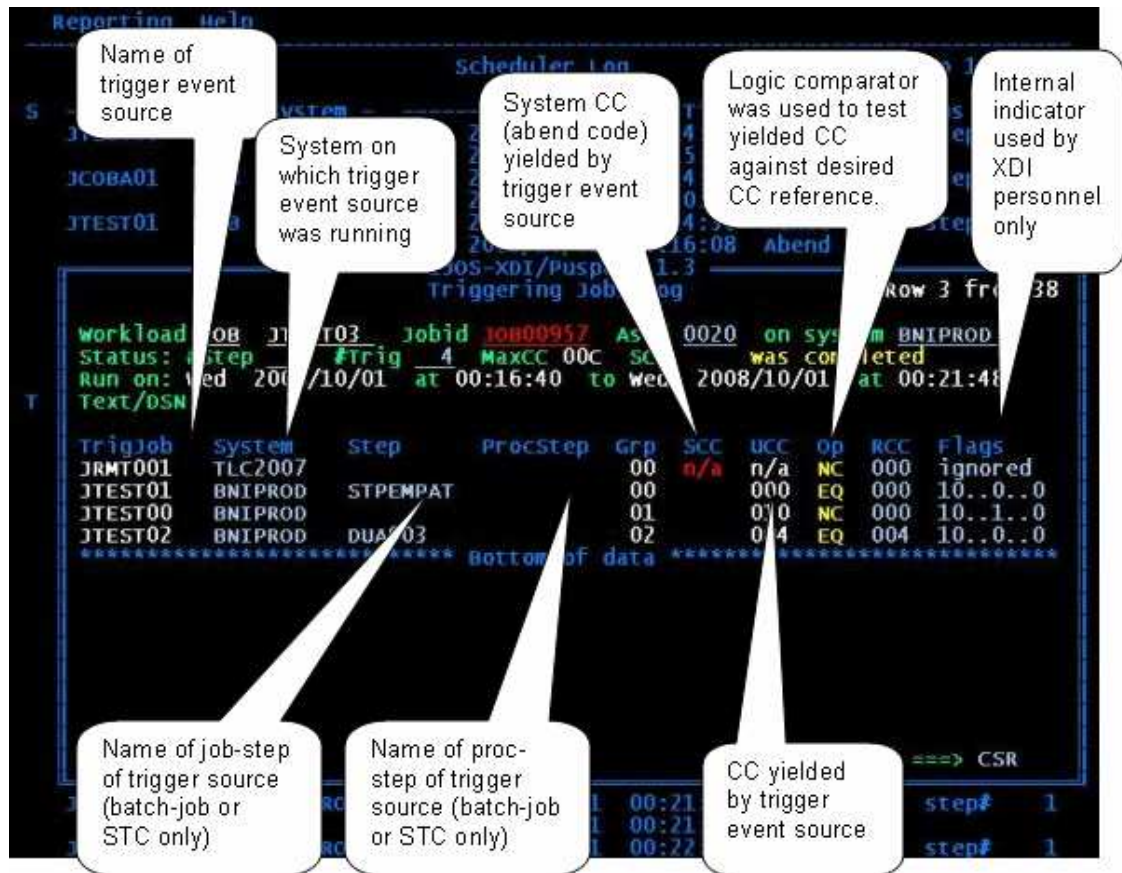


Figure 6.13: Brief description of effective triggers list window

6.3.4 Successors List Information

You can display successors list of the selected workload by entering F prefix command on scheduler log panel. On the schedule table panel, as discussed in chapter 5, this command is used to display forecast of successors list. The appearance of the displayed result is also almost similar. The only difference in its appearance is an additional column inserted prior to the last column which may contains remark 'OK'. Nevertheless, successors list here is not a forecast, rather, it is a fact of successors. The 4th column shows whether the returned condition of the workload is complied to trigger each successor. Only successor with remark 'OK' was complied. Remark 'OK' indicates compliance of returned condition of this particular workload. It does not mean that triggering was done, since the final triggering decision is compiled from all trigger events. Figure 6.14 shows an example of F prefix command response.



Reporting Help		Scheduler Log						Row 1 to 18 of 26	
S	Workload	System	Sta	Date	Time	Last	Status		
JTEST00	JOB	BNIPROD	Sta	2008/10/01	00:14:24	MaxCC	010	step#	4
			End	2008/10/01	00:15:27	Abend			
JCOBA01	JOB	BNIPROD	Sta	2008/10/01	00:14:24	MaxCC	008	step#	5
			End	2008/10/01	00:20:33	Abend			
JTEST01	JOB	BNIPROD	Sta	2008/10/01	00:14:36	MaxCC	00c	step#	5
			End	2008/10/01	00:16:08	Abend			
JIMWD	zJOS-XDI/Puspa 2.1.3								
F	JTEST0	Successors List							Row 1 to 8 of 8
JIWIM	wkld JTEST02 JOB on system BNIPROD								
JTEST0	wkld	Type	System	Trigger	Condition				
JTEST03	JOB	BNIPROD	Step=DUA003	OK	CC EQ 004				
JTEST04	JOB	BNIPROD	Step=DUA004	OK	CC EQ 000				
TSORXCLT	JOB	BNIPROD	End of job	OK	Any CC				
TSORXCL2	JOB	BNIPROD	End of job	OK	Any CC				
JIWWD	JOB	BNIPROD	End of job	OK	Any CC				
JNOIWF	JIWIM	JOB	BNIPROD	Step=DUA002	OK	CC EQ 00c			
JEMON	JOB	BNIPROD	Step=DUA003	OK	CC EQ 004				
JFWWD	JOB	BNIPROD	End of job	OK	Any CC				
***** Bottom of data *****									
JNOEMO									
TSORXC									
JTEST0									
JNOFWW	Command ==> Scroll ==> CSR								
JNOFMWD	JOB	BNIPROD	Sta	2008/10/01	00:21:49	MaxCC	000	step#	1
			End	2008/10/01	00:21:55	Abend			
JNOFMFW	JOB	BNIPROD	Sta	2008/10/01	00:22:11	MaxCC	000	step#	1

Figure 6.14: Display list of successors

6.4. Halt and Restart Schedule

scheduled-jobs Log							Row 4 to 13 of 13	
S	Jobname	System	Date	Time	Condition	Misc. Info		
JTEST01	MFOC	Start	2007/09/28	05:32:00	MaxCC	00c	#steps	5
Type S=(Show detail), T=(Trigger list), H=(Halt schedule at this job) or R=(Restart schedule at this job)								
JCOBA03	MFOC	End	2007/09/28	05:36:22	Abend	000	#trigs	1
		Start	2007/09/28	14:25:05	MaxCC	00c	#steps	4
		End	2007/09/28	14:25:35	Abend	000	#trigs	2

Figure 6.15: HALT and RESTART requests available on schedule log panel

As discussed on par 4.3 chapter 4, during scheduling progress, you may halt, then resume or restart at a certain position of schedule flow. Such facilities can be done within schedule log panel. However, as discussed in sub-par 6.3.1 this chapter and shown in figure 6.15, the schedule log panel provides only H (halt) and R (restart) action characters. Hence, only halt and restart functions are available on the schedule log panel.



Example

The following figures describe an example of restarting halted schedule from the schedule log panel. Schedule is being halted. In figure 6.15, shown R request is issued to restart schedule at job JTEST02.

Scheduled-jobs Log									
Row 4 to 17 of									
S	Jobname	=	System	=	Date	Time	Condition	=	Misc. Info
-	JTEST01		MFOC		Start	2007/09/28 05:32:00	MaxCC	00C	#steps 5
					End	2007/09/28 05:36:37	Abend	000	#trigs 1
-	TSORXSRV		MFOC		Start	2007/09/28 05:33:34	MaxCC	000	#steps 1
					End	2007/09/28 05:33:38	Abend	000	#trigs 1
r	JTEST02		MFOC		Start	2007/09/28 05:34:53	MaxCC	00C	#steps 4
					End	2007/09/28 05:37:32	Abend	000	#trigs 2
-	TSORXCL1		MFOC		Start	2007/09/28 05:36:18	MaxCC	000	#steps 1
					End	2007/09/28 05:36:22	Abend	000	#trigs 1

Figure 6.16: R request is issued to restart schedule at JTEST02

Puspa immediately confirm by sending message DERSCS348 on console as in figure 6.17. At the same time, job JTEST02 is restarted and all its successors are cleaned up for restarting. Appearance of log is then changes as shown in figure 6.18.

```
*14.34.53 STC09591 *DERSCS348I scheduler is restarted at job of JTEST02
MFOC          DEL=RD  RTIME=1  RNUM=19  SEG=14  CON=N
```

Figure 6.17: Response on console confirming that schedule is restarted at JTEST02

Scheduled-jobs Log									
Row 4 to 8 of									
S	Jobname	=	System	=	Date	Time	Condition	=	Misc. Info
-	JTEST01		MFOC		Start	2007/09/28 05:32:00	MaxCC	00C	#steps 5
					End	2007/09/28 05:36:37	Abend	000	#trigs 1
-	TSORXSRV		MFOC		Start	2007/09/28 05:33:34	MaxCC	000	#steps 1
					End	2007/09/28 05:33:38	Abend	000	#trigs 1
-	TSORXCL1		MFOC		Start	2007/09/28 05:36:18	MaxCC	000	#steps 1
					End	2007/09/28 05:36:22	Abend	000	#trigs 1
-	JCOBA03		MFOC		Start	2007/09/28 14:25:05	MaxCC	00C	#steps 4
					End	2007/09/28 14:25:35	Abend	000	#trigs 2
-	JTEST02		MFOC		Start	2007/09/28 14:34:53	MaxCC	...	#steps 2
					End	Abend	...	#trigs 2
***** Bottom of data *****									

Figure 6.18: Appearance of schedule log when JTEST02 is restarted

6.5. Force a Workload to Run

Automatic scheduling system requires correctness in both scheduled workload flow and workload content definitions to run perfectly. Incomplete flow definition may cause the flow stuck when unexpected condition occurred. Incorrectness content of workload may cause error condition such as abend, JCL error or out of



units range which potentially cause unexpected flow stuck unless anticipated in the schedule flow definition. Understand that such situation may occur in either dynamic environment or newly implemented automation, zJOS/Puspa provides a chance to avoid flow stuck by forcing a workload rerun or run unconditionally.

6.5.1 Rerun a Workload

zJOS/Puspa allows you (if necessary) to rerun workload that has been finished, by using X prefix command. You must, however, be aware It would recheck its successors that have not been scheduled. For instant, let say a workload job JTEST7 is defined in the table. Job JTEST8 is defined to be scheduled when JTEST7 ended with CC = 0 and job JTEST9 is defined to be scheduled when JTEST7 ended with CC = 8. In the previous run, job JTEST7 was ended with CC = 8 and JTEST9 was triggered. Job JTEST8 of course was not be triggered. If you then rerun JTEST7 and ended with CC = 0, then JEST8 is triggered. If you want rerun JTEST7 and don't want JTEST8 run, you must not rerun job JTEST7 this way. Rather, you should execute JTEST7 from outside Puspa.

6.5.2 Unconditionally Run a Workload

Unconditionally run or schedule workload is a facility to give you chance to run a workload without care of its dependency with predecessors. This actually breaks the scheduler concept. Such facility is provided only for emergency situation.

Newly implemented automatic scheduling system sometime face an unexpected situation where schedule flow of a workload gets stuck. If you forget to define exception handling correctly for a workload, this potentially possible to place the workload in stuck situation when no successor meets with returned CC. You may fix and rerun it. If it still returns unexpected CC, the flow will still stuck. In such situation, if you expect the flow is proceeded unconditionally at this point, you may use this facility as an emergency action to schedule some or all its successors by entering Z prefix command in front of each selected successor.

For example, a new workload job JOB1 is added in the schedule table without exception handling. Its defined successors are job JOB1A and JOB1B. Job JOB1A is only triggered when JOB1 is ended with CC = 0 and job JOB1B only when CC = 8. One day, JOB1 is abnormally ended with system code 0C4. Of course neither JOB1A nor JOB1B is triggered. If no chance at the moment to fix JOB1 to only return CC equal to 0 or 8, the only way to proceed the flow to either JOB1A or JOB1B, is forcing either JOB1A or JOB1B to be scheduled by entering Z prefix command.

6.6. Scheduling Report

Puspa provides a facility to produce report and download it onto the workstation computer. Report contains information which exactly the same as shown interactively in scheduling log panel. Rather, it is formed in 132-byte length of VBA record flat dataset. If AutoXfer is active, report will directly be downloaded onto workstation unless LOCAL destination is specified.

Reporting facility can be found in reporting-bar of scheduling log panel as shown in figure 6.19. When you select this bar, pop-down menu appears as shown in figure 6.20.



Jobname	System	Date	Time	Condition	Misc. Info
JTEST00	MFOC	Start 2007/11/20	21:54:56	MaxCC 010	#steps 3
		End 2007/11/20	21:56:51	Abend 000	#trigs 0
JCOBA01	MFOC	Start 2007/11/20	21:54:57	MaxCC 008	#steps 5
		End 2007/11/20	21:59:48	Abend 000	#trigs 0
JCOBA02	MFOC	Start 2007/11/20	21:55:03	MaxCC 000	#steps 2
		End 2007/11/20	21:56:56	Abend 000	#trigs 1
JTEST01	MFOC	Start 2007/11/20	21:55:08	MaxCC 00C	#steps 5
		End 2007/11/20	22:00:10	Abend 000	#trigs 1
TSORXSRV	MFOC	Start 2007/11/20	21:56:49	MaxCC 000	#steps 1
		End 2007/11/20	21:57:00	Abend 000	#trigs 1

Figure 6.19: Reporting-bar supplied in scheduling log panel



Jobname	System	Date	Time	Condition	Misc. Info
JCOBA01	MFOC	Start 2007/11/20	21:54:57	MaxCC 008	#steps 5
		End 2007/11/20	21:59:48	Abend 000	#trigs 0
JCOBA02	MFOC	Start 2007/11/20	21:55:03	MaxCC 000	#steps 2
		End 2007/11/20	21:56:56	Abend 000	#trigs 1

Figure 6.20: Reporting-bar menu

6.6.1 Setting up Report

Before you work with reporting facility, the first step is setting up the report to meet your specific requirement. Not much thing to set, just specifying the name, destination, dataset prefix and number of lines per page. Click the reporting-bar, then reporting pop-down menu appears as in figure 6.20. Then select choice 1 to obtain report setup panel. When you click choice 1, a small window is popped up as shown in figure 6.21, with default information values.

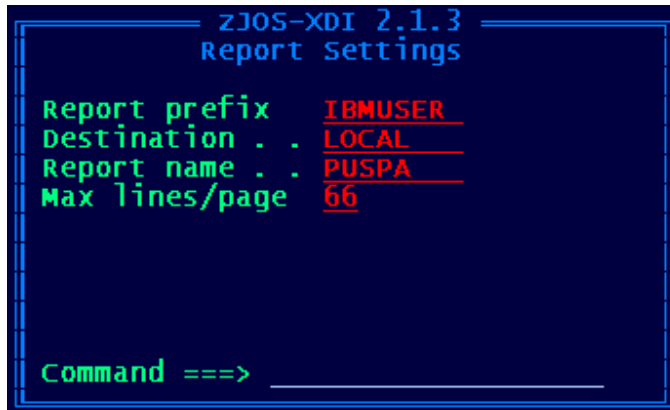


Figure 6.21: Report setting panel

The important thing here is destination. Default destination is LOCAL, which means you tell Puspa not to download the report dataset. If you want AutoXfer to download it onto your workstation, you have to change this to the destination name in which your workstation is included according to your current AutoXfer configuration. Although AutoXfer can use string "LOCAL" as a name of a true destination, Puspa, however, will ignore it.

Prefix, destination and report name will be combined with current date and time to produce flat dataset name into which report is printed out. Hence, all the above information must comply qualifier format of MVS dataset name.

The last field, number of lines per page is the number of physical line per page of hardcopy when report is printed to the paper. Since the report uses ASA code to control the printing mechanism, hence the true number of lines per page will be adjusted according to ASA code on each line.

To confirm your update, press enter-key. Previous panel is then resumed and update is effective. To abort the update, press F12 key, or issue CANCEL command.

6.6.2 Producing Report

To produce the report, click choice 2 on reporting-bar menu as shown in figure 6.20. Puspa take few amount of time while TSCSV module is producing report dataset. Once dataset is complete, Puspa then browse it using standard ISPF browse panel as shown in figure 6.22, to show you the content of the report.



```
BROWSE IBMUSER.JAKARTA.SKEDULER.D2007326.T0012245 Line 00000069 Col 001 080
JOS/Puspa (Scheduler) Report
wkld TSORXCL1 type=JOB id=JOB04764 ASID=0026 on system MFOC run on Tue 2007
  detail steps:
    No. Jobstep Procstep UCC SCC End-time Elapsed
    001 TSO 000 000 21:59:52.45 000004.22
  Triggered by:
    Trigger-src System Step ProcStep SCC UCC Op RCC Grp
    JOB JC0BA01 MFOC 000 008 NC 000 00
wkld JTEST03 type=JOB id=JOB04765 ASID=0024 on system MFOC run on Tue 2007
  detail steps:
    No. Jobstep Procstep UCC SCC End-time Elapsed
    001 COBA01 000 000 22:01:20.96 000010.49
    002 COBA02 000 000 22:01:22.32 000001.36
    003 COBA03 004 000 22:02:14.49 000052.17
    004 COBA04 000 000 22:02:21.70 000007.21
  Triggered by:
    Trigger-src System Step ProcStep SCC UCC Op RCC Grp
    JOB JTEST01 MFOC STPDUA n/a n/a LE 004 00
    JOB JTEST02 MFOC DUA003 000 004 LT 008 00
    JOB JTEST00 MFOC 000 010 GT 000 01
```

Figure 6.22: Upon completion, report is then browsed

```
Report was filed in IBMUSER.JAKARTA.SKEDULER.D2007326.T0012245
```

```
Report was downloaded to JAKARTA
```

Figure 6.23: Completion messages which appear when you leaving browse panel.

Content of the report is exactly the same log shown in scheduling log panel. It is exactly produced from the logged information. Hence you should care when the scheduling is still in progress, both log panel and report will only contain finished workloads. To have complete report, you have to produce again when scheduling cycle is completed.

Report dataset name is qualified combination of prefix, destination name, report name as you specified in report setup panel, and current date (Dyyyymmdd) and time (Thhmmssd). In figure 6.23 shown that the produced report dataset name is IBMUSER.JAKARTA.SKEDULER.D2007326.T0012245. Finally, then message "Report was downloaded to JAKARTA" is shown to acknowledge you that report was downloaded to specified destination by using AutoXfer.

Each time report is produced, regardless it is downloaded, it will remain on your DASD volumes until you delete it. On your choice and responsibility to either keep and maintain all produced reports or just scratch them.



6.6.3 Re-downloading Report

Choice 3 of reporting menu in figure 6.20 is for download report. This gives you a chance to download previously produced report. When you click it, a small panel is popped up as shown in figure 6.24.



Figure 6.24: Download report panel.

On this panel, the most recent report dataset and current destination and report names appear. You can just hit enter-key to re-download the most recent report, or alter them and hit enter-key to download the different thing. When you change either destination or report name (xwtr), it will affect your current setting.



Chapter 7 Integrated Scheduling

System on which workload is running, system on which triggering event source is running have been many times explained in several previous discussion. These indicate that the scope of workload scheduling with Puspa is not only involving a single system. As a modern scheduling system, Puspa provides cross systems workloads scheduling feature. This is called integrated scheduling.

7.1. Integrated zJOS Network

Integrated zJOS network means, zJOS installation which is distributed among host on networked-z/OS. Networked-z/OS in this manner is not a multi-sysplex in cross-coupled (XCF) configuration, instead, is a network of 2 or more interconnected z/OS hosts in TCP/IP protocol. Both hardware and software must meet TCP/IP base network requirements. zJOS server on one z/OS system and several zJOS agents on other hosts which may either z/OS or OS/390 system

7.1.1 Hardware Requirements

To establish TCP/IP network among several z/OS hosts, the following hardware requirements must be complied.

1. z/Series compatible system processor complete with minimum host basic configuration, including DASD, console, terminal display station, tape drive and so forth.
2. TCP/IP capable connection station, such as OSA channel, CTC paired channel, or ordinary channel with XCA attached and so forth.
3. TCP/IP capable connection media, such as ESCON or FICON optical cable, satellite sender/receiver equipment, or ordinary telecommunication cable.
4. IP routing facility as necessary.

All the above materials must be physically installed, connected, setup and well tested. Review each of them and ask vendor support to make sure everything is ready.

7.1.2 Software Requirements

To establish TCP/IP network among several z/OS hosts, the following software requirements must be complied for each z/Series host.

1. Copy of licensed IBM z/OS complete with minimum host basic program configuration, including JES2, TSO, SDSF, ISPF and so forth.
2. Copy of licensed IBM Communication Server (CS) for z/OS which minimum consist of VTAM, TCPIP, VMCF and IUCV.

All the above software materials must be physically installed, well setup and well tested. Review each of them and ask vendor support to make sure everything is ready and comply with the following states:

- Complete z/OS copy is well setup on each z/Series machine, each with a unique system name defined in IEASYSxx parameter.
- Complete CS for z/OS is well setup on each z/Series machine, each with host name (defined in TCP/IP profile) equal to system name (defined in IEASYSxx).
- When z/OS is booted, make sure the following states are complied:
 - JES2, OMVS, VTAM and TCPIP are up.
 - API for socket programming is available.
 - Ping and/or signon (telnet) is well verified.

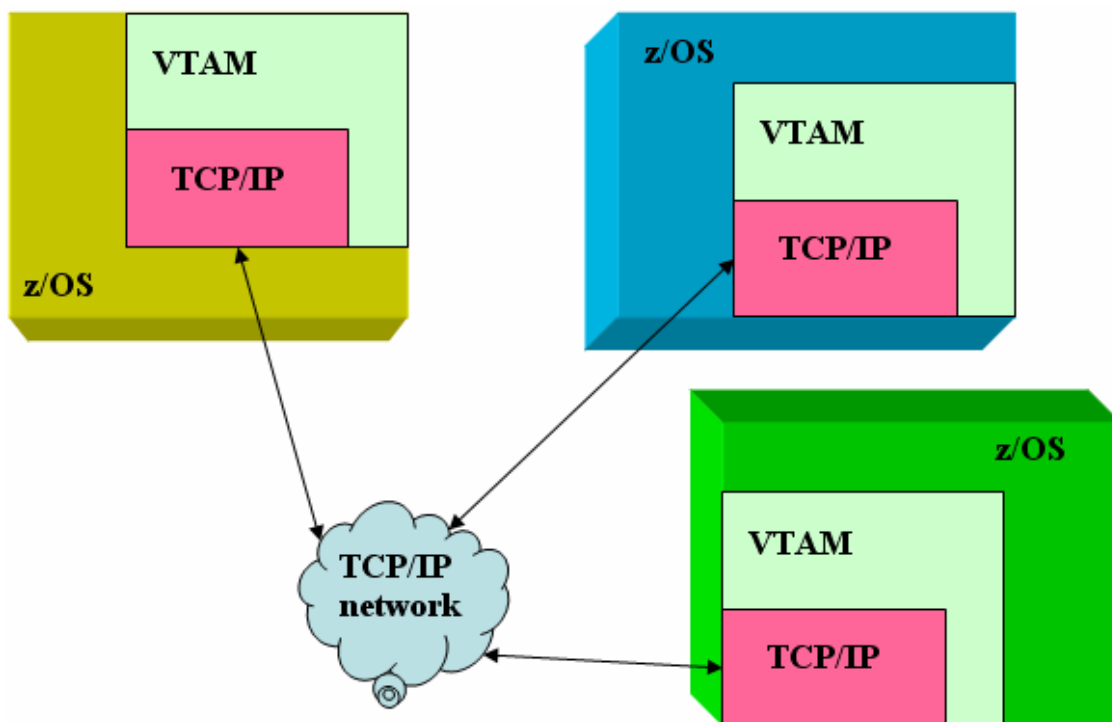


Figure 7.1: Networked-z/OS



7.2. Puspa for Integrated zJOS Network

Starting at zJOS version 2.1.3, Puspa is featured with capability to establish an integrated scheduling on an integrated zJOS network environment. You don't need complete zJOS configuration on each z/OS host in the network, instead the following:

- Complete copy of licensed zJOS/Puspa package on one z/OS system host which is assigned as scheduling server.
- Copy of licensed zJOS agent on each z/OS system host which is assigned as scheduling member or client.

All the above software materials must be physically installed, well setup and well tested. Review each of them and ask vendor support to make sure everything is ready and comply with the following states:

- zJOS address space (XDI) up with Puspa and zJOS socket server active with well-setup parameters on scheduling server machine. Make sure Puspa schedule tables involve all scheduling client machines.
- zJOS agent address space (XDA) which represent Puspa agent up and active on each scheduling client machine.

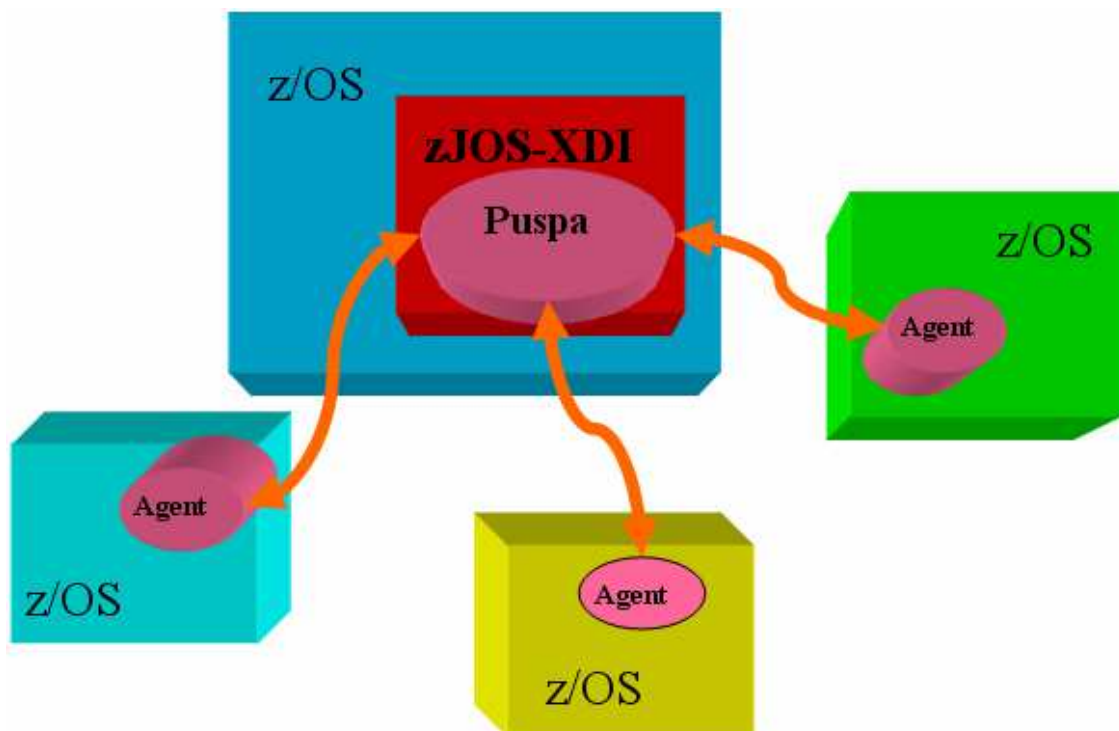


Figure 7.2: Puspa for integrated zJOS network



7.2.1 Preparing zJOS Server

If you have already prepared zJOS Server for Sekar, you don't need to do it again for Puspa. Because, once it is prepared, server will ready for both Sekar and Puspa. To implement an integrated workload scheduling, Puspa must be prepared to accept connection request from each zJOS agent on each z/OS host in the network which are:

1. Make sure your current zJOS-XDI is version 2.1.3 or higher. Else, you have to upgrade it to version 2.1.3 or higher.
2. Make sure zJOS agent is ready on each connected z/OS host which designated to be a member of integrated EMS.
3. Start zJOS server. By default, server is initially down. You can bring it up manually or automate it later. The command is:

```
.SVR START
```

or

```
F XDI,SVR START
```

Or issue START request on control panel as shown in figure 7.3:

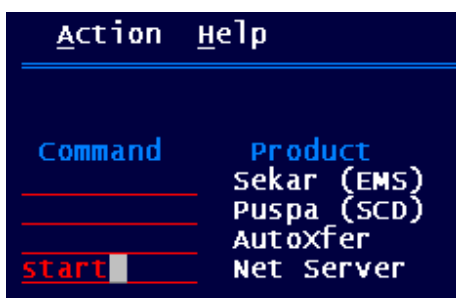


Figure 7.3: Starting zJOS Server

On control panel, zJOS server then indicated as UP and ACTIVE as shown in figure 7.4

Action Help		zJOS-XDI Control Panel						Row 27 to 28 of	
Command	Product	State	Table	Suf	works	Usage	Day		
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LICENSED	none		
	Puspa (SCD)	<UP> READY	loaded	00	000000	**DEMO**	..?!		
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!		
	Net Server	<UP> ACTIVE	N/A	**	000000	standard	none		

Figure 7.4: zJOS Server is in active state

In respond to the above command, zJOS server is brought up as shown in figure 7.5. This figure shows host IP and port number (7777) on which server is

listening agent connection request. Although an IP address is displayed, it does not mean that server will only use this IP. It just shows first IP found in socket address control block. Server will rather use all available IP addresses.

```

00.04.51 STC09591 *DERSVR500I zJOS server initialization in progress...
- 00.04.51 STC09591 DERSVR543I A NETCCB was built for xDI agent on system
- DYAH2003
- 00.04.51 STC09591 DERSVR543I A NETCCB was built for xDI agent on system
- ADI2007
- 00.04.51 STC09591 DERSVR544I 002 NETCCBs were built for xDI agents.
- 00.04.53 STC09591 DERSVR509I Socket 000 and max=099 was obtained for
- task DERSVR
- 00.04.53 STC09591 DERSVR510I server host is dewi2007 IP=100.99.122.2
- 00.04.53 STC09591 DERSVR511I Socket 000 bound to port 07777, IP
- 100.99.122.2
- 00.04.53 STC09591 DERSVR512I server is listening on port 07777
- 00.04.53 STC09591 DERSVR513I MAINTask has confirmed for nonblocking I/O.
- 00.04.53 STC09591 DERSVR515I DERSVR pid=**none** is confirmed as
- TCP/IP client.
- 00.04.53 STC09591 DERSVR501I zJOS server initialization complete.
  
```

Figure 7.5: Console message in respond to zJOS Server activation

Message DERSVR543I shows network connection control block (NETCCB) was built to accommodate agent connection request. Each NETCCB represent one agent connection. To display list of NETCCBs, issue the following command:

```
.LIST NETCCB
```

or

```
F XDI,LIST NETCCB
```

Or issue LIST request on control panel as shown in figure 7.6:

Action	Help
Command	Product
	sekar (EMS)
	Puspa (SCD)
	AutoXfer
List	Net Server

Figure 7.6: Issue LIST request to zJOS Server

In respond to the above command, all NETCCBs are then listed as shown in figure 7.7 and 7.8 Each message DERCMD094I shows remote host name, agent id, connection status, EMS parameter status, scheduler (SCD) parameter status and host coding type (E for EBCDIC and A for ASCII). Conn=N describes no connection yet.



Action	Help	zJOS-XDI Control Panel						Row 29 to 30 of
Command	Product	State	Table	Suf	Works	Usage	Day	
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LICENSED	none	
	Puspa (SCD)	<UP> READY	loaded	00	000000	**DEMO**	..?!	
	Autoxfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!	
*LIST	Net Server	<UP> ACTIVE	N/A	**	000000	standard	none	
Date	Time	Log						
07.271	00:49:00	Agent 01dc8428 on DYAH2003 conn=N EMS=N SCD=N type=E.						
07.271	00:49:00	Agent 01dc80c0 on ADI2007 conn=N EMS=N SCD=N type=E.						
***** Bottom of data *****								

Figure 7.7: List of zJOS agent in respond to LIST request on control panel

```
.LIST NETCCB
DERCMD094I Agent 01DC8428 on DYAH2003 conn=N EMS=N SCD=N type=E.
DERCMD094I Agent 01DC80C0 on ADI2007 conn=N EMS=N SCD=N type=E.
DERCMD201I zJOS is ready to accept command.
```

Figure 7.8: List of zJOS agent in respond to console command .LIST NETCCB

To know whether zJOS socket server is up, issue **.STATUS** console command, or just press enter-key on zJOS control panel. Figure 7.9 shows all zJOS component status and statistics with server up and active.

Action	Help	zJOS-XDI Control Panel					Row 230 to 242 of	
Command	Product	State	Table	Suf	Works	Usage	Day	
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LICENSED	none	
	Puspa (SCD)	<UP> READY	loaded	00	000000	**DEMO**	..?!	
	Autoxfer	DOWN INACTIVE	unloaded	00	000000	**DEMO**	..?!	
	Net Server	<UP> ACTIVE	N/A	**	000000	standard	none	
Date	Time	Log						
07.271	01:06:21	Statistics:						
07.271	01:06:21	Config: SSN=XDI Load=LPA COM=0803B3A0 WSA=00C42F90						
07.271	01:06:21	Subtasks: Major=010 EVX=000 SVR=000 SCD=000 Abn=000						
07.271	01:06:21	Network agents: total=0002 active=0000 local=N/A						
07.271	01:06:21	Network traffic: Snd=00000000 Rcv=00000000 que=00000						
07.271	01:06:21	JES I/F: Up=Y PIT=Y Conn=Y Irdr=Y FR(5=N,12=Y,22=N)						
07.271	01:06:21	Queues: ARQS=00000 SQBS=00000 EOTS=00000 RMG=00000						
07.271	01:06:21	State: NORMAL Parm: SYS=00 EMS=00 SCD=00 DEST=00						
07.271	01:06:21	SCD: Lib=0 O=EVXMS M=EVXMS Pos=EOS/EOJ-RMPL ENQ=FREE						
07.271	01:06:21	SCD Free-pool: SCT=009588 TRG=0199479 EOT=0500000						
07.271	01:06:21	SCD Used-pool: SCT=000412 TRG=0000521 EOT=0000000						
07.271	01:06:21	SCD Curr-pool: SCT=000024 TRG=0000047 EOT=0000000						
***** Bottom of data *****								

Figure 7.9: Status and statistics information on control panel when Server up

zJOS IP port

By default zJOS server uses port 7777. Unless it conflicts with your existing application, you are strongly recommended to leave this default. To change it, you have to manually update your current XDISYSxx member in zJOS parameters library, insert the following parameter:

PORT=nnnn

To make new port number effective, you have to recycle zJOS address space.

7.2.2 Preparing zJOS Agent for z/OS

If you have already prepared zJOS Agent for Sekar, you don't need to do it for Puspa. Because, once it is prepared, agent will ready for both Sekar and Puspa.

In an integrated workload scheduling, Puspa runs only in one z/OS system host, which is designated as scheduling server. Other hosts are called as scheduling member or client. Each scheduling member needs zJOS agent which runs as Puspa partner. To have zJOS agent ready on scheduling member, perform the following 2 simple steps:

1. Install copy of zJOS agent
2. Customize XDA procedure

Install copy of zJOS agent

zJOS agent for z/OS is shipped together in the same zJOS-XDI package. Once zJOS-XDI package is installed in scheduling server, all products including agent are installed. Though, you don't need agent in scheduling server. You rather, need it for scheduling member. To install it on scheduling member, you can easily put its copy onto scheduling member as follow:

- If zJOS load library is resided in shared volume, you only need to catalog it into scheduling member and register it as an APF library in scheduling member.
- If zJOS load library is resided in non-shared volume, you need to copy and catalog it into scheduling member and register it as an APF library in scheduling member.

Customize XDA procedure

zJOS agent for z/OS runs as XDA address space on scheduling member, which is based on XDA procedure JCL generated during zJOS-XDI installation steps. Below is an example of XDA procedure:

```
//XDA PROC V=V2,LVL=19,HLQ=SYS5,SSN=XDA,
//      IP=100.99.125.3,PORT=7777
//AGENT EXEC PGM=DERJXA,REGION=0M,DYNAMNBR=99,
//      TIME=1440,PARM='SSN=&SSN,PORT=&PORT,IP=&IP'
//STEPLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LINKLIB
//          DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LPALIB
//JCLLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..SAMPJOBS
```




PROC card consists of 6 parameter keywords. Keyword V, LVL and HLQ are just for JCL substitution, not mandatory to zJOS-XDI programs. You can modify them as necessary, or eliminate them if you prefer to use fixed datasets names, or just leave it as it is (recommended).

Keyword SSN, IP and PORT are mandatory, since their parameter values are passed to zJOS agent programs. You can change their default value, but you can not eliminate each of them. SSN assigns subsystem name for zJOS agent. Default subsystem name is XDA.

IP keyword specifies scheduling server host IP address. This is the most important point of which you have to customize. This keyword must be exactly scheduling server host IP address.

PORT keyword specifies scheduling server host port number which is used by zJOS server to listen connection request. Although this is the most important point of customization, unless you have to use another port number, you are strongly recommended to leave its default, which is 7777. This keyword must be exactly port number on which zJOS server listen connection requests.

Next card is EXEC card. It specifies that storage and time are unlimited. You should not change anything in EXEC card. Just leave it as it is.

The rest are 2 DD cards for STEPLIB and JCLLIB. STEPLIB must point to load library which contains all zJOS-XDI program modules. Since zJOS Agent for z/OS runs as a privileged program, this load library must be registered as an APF library. Although most of zJOS-XDI modules reentrant even refreshable, you may not place them in LPA nor LNKLIST concatenation. Both can result unpredictable problems. Initialization routine of zJOS agent manages them in very unique way instead. Some modules loaded onto the dynamic LPA, and some others onto the common segment, by means of CSA. The rest must remain in the STEPLIB.

To avoid maintenance redundancy, since zJOS agent modules are placed and maintained together with all zJOS-XDI modules libraries, the best configuration is when STEPLIB address to shared zJOS-XDI load library with zJOS address space in scheduling server. Otherwise, you must only maintain zJOS load library in scheduling server, and you have to clone it to all scheduling member.

JCLLIB DD card is library or concatenated libraries of scheduled workloads, which is used only by Puspa for automatic workloads scheduling. You have to prepare each JCLLIB on each scheduling member to contain a number of batch-job exactly as you defined in schedule table. Note, unlike batch-job workload in server site which can have member name different from jobname, member name and jobname of workload in client site must be the same.

7.3. Puspa Agent for z/OS

Puspa agent for z/OS is actually zJOS Agent for z/OS, which supports both Sekar (EMS) and Puspa (automatic scheduling). Once agent is well setup and up in a z/OS host, this host is then eligible to EMS server as an EMS member, as well as eligible to scheduling server as scheduling member. Refers to paragraph 7.2.2 to install and setup zJOS Agent for z/OS.

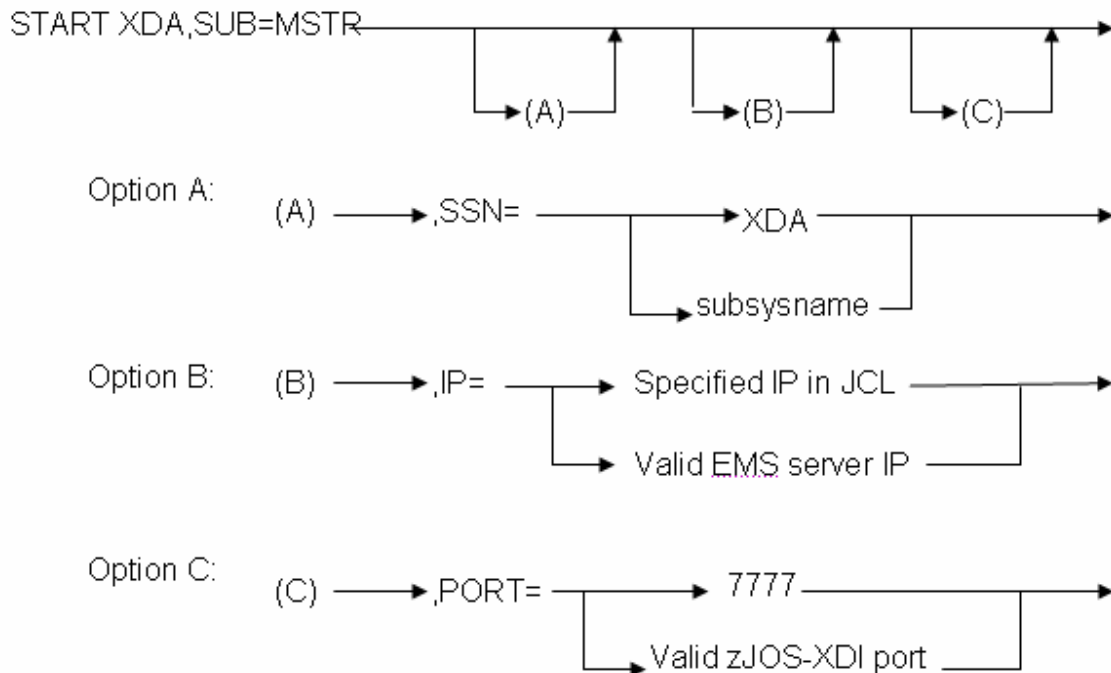
zJOS Agent for z/OS consist of 2 categories of major components, system event listener and IP socket client. System event listener mostly runs as subsystem functions and resource managers to capture occurrence of any event including job status events. Captured events are then posted to socket client task on the agent address space (XDA) to be transferred to Puspa in scheduling server machine. Although runs as a subsystem, system events listener initialization is done on agent address space at the first time agent is started up. Once it get initialized, it will remain exist on the system until next IPL. When agent is brought down, agent subsystem is actually still active, rather, it is placed in on hold status to limit its activities. Hold flag is released when agent is brought back up.

Socket client runs on agent address space to interact locally with system event listener and remotely with Puspa in scheduling server. When event information is posted by system events listener via ECB, socket client task forwards it to Puspa on scheduling server machine via socket.

Besides, in either way, socket client task also executes instruction received from Puspa. The way Puspa instructs all scheduling members is using socket stream and sent to each socket client task. Some certain instructions are interpreted by the socket client task. Some others are forwarded to agent main task for further interpretation

7.3.1 Starting and Stopping zJOS Agent

To start zJOS Agent for z/OS, issue the following command:



SUB=MSTR argument specifies that agent must run under z/OS MVS master scheduler. This parameter is required. You must specify this argument explicitly, exactly as it is. Otherwise, it will result unpredictable situation.

SSN= is an optional argument, to specify subsystem name for zJOS Agent for z/OS. zJOS agent requires to run as a z/OS MVS subsystem. The default subsystem name is XDA. Use this keyword if you prefer to use different name. Valid subsystem name must be 1 to 4 alphameric.

IP= is an optional argument, to specify EMS server IP address on which zJOS server address space runs. The default IP address is one you have specified in XDA procedure as explained in paragraph 7.2.2.

PORT= is an optional argument, to specify port number on which zJOS Server is listening for connection request. The default port number is 7777.

Once the above start command issued, the following steps are then performed:

1. Initialize zJOS Agent subsystem.
2. Attach socket client program as a subtask.
3. Socket client program then does the following steps:
 - a. Login to zJOS server
 - b. Request EMS and scheduler parameters to zJOS server
 - c. Receive EMS and scheduler parameters from zJOS server
 - d. Tell zJOS Agent subsystem that socket is ready.



```
S. XDR, SUB=MSTR
*DERAGT600I zJOS MVS agent initialization in progress...
DERAGT632I Agent successfully done INITAPI
DERAGT632I Agent successfully done GETHOSTNAME
DERAGT632I Agent successfully done GETADDRINFO
DERAGT609I Socket 000 and max=049 was obtained for task DERAGT
DERAGT632I Agent successfully done SOCKET
DERAGT610I Agent host is DYAH2003 IP=100.99.125.2
DERAGT615I DERAGT pid=**none** is confirmed as TCP/IP client.
DERAGT611I Connecting to server port 07777 IP 100.99.122.3
DERAGT612I Socket 000 connected to port 07777, IP 100.99.122.3
DERAGT613I Agent has confirmed for nonblocking I/O.
DERAGT601I zJOS MVS agent initialization complete.
DERAGT616I Logging in to zJOS XDI server...
DERAGT617I AGT task is logged in as agent 022E9730.
DERAGT619I Requesting Event-Mgr parameters.
DERAGT619I Requesting Scheduler parameters.
DERAGT620I Downloading Event-Mgr parameters from zJOS XDI server.
DERAGT621I 0003 EVBs of Event-Mgr parameters were completely downloaded.
DERAGT622W Scheduler parameters are not available.
DERXDM489I Job status listener is being shutted down.
DERXDM490I Job status listener shutdown complete.
```

Figure 7.10: zJOS agent startup on DYAH2003

Figure 7.10 shows agent startup messages when started on DYAH2003 host, where Puspa is running on MFOC host at 100.99.122.3. Shown in message DERAGT621I there are only 3 EMS parameters were sent by zJOS server. No scheduling parameter is available.

In respond to agent startup on DYAH2003 host, zJOS server on MFOC host assigns a subtask named zJOS#001 as a server's worker to be a communication partner to the agent. In common, subtask name is zJOS#nnn, where nnn is a sequent number based on its occurrence.

```
DERSVR514I Contact on socket 099 accepted from port 01027 IP 100.99.125.2
DERSVR516I Socket 099 is being given to worker task.
DERSVR502I zJOS Worker zJOS#001 initialization in progress...
DERSVR517I Socket 099 is taken by zJOS#001 as 001.
DERSVR529I Worker zJOS#001 is responding to up.
DERSVR530I zJOS#001 at TCB=008D0E88 is granted to take over job.
DERSVR513I zJOS#001 has confirmed for nonblocking I/O.
DERSVR503I zJOS worker zJOS#001 initialization complete.
DERSVR518I Socket 099 is disassociated from maintask.
DERSVR545I zJOS#001 is joined to agent on DYAH2003 EBCDIC
DERSVR548I DYAH2003 received NACCB req=ACQR obj=EMS from agent 022E9730 on DYAH2003
DERSVR551I 0002 EVB entries are sent to DYAH2003
DERSVR548I DYAH2003 received NACCB req=ACQR obj=SCD from agent 022E9730 on DYAH2003
DERSVR549W DYAH2003 got error NACCB from DYAH2003 reason: table not loaded yet
DERSVR548I DYAH2003 received NACCB req=SAVE obj=EMS from agent 022E9730 on DYAH2003
DERSVR548I DYAH2003 received NACCB req=ERRD obj=SCD from agent 022E9730 on DYAH2003
```

Figure 7.11: zJOS server accepting agent connection request from DYAH2003

As shown in figure 7.11, when login request accepted, server then change subtask name zJOS#001 to DYAH2003 which represent host name on which the agent is running, and assign agent ID 022E9730 as shown in figure 7.10.



Next step, server receives request from agent to provide EMS and scheduling parameters associated with DYAH2003 host. In this case/example, server sent EMS parameters only, since scheduling (Puspa) is not activated yet.

Once agent up, you do not actually need to stop it unless you want to perform maintenance tasks or because of regular IPL schedule.

To terminate zJOS agent address space (XDA), issue the following command:

-STOP

Although agent address space is now down, XDA subsystem will still remain in memory with status active held. Hence to bring it back up, you can just issue the following command:

-START

7.3.2 Connecting and Disconnecting Agent

As explained in the previous paragraph, once agent is started, it automatically tries to connect to zJOS server. When zJOS server is already up and host on which agent is running is already connected to the TCP/IP network in which zJOS server is connected, as long as IP address and port number are correctly specified in XDA procedure or arguments of START command, you should find agent automatically connected. If not, you should check and make sure all the above stuffs are complied, and then issue the following command:

-CONNECT

Figure 7.12 shows console messages in response to connection request. Upon connection, agent is then receiving EMS and scheduling parameters sent by the server.

To connect agent to other than specified server IP and port in XDA procedure, issue the following command:

-CONNECT IP=xxx.xxx.xxx.xxx PORT=nnnn

Once issued, zJOS Agent remembers the recently used IP and/or port, and become default for next CONNECT command issuance.



```
- 05.32.50 -connect
- 05.32.50 DEXDA476I AGT major subtask is being started.
| 05.32.50 *DERAGT600I ZJOS MVS agent initialization in progress..
- 05.32.54 STC02987 DERAGT609I socket 000 and max=049 was obtained for
- task DERAGT
- 05.32.54 STC02987 DERAGT610I Agent host is DYAH2003 IP=100.99.125.2
- 05.32.54 STC02987 DERAGT615I DERAGT pid=**none** is confirmed as
- TCP/IP client.
- 05.32.54 STC02987 DERAGT611I connecting to server port 07777 IP
- 100.99.122.3
- 05.32.54 STC02987 DERAGT612I socket 000 connected to port 07777, IP
- 100.99.122.3
- 05.32.54 STC02987 DERAGT613I Agent has confirmed for nonblocking I/O.
- 05.32.54 STC02987 DERAGT601I ZJOS MVS agent initialization complete.
- 05.32.54 STC02987 DERAGT616I Logging in to ZJOS XDI server...
- 05.32.54 STC02987 DERAGT617I AGT task is logged in as agent 01092F50.
- 05.32.54 STC02987 DERAGT619I Requesting Event-Mgr parameters.
- 05.32.54 STC02987 DERAGT619I Requesting Scheduler parameters.
- 05.32.54 STC02987 DERAGT620I Downloading Event-Mgr parameters from ZJOS
- XDI server.
- 05.32.54 STC02987 DERAGT621I 0004 EVBS of Event-Mgr parameters were
- completely downloaded.
- 05.32.55 STC02987 DERAGT620I Downloading scheduler parameters from ZJOS
- XDI server.
- 05.32.55 STC02987 DERAGT621I 0004 EOTS of scheduler parameters were
- completely downloaded.
00- 05.32.55 STC02987 DEXDA456I Job status listener initialization in
- progress.
- 05.32.55 STC02987 DEXDA457I Job status listener initialization complete.
```

Figure 7.12: Connecting agent to server

To stop agent interaction activities, issue the following command:

-DISCONNECT

This will cause socket client subtask logoff from server and terminate connection. Agent address space remains up and ready for next connection request. Figure 7.13 shows console messages in response to disconnection request, which is responded by server as shown in figure 7.14.

```
- 05.24.04 -disconnect
- 05.24.04 STC02986 DERAGT614I socket 000 was closed.
- 05.24.04 STC02986 DERAGT611I Agent termination complete.
00- 05.24.05 DEXDA474I AGT major subtask TCB=008EE258 was
- terminated normally.
```

Figure 7.13: Disconnecting agent from server

```
- 05.11.57 STC09591 DERSVR539I Traffic on socket 001 of worker DYAH2003
- was shutdown.
- 05.11.57 STC09591 DERSVR520I socket 001 of worker DYAH2003 was closed.
00- 05.11.58 STC09591 DERSVR531I worker DYAH2003 termination complete.
```

Figure 7.14: Server response when agent is signing off.



In case there is a problem with networking, for example if TCP/IP stack (address space) is unexpectedly down, DISCONNECT request will not be responded correctly, you have to use DROP to force detach socket client subtask as follow:

-DROP

Take a note that DROP command is for emergency case only. It just to issue DETACH to detach socket client subtask, to give you a chance to shutdown the agent normally. Without dropping the socket, agent will not be able to shutdown normally.

You should not issue DROP to do normal disconnection. Once DROP is issued, you should not issue CONNECT to reconnect to the server. No guarantee that stable interaction activities will be achieved. To have better reconnection, you should issue STOP then START to recycle agent address space.

7.3.3 Controlling zJOS Agent

Display agent status information

To get agent status information summary, issue the following command:

-STATUS

Then agent current status is displayed as shown in figure 7.15. Agent identifier, subsystem name and worker (work partner) name are displayed as well as server IP address and port number. Current interaction status with and number of received parameters from Sekar and Puspa are also displayed.

```
00- 05.36.05      -status
- 05.36.05 STC02987 DERSPY453I zJOS XDI agent status:
- Agent ID=01D92F50 SSI=XDA name=DYAH2003
- Server address 100.99.122.3 port=7777
- Sekar (Event-Mgr) active|ready #EVBS=0004
- Puspa (Scheduler) active|ready #EOTS=0004
- SSI-FC12=SHARED Trace=OFF #NACCB-in-Q=0000
- Config Load=CSA ZCA=08024098 COM=0803B3A0
```

Figure 7.15: Agent summary of status information.

Display list of parameters received from server

There are 2 types of parameters which are received from zJOS server, EMS and scheduling parameter. Each scheduling parameter is represented by end-of-task control block (EOT). Issue the following command to display all EOTs:

-LIST SCD or -LIST EOT

XDA then display response as shown in figure 7.16.

```
- 05.37.55 -list scd
- 05.37.55 STC02987 DERSPY494I EOT table of content:
- 05.37.55 STC02987      0001 JOB=JRMT001 Jstep=(blanks) Pstep=(blanks)
- 05.37.55 STC02987      0002 JOB=JRMT002 Jstep=(blanks) Pstep=(blanks)
- 05.37.55 STC02987      0003 JOB=JRMT003 Jstep=(blanks) Pstep=(blanks)
- 05.37.55 STC02987      0004 JOB=JRMT003 Jstep=RSTEP2  Pstep=(blanks)
```

Figure 7.16: Agent show list of scheduling parameter

Each EMS parameter is represented by event parameter control block (EVB). Issue the following command to display all EVBs:

-LIST EMS or -LIST EVB

Requesting scheduling parameters

When agent is started, connection to server normally establish automatically. Then server also automatically send portion of EMS parameters designated to this agent, when one ready by the time. Else, zJOS server will send later as soon as one ready.

In case server got missed, you can ask server to send scheduling parameter to this agent by issuing the following command:

-GET SCD

Agent then reissue request to the server.

7.3.4 Remote Command

Remote command is actually a part of EMS functions, instead of scheduling. Though, in an integrated scheduling system, such facility could be useful to either to support scheduling functions or just a tool to control scheduling agent from server site. Because, by using remote command facility, you can execute any valid console command (including agent command) on any connected agent site. Note that, remote command facility is just to send command text and request to agent to execute it. It does not return the respond nor condition code back to the server.

When an integrated EMS is established on your integrated zJOS network environment, by means zJOS/Sekar, you then can pass any command from EMS server to any connected EMS client. This facility called remote command facility. To issue remote command, use the following syntax:



RCMD hostname command_text

or

RC hostname command_text

Where:

Hostname must be a valid EMS member host name

Command_text is a string containing command verb and its arguments.

```
RC DYAH2003 -STATUS
*DERCMD088I NACCB is posted to schedule on DYAH2003 action XDA STATUS
DERCMD201I zJOS is ready to accept command.
RC DYAH2003 D A,L
*DERCMD088I NACCB is posted to schedule on DYAH2003 action D A,L
DERCMD201I zJOS is ready to accept command.
```

Figure 7.17: Issuing remote command to DYAH2003 from EMS server

Figure 7.17 shows RC DYAH2003 -STATUS and RC DYAH2003 D A,L are issued on EMS server console. Both command texts are then sent to DYAH2003 site for execution. On DYAH2003 site, both command are received and then executed as shown in figure 7.18.

```
DERAGT635I Executing 116 bytes cmd> -STATUS
DERSPY453I zJOS XDI agent status: 388
  Agent ID 022E9730 SSI XDA name DYAH2003
  Server address 100.99.122.3 port=7777
  Sekar (Event-Mgr) active|ready #EVBS=0003
  Puspa (Scheduler) unavailable #EOTS=0000
  Trace OFF Number of enqueued NACCB=0000
DERAGT635I Executing 116 bytes cmd> D A,L
IEE114I 05.53.43 2007.072 ACTIVITY 390
JOBS  M/S  TS USERS  SYSAS  INITS  ACTIVE/MAX VTAM  ORS
00002  00013  00001  00030  00012  00001/00040  00008
LLA  LLA  LLA  NSW S  XDI  XDI  XDIEXEC  NSW S
VLE  VLE  VLE  NSW S  RACE  RACE  RACE  NSW S
JES2  JES2  IEFPROC  NSW S  DLF  DLF  DLF  NSW S
VTAM  VTAM  ACEVTAM  NSW S  TSO  TSO  TCAS  OWT S
SDSE  SDSE  SDSE  NSW S  TCPIP  TCPIP  TCPIP  NSW SO
HTTPD1  HTTPD1  WEBSRV1  IN  SO INETD4  STEP1  OMVSKERN  OWT RO
PORTMAP  PORTMAP  PMAP  OWT SO FTPD1  STEP1  FTPD  OWT RO
XDA  XDA  AGENT  NSW SO
DERU  OWT
```

Figure 7.18: Agent on DYAH2003 responding remote command from MFOC (server)

Remote command is actually one of basic functions in integrated EMS feature of zJOS/Sekar. This function is used by Sekar to perform actions in EMS member. Hence, remote command is useful indicator to verify whether you have setup an integrated EMS correctly.



7.3.5 Remote Job Submission

Remote job submission is another EMS facility which is actually a special form of remote command to ask agent to submit a job. To issue remote job submission, use the following syntax:

```
RJOB hostname jobname
```

Where:

Hostname must be valid EMS member host name

Jobname is a string containing jobname. Jobname must be a member name of job JCL library pointed by JCLLIB in XDA procedure.

7.4. Planning The Integrated Scheduling

Mainframe is normally used to handle large scale business computing, especially when extra security and high transaction intensity are prioritized. In modern mainframe system application, such objectives could be implemented in several ways. The most popular ways are:

- Conventional disaster recovery system
- Geographically dispersed parallel sysplex (GDPS)

Conventional disaster recovery system

This way is very simple. One mainframe is connected remotely to production mainframe and act as disaster recovery (DR) machine. Connection is based on I/O channel protocol and implemented using high speed optical channel cable.

Workload movement from production system to DR machine is handled by either one of special data mover, remote copy or mirroring technology. Remote copy, as illustrated in figure 7.19, is older technology which is a program on either one of both CPUs. Whereas, mirroring, as illustrated in figure 7.20, is an integrated modern disk storage technology, which is running on disk storage subsystem and independent from CPU. DR machine is just like insurance, sleeping all the time until disaster on production site happens.

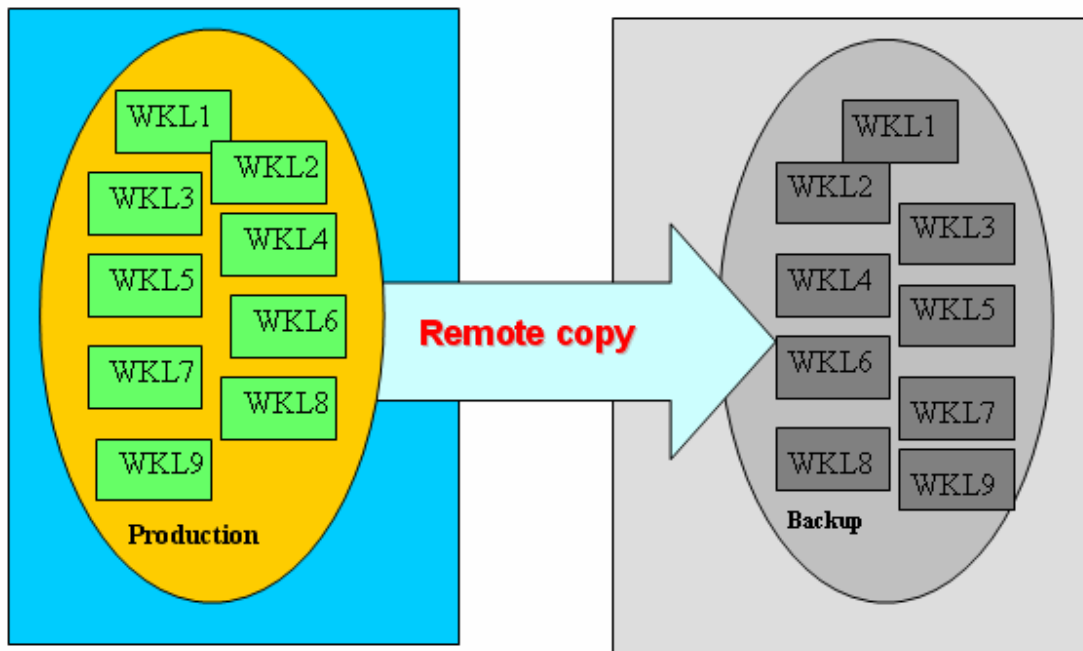


Figure 7.19: Conventional DR based on remote copy data mover

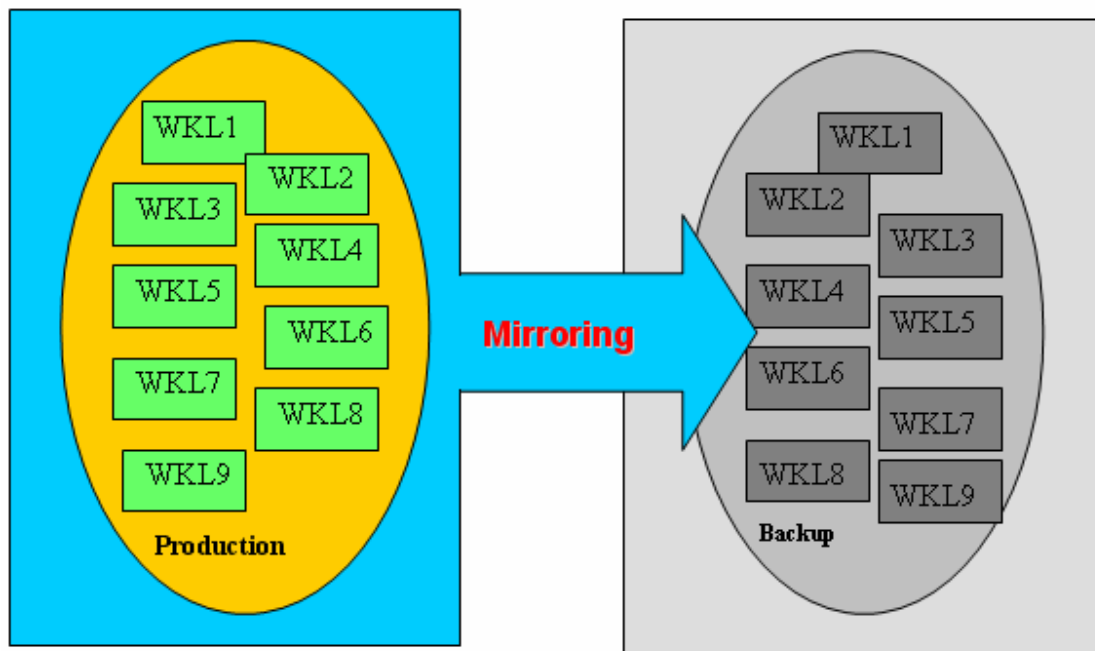


Figure 7.20: Conventional DR based on mirroring data mover



Geographically dispersed parallel sysplex (GDPS)

GDPS is a modern mainframe configuration which is supported by parallel sysplex technology. Unlike in conventional mainframe system where DR host appears as a separate system. Within GDPS system there is no separate DR machine. Each machine contributes as a part of production system. The whole GDPS configuration appears as a single system.

Workload movements are managed perfectly by workload manager (WLM) and supported by advanced copy/mirror technology which runs independently on disk storage subsystem. Data and backup are perfectly managed on each machine. The way workloads are managed illustrated in figure 7.21. When one machine disaster, workloads on that machine are automatically moved to and brought up on other machines.

GDPS guarantees no machine is sleeping. All machine are working together to support production. Hence, when full capacity of DR is recruited as production image in GDPS, it becomes an additional extra power, and of course the whole performance is drastically increased. You don't pay extra cost for just a sleeping machine.

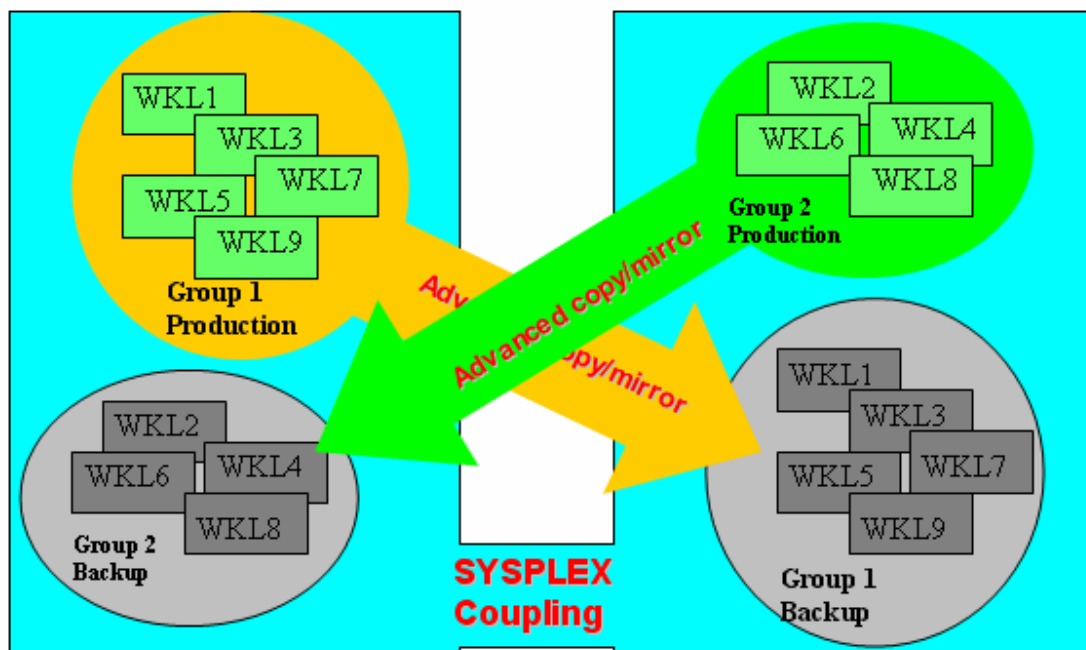


Figure 7.21: GDPS system

7.4.1 Puspa Concept versus GDPS

As discussed previously in this chapter, GDPS is a way to implement an efficient and secured mainframe system. No separate DR machine, but each machine is production as well as DR machine.

zJOS/Puspa concept of integrated scheduling is one of GDPS simpler approach. With Puspa, you will be able to manage workload on all hosts within integrated zJOS network centrally. Relationships and dependencies between a workload with either its predecessors or successors are dispersed inter-hosts as illustrated in figure 7.22. Combined with integrated EMS (by means zJOS/Sekar), you will have a fully integrated automation system.

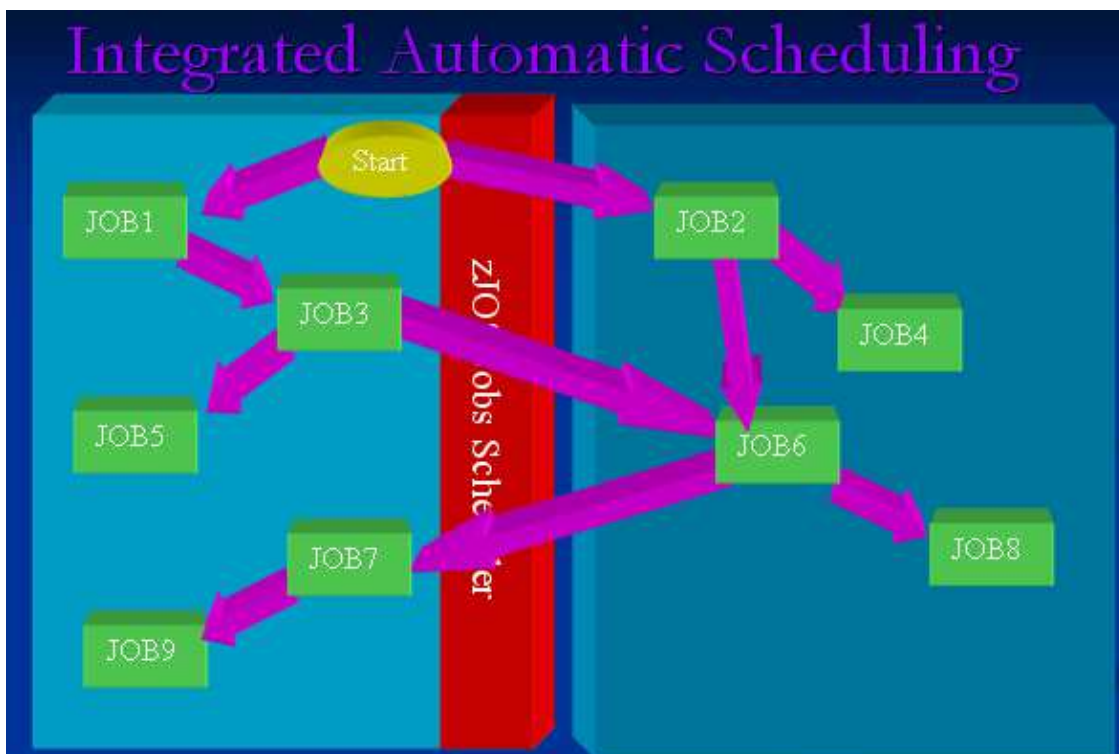


Figure 7.22: Puspa concept of integrated scheduling

Although workloads movement in the integrated zJOS network is not as flexible as GDPS which is supported by parallel sysplex technology, Puspa is easier and cheaper to implement. Nevertheless, Puspa is not alternative for GDPS, rather, just a simpler approach to GDPS. You will even achieve much better integrated scheduling system when you implement Puspa in the GDPS system.

Besides, not every system is feasible for GDPS. If your system has unique workloads characteristics which is impossible to be distributed, then GDPS is not feasible for you. You will pay high cost for nothing.

With Puspa, you get lucky if your workloads are distributable. Otherwise, nothing to loose since you can still use Puspa for normal workloads scheduling system. You don't use zJOS agent, so you don't need to pay it.

7.4.2 Modernize Conventional DR with Puspa

GDPS is a new technology in this decade, whereas most of mainframe shops are already using conventional system. To migrate existing conventional system to GDPS is not a simple effort. New parallel sysplex hardware is installed. System hardware configuration totally changed. Software on each particular machine is also reconfigured to accommodate MULTIPLEX configuration. All efforts need high level expertise to handle.

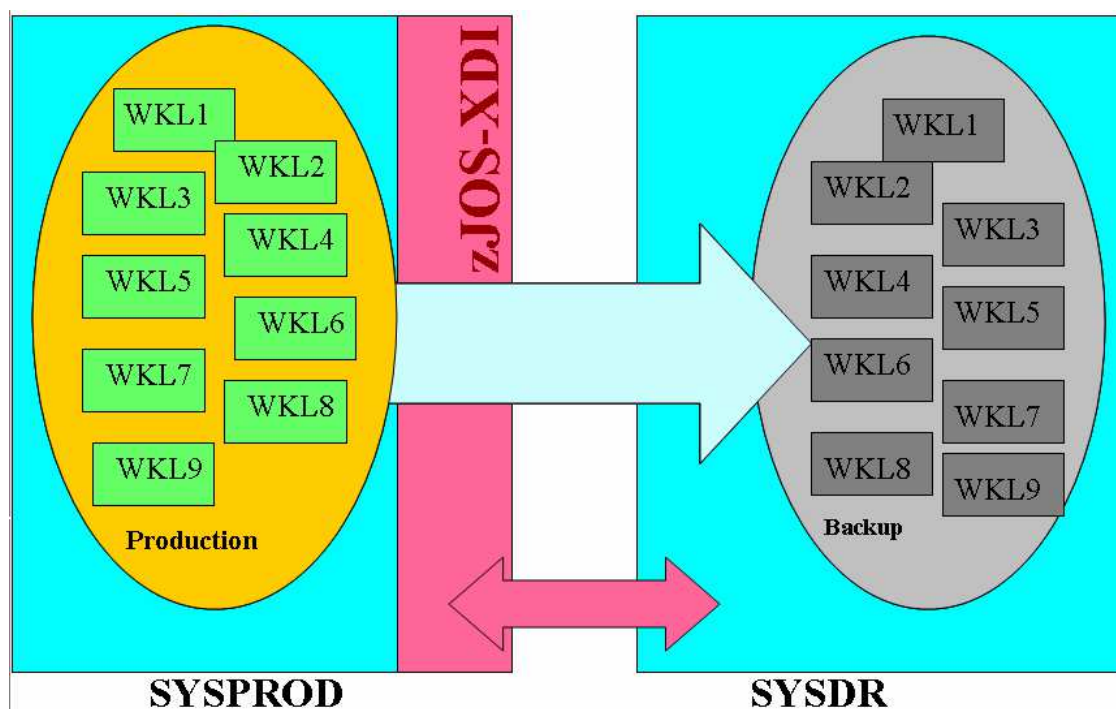


Figure 7.23: Conventional system configuration

Unless you feel very urgent to migrate to GDPS system, you can modernize your conventional without too much critical efforts with zJOS/Puspa. *The first step* is installing and implementing workload scheduling system with Puspa on your production machine. Have zJOS agent ready on DR machine, hence, logically zJOS network is ready on your conventional system. Tune schedule flow with better triggering mechanism (by means step level triggering) to obtain the best turn-around time.

Second step is learning your workloads characteristics to prepare workloads grouping. Which workloads are logically and physically possible to be separated as illustrated in figure 7.24. This step is a key and also required even when you implement GDPS.

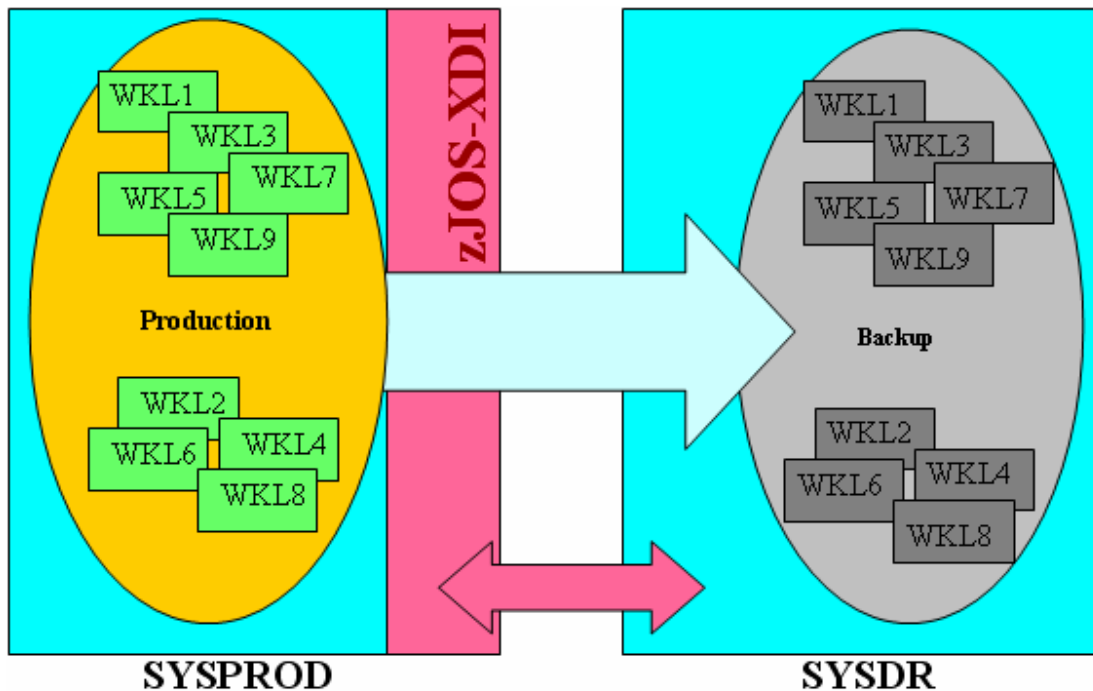


Figure 7.24: Preparing for workload grouping

Third step is workloads grouping. All separable workloads are then moved onto separate area in production system as illustrated in figure 7.25. This to make you easier to move them onto DR machine. Meanwhile, on DR machine you do the same thing.

Fourth step is distributing workloads groups and update Puspa schedule table. It does not mean you need to physically move the all content of desired group onto DR machine. Rather, just make data of desired group on DR machine as current data. Because, similar groupings should already exist on DR machine since you have grouped both sites during third step. At the same time, you have to update Puspa schedule table for all moved workloads. Unless the content of some moved workloads are changed, this is a very simple effort. All you need is just change the name of system for all moved workloads.

Unlike flexibility of GDPS, with Puspa, workloads are distributed permanently on each host. For balancing, you need to measure each particular workload.

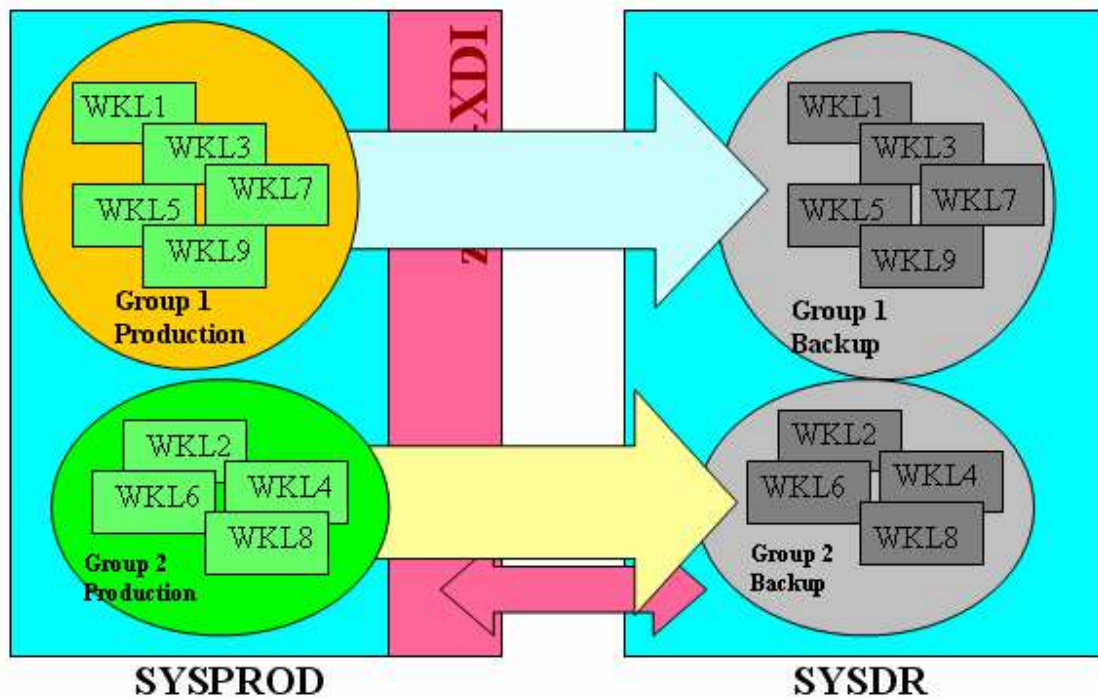


Figure 7.25: Workload grouping

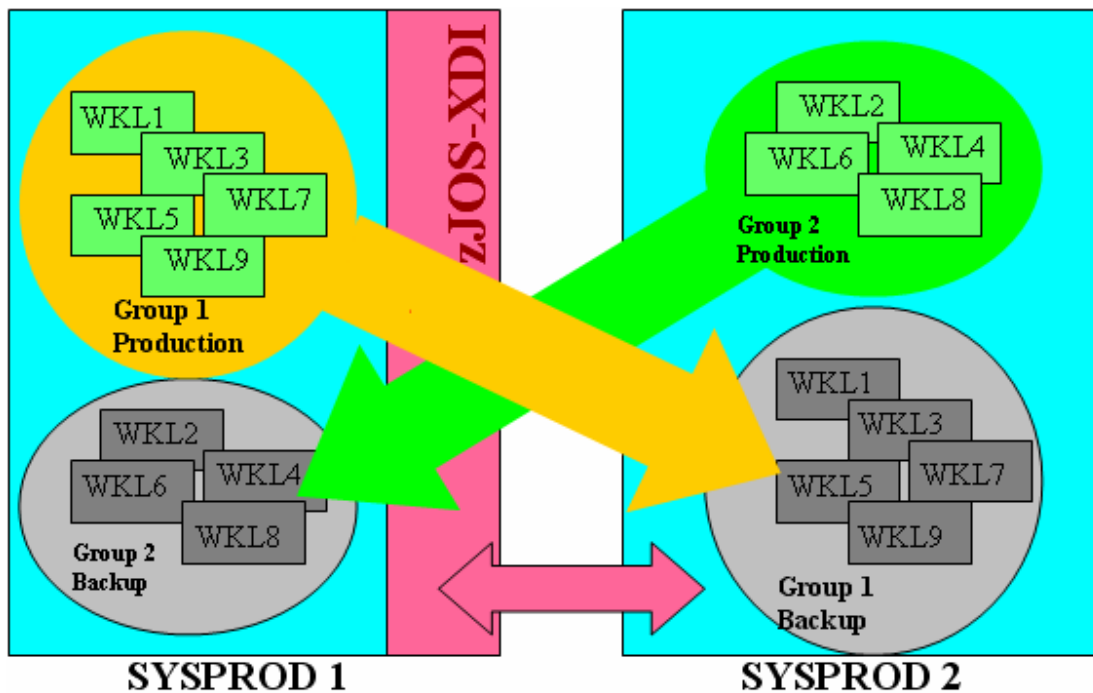


Figure 7.26: Final of workload grouping

Fifth step is change data moving direction. Once moved group of workloads was accommodated in the scheduling system, the moved workloads group then will be operated on DR machine in next operation cycle. Therefore, in next cycle, the



separate DR machine will no longer exist. Both sites become production sites. Therefore, moved workloads group must be backed up by the origin machine as illustrate in figure 7.26. As consequence, data moving direction of moved group must be changed to the opposite way.

Finally, your conventional system is now changed look like GDPS. Both sites are production systems and both sites are DR systems. Now you will achieve much better system performance since the production power is now doubled.



Chapter 8 Operate Integrated Scheduling System

Operating an integrated scheduling system with Puspa is vary similar as when you operate standalone scheduling system as discussed in chapter 6. The only different is just starting zJOS server task on server site and starting zJOS agent address space on each client site. See figure 7.3 and 7.4 in chapter 7 to review how to start zJOS server task. And see also par 7.3 in chapter 7 to review how to start zJOS agent address space.

8.1. Reviewing Agent-Server Connection

Assuming that zJOS server is up on scheduling server site, then when you start zJOS agent address space on scheduling client site, agent automatically sign on to server as shown in figure 8.1. Once agent is connected, server then sent all defined EMS and scheduling parameters to the agent as shown in figure 8.2. This the critical point that you should review.

```
- 04.10.03 STC02986 DERAGT615I DERAGT pid=**none** is confirmed as
- TCP/IP client.
- 04.10.03 STC02986 DERAGT611I Connecting to server port 07777 IP
- 100.99.122.3
- 04.10.03 STC02986 DERAGT612I socket 000 connected to port 07777, IP
- 100.99.122.3
- 04.10.03 STC02986 DERAGT613I Agent has confirmed for nonblocking I/O.
- 04.10.04 STC02986 DERAGT601I zJOS MVS agent initialization complete.
- 04.10.04 STC02986 DERAGT616I Logging in to zJOS XDI server...
- 04.10.04 STC02986 DERAGT617I AGT task is logged in as agent 01d92FA8.
- 04.10.04 STC02986 DERAGT619I Requesting Event-Mgr parameters.
- 04.10.04 STC02986 DERAGT619I Requesting Scheduler parameters.
- 04.10.04 STC02986 DERAGT620I Downloading Event-Mgr parameters from zJOS
- XDI server.
- 04.10.04 STC02986 DERAGT621I 0004 EVBs of Event-Mgr parameters were
- completely downloaded.
- 04.10.04 STC02986 DERAGT620I Downloading scheduler parameters from zJOS
- XDI server.
- 04.10.04 STC02986 DERAGT621I 0007 EOTs of scheduler parameters were
- completely downloaded.
- 04.10.04 STC02986 DERXDA456I Job status listener initialization in
- progress.
- 04.10.04 STC02986 DERXDA457I Job status listener initialization complete.
```

Figure 8.1: Messages in agent site showing agent startup and sign on to server



```

- 03.57.56 STC09591 DERSVR514I Contact on socket 099 accepted from port
- 01027 IP 100.99.125.2
- 03.57.56 STC09591 DERSVR516I Socket 099 is being given to worker task.
- 03.57.56 STC09591 DERSVR502I zJOS worker zJOS#001 initialization in
- progress...
- 03.57.56 STC09591 DERSVR517I Socket      is taken by zJOS#001 as 001.
- 03.57.56 STC09591 DERSVR529I Worker zJOS#001 is responding to up.
- 03.57.56 STC09591 DERSVR513I zJOS#001 has confirmed for nonblocking I/O.
- 03.57.56 STC09591 DERSVR530I zJOS#001 at TCB=008D6378 is granted to take
- over job.
- 03.57.56 STC09591 DERSVR518I Socket 099 is disassociated from maintask.
- 03.57.56 STC09591 DERSVR503I zJOS worker zJOS#001 initialization
- complete.
- 03.57.56 STC09591 DERSVR545I zJOS#001 is joined to agent on DYAH2003
- EBCDIC
- 03.57.56 STC09591 DERSVR551I 0016 EVB entries are sent to DYAH2003
- 03.57.56 STC09591 DERSVR551I 0007 EOT entries are sent to DYAH2003
    
```

Figure 8.2: Messages in server site showing agent startup and sign on to server

8.1.1 Reviewing Server Site

On server site, connection can easily be reviewed from zJOS control panel. On the product status region, Net Server component should be stated as UP and ACTIVE as shown in figure 8.3. On the statistics logs region, see highlighted part. Major subtasks are at least 10. When Puspa state ACTIVE, at least is 11. You can also check that the number of active network agents should increase as much as number of newly connected agents.

```

Action Help
-----
zJOS-XDI Control Panel
Row 159 to 171 of

Command      Product      State      Table      Suf  works  Usage  Day
-----
          Sekar (EMS)  <UP> ACT(SS) loaded    00  000000 LICENSED none
          Puspa (SCD)  <UP> READY  loaded    00  000000 **DEMO** ..?!
          Autoxfer     DOWN INACTIVE unloaded  00  000000 **DEMO** ..?!
          Net Server   <UP> ACTIVE   N/A      **  000014 standard none

Date      Time      Log
07.271    05:27:48
07.271    05:27:48 Statistics:
07.271    05:27:48 Config: SSN=XDI Load=LPA COM=0803B3A0 WSA=00C42F90
07.271    05:27:48 Subtasks: Major=010 EVX=000 SVR=001 SCD=000 Abn=000
07.271    05:27:48 Network agents: total=0002 active=0001 local=N/A
07.271    05:27:48 Network traffic: snd=00000006 rcv=00000008 que=000000
07.271    05:27:48 JES I/F: Up=Y PIT=Y Conn=Y IrdR=Y FR(5=N,12=Y,22=N)
07.271    05:27:48 Queues: ARQS=00000 SQBS=00000 EOTs=00000 RMG=00000
07.271    05:27:48 State: NORMAL Parm: SYS=00 EMS=00 SCD=00 DEST=00
07.271    05:27:48 SCD: Lib=0 o=EVXMS M=EVXMS Pos=EOS/EOJ-RMPL ENQ=FREE
07.271    05:27:48 SCD Free-pool: SCT=009588 TRG=0199479 EOT=0500000
07.271    05:27:48 SCD Used-pool: SCT=000412 TRG=0000521 EOT=0000000
07.271    05:27:48 SCD Curr-pool: SCT=000024 TRG=0000047 EOT=0000000
***** Bottom of data *****
    
```

Figure 8.3: Connection status and statistics on control panel.



To see more detail regarding agent connection, you can obtain list of agents by issuing LIST request on Net Server command slot as shown in figure 8.4. List of agents is then displayed on logs of control panel as shown in figure 8.5.

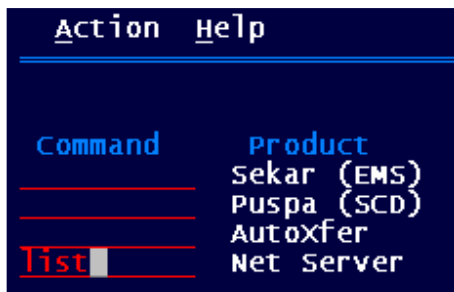


Figure 8.4: Requesting list of agents.



Figure 8.5: List of agents.

On the list, status of each agent is shown discussed in sub par 7.2.1 in chapter 7. For integrated scheduling, each desired agent must have conn=Y and SCD=Y, telling that agent is connected and scheduling parameters has already been received.

8.1.2 Reviewing Agent Site

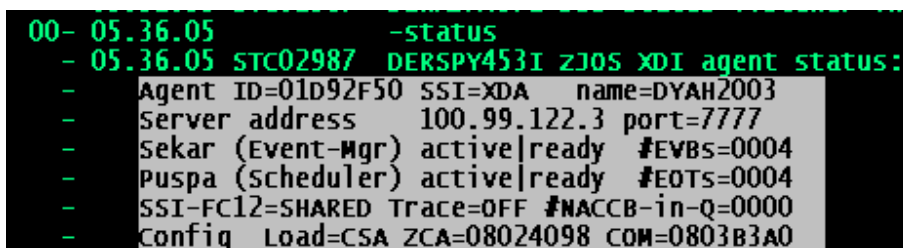


Figure 8.6: Agent status displayed by agent on agent site.

To see detail status of each agent, you need to issue **-status** command on console of system host on which agent is running. Figure 8.6 shows agent status in respond to **-status** command. Status information involves agent ID, name

of agent subsystem, name of agent (IP host name), server IP address and port number, and agent readiness to work with Sekar and Puspa.

#EVB – number of EMS parameters

Since each EMS parameter is contained in event parameter control block (EVB), so number of EVBs represents number of EMS parameters. This value indicates whether agent acts as Sekar agent. To list all EMS parameters on this agent site, issue `-list ems` command on agent site console as shown in figure 8.7.

#EOT – number of scheduling parameters

Since each scheduling parameter is contained in end-of-task control block (EOT), so number of EOTs represents number of scheduled workload resided on this agent site. This value indicates whether agent acts as Puspa agent. To list all scheduling parameters on this agent site, issue `-list scd` command on agent site console as shown in figure 8.7.

```
- 05.37.55 -list scd
- 05.37.55 STC02987 DERSPY494I EOT table of content:
- 05.37.55 STC02987 0001 JOB=JRMTO01 Jstep=(blanks) Pstep=(blanks)
- 05.37.55 STC02987 0002 JOB=JRMTO02 Jstep=(blanks) Pstep=(blanks)
- 05.37.55 STC02987 0003 JOB=JRMTO03 Jstep=(blanks) Pstep=(blanks)
- 05.37.55 STC02987 0004 JOB=JRMTO03 Jstep=RSTEP2 Pstep=(blanks)
00- 05.38.16 -list ems
- 05.38.16 STC02987 DERSPY494I EVB table of content:
- 05.38.16 STC02987 0001 Event=MSG Key(08)=TESTONLY
- 05.38.16 STC02987 0002 Event=CMD Key(05)=JAJAL
- 05.38.16 STC02987 0003 Event=CMD Key(04)=DSMF
- 05.38.16 STC02987 0004 Event=??? Key(00)=
```

Figure 8.7: List of EMS and scheduling parameters on agent site.

8.2. Monitoring Integrated Schedule

Once agents-server connections are establish and each agent is ready to act as Puspa agent, an integrated scheduling system is ready to operate. Although internally some networking activities are added, to operate integrated scheduling system is nothing specific. You can refer to chapter 6 for detail guidance.

8.2.1 Monitoring Syslog

Networking activities are logged in both agent and server syslog. Figure 8.8 shows scheduling activities logged into server site syslog. Shown that each time a remote workload is scheduled, an NACCB (network access communication control block) is sent to targeted agent. NACCB is a common vehicle in zJOS



network environment to exchange information including requests and responses between agents and server in IP streaming mode.

```

DERSCD087I NACCB is posted to schedule job JRMT001 on DYAH2003
DERSCS318I Job JRMT001 is triggered at 05:35:07 on Fri 2007/09/28.
DEREVX890I 01 EOS JCOBA03 (JOB09609) S(COBA03 /**none**) S000-U004
HSDV/HDIV on MFOC
DERRMG799I EOS A=0023/JCOBA03 Tr=Y St=(*noname*/COBA03 ) S#=003
CC=S000-U004 I=N Sys=Local
DEREVX890I 01 EOS JCOBA03 (JOB09609) S(COBA04 /**none**) S000-U000
HSDV/HDIV on MFOC
DERSCD087I NACCB is posted to schedule job JRMT002 on DYAH2003
DERSCS318I Job JRMT002 is triggered at 05:35:22 on Fri 2007/09/28.
DERRMG799I EOS A=0023/JCOBA03 Tr=Y St=(*noname*/COBA04 ) S#=004
CC=S000-U000 I=N Sys=Local
IEF404I JCOBA03 - ENDED - TIME=05.35.27
DERSCD085W Agent on ADI2007 unconnected, so can not schedule job
JADI002
DEREVX890I 01 EOJ JCOBA03 (JOB09609) S(**all**/**all**) S000-U00C
SCUT/EOJF on MFOC
DERSCD085W Agent on ADI2007 unconnected, so can not schedule job
JADI002
DERSCD085W Agent on ADI2007 unconnected, so can not schedule job
JADI001
DERSCD085W Agent on ADI2007 unconnected, so can not schedule job
JADI001

```

Figure 8.8: Logged activities in server site syslog.

```

DERAGT636I Submitting job JRMT001
$HASP100 JRMT001 ON INTRDR FROM STC02987 XDA
IRR010I USERID START2 IS ASSIGNED TO THIS JOB.
ICH70001I START2 LAST ACCESS AT 04:09:59 ON FRIDAY, SEPTEMBER 28, 2007
$HASP373 JRMT001 STARTED - INIT 1 - CLASS A - SYS SYS1
IEF403I JRMT001 - STARTED - TIME=05.47.18
DERAGT636I Submitting job JRMT002
$HASP100 JRMT002 ON INTRDR FROM STC02987 XDA
IRR010I USERID START2 IS ASSIGNED TO THIS JOB.
ICH70001I START2 LAST ACCESS AT 05:47:18 ON FRIDAY, SEPTEMBER 28, 2007
$HASP373 JRMT002 STARTED - INIT 2 - CLASS A - SYS SYS1
IEF403I JRMT002 - STARTED - TIME=05.47.31
DERARM799I EOS A=0022/JRMT001 Tr=N St=(*noname*/JR01S01 ) S#=001
CC=S000-U000 I=N Send=Y
IEF404I JRMT001 - ENDED - TIME=05.48.20
DERRMJ799I EOJ A=0022/JRMT001 Tr=M St=(*noname*/*noname*) S#=***
CC=S000-U004 I=J Sys=Local
DERARJ799I EOJ A=0022/JRMT001 Tr=Y St=(*noname*/*noname*) S#=***
CC=S000-U004 I=J Send=Y
$HASP395 JRMT001 ENDED
SE '05.48.21 JOB02988 $HASP165 JRMT001 ENDED AT N1 MAXCC=4', LOGON,
USER=(START2)

```

Figure 8.9: Logged activities in agent site syslog.

In agent site syslog, networking activities are not explicitly described as on server site. Most messages only explaining workload activities. The only indication is a remark of **send=Y** on message DERARM799I as shown in figure 8.9, explains that the workload status information is being sent to server. Although not explicit, all agent activities are supposed to be networking activities, since agent has no initiative unless requested by zJOS server.

8.2.2 Monitoring Schedule Log

Although the way remote workload information collected is totally different, but it is logged into the same place and formatted with the same template as local workload. Hence its appearance and the way you monitor on schedule log panel, almost no different. The only thing to indicate is system name, which is not local system name. In figure 8.10, MFOC is the name of local system, on which zJOS server is running. Hence, any workload with system name not MFOC must be a remote workload. Figure 8.10 shows remote jobs, JRMT001 and JRMT002 while in progress. They look the same as local jobs.

Scheduled-jobs Log										Row 1 to 9 of	
S	Jobname	= System =	Date	Time	Condition	= Misc.	Info =				
	JTEST00	MFOC	Start	2007/09/28 05:31:43	MaxCC	010	#steps	3			
			End	2007/09/28 05:33:36	Abend	000	#trigs	0			
	JCOBA01	MFOC	Start	2007/09/28 05:31:43	MaxCC	...	#steps	3			
			End	Abend	...	#trigs	0			
	JCOBA02	MFOC	Start	2007/09/28 05:31:50	MaxCC	000	#steps	2			
			End	2007/09/28 05:33:40	Abend	000	#trigs	1			
	JTEST01	MFOC	Start	2007/09/28 05:32:00	MaxCC	...	#steps	2			
			End	Abend	...	#trigs	1			
	TSORXSRV	MFOC	Start	2007/09/28 05:33:34	MaxCC	000	#steps	1			
			End	2007/09/28 05:33:38	Abend	000	#trigs	1			
	JCOBA03	MFOC	Start	2007/09/28 05:33:40	MaxCC	...	#steps	4			
			End	Abend	...	#trigs	2			
	JTEST02	MFOC	Start	2007/09/28 05:34:53	MaxCC	...	#steps	0			
			End	Abend	...	#trigs	2			
	JRMT001	DYAH2003	Start	2007/09/28 05:35:07	MaxCC	...	#steps	0			
			End	Abend	...	#trigs	3			
	JRMT002	DYAH2003	Start	2007/09/28 05:35:22	MaxCC	...	#steps	0			
			End	Abend	...	#trigs	4			
***** Bottom of data *****											

Figure 8.10: Remote jobs look no different as local workloads while in progress

Figure 8.11 shows completed remote workloads. They also look no different to local workloads. Regardless the status is in progress or completed, you can explore deeper information of both local and remote workload in the same way. Type S on selected workload and press enter-key to obtain its detail information, as shown in figure 8.12. .



Scheduled-jobs Log								Row 1 to 17 of	
S	Jobname	System	Date	Time	Condition	Misc.	Info		
-	JTEST00	MFOC	Start	2007/09/28 05:31:43	MaxCC	010	#steps	3	
-			End	2007/09/28 05:33:36	Abend	000	#trigs	0	
-	JCOBA01	MFOC	Start	2007/09/28 05:31:43	MaxCC	008	#steps	5	
-			End	2007/09/28 05:36:18	Abend	000	#trigs	0	
-	JCOBA02	MFOC	Start	2007/09/28 05:31:50	MaxCC	000	#steps	2	
-			End	2007/09/28 05:33:40	Abend	000	#trigs	1	
-	JTEST01	MFOC	Start	2007/09/28 05:32:00	MaxCC	00c	#steps	5	
-			End	2007/09/28 05:36:37	Abend	000	#trigs	1	
-	TSORXSRV	MFOC	Start	2007/09/28 05:33:34	MaxCC	000	#steps	1	
-			End	2007/09/28 05:33:38	Abend	000	#trigs	1	
-	JCOBA03	MFOC	Start	2007/09/28 05:33:40	MaxCC	00c	#steps	4	
-			End	2007/09/28 05:35:27	Abend	000	#trigs	2	
-	JTEST02	MFOC	Start	2007/09/28 05:34:53	MaxCC	00c	#steps	4	
-			End	2007/09/28 05:37:32	Abend	000	#trigs	2	
-	JRMT001	DYAH2003	Start	2007/09/28 05:35:07	MaxCC	004	#steps	1	
-			End	2007/09/28 05:36:14	Abend	000	#trigs	3	
-	JRMT002	DYAH2003	Start	2007/09/28 05:35:22	MaxCC	004	#steps	1	
-			End	2007/09/28 05:36:40	Abend	000	#trigs	4	
-	JRMT003	DYAH2003	Start	2007/09/28 05:36:14	MaxCC	004	#steps	3	
-			End	2007/09/28 05:37:19	Abend	000	#trigs	3	
-	TSORXCL1	MFOC	Start	2007/09/28 05:36:18	MaxCC	000	#steps	1	
-			End	2007/09/28 05:36:22	Abend	000	#trigs	1	
-	JTEST03	MFOC	Start	2007/09/28 05:37:14	MaxCC	00c	#steps	4	
-			End	2007/09/28 05:38:24	Abend	000	#trigs	3	
-	TSORXCL2	MFOC	Start	2007/09/28 05:37:15	MaxCC	000	#steps	1	
-			End	2007/09/28 05:37:18	Abend	000	#trigs	1	
-	JTEST04	MFOC	Start	2007/09/28 05:37:26	MaxCC	008	#steps	3	
-			End	2007/09/28 05:38:26	Abend	000	#trigs	4	
-	TSORXCLT	MFOC	Start	2007/09/28 05:37:31	MaxCC	000	#steps	1	
-			End	2007/09/28 05:37:35	Abend	000	#trigs	1	
-	JTEST05A	MFOC	Start	2007/09/28 05:38:24	MaxCC	...	#steps	2	
-			End	Abend	...	#trigs	2	
-	JTEST06	MFOC	Start	2007/09/28 05:38:26	MaxCC	000	#steps	2	
-			End	2007/09/28 05:40:14	Abend	000	#trigs	2	
***** Bottom of data *****									

Figure 8.11: Remote jobs look no different as local workloads when completed

```

                ZJO5-XDI/Puspa 2.1.3
Action
-----
                scheduled-job Log detail                                Row 41 from 43

Job JRMT003   Jobid JOB02990  Asid 0022  on system DYAH2003
Statistics: #Step 3 #Trig 3 MaxCC 004 SCC 000
Run on : Fri 2007/09/28 at 05:36:14 to Fri 2007/09/28 at 05:37:19
No.  Jobstep      Procstep    UCC   SCC  End-time      Elapsed
  1  RSTEP1                000   000  05:48:31.22
  2  RSTEP2                000   000  05:49:12.61
  3  RSTEP3                004   000  05:49:26.27
***** Bottom of data *****

Command ==> _____ Scroll ==> CSR

```

Figure 8.12: Detail information of a remote job looks no different as local workloads



All detail information that you found on local workload panel are also provided on remote workload panel.

To explore triggering information, type T on selected workload and hit enter-key. Triggering information is then obtained as shown in figure 8.13.

```

zJOS-XDI/Puspa 2.1.3
Triggering Jobs Log
Row 36 from 38

Job JRMT003 Jobid Job02990 Asid 0022 on system DYAH2003
Statistics: #Step 3 #Trig 3 MaxCC 004 SCC 000
Run on : Fri 2007/09/28 at 05:36:14 to Fri 2007/09/28 at 05:37:19

TrigJob System Step ProcStep SCC UCC Op RCC Flags
JRMT001 DYAH2003 000 004 NC 000 10..1..0
JRMT002 DYAH2003 000 004 NC 000 10..1..0
JCOBA04 MFOC n/a n/a LE 00C 00..1..0
***** Bottom of data *****

Command ==> scroll ==> CSR
    
```

Figure 8.13: Triggering information of a remote job looks no different as local workloads



Chapter 9 Commands and Messages Reference

9.1. Puspa Commands Facilities

Sekar console commands can be issued in 3 ways:

1. Via zJOS subsystem (on console)
2. Via MODIFY command to zJOS address space (on console)
3. Via zJOS control panel (in ISPF session on TSO)

9.1.1 Entering Command via zJOS Subsystem

zJOS subsystem provides a gate for you to enter Puspa command on z/OS MVS console. Subsystem recognize all zJOS command when either prefixed by dot sign (.) or XDI with a blank ("XDI "). The common command syntax is

```
prefixSCD request
```

or for more specific is:

```
.SCD request
```

or

```
XDI SCD request
```

Where **request** is a service you want to obtain.

9.1.2 Entering Command via MODIFY

:

Alternatively, you can also pass command to Puspa via z/OS MVS MODIFY system command on console to zJOS address space. The command syntax is:

```
MODIFY XDI,SCD request
```



or

F XDI,SCD request

Where **request** is a service you want to obtain.

9.1.3 Entering Command via zJOS Control Panel

The second alternative to issue Puspa command is via control panel. This facility available on TSO/E terminal while in ISPF session as discussed in chapter 7. To issue command, just enter the request on Puspa command slot as shown in figure 9.1. See chapter 6 for further explanation.

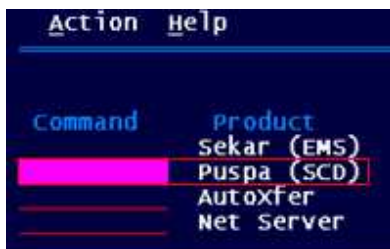


Figure 9.1: Puspa command slot on zJOS control panel

9.2. Puspa Commands Reference

This paragraph only explains **request** verb instead of full command text.

9.2.1 HALT request

Syntax on console:

HALT [JOB=jobname]

Syntax on control panel:

HALT or H [jobname]

Where **jobname** is name of workload you optionally desire to be halting point.



Function:

Halt the progress of scheduling system. . .

9.2.2 HOLD request

Syntax on console:

```
HOLD JOB=jobname
```

Where **jobname** is name of workload you optionally desire to be halting point.

Function:

Hold the scheduling progress of specified workload. . . .

9.2.3 INIT request

Syntax on console:

```
INIT
```

Syntax on control panel:

```
INIT
```

Function:

Load schedule table onto memory and prepare all scheduling related processes to support Puspa to READY state. .

Note:

INIT is required only at the first time schedule table is loaded. Once a table is loaded onto DIV, subsequent issue will be ignored.

9.2.4 LOAD request

Syntax on console:

```
LOAD
```

Syntax on control panel:

```
LOAD
```



Function:

Load schedule table onto memory.

Note:

LOAD effective only at the first time a schedule table is loaded. Once a table is loaded onto DIV, subsequent issue will only make addressed loaded table becomes a current table.

9.2.5 REFRESH request

Syntax on console:

REFRESH

Syntax on control panel:

REFRESH or REF

Function:

Cleanup all current scheduling status in schedule table and prepare it for next scheduling cycle. . .

9.2.6 RELOAD request

Syntax on console:

RELOAD

or

RELOAD TAB=nn

Where nn is 2-digit suffix to address schedule table XDISCDnn member of zJOS PARMLIB

Syntax on control panel:

RELOAD

Function:

The same as LOAD request, load schedule table onto memory.

Note



On control panel **reload** has the same effect as **load** request. Puspa uses specified **nn** in **Suf** column.

9.2.7 RESTART request

Syntax on console:

```
RESTART JOB=jobname
```

Syntax on control panel:

```
R jobname
```

Where **jobname** is name of workload you desire to be a starting point.

Function:

Restart the progress of halted scheduling system. . .

9.2.8 RESUME request

Syntax on console:

```
RESUME
```

Syntax on control panel:

```
RESUME
```

Function:

Resume the progress of halted scheduling system. . .

9.2.9 START request

Syntax on console:

```
START
```

Syntax on control panel:

```
START or S
```

Function:



Activate scheduling process.

Note:

START can only be issued when Puspa is in READY state

9.2.10 STOP request

Syntax on console:

STOP

Syntax on control panel:

STOP

Function:

Inactivate scheduling system.

Note:

Unless for maintenance purpose and really recommended by XDI support personnel, you should not stop scheduler.

9.3. zJOS System Commands Facilities

To manage and control the zJOS address space and subsystem, zJOS provides some commands. All zJOS commands can only be issued via MODIFY (F) command or zJOS subsystem. MODIFY command syntax is:

F XDI,request

Subsystem recognize all zJOS agent command when either prefixed by dot sign (.) or XDI with a blank ("XDI "). Hence, the command syntax is

.request

or

XDI request

Where **request** is a service you want to obtain.

9.4. zJOS System Commands Reference

This paragraph only explains `request` verb instead of full command text.

9.4.1 ASCB request

Syntax

`ASCB`

Function:

List all ASCBs and each with detail information.

9.4.2 HELP request

Syntax

`HELP`

Function:

Display zJOS command reference summary on console.

9.4.3 LIST request

Syntax

`LIST object`

Where:

`object` is either NETCCB, PIT, Q

Function:

LIST NETCCB → Lists existing chained NETCCB.

LIST PIT → Lists all captured JES PIT

LIST Q → List all enqueued zJOS resources.

9.4.4 RCMD request

Syntax

`RCMD hostname command_text`

Where:



hostname is host name of targeted system.

command_text is complete command text to be sent to agent site

Function:

Send command to agent site and ask agent to execute it.

Notes:

1. RCMD command only supported by zJOS subsystem
2. Command does not need prefix. Just issue as appeared in syntax.

9.4.5 RJOB request

Syntax

RJOB *hostname jobname*

Where:

hostname is host name of targeted system.

jobname is name of job which is a member of JCLLIB

Function:

Send request to agent site to submit a job which is addressed by jobname.

Notes:

1. RJOB command only supported by zJOS subsystem
2. Command does not need prefix. Just issue as appeared in syntax.

9.4.6 SHUTDOWN request

Syntax

SHUTDOWN

Function:

Bring zJOS address space down.

9.4.7 START request

Syntax

START

Function:

Bring zJOS address space up.



Notes:

1. START command not available at the first time startup of zJOS during IPL cycle.
2. START command can not be issued via MODIFY command.

9.4.8 WTO or MSG request

Syntax

WTO text

Function:

Issue WTO macro to send message text to console

Notes:

1. Message is highlighted.
2. Message is deleted by subsequent WTO issuance, by means of DOM macro. .

9.5. zJOS Agent Commands Facilities

To manage and control the agent, zJOS provides some commands for agent which available only on agent site. All agent commands can only be issued via zJOS agent subsystem. Subsystem recognize all zJOS agent command when either prefixed by minus sign (-) or XDA with a blank ("XDA "). Hence, the command syntax is

-request

or

XDA request

Where **request** is a service you want to obtain.

9.6. Agent Commands Reference

Since agent command is only prefix and **request** verb, this paragraph only explains **request** verb prefixed with minus sign. To use XDA prefix, you can easily just substitute minus sign with "XDA " string.



9.6.1 CONNECT request

Syntax

-CONNECT [IP=server_address] [PORT=server_port]

Where:

server_address is IP address or name of server, and
server_port is server port number (default is 7777)

Function:

Requesting connection (sign-on) to zJOS server.

9.6.2 DISCONNECT request

Syntax

-DISCONNECT

Function:

Requesting disconnection (sign-off) to zJOS server.

9.6.3 DROP request

Syntax

-DROP

Function:

Force socket task to be detached from zJOS agent address space. .

9.6.4 GET request

Syntax

-GET component_name

Where **component_name** is either EMS or SCD.

Function:

Requesting server to send EMS or scheduling parameters.



9.6.5 **HELP request**

Syntax

-HELP

Function:

Display zJOS Agent command reference summary on console.

9.6.6 **LIST request**

Syntax

-LIST component_name

Where **component_name** is either EMS or SCD.

Function:

List down EMS or scheduling parameters received from server.

9.6.7 **START request**

Syntax

-START

Function:

Bring up agent address space.

Note:

START command can not be used for first start along with IPL cycle. .

9.6.8 **STOP request**

Syntax

-STOP

Function:

Bring down agent address space.



9.7. Puspa Messages

All Puspa messages have the following common format:

DERXXXYYYZ Message_text

DER indicates product package of zJOS-XDI

XXX indicates component id, by which the messages is issued. The same message text could be issued by more than one component.

YYY is message number, indicates the message id.

Z is message suffix code, indicates the status of message.

- I – informational message
- W – warning
- E – error message
- A – needs user action
- T – logic tracing information

Message_text is information description of the message. Most of zJOS XDI messages have clear and simple information.

Complete messages reference can be found in zJOS-XDI Messages Reference manual.

Chapter 10 Advanced Tricks

Implementing automatic scheduling system sometime need more creative tricks instead of just using provided standard scheduler functions and features such as various triggering event types (including step-level pipelining using end-of-step event), basic and advanced condition filtering, timeframe filtering, calendar and special calendar filtering. Sometime, certain specific environment need more than that. For example, workload that runs on the off days needs to be varied in sequent rule. In certain situation, some workloads need to be scheduled outside its defined clock time range, is more extreme example.

Most of tricks in handling the above examples are programming. If you license zJOS/Sekar, you can use the advantage of system event automation functions and feature to help you find the simplest trick for such scheduling cases. It can be implemented as:

- Event entry which trigger action to trigger scheduled workloads.
- Event entry which trigger rule to trigger scheduled workloads.
- Event entry which trigger JOB submission or STC startup to trigger scheduled workloads.
- Scheduled JOB or STC which executing rexx program to create desired condition or event to trigger other scheduled workloads.
- Rexx program inserted as a step in either scheduled JOB or STC to create desired condition to control subsequent steps and/or create desired event to trigger other scheduled workloads.

10.1. Using Sekar Standard EMS Features

You may use standard EMS functions of zJOS/Sekar for advanced scheduling trick. For example, workload JOB1 is scheduled at 21:00:00 every working day. But, sometime in certain situation, JOB1 has to be started earlier. Since JOB1 is followed by some successors, it requires to be handled by scheduler by means Puspa. If it is defined in scheduler table with start time 21:00:00, you must use unconditional schedule function, by means Z prefix command, to schedule JOB1 earlier. This function, however, need full authorization which is not for operator, hence administrator has to come down and login to zJOS.

The easier way is using message event to trigger JOB1 as follow:

- You change schedule definition timeframe to unrestricted time 00:00:00 to 24:00:00.
- Define trigger for JOB1 with event type of MSG with unique message text, let say 'RUNJOB1'.



- Define TOD event in EMS table to fire .WTO command at every 21:00:00. Use the same text ('RUNJOB1') as you have defined in JOB1 triggering event definition for .WTO command argument.

Once EMS table is activated, .WTO command will be fired at every 21:00:00 and a message with designated text then occurred. The occurrence of this message then triggers JOB1. Hence, in normal situation, workload JOB1 is triggered at every 21:00:00. If certain situation you desire JOB1 to run at 19:00:00, you can easily call operator to issue .WTO RUNJOB1 command at 19:00:00. When the .WTO is issued (by operator), message 'RUNJOB1' is occurred and JOB1 is then triggered and run. Next, when the same message is occurred from EMS at 21:00:00, is ignored because JOB1 has already been done. The next day, JOB1 back to normal which run at 21:00:00 following 'RUNJOB1' message occurrence.

10.2. Using zJOS Rexx Functions

zJOS/Sekar provides package of 13 rexx functions for various automation application purposes, including scheduling system. The most popular functions which are used in advanced scheduling system are zjcal(), zjholday(), zjcmd(), zjwait(), zjwto() and zjwto(). Please refer to zJOS/Sekar chapter 6 par 6.4 for detail description of each function.

Example 1

This example show you how to schedule JOB1 at 20:00:00 everyday except on month-end day, which is as early as possible once all predecessors completed. Using standard scheduler function, you have to define JOB1 with start time 20:00:00 and select exclusion on EMON checkbox to avoid it from being scheduled on month end day. Then, define JOB1 using different name (eg. JOB1EMON for member name JOB1) with start time 00:00:00 and select EMON to make it only runs on month end day.

As member name is only supported for JOB type of workload, the above example is applicable only if workload type of JOB1 is JOB (batch job). If JOB1 is an STC, this case becomes more complex. Using Sekar rexx function, however, you would have the simplest solution by coding the following one line rexx:

```
If zjcal('EMON') = 'N' then X = zjwait('20:00:00')
```

Use either batch rexx or batch TSO step to execute the above single line rexx and insert it as the first step of JOB1. Then, schedule JOB1 without start time filtering (00:00:00). Although JOB1 runs immediately once all its predecessors completed, its real step will be delayed until 20:00:00 everyday except on month end day. JOB1 can be STC or JOB.



If you don't want JOB1 shown in active workloads list before 20:00:00 except on month end day, you can easily pull this rexx step out from JOB1 and place it in dummy workload JOB0. Then, schedule JOB0 without start time and move out all JOB1 predecessors to trigger JOB0. JOB1 is then scheduled as successor of JOB0 with EOJ of JOB0 as the only trigger event source.

Example 2

All steps of JOB2 were designed to produce datasets DATA.AB1 and DATA.AB2 for use by JOB2A and JOB2B on everyday as input. Newly added application feature requires a new job, JOBX, to be added into existing schedule flow with some specific requirements. On every working day, JOB2A and JOB2B require an additional input dataset DATA.ABX which is produced by JOBX. Both job JOB2A and JOB2B have already been plotted in the schedule table to trigger complex job flow at step level pipelining mechanism which impossible to be modified. As user doesn't want to separate both JOB2A and JOB2B each into 2 jobs for working and off days for maintenance and complex schedule flow reasons, there are 2 problems raised here:

1. During off days, job JOB2A and JOB2B are triggered by end of job JOB2, but, during working days by job JOB2 and JOBX. You can not place JOB2 and JOBX in the same group as both predecessors must be ANDed to have final compliance during working days. If them ANDed, however, they will never comply to trigger job JOB2A and JOB2B during off days since no JOBX involve.
2. To avoid JCL error, dataset DATA.ABX must always exist unless the way job JOB2A and JOB2B allocate dataset DATA.ABX are dynamically by using SVC 99. In this case, they use JCL instead.

Such case is too hard for ordinary scheduling technique. The only way to do so is by separating JOB2A becomes JOB2A and JOB2AX, and JOB2B becomes JOB2B and JOB2BX. Job JOB2A and JOB2B are only triggered by JOB2 and only access DATA.AB1 and DATA.AB2. Job JOB2AX and JOB2BX are triggered by combined JOB2 and JOBX (ANDed) and access DATA.AB1, DATA.AB2 and DATA.ABX. Unfortunately, subsequent successors are complex flows and user doesn't agree that way.

Being an automation administrator, you should more creative to safely add JOBX into the existing schedule flows. Your best chance is JOBX. Define JOBX in schedule table as necessary which is also as a predecessor to trigger job JOB2A and JOB2B together with JOB2 (ANDed) on everyday. Insert batch rexx program step in between dataset DATA.ABX allocation step and other subsequent steps. Inserted rexx step is to abort all subsequent steps if current day is off day. There are many ways to achieve such mechanism. Here is one of possible ways:



```
If zjcal('OFFD') = 'Y' then x = zjcmd('CANCEL JOBX')
```

The above one line rexx program cancels JOBX if run on the off days. Hence JOBX might be ended with CC = 0 or abend 222. Then you update triggers list for JOB2A and JOB2B by adding a triggers group contains JOBX with CC = 0 and JOBX with abend 222. As both events are in a group, compliance of either one will be used as group compliance.

Notes