

**Nusantara Software Industry**

**zJOS/Sekar**



**Event Management  
User Guide**



## Table of Content

<b>CHAPTER 1</b>	<b>INTRODUCTION .....</b>	<b>1</b>
1.1.	Event.....	3
1.2.	Events Management System.....	4
1.3.	EMS with Sekar.....	6
<b>CHAPTER 2</b>	<b>GETTING STARTED .....</b>	<b>9</b>
2.1.	Preparing zJOS-XDI System.....	9
2.1.1	zJOS-XDI Datasets.....	9
2.1.2	zJOS-XDI JCL Procedures.....	13
2.2.	Starting and Stopping zJOS .....	18
2.3.	Preparing Sekar .....	21
2.4.	Starting and Stopping Sekar .....	23
2.5.	Preparing National Holiday Table.....	24
2.6.	Preparing System Startup Table.....	27
<b>CHAPTER 3</b>	<b>WORKING WITH SEKAR PARAMETERS .....</b>	<b>31</b>
3.1.	Managed Event Table .....	32
3.2.	Event Entry .....	35
3.2.1	Event verb and System .....	37
3.2.2	Timeframe and Timespec .....	38
3.3.	Action Table.....	42
3.4.	Action Entry .....	44
3.5.	Action Variables.....	47
3.5.1	&ARG Variable.....	47
3.5.2	&UID and &JOB Variables.....	50
3.6.	Applying Sekar Parameters .....	50
<b>CHAPTER 4</b>	<b>CONTROLLING SEKAR .....</b>	<b>51</b>
4.1.	Status Information .....	51
4.1.1	Products Status Information .....	52



4.1.2	Statistics Information .....	54
4.2.	Reloading Sekar Parameters.....	56
<b>CHAPTER 5 INTEGRATED AUTOMATION .....</b>		<b>59</b>
5.1.	Integrated zJOS Network.....	59
5.1.1	Hardware Requirements .....	59
5.1.2	Software Requirements .....	60
5.2.	Sekar for Integrated zJOS Network.....	61
5.2.1	Preparing zJOS Server.....	62
5.2.2	Preparing zJOS Agent for z/OS.....	65
5.3.	Sekar Agent for z/OS .....	67
5.3.1	Starting and Stopping zJOS Agent .....	68
5.3.2	Connecting and Disconnecting Agent .....	70
5.3.3	Controlling zJOS Agent .....	71
5.3.4	Remote Command.....	72
5.3.5	Remote Job Submission .....	74
5.4.	The Goal of Integrated Automation .....	74
<b>CHAPTER 6 IMPLEMENTING YOUR INNOVATION WITH SEKAR .....</b>		<b>77</b>
6.1.	Innovation with Command.....	77
6.2.	Standard XDI Rule .....	79
6.2.1	Example.....	81
6.3.	Innovation with Rule.....	82
6.3.1	Rule Programming .....	83
6.3.2	Creating Your Own XDIRULE.....	87
6.4.	zJOS Supported Rexx Functions .....	87
6.4.1	zjaxfer() .....	88
6.4.2	zjcal().....	88
6.4.3	zjcmd() or xcommand() .....	90
6.4.4	zjevent().....	91
6.4.5	zjholday() .....	93
6.4.6	zjpuspa().....	94
6.4.7	zjsekar().....	95
6.4.8	zjserver().....	96
6.4.9	zjset().....	97
6.4.10	zjstate() .....	100
6.4.11	zjwait().....	101
6.4.12	zjwto().....	102
6.4.13	zjwto() .....	103
6.5.	Using zJOS Rexx Functions .....	104
6.5.1	Non-rule Rexx Programming .....	104
6.5.2	Using zjset() and zjevent() in Your Program.....	105



<b>CHAPTER 7</b>	<b>USING ZJOS CONTROL PANEL .....</b>	<b>107</b>
7.1.	Starting and Stopping zJOS .....	108
7.2.	Issuing Sekar Command.....	112
7.3.	Obtaining Helps.....	113
7.3.1	Common Help Information .....	114
7.3.2	Field-specific Help Information .....	115
<b>CHAPTER 8</b>	<b>COMMANDS AND MESSAGES REFERENCE .....</b>	<b>117</b>
8.1.	Sekar Commands Facilities .....	117
8.1.1	Entering Command via zJOS Subsystem .....	117
8.1.2	Entering Command via MODIFY .....	117
8.1.3	Entering Command via zJOS Control Panel .....	118
8.2.	Sekar Commands Reference .....	118
8.2.1	LOAD request .....	118
8.2.2	RELOAD request .....	119
8.2.3	START request.....	119
8.2.4	STOP request.....	120
8.3.	zJOS System Commands Facilities.....	120
8.4.	zJOS System Commands Reference.....	120
8.4.1	ASCB request.....	121
8.4.2	HELP request .....	121
8.4.3	LIST request.....	121
8.4.4	RCMD request.....	121
8.4.5	RJOB request.....	122
8.4.6	SHUTDOWN request.....	122
8.4.7	START request.....	122
8.4.8	WTO or MSG request .....	123
8.5.	zJOS Agent Commands Facilities.....	123
8.6.	Agent Commands Reference .....	123
8.6.1	CONNECT request.....	123
8.6.2	DISCONNECT request .....	124
8.6.3	DROP request.....	124
8.6.4	GET request.....	124
8.6.5	HELP request .....	124
8.6.6	LIST request .....	125
8.6.7	START request.....	125
8.6.8	STOP request.....	125
8.7.	Sekar Messages.....	125





## Chapter 1 Introduction

This topic introduces you to the concept of automatic events management and how zJOS/Sekar® do it for you. Before going further, probably want to know what event management system (EMS) is and why we need it. For instance, for you who manage in day-to-day z/OS operation, thousands messages may occur in your system consoles. Some of them probably do not very important. But, there some critical messages which need you to do certain actions, such as replying to the message (for WTOR message), issuing certain commands, allocating or deleting certain datasets or whatever. For example, when a certain job is ABENDING that causing SVC dump while no more enough space in all assigned disk volumes, then message IEA793A occurs asking you to either abort the dump (reply D) or prepare enough disk space.

Without EMS support, your staff has to do it manually. Such task is not too hard to do. Just stick his/her eyes to the console screen and soonest IEA793A occurs, do all those necessary actions. The problem is, message IEA793A is not the only thing to do. There are some other similar mechanisms. Moreover, their occurrences are totally unsolicited, can not be predicted. No guarantee when the message IEA793A is coming, tonight? Tomorrow? Next week? Or even next year? Some very frequent, whereas some others rare. Hence, difficult for your staffs to familiar what they have to do, especially for the rare one. Although special/local manual is provided, they need awhile to select which page to read, need time to read it and need time to play with console and/or terminal keyboard, even though no guarantee all are done correctly. Meanwhile, total consumed time sometime is intolerable delay.

Smart systems programmer can help such situation by customizing console exit routines such as IEAVMXIT. Nevertheless, such solution is not very flexible and high skilled people to maintain. As most of z/OS exit routines are written in assembly, they need to be customized and maintained by only person who has assembly programming skill and deeply understand z/OS architecture. Not easy to find them. Moreover, most of exits need IPL to make them effective. It means, you have to schedule IPL every time the exit(s) are updated.

Exit routines can read system TOD, so, smarter systems programmer can make them to work differently in certain year, month, date, clock time and/or week day according to the need. The need, however, sometime also require the different actions in the holiday. In such case, a national calendar management has to be added in the exit routines. This will be a very serious constraint and only very expert people can do it efficiently.

In fact, not only messages need to be responded. Sometime commands or status of jobs execution also need to be responded. Some environments even require



executing sets of commands periodically. Job execution status (such as job termination and job step termination) probably can be handled using JES or SMF exit routines. Exit routines, however, are not applicable for commands and timer, except for JES commands. No sample is provided, so nothing to customize. They need deeper way to handle which need user program to manage the timer and/or intercept command processing. If all are done by exit routines and user programs, your system will become a ranch of system-level user's modules. You are facing big problem in the future.

In the other hand, if all have to be done manually, e.g. you don't have the expert, you can imagine how they work. These are not just a matter of time consuming which result quite significant delay in total. These are also potential risks which may come from human error.

Occurrences of messages, command issuances, status of jobs termination or job step termination and time up of certain TOD are in common called as system events or just event for short. All the above cases are regarding how to manage events systematically and efficiently. That means, you need a technology which able to do it, called events management system (EMS).

With EMS, the above cases are very simple to handle. You don't need to write even any single exit routine nor user program. You don't even need systems programming expert to manage them. All you need is just register text or part of text of messages, commands, name of jobs or job with steps you are going to respond, and sets of actions (in text also) accordingly. EMS will work for you as what you expect, even more.

There are some EMS products in the IT market you can consider. Their capability in detail probably varies. Their basic concept of work, however, should be the same, capture specified events and fire specified associated actions. The way to specify or register event/action(s) pairs could be varies. Some products use a scripting, some use tabulation and the others offer both ways. Scripting is more flexible. You can put your logic for specific relationship between event and its associated set of actions. But, you need a specific skill to write script. Although much simpler than assembler coding for exit routines, scripting is a programming. You need another investment to have the skill.

Tabulation is simpler and easier to work with. You don't need to write script. All you need is just fill up a form in the panel to register each pair of event and its set of actions into EMS database. But, it is less flexible and not applicable for complicated conditional actions.

The moderate one is the product that support both ways. For simple cases, you can use tabulation. For complex cases, you are challenged to write script.



## 1.1. Event

Operating system provides formatted interruption called “event”. Although not all events reflect hardware interruption, the ways they occur, however, like hardware interruption. Some events are used internally by operating system to do some advance processing. For example, when a task is terminating, “end-of-task” event occurs to notify task manager to do necessary recovery or housekeeping. Other example is I/O processing. All data movement between CPU memory and external devices, including memory of other computer by means of data communication, are named as I/O, which is actually handled at primitive layer of operating system based on hardware interruption. However, at advance layer, each type of device need specific handling. Data from or to a disk needs to be handled as storage data or file by file manager. Data from, or to other computer needs to be handled as network data by network manager. Even each type of network needs separate network manager. Network data from or to SNA network needs SNA control point program, whereas data from or to IP network needs TCP/IP stack program. Primitive layer (by means first level interrupt handler or FLIH), catches interruption and does some control mechanism, then schedule I/O subsystem (IOS) for further handling. IOS then categorize to which manager data need to be processed. To notify selected manager, IOS uses “soft interruption” called event.

Both examples above describe events that used by OS components to trigger other OS components. Their exchange mechanisms are varies and mostly are not general programming interface (GPI). This means, there is no guarantee for their consistencies and compatibilities from release to subsequent release of OS. User program should not touch with non-GPI stuffs.

OS also provide GPI for some certain events exchanges for user programming. For example, in I/O and network programming, the way user program knows that its write(), send(), read() or recv() request has completed is because notified by I/O or network manager. Sometime you don't feel there is an event exchange between your program and the system. Your codes just issue send() as a part of synchronous process. This is an illusion of a magic trick. Event exchange is hidden behind the send() function. When your program got execution control at send() function, your program is actually held in wait state after the detail request is received by the system. System then schedules your request for processing then switch execution control to task in the dispatching cyclic. Your program's task is always skipped until your request is completely done. Upon completion of request processing, system then wake your program up. Such mechanism is caller blocking-I/O algorithm.

You may also involve in managing event exchange. In such case, your program is not held to wait for send() request completion. Execution control return to your program immediately after the detail request was received by the system and



subsequent instruction of your program got control. Such mechanism is called non-blocking-I/O algorithm. No more synchronous illusion. But, you must really understand that when control is returned after send(), the I/O is not done yet, so, subsequent instruction should not deal the result of this send() function. You are responsible to find out whether your send() request is completed by listening its associated event.

Events exchange that included in GPI stuff usually well documented in user's guide and reference manuals. The exchange uses a standard format and rule. Event information is encoded into 4-byte word aligned area called event control block (ECB). Source of event could be the system or user program, where the target in most cases is user program. OS provides a standard rule for source to post a signal and for target to listen the event of incoming signal. Hence, for user program to user program exchange, event can be anything.

Regardless the GPI category and the target of event sourced from the system, sometime standard processing is not enough for certain implementation. For example, when a job (user program) ends, standard action of the system is just sending a message to console telling that the job is ended. Whereas, in certain user's site, when a job ABC ends, user needs to run job PQR and transfer all resulted spool files to a Windows system at a certain IP address. Such action is very specific and not implemented in the OS. To have it done, user must do it manually every time message IEF404I or \$HASP395 regarding job ABC ended is occurred on the console. As both messages are scrolled out from console too fast, user need to watch them from SDSF. Worse, isn't it?

In heavy production sites, operators have to do tens or even hundreds of specific or even critical actions in respond of certain events. To avoid human error, user needs event management system, which is normally not included as a standard component of OS.

## 1.2. Events Management System

Events management system (EMS) is a software product that was designed to give a chance to user to manage any event types and their associated actions easily without touching internal OS nor programming effort. Users shouldn't care whether the events they manage are GPI or non-GPI. Users don't need to know which rule of the game is used internally by EMS, whether it is standard ECB rule or not. EMS product vendors should fully responsible to guarantee their products compatibility to their users' system environment. All users have to do is just fill up the table or write simple script.

Technically, EMS is a set of programs under a single coordination to manage each pair of event and associated actions stored in its database. It consists of at



least 4 basic major components: automation database manager, event listener, action manager and user interface. .

**Automation database manager** is a set of services to manage accesses to the database that contain events and actions definition. Its style, type of processing and algorithm are varies, depend on the taste and skill of its designer. It is tightly connected with all other 3 basic major components.

Database manager can either run as system- or user-level program, depends on its design. If it use OS standard database establishment, it can run in problem state as a user-level program. Privileged portion is hidden behind the OS. But, some products probably prefer to use their own establishment technique. Hence, consequently, it must run in supervisor state to authorize several controls of I/O and memory.

**Event listener** is a set of services to capture events exactly at the time of their occurrences. Its capabilities in capturing various types of events and algorithm used are varies, depend on the skill of its designer and developers. Targeted events mostly are system events, such as messages, commands, job termination and job step termination. It calls automation database manager to select which events table got to be used as a reference during listening events traffic. Captured events are then routed to action manager for firing.

Event listener is the most critical component. As the targets are system events, it must run in supervisor state all the time. Events traffics are dispersed around several locations in the system depend on their types. Whereas, most of them doesn't follow standard ECB rule, rather, each type uses its own protocol. Some types use subsystem interface (SSI), some other types use JES or SMF. Hence, to capture them, listener must adapt their ways and stay on their residences.

**Action manager** is a set of services to collect captured events received from event listener and manage their executions based on action tables provided by automation database manager. Selection of each actions-set for each captured event is done by automation database manager. Action manager responsibilities are only managing their firing processes which normally done simultaneously in multitasking algorithm.

Action manager is second critical component. As it deals with system command processors, it must runs in supervisor state. As the way events captured by the listener are asynchronously, action manager must also work simultaneously to get the shortest path for each event processing. So that is why, in most of EMS products, action manager uses multitasking algorithm.

**User interface** is a set of services to allow EMS administrators interact with each major component of EMS. With this interface, EMS admin can manage events and actions tables using automation database manager. With this interface,



EMS admin can control either event listener or action manager. Its art, style, ergonomic level, type of processing and algorithm are varies, depend on the taste and skill of its designer and developers. .

User interface is not very critical and in most cases it doesn't need to run in supervisor state unless database manager uses its own establishment method. Nevertheless, it must be able to enter to supervisor state, in case it needs to interact directly with either listener or action manager.

User interface that runs on TSO/ISPF needs special treatment when switched to supervisor state as ISPF doesn't authorize. Otherwise, it must provide its own panel and screen management.

## 1.3. EMS with Sekar

Sekar or zJOS/Sekar is a program to help you to manage your specific events and associated actions mechanism automatically, commonly called as event management system (EMS). Each event can be associated with one or more actions. Figure 1.1 below illustrates how event is associated with actions in Sekar EMS table structure. Every time event 2 occurs, Sekar event listener then instantly schedule action 2.1, action 2.2 and action 2.3 for execution.

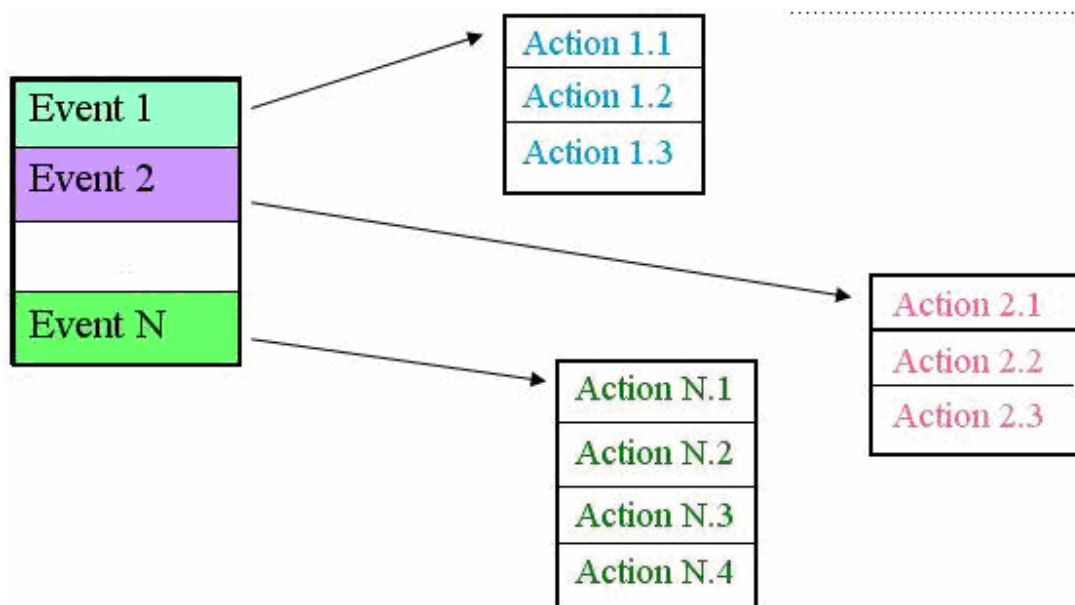


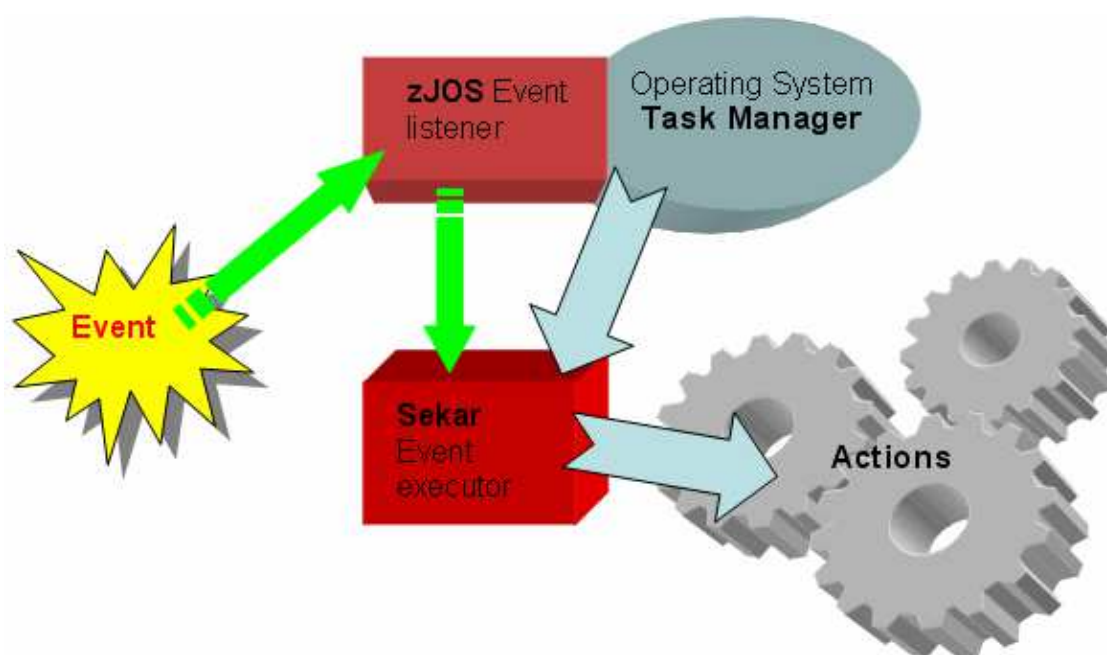
Figure 1.1: Structure of Sekar EMS table



At current level, Sekar supports 5 types of event, i.e.:

- Message (MSG) for both WTO and WTOR
- Console command (CMD)
- Time of day (TOD)
- End-of-jobstep (EOS)
- End-of-job (EOJ) for either job run on initiator or started job (EOM)

Part of Sekar, which is event listener, runs as subsystem on z/OS kernel. As illustrated in figure 1.2, event is caught by event listener, then posted to event executor. Then, when event executor is dispatched by z/OS task manager, event information is used to retrieve associated actions in action table. Matched actions are then executed serially.



*Figure 1.2: Event handling logic flow*

At current level, Sekar only supports 3 types of action, i.e.:

- Issue command
- Issue reply to message for WTOR only
- Start a rule. .

Command and rule actions are applicable for all of event types. Reply action to message only applicable for WTOR message event.

Before actions are performed, Sekar evaluates timeframe if one was defined. Timeframe is time based filtering, which consist of variable of start date and time, end date and time, and national holiday. Each variable is optional. If event is occurred within timeframe, it is considered as valid event, then, actions are



performed. Otherwise, actions are ignored. If timeframe was not defined, no timeframe checking is made. Matched event is always valid anytime.

Optionally for message event (for both WTO and WTOR), you can either suppress its appearance in console, syslog or both prior to actions execution.



## Chapter 2 **Getting Started**

zJOS/Sekar® is an EMS solution product which is bundled together with zJOS/Puspa® (automatic workload scheduler) 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-XDI main address space.

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

### **2.1. Preparing zJOS-XDI System**

Sekar, although most of its processes are run as subsystem functions embedded within z/OS system area, is booted and controlled from ordinary system-task that runs on an address space together with other zJOS-XDI products. This address space is called zJOS-XDI main address space. So therefore, to have Sekar runs properly, zJOS-XDI main address space must up properly. As a common service provider for all bundled products, zJOS process manager in certain circumstance needs specific additional services which need to be executed outside the main address space for performance reason. Hence to ensure the process manager, which is a part of Sekar, runs properly, main and all associated address spaces must be properly prepared. Preparing address spaces means preparing their JCL procedures and all associated stuffs.

Unless you need very specific changes, the main and all its associated zJOS-XDI address spaces have actually been completely prepared by INSTALLX exec during products installation steps. Once INSTALLX done, you would find all zJOS-XDI datasets and JCL procedures are ready in your system.

#### **2.1.1 zJOS-XDI Datasets**

Main materials of zJOS-XDI package are 2 sets of datasets, zJOS-XDI system datasets and application datasets. Both sets are prepared by INSTALLX exec during installation steps.



## zJOS-XDI System Datasets

System dataset is a datasets that contains zJOS-XDI system materials. zJOS-XDI installation generates some systems datasets such as module libraies, ISPF stuffs libraries, Rexx and Clist exec libraries and samples of parameters and other stuffs libraries etc. By default, all system dataset names follow naming standard as below:

**hlq.zJOSvrm.libname**

Where

- **hlq** is high level qualifier as specified during installation steps
- **vrm** is version, release and modification level. For example, 219 means version 2, release 1 and modification level 9.
- **libname** is a name that represents the content or functions contained in the dataset.

For example, if your zJOS-XDI is 210 and have chosen prefix SYS5, your Clist exec library name will be SYS5.ZJOS219.CLIST. Below is a list of library names of zJOS-XDI system datasets: To describe them easier, the names mentioned below are shorten to the last qualifier.

- **CUSTMAC** is an assembler macro library that contains macros for use as references when source codes in CUSTLIB are assembled.
- **LINKLIB** is a load library that contains non-LPA modules. These also in binary format. Some modules are non-executables (just for loaded only). Non-LPA doesn't mean non-reentrant. Most of executable modules of zJOS-XDI are reentrant. This library must be accessed in STEPLIB concatenation of all zJOS-XDI address spaces and TSO logon procedure for each userid that's assigned to work with zJOS control panel. Unless CUSTMOD is also concatenated, LINKLIB must be on top.
- **LPALIB** is a load library that contains LPA modules. All are in binary format and reentrant regardless executable or not. All LPALIB modules are designed to be loaded onto link pack area (LPA) unless you choose standard installation to let zJOS-XDI process manager itself to load its modules onto LPA dynamically. This library must be accessed in STEPLIB concatenation of all zJOS-XDI address spaces and TSO logon procedure for userid who needs to access zJOS control panel, following LINKLIB, unless zJOS-XDI is installed in LPA permanently.
- **DYNLPA** is a load library that contains LPA modules. All are in binary format and reentrant regardless executable or not. All DYNLPA modules must be loaded onto LPA. When you choose standard installation, zJOS-XDI process manager will load all DYNLPA modules dynamically onto LPA at the first start within IPL period. Subsequent start will ignore them unless you specify OPT=NEW on start command. Unless zJOS-XDI is installed in LPA permanently, DYNLPA must be accessed together with



LPALIB as an input file in zJOS-XDI main address space with DD name as specified in ZDD keyword.

- **CLIST** is a text library that contains Clist and Rexx programs for use as procedures to prepare and support zJOS-XDI control panel. This library must be accessed as SYSPROC file in TSO logon procedure for userid that's assigned to work with zJOS control panel.
- **TABLES** is an ISPF table library that contains several ISPF tables used by zJOS-XDI control panels. This library must be accessed as ISPTLIB file in TSO logon procedure for userid that's assigned to work with zJOS control panel.
- **PANELS** is an ISPF panel library that contains panel definitions for zJOS-XDI control panels. This library will be dynamically accessed by TSO users when zJOS control panel is in session.
- **MESSAGES** is an ISPF message library that contains message form definitions for zJOS-XDI control panel. This library will be dynamically accessed by TSO users when zJOS control panel is in session.
- **XMILIB** is a data library contains all installation materials of zJOS-XDI in TSO XMIT format. This library has to be kept for use by INSTALLX exec to maintain update track.

## zJOS-XDI Application Datasets

Application dataset is a dataset that contains zJOS-XDI user materials. zJOS-XDI installation generates some application datasets such as databases, logs, user module library, user's Rexx and Clist exec libraries, parameters library. Some could be inherited from package given samples, whereas some others are developed by users from scratch. Names of application data should follow user's naming standard. However, application datasets those are inherited from given samples, originally use zJOS-XDI naming format. But you are recommended to change them to follow your own standard instead. This to avoid unexpected replacement when you apply patches wrongly.

Application datasets which are developed by users from scratch are automation database and spool distribution logs.

- **Automation database** is a relational database into which all scheduler and EMS tables reside. Although the moment, this DB is only used for scheduler (Puspa) tables, format and structure for EMS tables are already prepared. This DB is supported by DIV technology to keep all tables floated on memory all the time for performance reason. Hence physically it must be allocated as a VSAM linear dataset (LDS). This VSAM must be accessed by zJOS-XDI main address space as DIVDATA file
- **Spool distribution logs** are logs for use by spool distribution, by means AutoXfer. These logs are supported by DIV technology to keep them floated on memory all the time for performance reason. Hence physically



it must be allocated as a VSAM linear dataset (LDS). This VSAM must be accessed by XDILGR address space as DIVDATA file

Application datasets which are inherited from package given samples are custom jobs, custom module, parameters, scheduled jobs, rules and rexx exec libraries. To describe them easier, names mentioned below are the last qualifier of their original given names.

- **CUSTLIB** is a text library that contains custom data source in text format, such as userid table, DIV capacity and all other definition source codes generated during installation steps. You also have chances to customize or make some changes to these codes. Binary load modules resulted from these source codes are stored into CUSTMOD.
- **CUSTMOD** is a load library that contains custom data modules in binary format, such as userid table, DIV capacity and all other binary object modules built during installation steps. This library must be accessed on top of STEPLIB concatenation of zJOS-XDI main address space and TSO logon procedure for userid who needs to access zJOS control panel.
- **PARMLIB** is a data library that contains samples parameters for all zJOS-XDI products. You can use them just to prove the functions only. You are responsible to customize them or add new parameters to meet your production requirements. This library must be accessed in all zJOS-XDI address spaces as PARMLIB file (DD name).
- **RULELIB** is a text library that contains samples of automation rules. Rule is a rexx exec program. You can use them just to prove the functions only. You are responsible to customize them or add new rules to meet your production requirements. This library must be accessed in XDIRULE address space as SYSEXEC file.
- **SAMPJOBS** is a text library that contains samples of jobs JCLs meet with sample of schedule table XDISCD00 member in PARMLIB. These jobs are just to prove scheduler (Puspa) functions only. Nevertheless, you need to make some corrections to meet your system environment before use them. This library must be accessed in zJOS-XDI main address space as JCLLIB file.
- **REXXLIB** is a Rexx exec library contains samples of Sekar Rexx function applications that can be use in either foreground or background job. To execute them, this library must be accessed in SYSEXEC file, where LINKLIB and LPALIB must also be accessed in STEPLIB file.
- **PROCLIB** is a text library contains JCL procedures for all zJOS address spaces. INSTALLX exec customizes and copies all these procedures into current system procedure library. You, however, could include this library into system procedure library in current MSTJCLnn instead.



## 2.1.2 zJOS-XDI JCL Procedures

Start Sekar means bring zJOS-XDI up. This means start zJOS-XDI address spaces and activate zJOS subsystem functions. If it is at the first time since IPL, zJOS subsystem initialization is done first prior to activate its functions and once only. zJOS subsystem remains floated on z/OS system area regardless address spaces are brought down.

Before you start zJOS-XDI, make sure all associated procedure JCLs resulted during installation steps correct. INSTALLX exec stores all procedures in your current system procedure library (e.g. SYS1.PROCLIB). There are 5 procedures you should verify; XDI, XDILGR, XDIAXFR, XDIRULE and XDISCD.

### XDI – Main procedure

Main procedure means a procedure to start zJOS-XDI main address space. Its given name is XDI. This because in the first design, address space was only for XDI part (e.g. AutoXfer). zJOS part (e.g. Sekar and Puspa) came later and most of them initially run in subsystem level. Although finally zJOS part was enhanced (since 2.1.3) with newly added 3 components, database manager, user interface and network server (for integration) which require to run on the main address space, its inherited procedure name, XDI, is still maintained to avoid confusion. .

The XDI member is fully commented, where more than 80% are remarked lines, so please read carefully. It actually consists of 12 JCL cards as shown below:

```
//XDI PROC START=00,V=2,LVL=19,
//          HLQ=NIT,SSN=ZJOS,OPT=,ZDD=ZJOSLIB
//DEREXEC EXEC PGM=DERJOS,REGION=0M,
//          DYNAMNBR=99,TIME=1440,
//          PARM='FILE=&START,SSN=&SSN,OPT=&OPT,ZDD=&ZDD'
//STEPLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..CUSTMOD
//          DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LINKLIB
//          DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LPALIB
//ZJOSLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..DYNLPA
//          DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LPALIB
//PARMLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..PARMLIB
//CMDLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..PARMLIB
//JCLLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..SAMPJOBS
//DIVDATA DD DISP=SHR,DSN=ZJOS.PUSPA.LDS
```

PROC card consists of 6 argument keywords and labeled as XDI. You can change the label, but, you must also change the member name accordingly. Keyword V, LVL and HLQ are just for JCL substitution. You can change them as necessary, or eliminate them, although leave them as they are much better.



Keyword START, SSN, OPT and ZDD are mandatory, since their values are passed to zJOS-XDI programs. These keywords will be discussed in par 2.2.

START=xx specifies member of XDISYSxx to be used as default main zJOS system parameter. Suffix xx can be any 2 digits valid member name. Take a look the content of this member. Most of content of XDISYSxx are used by AutoXfer, since it is first zJOS product. You can change value of xx to be set as default value, but associated valid XDISYSxx member must be exist. If you only run Sekar, the only important things in XDISYSxx member are CMDTAB=cc and KEY=AUTO:aaaaaa-aaaaaa-aaaaaa. Value cc for CMDTAB keyword specifies XDIEMSc member to be used as a source of EMS table for Sekar. Given value from installation package is 00. You can modify cc in XDISYSxx member directly or using XDI panel, but you must prepare associated XDIEMSc member in XDI parameter library. To prepare XDIEMSc member, however, you must not edit it directly. This member contains some sensitive binary values which can not be edited manually unless you have already very familiar with zJOS-XDI internal architecture, so you can use ISPF editor with HEX mode ON. Though, you are still not recommended to do this way.

KEY=AUTO:aaaaaa-aaaaaa-aaaaaa specifies license key of zJOS/Sekar®. If the key is valid, Sekar will work unlimited for you until the key is expired. You will be reminded since 30 days before expiration. For non-permanent license, you should ask new key to your zJOS support before your key is really expired.

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 6 DD cards, STEPLIB, ZJOSLIB, PARMLIB, CMDLIB, JCLLIB and DIVDATA. Although not all DDs used by Sekar, they must exist to avoid JCL error, and must be specified in procedure JCL to avoid program logical error, as this procedure is for an STC that run all zJOS-XDI products, not just Sekar alone.

STEPLIB DD must point to 3 load libraries, CUSTMOD, LINKLIB and LPALIB if zJOS standard installation is chosen. If zJOS is installed permanently in LPA, STEPLIB DD must only point to CUSTMOD and LINKLIB.

ZJOSLIB DD name actually can be any valid DD name matches with specified value in ZDD parameter keyword. ZJOSLIB must point to 2 zJOS-XDI load libraries, DYNLPA and LPALIB. DYNLPA library contains critical programs only modules that must be loaded onto LPA. If you choose to let zJOS-XDI to load itself onto LPA dynamically, it loads the whole DYNLPA and some modules from LPALIB onto Dynamic LPA via ZJOSLIB DD. That is why LPALIB must also be concatenated in STEPLIB to make the rest accessible. Don't worry about the search order. Although LPALIB is addressed in STEPLIB, some modules which are already in LPA will be search first by zJOS process manager.



If you choose to install zJOS-XDI onto LPA permanently, by means PLPA, MLPA or FLPA, however, you must remove DYNLPA and LPALIB from this procedure JCL and consequently you must put both libraries in LPALSTnn, IEALPAnn or IEAFIXnn member of system parameter library to let system load them during nucleus initialization progress (NIP). STEPLIB concatenation only consists of CUSTMOD and LINKLIB. ZDD parameter keyword must be set to either blank or STEPLIB. STEPLIB is the internal default, so when you specify blank, it will be changed to STEPLIB internally.

PARMLIB and CMDLIB DDs must address the same dataset, PARMLIB library. CMDLIB is actually inherited from earliest XDI/AutoXfer and it no longer use since version 2.1.2. It just to keep compatibility of long range AutoXfer releases.

JCLLIB DD is a library or concatenated libraries of scheduled jobs JCL which is used by Puspa only for automatic workloads scheduling. Initially it points to given SAMPJOBS library. Sekar doesn't use these libraries. Although you do not use Puspa in your environment, however, at least one dummy library must be specified to avoid JCL error problem.

DIVDATA DD must point to VSAM linear dataset (LDS) to hold most of zJOS-XDI databases. These databases are structured as multiple linked-list for relational access. Access method uses zJOS protocol based on DIV technology.

### **XDILGR – Procedure to start AutoXfer Logger**

Although Sekar doesn't depend on AutoXfer and its Logger, main address space always starts XDILGR automatically at initialization. If XDILGR is improper, error message will be appeared. You actually can ignore it if you don't use AutoXfer. However, if you don't like error message, you have to prepare a valid XDILGR procedure. Once it up, you can bring it down normally

The XDILGR member is fully commented, so please read carefully. The member is actually consists of 5 JCL cards as shown below:

```
//XDILGR PROC START=00,V=2,LVL=12,HLQ=NIT
//LOGGER EXEC PGM=DERLGR,REGION=0M,DYNAMNBR=99,
// TIME=1440,PARM='FILE=&START'
//STEPLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LINKLIB
// DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LPALIB
//PARMLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..PARMLIB
//DIVDATA DD DISP=SHR,DSN=NIT.ZJOSXLGR.LDS
```

If AutoXfer is not being used, nothing is important except to just make sure that XDILGR procedure is a correct procedure. All you need are; make sure all symbol variables are correct, all datasets pointed by each DD card exist, and especially for dataset name specified on DIVDATA DD card must be VSAM LDS.



If AutoXfer is being used in your environment, it very important that you must understand very well XDILGR procedure. If so, please read AutoXfer user guide.

## **XDIAXFR – Procedure to start AutoXfer externally**

AutoXfer by default is a subtask that runs within zJOS-XDI main address space. As it somehow deal too much I/O, AutoXfer can potentially degrade the whole address space which directly affect Sekar and Puspa performance as well. To avoid such situation, an option is given to run AutoXfer externally in a separate address space.

XDIAXFR is a procedure used by zJOS process manager to do it. You must neither change its name nor start it from outside zJOS-XDI control panel. This consists of 5 JCL cards as shown below:

```
//XDIAXFR PROC V=2,LVL=19,HLQ=NIT
//AUTOXFER EXEC PGM=DERAXF,REGION=0M,
//          DYNAMNBR=99,TIME=1440
//STEPLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LINKLIB
//          DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LPALIB
//PARMLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..PARMLIB
```

All argument keywords are subset of main procedure argument, so you can refer to main procedure to learn the detail.

If AutoXfer is not being used, XDIAXFR actually never be used. However, in case you want to do some trials, verify this procedure is a nice to have.

## **XDIRULE – Procedure to start an automation rule**

XDIRULE is actually a procedure to start TSO background job to run automation rule and executed in conjunction to rule action. Normally rule is a Rexx exec program. However, you can use any program that run on TSO as a command.

XDIRULE procedure consists of 8 JCL cards as shown below:

```
//XDIRULE PROC M=,ARG=,V=2,LVL=19,HLQ=NIT
//XDITSO EXEC PGM=IKJEFT01,
//          DYNAMNBR=20,REGION=0M,PARM='%&M &ARG'
//STEPLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LINKLIB
//          DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LPALIB
//SYSEXEC DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..RULELIB
//SYSTSIN DD DUMMY
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
```



Most of argument keywords are common, for JCL substitution. The only different are keyword M and ARG. Both will be filled up by Sekar when an automation rule is executed. M will receive rule name, which is a member name of RULELIB.

ARG will receive arguments for the rule. ARG is the most important thing that is used by Sekar to pass associated event information to your rule program. It will be discussed further in chapter 5, especially par 5.2 and 5.3.

Most of JCL cards are standard I/O table for TSO background execution with RULELIB accessed as SYSEXEC file. As RULELIB is an application dataset, you may change it to your own instead. STEPLIB DD must address zJOS-XDI LINKLIB and LPALIB for standard installation. If permanent LPA is chosen, you must remove LPALIB. If you use your non-rexx program as a rule, your library must be concatenated on top of STEPLIB.

### **XDISCD – Procedure to submit a job**

The way Puspa schedules each job is directly unload the job JCLs into internal card reader through SSI connection to JES2. In the most cases, connection to JES2 always granted. However, in case somehow the connection is not granted, the alternate way is provided. Puspa will automatically start XDISCD to submit a job.

As it is started internally, the name of XDISCD must not be changed. XDISCD is actually not a zJOS-XDI program. It rather is IEBEDIT utility to copy a member in a PDS into JES2 INTRDR spool. The source PDS will be substituted to a library name that is on top of JCLLIB DD concatenation in zJOS-XDI main procedure.

XDISCD procedure consists of 7 JCL cards and no zJOS-XDI system dataset involve as shown below:

```
//XDISCD PROC JOB=,LIB=
//SCHED EXEC PGM=IEBEDIT
//SYSPRINT DD SYSOUT=(A)
//SYSUT1 DD DDNAME=XDILIB
//SYSUT2 DD SYSOUT=(A,INTRDR),DCB=BLKSIZE=80
//SYSIN DD DUMMY
//XDILIB DD DISP=SHR,DSN=&LIB.(&JOB)
```

In PROC card, there are 3 keyword arguments, JOB and LIB. Nothing to bother, both arguments will be filled up by Puspa without your involvement.

As it is just an ordinary IEBEDIT utility, you can actually use it for your own work regardless any zJOS-XDI product is active. Nevertheless, copy it with different name is strongly recommended.



## **XDA – Procedure to start zJOS Agent for z/OS**

zJOS-XDI is designed as an integrated automation and scheduling solution. This means, if you have multiple hosts or servers, you have chances to automate them integrally using zJOS-XDI. To do this, you only need one host or server run full zJOS-XDI system and the rest just run zJOS Agent each. At the moment, only Agent for z/OS platform is provided.

See chapter 5 especially sub par 5.2.2 to learn further about XDA procedure JCL and zJOS Agent .

## **2.2. Starting and Stopping zJOS**

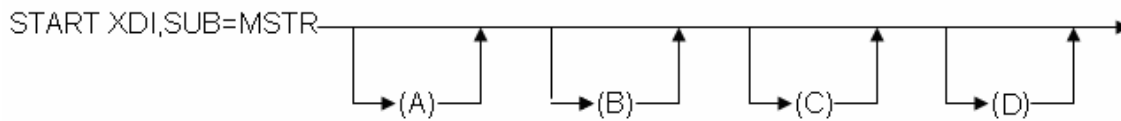
XDI (zJOS main address space) is an address space used by zJOS to initialize zJOS subsystem and to place all zJOS subtasks. Hence, starting zJOS address space at initial time performs both, initializing zJOS subsystem and attaching all zJOS major subtasks. Starting zJOS address space at subsequent time will perform attaching all zJOS major subtasks only. zJOS subsystem needs to be initialized once only in each IPL period.

zJOS major subtasks are categorize into 5 zJOS-XDI major components, Sekar, Puspa, AutoXfer, command processor and Socket server. Except for Sekar and command processor, all other major subtasks up only when each associated component is activated.

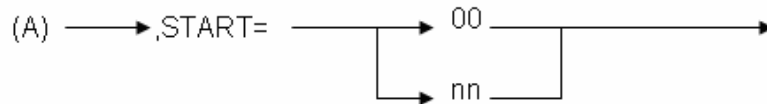
Sekar functions are supported by 8 major subtasks plus zJOS subsystem, which always up regardless Sekar is active. Once they up, they remain up along with zJOS address space. In each major subtask, ESTAE type recovery routine is provided to keep it remains up in all situations. This means, in case an exception condition is encountered, recovery routine will take action to avoid the task ended abnormally. Actually such mechanism is applicable for all zJOS major and minor subtasks.

During performing the works, each major subtask may generate several minor subtasks depend on its workload. This is to avoid workloads from being queued in ECB level. Most of minor subtask is non-loop task, which is up when attached by its major subtask, performs the works and then terminate upon completion of its works.

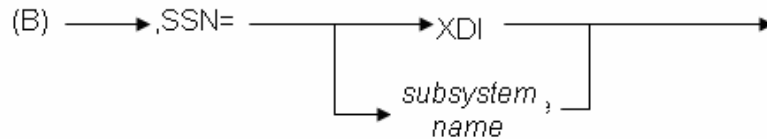
Since all zJOS major and minor subtasks and subsystem represent all zJOS-XDI products, including Sekar, to bring Sekar up, zJOS address space must be brought up first. Use START command to bring zJOS up with the following syntax:



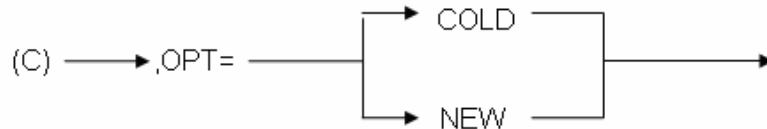
Option A:



Option B:



Option C:



Option D:



**SUB=MSTR** argument specifies that zJOS 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.

**START=** is an optional argument, to select zJOS-XDI system parameter, by means XDISYSxx member of zJOS parameter library, to be use to initialize zJOS main address space. An 'xx' value in START=xx argument reflects 2-digit suffix of XDISYSxx member. Supplied member in zJOS-XDI installation package is XDISYS00, which is the default when you ignore this argument.

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

**OPT=** is an optional argument, to specify whether zJOS-XDI is to refresh all its parameters from parameter library (COLD) or to refresh all its LPA modules in dynamic link pack area (NEW). If NEW is selected, COLD is also in effect. NEW option is recommended only when you have applied patches or updates to your zJOS-XDI which change several core modules. If you avoid this OPT= argument, warm start is selected, which means zJOS reuses all of its in-storage parameters. Since no I/O is in effect, zJOS initialization will be done much faster. This is the best way to bring zJOS up in the normal situation. You need COLD



start option only when you have made changes zJOS parameters and you want it to take effect. Refreshing parameters can also be done on the fly.

**ZDD=** is an argument to inform zJOS process manager the DD name in which zJOS LPA modules reside. Specified DD name must address the concatenation of 2 zJOS system datasets, DYNLPA and LPALIB and applicable only if zJOS-XDI is installed outside LPA (standard). If both libraries are in permanent LPA, ZDD must be avoided or specified as ZDD=STEPLIB.

Regardless your license, Sekar will automatically be brought up when zJOS address space initialization complete. If you don't license Sekar, EMS will still active on your zJOS address space, however, it only works 30 times a day. When 30 works has been reached, Sekar ignores all subsequent work until 0:00:00 o'clock.

Once zJOS address space is up, all zJOS subsystem commands then available for you. All zJOS subsystem commands must be prefixed with dot (".") or verb of "XDI" followed with a blank space. For example, to display operation status, issue

**.STATUS**

or

**XDI STATUS**

Then status information appears on console screen or syslog as shown in figure 2.1 below.

```
00- 16.22.24      .status
- 16.22.24 STC09248  DERCMD068I zJOS XDI status:
- Component- Stat- -Agent-- Tbl  works -Usage-- #dayX
- Sekar (EMS)  UP   ACT(SSI) IN   00007 LICENSED none
- Puspa (SCD)  UP   READY  IN   00000 **DEMO**  ..?!
- AutoXfer     DOWN INACTIVE OUT  00000 **DEMO**  ..?!
- Net-Server   DOWN INACTIVE N/A  00000 standard none
- zJOS XDI statistics:
- Config: SSN=XDI  Load=LPA COM=0802A3A0 WSA=00C4EF90
- 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=00000 SQBS=00000 EOTS=00000 RMG=00000
- State: NORMAL  Parm: SYS=00 EMS=00 SCD=00 DEST=00
- SCD: Lib=0 0=EVXMS M=EVXMS Pos=ASID-STACK  EnQ=FREE
- 16.22.24 STC09248  DERCMD201I zJOS is ready to accept command.
```

Figure 2.1: Appearance of zJOS status information

If dot (".") is already used by other subsystem in your environment, you should not use dot for zJOS. This will cause inter-subsystem conflict. In such case, please inform your XDI representative

You can also invoke zJOS command via MODIFY command to zJOS address space. Such way will not involve zJOS subsystem. For example, to display zJOS status, issue

```
F XDI,STATUS
```

It results the same effect as when you use zJOS subsystem command. Other non-subsystem way is invoking zJOS command from zJOS console interface panel in TSO. To obtain complete status information as shown in figure 2.2, just press enter on zJOS primary panel.

To stop zJOS address space, issue 'SHUTDOWN' command to zJOS-XDI. To do this, you can either use zJOS subsystem command;

```
.SHUTDOWN
```

Or MODIFY command:

```
F XDI,SHUTDOWN
```

When shutdown process is complete, all major and minor subtasks are down, and zJOS address space memory is end. Although, all in-storage parameters are kept and zJOS subsystem remains active held unless an exception condition was encountered during address space was up. Nothing is done by zJOS-XDI during active held state. To bring zJOS address space back up, simply issue;

```
.START
```

## **2.3. Preparing Sekar**

By default, Sekar parameter sample is provided in zJOS-XDI product installation package, in XDIEMS00 member. You should not tailor directly to any zJOS-XDI PARMLIB member, including XDIEMSxx, which can destroy their sensitive binary information. You should use XDI ISPF panel instead. Issue "XDI" in any ISPF session, zJOS-XDI primary control panel will appear on your terminal screen as shown in figure 2.2 below. Then click action bar, zJOS-XDI action bar menu will appear in small window as shown in figure 2.2a. To reach the parameters, select option 1 of action bar menu.

You have to customize Sekar provided sample parameter to fit your environment prior to bring zJOS up at initial time. This to avoid unexpected situation, since Sekar will be automatically brought up when zJOS address space initialization is complete.





```

Action Help
zJOS-XDI Control Panel Row 561 to 582 of 582

Command Product State Table Suf Works Usage Day
-----
Sekar (EMS) <UP> ACT(SS) loaded 00 000644 LCNSD/YR 0817
Puspa (SCD) <UP> ACTIVE loaded 02 000236 LCNSD/YR 0817
AutoXfer EXTR ACTIVE loaded 00 000000 LCNSD/YR 0817
Net Server <UP> ACTIVE N/A ** 000000 standard none

Date Time Log
11.278 04:25:21
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=000000
11.278 04:25:21 3ES I/F: Up=Y PIT=Y Conn=Y Irdr=Y FR(5=N,12=N,22=N)
11.278 04:25:21 Q's: ARQ=021 SQB=002 EOT=002 RMG=000 RML=010 QTR=000
11.278 04:25:21 State: NORMAL Parm: SYS=00 EMS=00 SCD=02 DEST=00
11.278 04:25:21 SCD: Lib=0 O=EVX M=EVX Pos=SCHEDULE-SCT EnQ=FREE
11.278 04:25:21 SCD Free-pool: SCT=001976 TRG=0058898 EOT=0059582
11.278 04:25:21 SCD Used-pool: SCT=001024 TRG=0001102 EOT=0000418
11.278 04:25:21 SCD Curr-pool: SCT=000389 TRG=0000439 EOT=0000418
11.278 04:25:21 SCD: Exec=000237 Wait=000024 Void=..... Rest=000128
11.278 04:25:21 SCD: Cur=003 Max=013 S=11278/01:24:17 Elaps=00:25:26
11.278 04:25:21 SCD: TRV=00025 Max peak:(SCD=0080 RMG=0002 EVX=0001)
11.278 04:25:21 Exits: EOJ=(Up,ACTIV) WTO=(Up,ACTIV) DSM=(Up,ACTIV)
11.278 04:25:21 MSG:(RMG=Y,EVX=Y,SCD=Y) Trace:(SSI=N,XDI=N) Timer=N
11.278 04:25:21 ParmLib=NSI.ZJOS219.PARMLIB
11.278 04:25:21 LDS=ZJOS.PUSPA.LDS,Vol=Z195SHR
11.278 04:25:21 Your (DERU ) authorities: Oper=Y Setting=Y Update=Y
11.278 04:25:21 Genlevel=20100101 CPUid=00AAF4/2098 System=NSILAB /SYS1
***** Bottom of data *****

Command ==> Scroll ==> CSR

```

Figure 2.2: zJOS primary control panel

Although you don't license Sekar nor need Sekar, you have to activate Sekar with minimum EMS settings to have zJOS subsystem active. Select option 1 of action bar menu.

## Notes:

1. All zJOS-XDI ISPF panels have more than 24 rows and in some cases, certain information indicated by color. Hence you must use extended IBM 3270 terminal model 3 or 4.
2. Some menus are placed in action bar. Hence before you entering zJOS session, make sure action bar is enabled in your ISPF settings.



```

Action Help
1. Administering zJOS
2. Viewing Scheduler Activities
3. Managing In-memory Scheduler Tables
4. Viewing AutoXfer Logs
* Starting AutoXfer Logger
6. Stopping AutoXfer Logger
* 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=000000
11.278 04:25:21 JES I/F: Up=Y PIT=Y Conn=Y Irdr=Y FR(5=N,12=N,22=N)
11.278 04:25:21 Q's: ARQ=021 SQB=002 EOT=002 RMG=000 RML=010 QTR=000
11.278 04:25:21 State: NORMAL Parm: SYS=00 EMS=00 SCD=02 DEST=00
11.278 04:25:21 SCD: Lib=0 O=EVX M=EVX Pos=SCHEDULE-SCT EnQ=FREE
11.278 04:25:21 SCD Free-pool: SCT=001976 TRG=0058898 EOT=0059582
11.278 04:25:21 SCD Used-pool: SCT=001024 TRG=0001102 EOT=0000418
11.278 04:25:21 SCD Curr-pool: SCT=000389 TRG=0000439 EOT=0000418
11.278 04:25:21 SCD: Exec=000237 Wait=000024 Void=..... Rest=000128
11.278 04:25:21 SCD: Cur=003 Max=013 S=11278/01:24:17 Elaps=00:25:26
11.278 04:25:21 SCD: TRV=00025 Max peak:(SCD=0080 RMG=0002 EVX=0001)
11.278 04:25:21 Exits: EOJ=(Up,ACTIV) WIO=(Up,ACTIV) DSM=(Up,ACTIV)
11.278 04:25:21 MSG:(RMG=Y, EVX=Y, SCD=Y) Trace:(SSI=N, XDI=N) Timer=N
11.278 04:25:21 ParmLib=NSI.ZJOS219.PARMLIB
11.278 04:25:21 LDS=ZJOS.PUSPA.LDS,Vol=Z19SHR
11.278 04:25:21 Your ZJOS is ready to accept command.

```

Figure 2.2a: zJOS primary action bar menu

## 2.4. Starting and Stopping Sekar

You do not need to start Sekar at initial time, since it will up automatically. You, however, need to restart it to work when it was down. Once you bring Sekar down, it will remain down until you restart it back. When Sekar down, zJOS subsystem function is inactive, so zJOS subsystem command is not available. To bring it up back, issue the following MODIFY command:

```
F XDI,AUTO SSI
```

The following response will appear:

```

DERCMD201I zJOS is ready to accept command.
DERSIP017I zJOS subsystem activation successful.
DERSIP035I zJOS/Sekar (Event Manager) is now active.

```

Figure 2.3: zJOS response to F XDI,AUTO SSI

Do not be mistaken to issue F XDI,AUTO START. It will activate MCS function instead of subsystem, which is legacy from old zJOS version.

To bring Sekar down, issue the following command:

## 2. Getting Started



## **.AUTO STOP**

Then the following response will appear:

```
DERSIP038I zJOS subsystem deactivation successful.  
DERCMD201I zJOS is ready to accept command.  
DERSIP040I zJOS/Sekar (Event Manager) is now inactive.
```

Figure 2.4: zJOS response to AUTO STOP

### **Notes:**

1. As Sekar is central control of all zJOS subsystem functions, zJOS subsystem is only active when Sekar up. Hence bringing Sekar down is not recommended.
2. Although facility to stop Sekar is provided, this only for maintenance purpose.
3. When other zJOS function active, for example, when scheduler (Puspa) is active, stopping Sekar causes unpredictable result.

## **2.5. Preparing National Holiday Table**

In order to validate either event – action relationship in EMS or job triggering in automatic scheduling, national holiday sometime is an important factor. Once event/actions is set for certain date/time range, Sekar will automate it everyday unless you unselect some certain days of week. Sometime you want Sekar to do actions everyday to certain event except on the national holiday. Since it sound very specific requirement, Sekar won't know which days are your national holiday until you provide it. zJOS-XDI provides national holiday table into which you need to fill up all your national holidays.

To reach holiday calendar setup facility, select option 1 of action bar menu on zJOS console interface (zJOS primary panel), then, zJOS parameter panel appears as shown in figure 3.1. Before reach this panel, a window as shown in figure 2.5a appears asking which PARMLIB dataset and suffix you are going to use. You have to fill it if empty or change it or confirm it then press enter to let it continue to zJOS parameter panel.

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 2.5a: Asking/confirming suffix and library you are going to use.

Once suffix and PARMLIB are filled, and press enter key, then zJOS parameters panel (figure 3.1) appears. On the zJOS parameter panel, then click option bar (figure 2.5.b) and then select option 1 (holiday calendar).

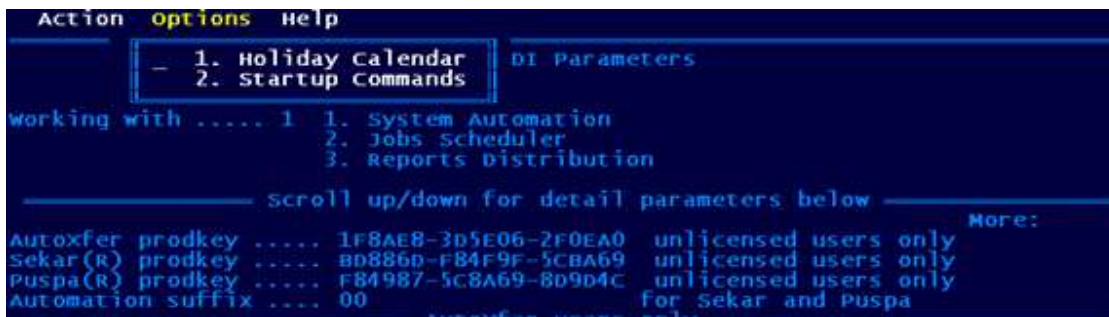


Figure 2.5b: Option bar

Then, larger window is popped up and holiday table appears as shown in figure 2.6. Type S in front of certain entry to obtain its detail, and then you can update it. To delete an entry, type D in front of selected entry, and then hit enter key. To add a new entry, type A anywhere in S column or use menu in action bar.

## 2. Getting Started

Action Options Help		
zJOS-XDI/Sekar 2.1.9		
Holiday Calendar Row 1 to 5 of 5		
Calendar year :: 2011		
S	Date	Description
-	20110420	hari Kartini
-	20110817	hari Kemerdekaan
-	20110818	hari Pramuka
-	20111005	hari Angkatan Perang
-	20111110	hari Pahlawan
***** Bottom of data *****		
Command ==> F3=Save/End F7=Up Scroll ==> CSR F8=Down		
F1=Help F12=Cancel		
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.ZJOS213.Z19LNK		
NETRC dataset ..... 00 xx of NETRCxx		

Figure 2.6: National Holiday table

When you type S, selected entry is then displayed in subsequent popped up window as shown in figure 2.7. You can make any changes as necessary on this panel, and then hit F3 save it or F12 to abort it.

When you type A or select an option 1 of action bar menu, the same panel as in figure 2.7 is then popped up with no entry. Fills up all fields, and then hit enter to insert this newly added entry. The panel remains on the window until you hit F3 or F12. This to give you chances for next new entry. To fill up next new entry, modify newly added fields, and then hit enter key again. When all new entries were added, hit F3 to finish it and return to previous panel. Hitting F12 will abort all newly added entries.

You can not update newly added entry straightly in this panel. Take a note that every time you modify fields on this panel, a new entry will be added instead of update existing entry. Hence to update newly added entry, hit F3 to return to previous panel (holiday table), then type S in front of it as ordinary update procedure.

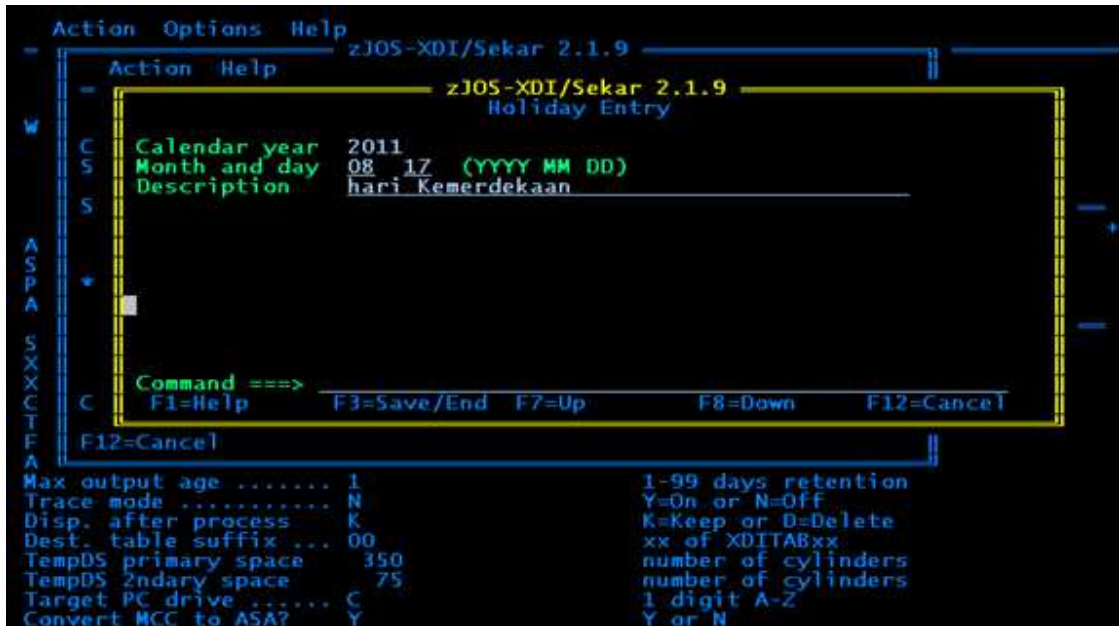


Figure 2.7: Holiday table entry

## 2.6. Preparing System Startup Table

IBM z/OS MVS operating system provides system startup facility to do all necessary works immediately after nucleus initialization process (NIP) complete, by means COMMNDxx member of system parameters library. Each entry of COMMNDxx is assumed as a valid MVS command text then passed to MVS command processor for execution. Normally, you use COMMNDxx to bring some application support tasks up, such as JES2, VTAM, TSO etc., and perform some necessary preparation, such as allocating system dump datasets, and so forth. Although any valid command can be executed, COMMNDxx does not care of the affect. Hence, tasks that have inter-dependencies should not be started together in the same COMMNDxx. For example, JES2 must up prior to VTAM, and VTAM must already up prior to TSO (TCAS). Hence, COMMNDxx should not be used to start JES2, VTAM and TSO together. Use it to start JES2 instead. When JES2 is up, start VTAM manually. Then start TSO manually when VTAM is up.

Although such effort does not really matter, sometime it causes potential problem when operators forget the sequent. In modern era right now, such case must be avoided, and automation is the most effective solution.

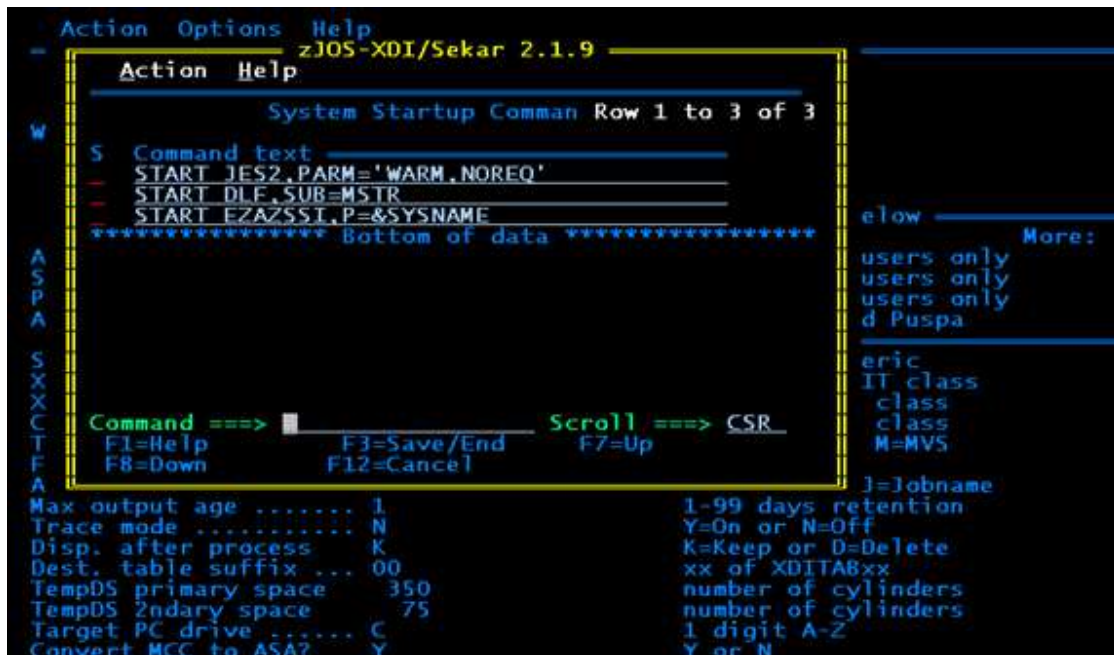


Figure 2.8: System startup command table

To have fully automated system, you should automate system startup either. Let Sekar to do complete system startup. To make sure all states can be monitored by Sekar, zJOS address space must be the only system task up immediately after NIP completed. Use COMMNDxx to start zJOS address space only instead. All other tasks are started by Sekar, by means zJOS address space.

zJOS-XDI provides such facility as COMMNDxx, called system startup command (SSC) table, which is executed immediately after zJOS-XDI base initialization complete, and only if zJOS is started immediately after NIP. This means, such mechanism not applicable when recycle zJOS, to guarantee SSC is executed once during IPL period.

Since COMMNDxx only to start zJOS, all original valid content of COMMNDxx should be migrated to SSC and EMS tables. SSC table should only contains startup command for independent tasks. In most of MVS system, SSC contains startup for JES, LLA, VLF, RACF and other independent tasks. The rest should be migrated to EMS action table as action in respond to completeness of their prerequisite tasks. For example, startup for VTAM is placed as action against JES establishment message (\$HASP492). Startup for TSO and TCPIP are placed as action against VTAM establishment message (IST020I).

Setting up EMS tables will be discussed in next chapter. Discussion here is only involving setting up SSC table. To do this, select option 2 of option bar menu, then SSC table appears in popped up window as shown in figure 2.8. To insert or add a new entry, type A in front of any entry or select option 1 of action bar





menu, then hit enter key. SSC entry then appears in subsequent popped up window. You need to fill up only one field, a command text. When you hit enter key, entry is inserted and panel remains in window for next new entry until you hit F3 to save all newly added entries or F12 to abort them all. You can not update newly added entry during this session.

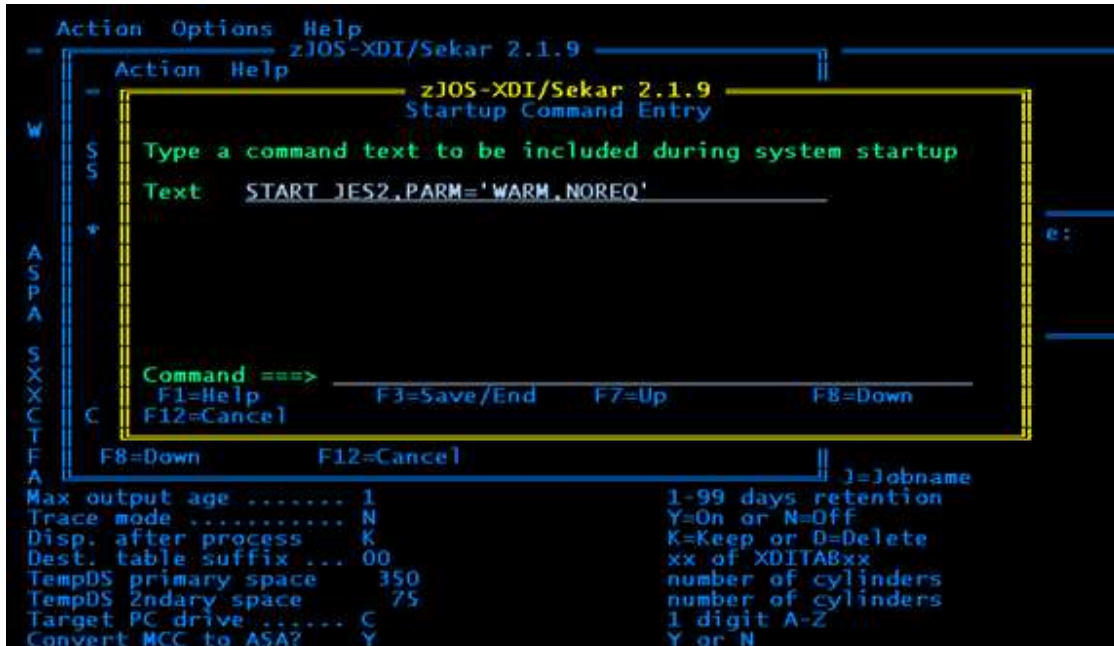


Figure 2.9: System startup command table entry

To update the entry, type S in front of selected entry and hit enter key, then the entry is popped up as shown in figure 2.9. Anything you overwrite on the panel will update the entry when you hit enter key. Hit F3 to accept the update and back to previous panel. If you hit F12, session is back to previous panel and update is aborted.

To delete the entry, type D in front of selected entry and hit enter key. Since deletion function is done straightly on table session, once enter key is hit, you can not abort it particularly. The only chance is, abort whole table.

To finish the SSC session, hit either F3 or F12. Hitting F3 will accept all you have done to SSC table permanently and close SSC window. No way to restore previous SSC table once F3 is hit. Whereas, hitting F12 will abort all you have done to SSC table. Previous SSC table is then restored and SSC window is closed.

Back to system startup, since you are using SSC table instead of COMMNDxx, Sekar up prior to all non-NIP system tasks startup, which means, Sekar has a chance to fully monitor all non-NIP system tasks. All independent non-NIP tasks



are started by SSC facility. Whereas all dependent non-NIP tasks are started by action in respond to event which represent state of each their predecessor tasks. For example, as independent task, JES2 is started by SSC. VTAM which depend on JES2 is then started by action in respond to \$HASP492 message which indicates that JES2 is up. TCPIP and TSO which depend on JES2 and VTAM, are started by action in respond to IST020I message which indicates that VTAM is already up. Finally, system startup is fully automated.



## Chapter 3 Working with Sekar Parameters

To work with Sekar parameters, you have to login to zJOS 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 on ISPF window as shown in figure 2.2. Reach action bar with cursor and click it (hit enter key) to open corner menu as shown in figure 2.2a. Then, select option 1 (administering zJOS) to reach to parameters panel as in figure 3.1. Before it is reached, a window is popped up asking 2-digit suffix and zJOS PARMLIB you want to manage as discussed in 2.5 on chapter 2.

On the top of this panel is a menu to which product you want to go. To reach Sekar parameters panel, select option 1 (system automation), then press enter-key. At initial time, before you do it, you have to complete Sekar product key and EMS suffix in this panel first. Ask your XDI support personnel to provide key.

The screenshot shows the 'zJOS-XDI Parameters' panel with a menu at the top for 'Working with' (1. System Automation, 2. Jobs Scheduler, 3. Reports Distribution). Below is a list of parameters with their current values and descriptions. The parameters are grouped into sections: 'AutoXfer users only', 'SSRF users only', and 'CA-Xcom users only'. The bottom of the panel shows function key shortcuts: F1=Help, F3=Save/End, F7=Up, F8=Down, F12=Cancel.

Parameter	Value	Description
AutoXfer prodkey	1F5C89-3D7B5C-6E89E4	unlicensed users only
Sekar(R) prodkey	EC0E0C-F89AC5-5C9D2D	unlicensed users only
Puspa(R) prodkey	F89CDD-5CA02D-CC1B2D	unlicensed users only
Automation suffix	00	for Sekar and Puspa
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	0	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
Splitting parameter	Y	Y if SSRF is installed
SSRF library	NIT.Z10S213.SSRFLIB	
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		

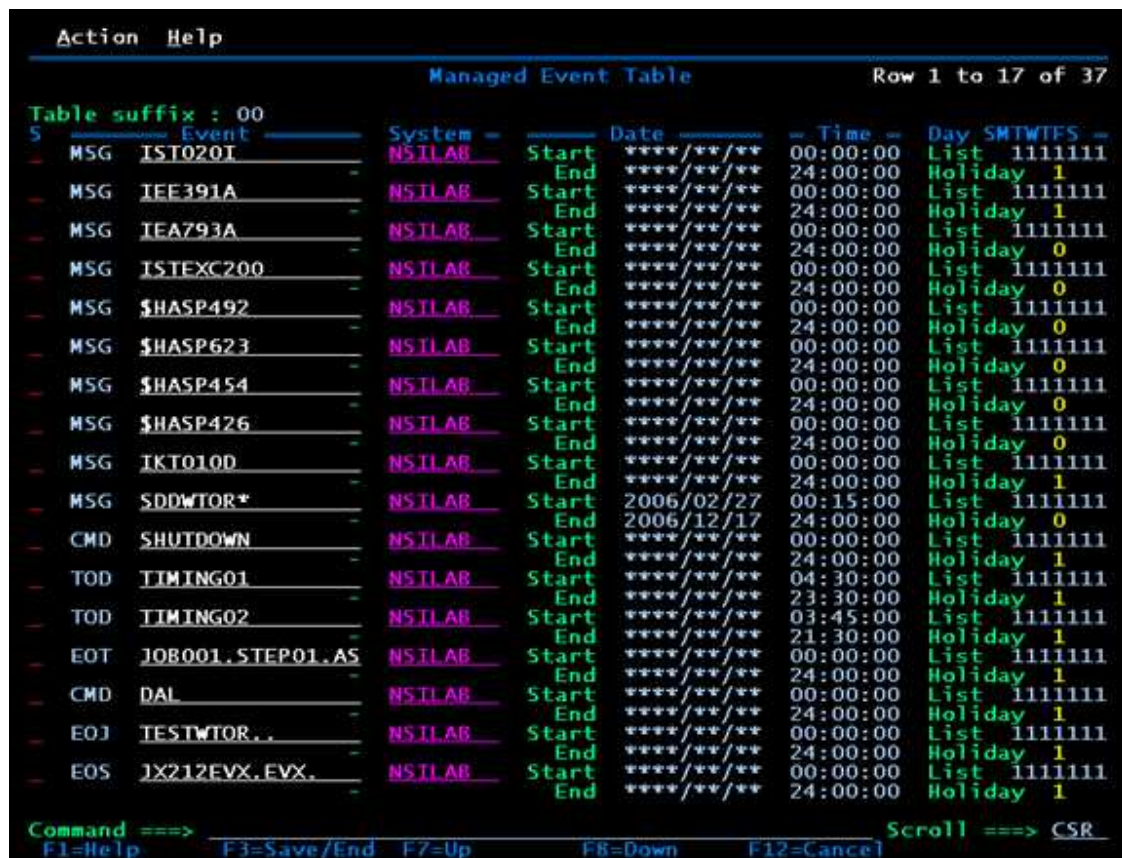
Command ==>

F1=Help F3=Save/End F7=Up F8=Down F12=Cancel

Figure 3.1: zJOS parameters panel

## 3.1. Managed Event Table

Managed event table as shown in figure 3.2, is Sekar primary EMS table panel. To reach this panel, select option 1 on zJOS-XDI parameters panel in figure 3.1. The panel show you current EMS table suffix as you selected in previous panel.



Event	System	Date	Time	Day SMTWTFS
MSG IST020I	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1
MSG IEE391A	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1
MSG IEA793A	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 0
MSG ISTEFC200	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 0
MSG \$HASP492	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 0
MSG \$HASP623	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 0
MSG \$HASP454	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 0
MSG \$HASP426	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 0
MSG IKTO100	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1
MSG SDDWTOR*	NSILAB	Start	2006/02/27 00:15:00	List 1111111
		End	2006/12/17 24:00:00	Holiday 0
CMD SHUTDOWN	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1
TOD TIMING01	NSILAB	Start	04:30:00	List 1111111
		End	23:30:00	Holiday 1
TOD TIMING02	NSILAB	Start	03:45:00	List 1111111
		End	21:30:00	Holiday 1
EOT JOB001.STEP01.AS	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1
CMD DAL	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1
EOT TESTWTOR..	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1
EOS JX212EVX.EVX.	NSILAB	Start	00:00:00	List 1111111
		End	24:00:00	Holiday 1

Figure 3.2: Sekar event table panel

### Managing event table

Managed event table panel (figure 3.2) provides facilities to manage event table as described in figure 3.3, e.g.: Function keys, action bar menu and S column prefix command. Function keys consist of 5 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.



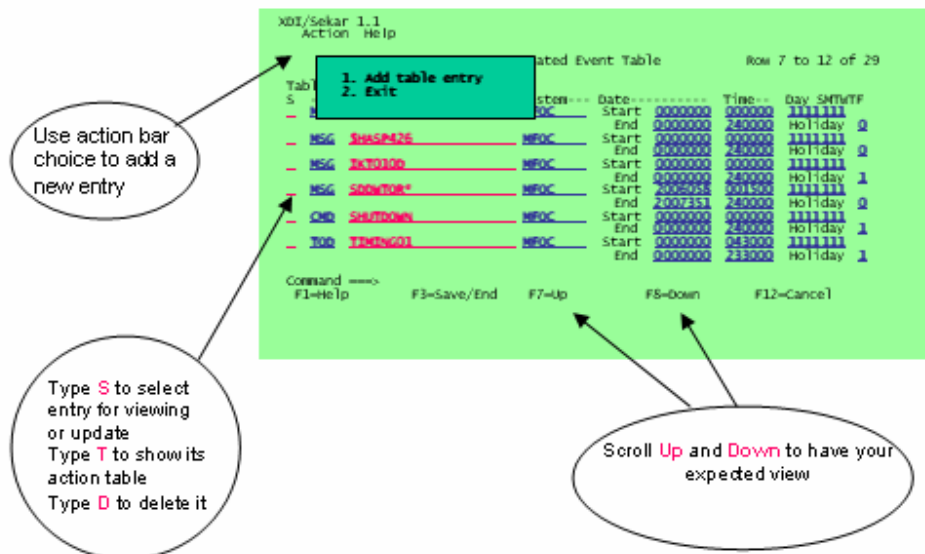


Figure 3.3: Managing event table

Action bar provides facility to add a new entry. Exit choice in action bar menu is to save and close the table, the same effect as hitting F3 key. Other facilities can be done in S column. S column is an input column for 1-digit action character. This gives you chance to manage the table. Valid action characters are:

- S – Select particular event entry in detail as shown in figure 3.4.
  - This gives you chance to update the detail of selected entry
- T – Show associated action table as shown in figure 3.5.
  - This gives you chance to manage action table of selected 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.

## Event columns

These are 2 columns describe event type and verb. Event type can be any one of 5 valid event types, (MSG, CMD, TOD, EOS or EOJ). Event verb can be an actual event verb, a part of event information or just a name to identify the event, depend on the type of event. For MSG type event, verb is the first substring of message text which is usually called message id. For CMD type event, verb is the command verb. For TOD type event, verb is just a unique string. You can use anything unique, for example "MORNING1". For EOJ type event, verb is job name of which to be detected. For EOS type event, verb is a combined job name and job step name of which to be detected.



## **System column**

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

Remote event can only be managed by Sekar if zJOS-XDI agent is active on that associated remote host. Event will be detected by agent and reported to Sekar, for actions. You can define associated actions in Sekar action table to be done in local machine, in originating host machine, or even in other remote machine as long as zJOS-XDI agent is ready.

## **Date and time columns**

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

## **Day list and holiday column**

These are minor timeframe to do second filter. Day list is list of valid week day from Sunday to Saturday. Holiday means national holiday you have registered in holiday calendar table. To continue processing, an event must comply that current day is a valid day, means the day is selected day, including if a holiday. By default, all 7 week days are selected if not holiday.

## **Saving and aborting changes**

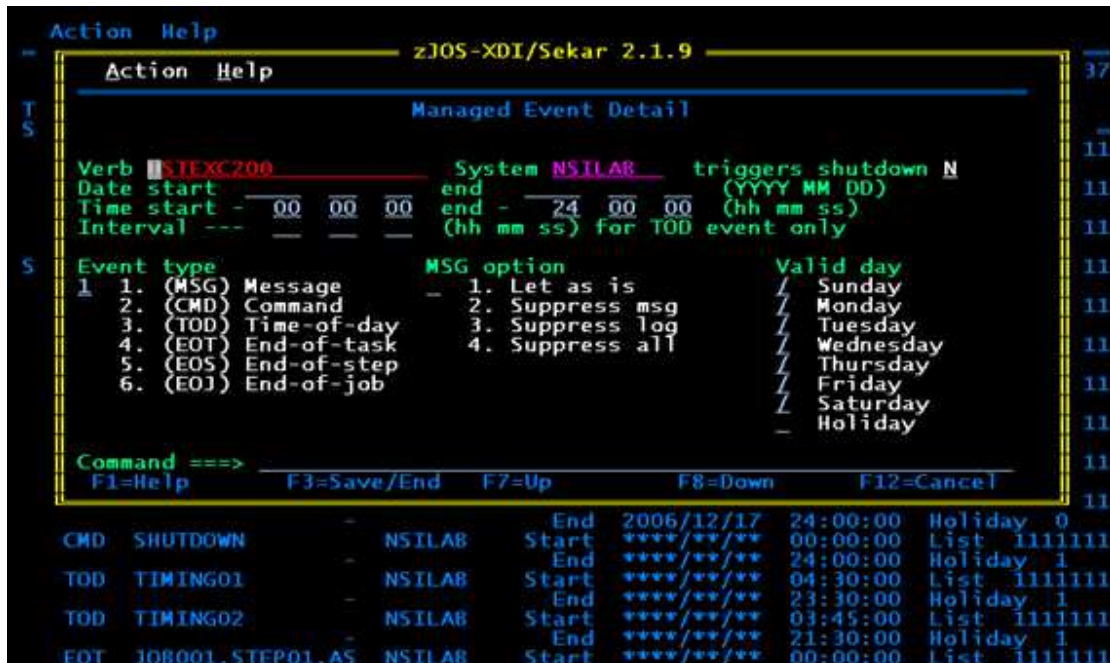
When you finish work with the event table and you want to save all changes you have made, press F3. Update progress appears in small window, then panel close and back to previous panel. XDIEMSxx member in XDI parameter library is then physically changed.

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. XDIEMSxx member in XDI parameter library is then remains unchanged. Take a note that abortion in this level is total cancellation. The XDI/ISPF interface will not remember what you have done so far. 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 an event entry. Delete command doesn't have detail level panel. Once "D" was invoked, selected event entry then instantly deleted from the table. To abort deletion you have done, the only way is total abortion.

### 3.2. Event Entry

Each event entry listed on the table as shown in event table (figure 3.2 or 3.3) can be zoomed in detail as shown in figure 3.4 when selected. Type S in front of selected entry, then stroke enter key, then “Managed Event Detail” panel appears in window. You can update each detail field on this panel.

All information on the previous panel appears in different form. Event verb, system on which event occurs and major timeframe (date and time) fields appear as data entry form. The rest appear as multiple choices menu and check boxes forms to minimize human error.



Managed Event Detail

Verb **STEXC200** System **NSILAB** triggers shutdown **N**

Date start \_\_\_\_\_ end \_\_\_\_\_ (YYYY MM DD)

Time start - 00 00 00 end - 24 00 00 (hh mm ss)

Interval --- --- --- (hh mm ss) for TOD event only

Event type

1. (MSG) Message	1. Let as is	/ Sunday
2. (CMD) Command	2. Suppress msg	/ Monday
3. (TOD) Time-of-day	3. Suppress log	/ Tuesday
4. (EOT) End-of-task	4. Suppress all	/ Wednesday
5. (EOS) End-of-step		/ Thursday
6. (EOJ) End-of-job		/ Friday
		/ Saturday
		/ Holiday

Command ==> F1=Help F3=Save/End F7=Up F8=Down F12=Cancel

CMD SHUTDOWN	-	NSILAB	Start	2006/12/17	24:00:00	Holiday 0
			End	****/~/~	00:00:00	List 1111111
TOD TIMING01	-	NSILAB	Start	****/~/~	24:00:00	Holiday 1
			End	****/~/~	04:30:00	List 1111111
TOD TIMING02	-	NSILAB	Start	****/~/~	23:30:00	Holiday 1
			End	****/~/~	03:45:00	List 1111111
EOT JOB001.STEP01.AS	-	NSILAB	Start	****/~/~	21:30:00	Holiday 1
			End	****/~/~	00:00:00	List 1111111

Figure 3.4: Managed event detail

When you finish updating, press enter key, then update notification appear in small window. Panel then close and back to previous event table panel. If you want to abort the update you have made, press F12 instead. Abortion alert then appear in small window. Abortion will only affect to this entry.

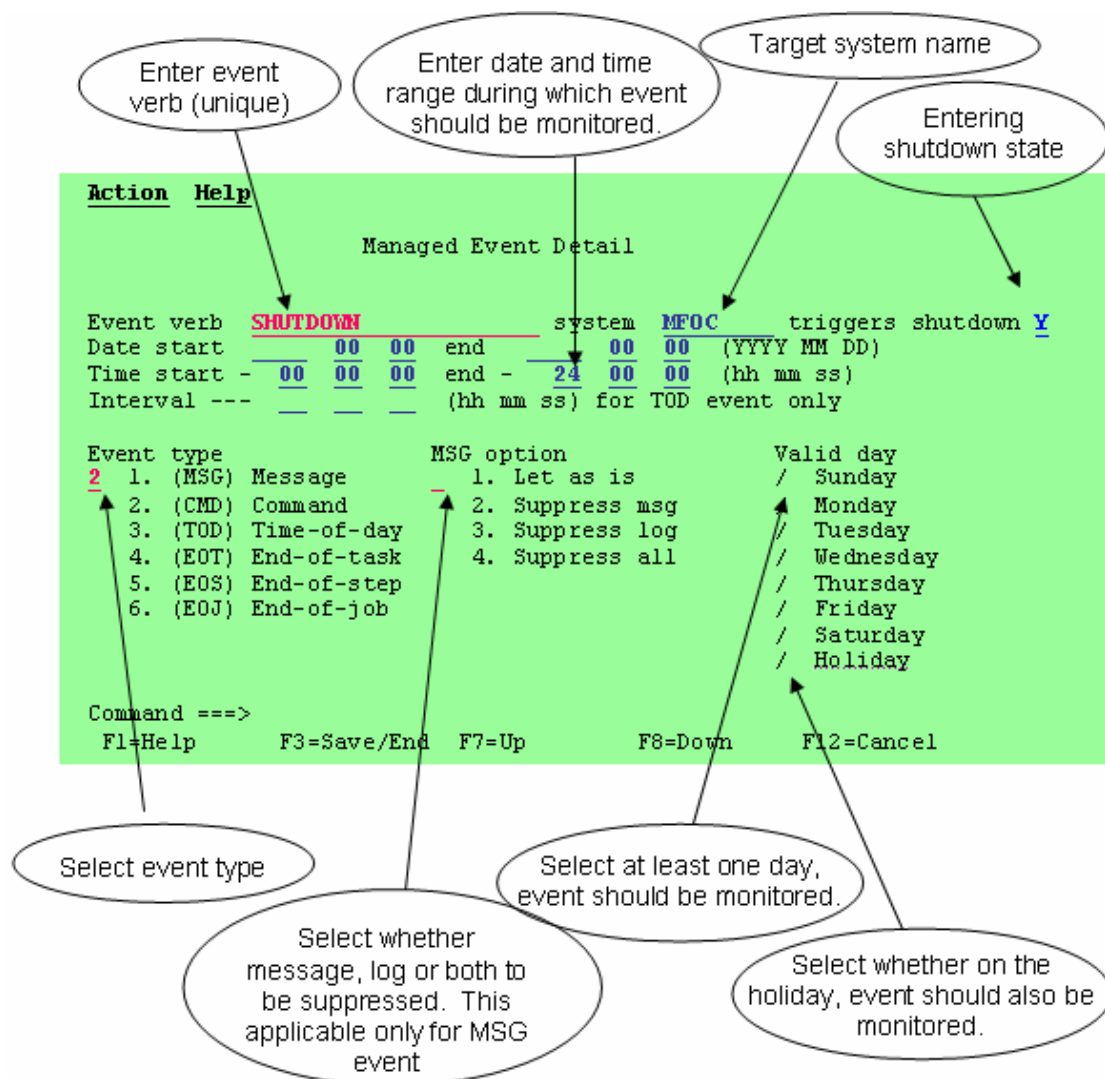


Figure 3.5: Detail event entry fields

Figure 3.5 explains in detail each field of event entry. In this example shows detail if SHUTDOWN event entry. The type of event is command. Actually MVS does not provide SHUTDOWN command. If you issue SHUTDOWN, normally system responds with message IEE305I explaining that command is invalid. With Sekar, you can make SHUTDOWN and other invalid commands become valid and execute certain process as what you want. Meanwhile, you can also change valid commands become invalid.

The process in respond to SHUTDOWN command is actually action or series of actions as you define in action table of EMS. When SHUTDOWN command is issued, Sekar captures it before MVS evaluating it, and followed by execution of defined actions in associated action table.



## 3.2.1 Event verb and System

Event verb and system name on which event occurs are a major part of event identifier. Except for TOD event, the text of verb is normally a part of information generated by or associated with the event. Verb text length is 1 to 20 bytes. For TOD event, the verb text is any unique string as explained in paragraph 3.1 in this chapter.

Minor part of event identifier is event type, which is message, command, TOD, EOJ or EOS.

### **MSG Event**

Event verb of MSG event normally message id. You can use first substring of message text starting from message id. Every occurrence of message from message processor, before it is reached in neither console buffer nor syslog, SSI part of Sekar catches it and evaluates it. The first evaluation is comparing it with event verb of each MSG event entry in event table. This step is done in address space of originating the event. Every matched entry found, event is then posted to action processor in zJOS address space, and message is returned to system with indicator whether to be suppressed according to suppression request stated in its entry in event table.

### **CMD Event**

Event verb of CMD event must be a command verb. Sekar doesn't care whether the command is a valid system command or just an abbreviation. As long as its verb is matched with any one of CMD event entries in event table, Sekar SSI then post a signal to action processor to continue event processing. Command is intercepted very soon after its issuance, before targeted command processor is reached. Matched command will never be returned back to the system.

Since command validity is not checked, you have a chance to make your own command without writing program. For example, "SHUTDOWN", as shown in figure 3.5, actually is not a valid command in z/OS environment. Regardless its validity, in Sekar, it is registered in event table and some actions are provided in associated action table. So, when verb "SHUTDOWN" is issued, it's matched and all associated actions are executed. The command itself is never returned back to the system, hence message IEE305I which telling that "SHUTDOWN" is an invalid command is never shown.

### **TOD Event**

Event verb of TOD event is not part of event information. It can be any unique string just to identify the event entry visually. The way TOD event is caught



internally very different with other event types. Timeframe information is used as timing specification which can be added with interval for periodic TOD event. To catch TOD event, actually during initialization Sekar sets timer according timing specification. Hence actually TOD event is created, instead of just intercepted.

## **EOJ Event**

Event verb of EOJ event is name of the job. Sekar doesn't care whether jobname is duplicated or used by wrong job, as long as matched with jobname of any one of defined EOJ entries. Though always unique, jobid can not be used, because it can not be predefined, since it is generated only when job is started.

## **EOS Event**

Event verb of EOS event is a combined of 8-byte jobname and 8-byte jobstep name. Sekar doesn't care whether combined jobname-stepname is duplicated or used by wrong job, as long as matched with jobname-stepname of any one of defined EOS entries.

## **3.2.2 Timeframe and Timespec**

Timeframe is combined of major and minor time filter during which event is monitored. Timeframe is applicable for all types of events, except for TOD. Whereas, timespec for TOD event only. Timespec is such timeframe during which TOD event is set to occur.

### **Start-date and end-date**

In most event types, start date and end date are major time filters during which event occurrence is eligible for further evaluation. No default for start- and end-date. If blanks or zeros, no filtering is done, rather, straightly to check clock time level filter. In event table panel appear as '\*\*\*\*/\*\*/\*\*' and in event detail as blanks. Date can also be specified partially, for example, '\*\*\*\*/09/\*\*'. This means, Sekar 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/\*\*'. Sekar grants the action 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 1<sup>st</sup> to 10<sup>th</sup> monthly, you must specify date range '\*\*\*\*/\*\*/01' to '\*\*\*\*/\*\*/10'. To get granted on every 1<sup>st</sup> to 10<sup>th</sup> monthly in every 1<sup>st</sup> 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 automation system if this entry is related to other entries.





For TOD timespec, start date and end date are major time range during which TOD event is set to occur. No default for start- and end-date. If not specified or set to zeros, no date range is set, rather, just clock time level setting. In event table panel appear as '\*\*\*\*/\*\*/\*\*' and in event detail as blanks.

For both timeframe and timespec, start date and end date, when specified, must be valid calendar date. Start date must be logically happen prior to end date. Otherwise, event entry will be marked as error entry and appropriate event will never be monitor, or TOD will never be set.

### **Start-time and end-time for non-TOD events**

In most event types, start time and end time are second major time filters during which event occurrence 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 event occurs during valid start- and end-date, will be considered as valid event and filtering is continued to minor timeframe.

Such time filtering is applicable only for MSG, CMD, EOJ, and EOS type events. Time interval is not applicable. If you fill it up, Sekar will ignore it.

### **Start-time, end-time and interval for TOD event**

For TOD timespec, start-time and end-time are second major time range during which TOD event is set to occur. Default start-time is 00:00:00, means timer is set to start at 0:00:00. Take a note that start-time for TOD timespec is not a time to begin event monitoring, rather, an exact start of timer. This means, TOD event occurs exactly at start-time.

Time interval (optional), when specified, means timer is set periodically. TOD event occurs at start-time, and reoccurs periodically in every time interval, until end-time is reached. Default end-time is 24:00:00, means timer is set to stop at 24:00:00. End-time is meaningful only for periodical TOD event. Unless interval is specified, TOD event occurs once only at start-time and end-time is ignored.

### **Valid day-list and holiday**

Beside first and second major timeframe or timespec, you are offered other chances to validate events based on day-of-week and holiday. Day-list offers which day-of-week during major timeframe/timespec, specified event is valid to be responded with associated actions. 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, event will be executed everyday during date and time range.



Holiday here means national holiday. It offers whether you want to treat event differently on the holiday. By default, holiday is selected. This means, national holiday will be ignored by event filtering process. If you unmark holiday check box, event will not be executed on 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. See par 2.5 in chapter 2 for detail about preparing holiday table.

## Triggers Shutdown

If you fill "Y", when this event occurs, zJOS then decides to enter shutdown state. This means, subsequent events processing will only execute actions that stated applicable during shutdown state.

```

Action Help
Managed Event Detail

Event verb MYJOB001 system SYSPROD1 triggers shutdown N
Date start 2004 01 01 end 2010 12 31 (YYYY MM DD)
Time start - 08 00 00 end - 17 30 00 (hh mm ss)
Interval --- (hh mm ss) for TOD event only

Event type MSG option valid day
6 1. (MSG) Message 1. Let as is Z Sunday
2. (CMD) Command 2. Suppress msg - Monday
3. (TOD) Time-of-day 3. Suppress log - Tuesday
4. (EOT) End-of-task 4. Suppress all - Wednesday
5. (EOS) End-of-step Z Thursday
6. (EOJ) End-of-job - Friday
- Saturday
Z Holiday

Command ===>
F1=Help F3=Save/End F7=Up F8=Down F12=Cancel
    
```

Figure 3.6: Example EOJ event detail for MYJOB001

## Examples

Figure 3.6 show you example of the detail of EOJ event. EOJ event of job MYJOB001 in system SYSPROD1 is monitored on every Monday and Thursday at 08:00:00 to 17:30:00 from January 1<sup>st</sup>, 2004 until December 31<sup>st</sup>, 2010. Holiday is ignored, means, monitoring is still performed although selected day is holiday. This event not trigger Sekar to enter to shutdown state.



Lets say today is Monday, March 5, 2007. If job MYJOB001 is running, when it ends at 08:10:00, generated EOJ event is captured and associated actions are then executed. But, when it ends at 07:59:59, is ignored, no action is taken.

Tomorrow is Tuesday, March 6, 2007, although within valid date range, is not valid day, so such event is not monitored. Next valid day is Thursday, hence EOJ event for job MYJOB001 is back monitored on Thursday, March 8, 2007.

If system SYSPROD1 is not local system, event monitoring is not performed unless zJOS-XDI agent is active on that system. If agent is active, monitoring is performed in the way as local system, except, event information is not processed locally, rather, is sent to zJOS/Sekar server. Timeframe up to final evaluation is performed by Sekar server, and then actions execution is triggered. Each action will be executed on each specified system. If targeted system on action entry is not local system, action entry is then sent to targeted system. Agent must targeted system must already active.

If you want, you can update each field on that panel screen, and then press enter key. Update is aborted if you press F12 key. All fields are updatable, including event verb and type. Hence, when you want delete an event entry and at the same time you want to add a new entry, you can do this way. Select (type S) the entry you want to delete, and then update its all fields with new entry information. When you press enter key, old entry is then replaced with new entry.

```

Action Help
Managed Event Detail

Event verb  MIDNIGHT          system  SYSPROD1 triggers shutdown  N
Date start  2007  01  01      end    2010  12  31  (YYYY MM DD)
Time start  - 23  55  00      end    - 24  00  00  (hh mm ss)
Interval --- 00  01  00      (hh mm ss) for TOD event only

Event type          MSG option          valid day
3  1. (MSG) Message      - 1. Let as is      / Sunday
  2. (CMD) Command      - 2. Suppress msg    / Monday
  3. (TOD) Time-of-day   3. Suppress log    / Tuesday
  4. (EOT) End-of-task   4. Suppress all    / Wednesday
  5. (EOS) End-of-step   / Thursday
  6. (EOJ) End-of-job    / Friday
                          / Saturday
                          - Holiday

Command ===>
F1=Help      F3=Save/End  F7=Up        F8=Down      F12=Cancel
```

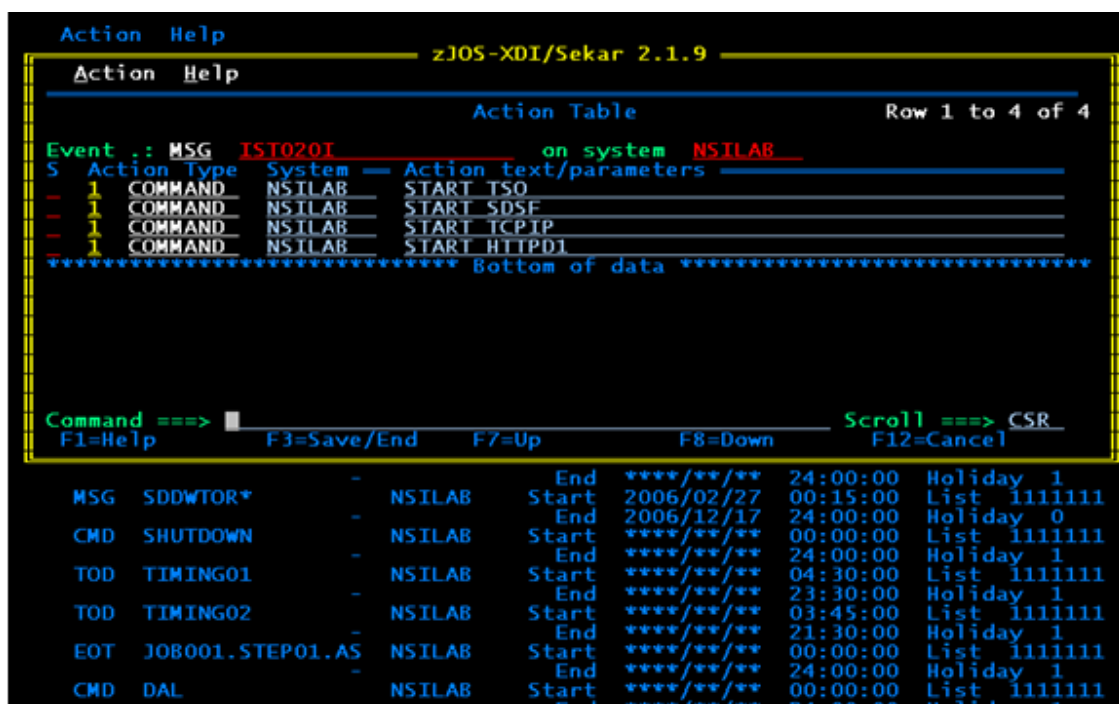
Figure 3.7: Example TOD event detail for MIDNIGHT

Figure 3.7 shows other example, TOD event, in this setting called MIDNIGHT. The name of MIDNIGHT is not real event name, rather, just a unique name to identify the event. Event is set to occur at 23:55:00 and reoccurs in every minute until 24:00:00, everyday except holiday, since Jan 1, 2007 until Dec 31, 2010.

Note:

TOD is very common information, so, to capture TOD event on remote system, you don't need agent to send TOD event. You can just easily use local TOD to trigger actions on remote system. Based on this perception, TOD event entry definition must be local event, so specified system name SYSPROD1 on this example, must be local system name.

## 3.3. Action Table



Event	Action Type	System	Action text/parameters
MSG IST020I	COMMAND	NSILAB	START TSO
	COMMAND	NSILAB	START SDSF
	COMMAND	NSILAB	START TCPIP
	COMMAND	NSILAB	START HTTPD1

\*\*\*\*\* Bottom of data \*\*\*\*\*

Command ==>  F1=Help F3=Save/End F7=Up F8=Down Scroll ==> CSR F12=Cancel

MSG	SDDWTOR*	NSILAB	End	****/**/****	24:00:00	Holiday	1
			Start	2006/02/27	00:15:00	List	1111111
			End	2006/12/17	24:00:00	Holiday	0
CMD	SHUTDOWN	NSILAB	Start	****/**/****	00:00:00	List	1111111
			End	****/**/****	24:00:00	Holiday	1
TOD	TIMING01	NSILAB	Start	****/**/****	04:30:00	List	1111111
			End	****/**/****	23:30:00	Holiday	1
TOD	TIMING02	NSILAB	Start	****/**/****	03:45:00	List	1111111
			End	****/**/****	21:30:00	Holiday	1
EOT	JOB001.STEP01.AS	NSILAB	Start	****/**/****	00:00:00	List	1111111
			End	****/**/****	24:00:00	Holiday	1
CMD	DAL	NSILAB	Start	****/**/****	00:00:00	List	1111111
			End	****/**/****	24:00:00	Holiday	1

Figure 3.10: Action table

Action table is a list of one or more actions associated to an event entry. When an event occurs, all actions in associated action table are performed. Although issuance of actions is done serially, there is no guarantee execution is performed serially by MVS. Hence, strongly recommended not to put action definitions which have inter-dependencies each other in an action table.

Action table can be reached from either event table (figure 3.2) or event detail panel. If you are in event table panel and want to reach action table of selected event directly, type T in front of selected event and then hit enter. Action table is then popped up in a window. Figure 3.10 shows action table window popped up from event detail panel when option 1 of action bar menu is selected.



Action table consist of 3 major columns, action type, system and action text or parameter. To avoid losing track, short information of event is displayed on heading region of this panel. All actions information on this panel are protected. To make changes, you have to follow its update role using prefix command.

## **Managing Action Table**

First column in action table panel is prefix command column, is not table column, rather, column in which you can issue commands to manage the action table, called prefix command. Valid prefix commands are:

1. S – Select an action entry and display in detail as shown in 3.11.
2. D – Delete an action entry.
3. A – Add or insert a new action entry.

You can not directly overwrite the content of action table on this panel. Instead, you must select a specific entry by typing S and Sekar will obtain the detail action in next window.

This panel is scrollable, so, you can scroll it up by hitting F7 and down by hitting F8 as normally applied in most of ISPF applications. To exit from action table panel session and save all changes you have done, hit F3. To abort all changes and restore whole table, hit F12.

## **Action type column**

This column shows both action type code and description, and must be either one of 3 valid action types. As shown in figure 3.10, action type is REPLY which is internally coded as 2.

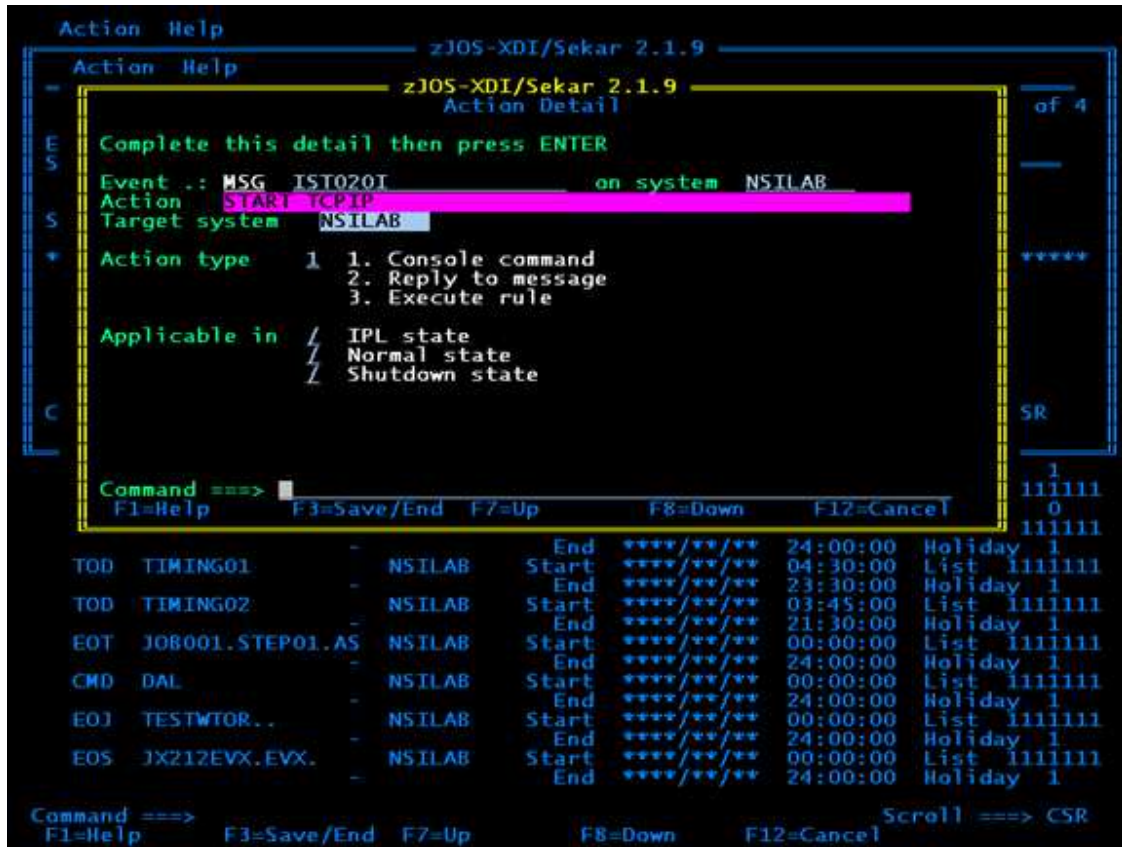
## **System column**

This column shows name of system on which this particular action to be done. For local system, name must be system name as defined in IEASYSxx member of system parameter library, or blank. For remote system, name must be network host name as defined in HOSTNAME parameter in TCP/IP profile or node name specified for VMCF when you start EZAZSSI procedure. System name here can either the same as or different from system name specified in event entry. Such facility gives you a chance to capture an event on certain system in the network and perform actions on any system in the same network.

## **Action text column**

This column shows action text or parameter. For command type action, action text is complete command text. For reply type action, action text is reply text. For rule type action, action text is member name of designated rule in rule library.

## 3.4. Action Entry



zJOS-XDI/Sekar 2.1.9

Action Detail

Complete this detail then press ENTER

Event :: MSG IST020I on system NSILAB

Action START TCP/IP

Target system NSILAB

Action type 1 1. Console command  
2. Reply to message  
3. Execute rule

Applicable in / IPL state  
/ Normal state  
/ Shutdown state

Command ==>

F1=Help F3=Save/End F7=Up F8=Down F12=Cancel

Event	Action	Target system	Start	End	List
TOD TIMING01	-	NSILAB	Start	End	111111
TOD TIMING02	-	NSILAB	Start	End	111111
EOT JOB001.STEP01.AS	-	NSILAB	Start	End	111111
CMD DAL	-	NSILAB	Start	End	111111
EOT TESTWTOR..	-	NSILAB	Start	End	111111
EOS JX212EVX.EVX.	-	NSILAB	Start	End	111111

Command ==> Scroll ==> CSR

F1=Help F3=Save/End F7=Up F8=Down F12=Cancel

Figure 3.11: Action detail

As described in previous paragraph, action entry consists of 4 major fields, action type, targeted system and action text/parameter. Figure 3.11 shows action entry displayed on action detail panel. To reach this panel, type S in front of selected entry on action table panel (figure 3.10), then hit enter key.

The heading line shows associated event definition to help you keep on track. Detail action consists of action text, targeted system name, action type and state in which action is applicable.

### Action text/parameter

As describe on the above par, the mean of action text or parameter depends on the type of action. Note that for any type of action, Sekar will not validate the text prior to execution. Hence, you have to care that the text must be valid as what you expect.



For command type action, the text is complete command text. Sekar will execute the text as it is, hence you have to make sure the text you specified must be a valid command text. Otherwise, command will be rejected by the system.

For reply type action, the text is reply text only. Sekar will build complete reply command text which consists of "REPLY" verb and reply id and followed with reply text exactly as what you specified. Hence, the text must be valid reply text against WTOR you expect to receive. Otherwise, reply will be rejected by the program issuing WTOR or it could cause unexpected result.

For rule type action, the text is a name of member of designated rule in your rule library. Sekar will not check whether the member exists. Instead, it just issues START XDIRULE,M=argument with specified action text as argument. If your specified text does not match any member, the rule executor terminated with IKJ56500I message explaining that the command not found.

### **Target system**

This field is a name of the system on which action is targeted to be executed, as explained on the above par. In respond to event that occurs in a certain system, action can be executed on any connected z/OS system within zJOS integrated network configuration. For multiple actions, each action can be addressed either to the same or different system.

At the moment Sekar only supports z/OS and OS/390 system. XDI development team is preparing and designing zJOS agents for some other platform, includes Linux, Unix and Windows. Next version of zJOS, Sekar will support some other platforms.

### **Action type**

As mentioned above, at the moment, Sekar only provides 3 action types. Which are command, reply and rule. Please refer to action text for detail explanation for all action types.

To minimize mistake, action type field appears as multiple choice menu. To fill it up, you have to code selection number which is actually internal code of action type. Currently, valid input is 1 for console command type, 2 for reply type, or 2 for rule type.

### **State applicability**

To establish perfect automation, Sekar provides 3 type of state applicability for each action. They are: IPL, normal and shutdown states. You must specify whether action is applicable during a certain state, two of them or all states. Only



actions that match with the state during which the event is occurred are eligible for execution.

## ***IPL state***

Is a short time range that follows IPL process. IPL state is ranged since NIP (nucleus initialization process) completion up to standard system startup complete. This state will only recognized by Sekar when zJOS is started exactly following the NIP (registered in COMMNDxx). During this state, Sekar executes all specified commands in SSC table. When defined events occur, Sekar will only execute associated actions which stated applicable in IPL state.

Sekar automatically change the state to normal state when system startup completes. Normally, completion of system startup is indicated by readiness of all system access facilities, such as, JES for batch job, VTAM and TCPIP for online interface in SNA and TCP/IP network environment, and OMVS for USS processes.

If you start or restart zJOS not follow NIP, for example you issue START XDI manually, Sekar will not experience IPL state, instead, entering normal state directly. If so, SSC table will not be executed.

## ***Normal state***

Is a long time range since completion of system startup until a certain event that you defined to trigger shutdown state is occurred. During this state, when defined event occurs, Sekar will only execute associated actions which stated applicable in normal state.

## ***Shutdown state***

When an event that you defined to trigger shutdown state is occurred, Sekar then change the state to shutdown state. When defined events occur, Sekar will only execute associated actions which stated applicable in shutdown state. Some EMS internal (subsystem) functions are disabled immediately as well.

Once state is changed to shutdown state, it will never change again to other state unless you recycle zJOS address space. Shutdown state is design to isolate some certain internal processing and establish a specific situation which supports shutdown procedure. Hence all actions you define to respond event that triggers shutdown state must perform any relevant processing to shutdown steps.



## **Example**

In figure 3.11, command action of “START TCPIP” is defined to be executed on system MFOC to respond against message IST020I on system MFOC too. This action is applicable during IPL state only. This is a good example to understand real application of EMS. Message IST020I indicates VTAM startup is complete, which normally happen during system startup. One of actions to respond such situation is bringing TCPIP up, since TCPIP address space needs some VTAM services. Nevertheless, VTAM might be recycled for a certain case. When VTAM is brought down, TCPIP may remain although its functions are suspended until VTAM is back up. Without state classification, you have to stop TCPIP prior to recycle VTAM, because, TCPIP will be brought up again when message IST020I occurs upon completion if VTAM recycling. Such case causes constrained policy in automation project.

With state categorization, such constrained policy is eliminated. By specifying this action is applicable only during IPL state, you can recycle VTAM anytime outside IPL state without impacting TCPIP. Because, this action is ignored when IST020I message occurs in normal or shutdown state.

## **3.5. Action Variables**

Variables are provided by Sekar to give you chance to substitute action text “on the fly” just before fired. Substitution is done by Sekar based on caught event information. At the moment, there are 3 action variables supplied by Sekar; &ARG, &JOB and &UID.

### **3.5.1 &ARG Variable**

&ARG variable is an action variable to hold command arguments from a caught command event. When an expected command event is caught by Sekar, you can use its arguments in your action text by inserting &ARG variable. However, take a note that &ARG will only contain arguments outside command verb you specified in event entry. For example, if specified command verb is SHOW, then &ARG will hold anything following verb SHOW. When someone issues SHOW MY DATA, then &ARG will contain MY DATA. However, if specified verb in the event entry is SHOW MY, &ARG will contain DATA.

Figure 3.12 to 3.12e show a case that you want to define your own command called KILL. KILL is actually not a command. When you issue KILL command on the console in a standard z/OS environment, message IEE305I is then prompted as shown in figure 3.12 to alert you that KILL is an invalid command.



```

Display Filter View Print Options Help
-----
SDSF SYSLOG 5558.101 SYS1 SYS1 10/06/2011 0W 14,675 COLUMNS 52- 131
COMMAND INPUT ==> █ SCROLL ==> CSR
0290 KILL ABC
0090 IEE305I KILL COMMAND INVALID
0290 DERSCD085W Agent on BNIPROD unconnected, so can not schedule job
JREMOTE0
0290 DERSCD085W Agent on BNIPROD unconnected, so can not schedule job
JREMOTE0
0290 KILL IBMUSER
0090 IEE305I KILL COMMAND INVALID
***** BOTTOM OF DATA *****

```

Figure 3.12: KILL is an unknown command

Using Sekar, you can use word KILL as a valid command. First, you define a command verb KILL in the event entry panel as shown in figure 3.12a. The verb is KILL and the type is 2 (CMD). In this case, KILL command capturing enable all the time any date and any day.

```

Action Help zJOS-XDI/Sekar 2.1.9
-----
Managed Event Detail
Verb KILL System NSILAB triggers shutdown N
Date start end (YYYY MM DD)
Time start - 00 00 00 end - 24 00 00 (hh mm ss)
Interval --- (hh mm ss) for TOD event only

Event type MSG option Valid day
2 1. (MSG) Message 1. Let as is / Sunday
2. (CMD) Command 2. Suppress msg / Monday
3. (TOD) Time-of-day 3. Suppress log / Tuesday
4. (EOT) End-of-task 4. Suppress all / Wednesday
5. (EOS) End-of-step / Thursday
6. (EOJ) End-of-job / Friday
/ Saturday
/ Holiday

Command ==> █
F1=Help F3=Save/End F7=Up F8=Down F12=Cancel

MSG ZJOSTEST - NSILAB Start ****/**/ 24:00:00 Holiday 1
End ****/**/ 00:00:00 List 111111
MSG IEF2380 - NSILAB Start ****/**/ 24:00:00 Holiday 1
End ****/**/ 00:00:00 List 110111

```

Figure 3.12a: Registering KILL command event entry into events table

```

Action Help zJOS-XDI/Sekar 2.1.9
-----
Action Help zJOS-XDI 2.1.9
Successful!
KILL added!
Interval --- triggers shutdown N
(hh mm ss) (YYYY MM DD)
(hh mm ss) 00 (hh mm ss)
(hh mm ss) for TOD event only

Event type MSG option Valid day
2 1. (MSG) Message 1. Let as is / Sunday
2. (CMD) Command 2. Suppress msg / Monday
3. (TOD) Time-of-day 3. Suppress log / Tuesday

```

Figure 3.12b: Entry detail of KILL is added to the event table.





When you hit enter key, system prompts that KILL was added as shown in figure 3.12b. Hit enter again, then action entry detail panel is popped up. Next is defining action text 'CANCEL U=&ARG' as shown in figure 3.12c. This means, when someone issue KILL IBMUSER, command will be trapped by Sekar and 'CANCEL U=&ARG' action text is then substituted to 'CANCEL U=IBMUSER' and fired. So KILL becomes synonym of 'C U=' or 'CANCEL U=' command.

```

Action Help
-----
zJOS-XDI/Sekar 2.1.9
Action Detail
-----
Complete this detail then press ENTER
Event.: CMD KILL on system NSILAB
Action: CANCEL U=&ARG
Target system: NSILAB
Action type: 1. Console command
              2. Reply to message
              3. Execute rule
Applicable in: / IPL state
               / Normal state
               / Shutdown state
Command: ==>
F1=Help F3=Save/End F7=Up F8=Down F12=Cancel
-----
TOD TIMING01 - NSILAB Start ****/**/ 24:00:00 Holiday 1
               End ****/**/ 04:30:00 List 111111
               End ****/**/ 21:30:00 Holiday 1

```

Figure 3.12c: Registering action for KILL cmd event

Now, KILL command and its action were stored into Sekar database. However, these are not effective until you issue LOAD request for Sekar. Soonest after LOAD request was issued, KILL command then become a valid command as shown in figure 3.12d. Refer to par 4.2 in chapter 4 for LOAD request.

```

Display Filter View Print Options Help
-----
SDSF SYSLOG 5558.101 SYS1 SYS1 10/06/2011 0W 14,824 COLUMNS 52- 131
COMMAND INPUT ==> SCROLL ==> CSR
JREMOTE0
0290 KILL ABC
0090 IEE324I ABC NOT LOGGED ON
0290 KILL IBMUSER
0281 IEA989I SLIP TRAP ID=X222 MATCHED. JOBNAME=IBMUSER, ASID=003D.
0090 IEE301I IBUSER CANCEL COMMAND ACCEPTED
0281 IEA989I SLIP TRAP ID=X13E MATCHED. JOBNAME=IBMUSER, ASID=003D.
0281 IEA989I SLIP TRAP ID=X13E MATCHED. JOBNAME=IBMUSER, ASID=003D.
0290 IEA631I OPERATOR IBUSER NOW INACTIVE, SYSTEM=NSILAB, LU=SCOTCP04
0281 IEF450I IBUSER ZJOSPROC ZJOSPROC - ABEND=S222 U0000 REASON=00000000 024
0281 TIME=05.19.33
0281 $HASP395 IBUSER ENDED
0290 DERSCD702I Incoming signal is 00CLOCK request
0281 IEA989I SLIP TRAP ID=X13E MATCHED. JOBNAME=*UNAVAIL, ASID=003D.
0290 DERSCD085W Agent on BNIPROD unconnected, so can not schedule job
JREMOTE0
***** BOTTOM OF DATA *****

```

Figure 3.12d: KILL becomes a valid command



The action table, when you enter T against event entry of KILL on event table panel will appear as shown in figure 3.12e.

The screenshot shows the 'Action Table' for the 'KILL' command event. The table has three columns: 'Action Type', 'System', and 'Action text/parameters'. The first row shows '1' as the action type, 'NSILAB' as the system, and 'CANCEL U=&ARG' as the action text/parameters. The table is titled 'Event :: CMD KILL on system NSILAB' and 'Row 1 to 1 of 1'. The bottom of the data is indicated by a line of asterisks. The command line at the bottom shows 'Command ==>' followed by a cursor, and 'Scroll ==> CSR'. Function keys are listed at the bottom: F1=Help, F3=Save/End, F7=Up, F8=Down, and F12=Cancel.

Action Type	System	Action text/parameters
1	NSILAB	CANCEL U=&ARG

Figure 3.12e: &ARG variable shown in action table entry for KILL cmd event

Now you understand that &ARG variable effectiveness for command event.

## 3.5.2 &UID and &JOB Variables

Unlike &ARG variable which is designed for command events, &UID and &JOB variables are for either command or message events. &UID variable will be substituted by userid of a TSU, a JOB or an STC from which a trapped message event was sent, or by which a trapped command was invoked. &JOB variable will be substituted by jobname of a JOB, a TSU or an STC from which a trapped message event was sent, or by which a trapped command was invoked. For a TSU, jobname and userid are the same, so &UID and &JOB will result the same.

## 3.6. Applying Sekar Parameters

All Sekar parameters we have discussed in previous paragraphs, automatically loaded at the first time Sekar is activated, e.g. when zJOS address space is started up. When you update them while Sekar is active, newly updated parameters will not automatically effective until you reload them or recycle zJOS address space. Refer to chapter 4 paragraph 4.2 for further explanation about how to reload Sekar parameters.



## Chapter 4 Controlling Sekar

As an automation system or event management system (EMS) solution, Sekar has complete capabilities to control the system. Most of interaction activities which usually performed by operators can be handled by Sekar. Combined with automatic workload scheduling (e.g. Puspa), 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 intervention. Though, it does not mean your system can totally operate itself. It will still need human intervention, at least to control Sekar.

Controlling Sekar is very 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 itself.

### 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=000000
JES I/F:  Up=Y PIT=Y Conn=Y Ird=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=0 O=EVXMS M=EVXMS Pos=EVALUATE-JMR EnQ=FREE
```

On TSO/ISPF XDI session, zJOS status information appears as in figure 2.2. 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:

Component-	Stat-	-Agent--	Tbl	Works	-Usage--	#dayX
<b>Sekar (EMS)</b>	<b>UP</b>	<b>ACT(SS)</b>	<b>IN</b>	<b>00000</b>	<b>LICENSED</b>	<b>none</b>
Puspa (SCD)	UP	READY	IN	00000	**DEMO**	..?!
AutoXfer	DOWN	INACTIVE	OUT	00000	**DEMO**	..?!
<b>Net-Server</b>	<b>DOWN</b>	<b>INACTIVE</b>	<b>N/A</b>	<b>00000</b>	<b>standard</b>	<b>none</b>

### Stat column

The above information describes whole status of zJOS-XDI products. Status of Sekar is indicated by **red** color. Stat 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. To activate Sekar, issue F XDI,AUTO SSI as described in chapter 2, paragraph 2.4. You can also issue F XDI,AUTO START, but Sekar will active supported by MCS agent, which is actually legacy from XDI version 2.1.1. Strongly recommended not to use MCS, unless, you are assisted by XDI support personnel

### Agent column

Agent column shows current internal agent activity. For Sekar, there are 2 type internal agents, SSI or MCS. MCS is legacy from XDI version 2.1.1 and should not active unless SSI agent got serious problem. MCS internal agent currently is

used by XDI support personnel for debugging purpose only. Normally internal agent shown as ACT(SSl) if its status is UP, which means Sekar is up and current activities is supported by SSI internal agent. If you find Sekar status UP and internal agent INACTIVE, please call XDI support personnel immediately.

### **Tbl column**

This column shows status of parameters table, which is indicated as IN or OUT. IN indicates parameters table is already loaded, and OUT indicates that table is not loaded yet or unsuccessfully loaded.

For Sekar, if status is UP, table must be IN. If you find Sekar status UP with table OUT, issue **.AUTO RELOAD** and recheck the status. If table remains OUT, you must recheck Sekar 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 Sekar. Issue **.AUTO STOP**, then issue **F XDI ,AUTO SSI**. Please call XDI support personnel immediately if you find the problem persist.

### **Work column**

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



## #dayX column

#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=0 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. 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 XDI address space was started. Each zJOS 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 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 shows number NACCB received by server. S=nnnnnnnn shows number of NACCB sent to agents. Q=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 zJOS 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

## **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. Reloading Sekar Parameters**

Sekar parameters, which are an event table and a number of action tables, each associated with one event entry, physically are placed into a single sequential file called XDIEMSxx, which is a member of XDI parameters library. The two-digit suffix 'xx' can be any valid numeric combination. You can have more than one XDIEMSxx, for example XDIEMS00, XDIEMS25 etc., but, only one XDIEMSxx effective at a time.

As an ordinary partitioned dataset or library (PDSE) member, you can copy or rename XDIEMSxx in XDI parameters library either interactively using ISPF/PDF tools or in batch using DFSMS tools. However, you must not straightly edit it. There are some binary and packed decimal fields which

When you update Sekar parameters will automatically loaded at the first time Sekar is activated, e.g. when XDI address space is started up.



At current product level (2.1.9), applying parameters can only be done in batch. If Sekar is already up and you make updates, Sekar will not automatically load them. Either, you can not load each particular entry. To make your update effective, you have to issue the following console command to ask Sekar to reload whole event and all associated action tables.

**.AUTO RELOAD**

or

**F XDI,AUTO RELOAD**

Then Sekar will reload all parameter tables pointed with current or most recent used suffix (default is 00).

```
DEREVX705I DERU      is granted to issue .AUTO RELOAD
DERCMD221I Scheduled commands are being purged.
DERCMD222I All timer services have been brought down.
DEREVX090I Shutting down all TMR components...
DERCMD217I Scheduled commands are being initialized.
DEREVA217I Scheduled commands are being initialized.
DEREVA333I Automation table XDIEMS00 is being loaded.
DERSIP090I Shutting down all EVX components...
DEREVA303I Loading source 00000037 event recs in progress..
DEREVA303I Loading source 00000117 action recs in progress.
DEREVA220I Scheduled commands initialization complete.
DEREVA334I Automation table XDIEMS00 is now loaded.
DEREVA273I Loading holiday calendar...
DEREVX090I Shutting down all TMR components...
DEREVA274I Holiday calendar initialization complete.
DERSVR544I 000 NETCCBs were added for zJOS agents
DERSIP074I EVP major subtask TCB=008CD270 was terminated normally.
DERTOD303I Day of week is 00000004 (Thu) based on LT-zone.
DERSIP074I EVX major subtask TCB=008D0AA0 was terminated normally.
DERTOD706I 2011/10/31 is the final working day this month.
DEREVX090I Shutting down all TMR components...
DERSIP074I EVS major subtask TCB=008CFC50 was terminated normally.
DERTOD706I 2011/10/07 is the final working day this week.
DERSIP074I EVC major subtask TCB=008D08F0 was terminated normally.
DERSIP074I EVT major subtask TCB=008CF9A8 was terminated normally.
DERSIP074I EVZ major subtask TCB=008CF700 was terminated normally.
DERSIP074I EVM major subtask TCB=008CD518 was terminated normally.
DERSCD702I Incoming signal is SCHEDULE reqst
DERSCD702I Incoming signal is 00CLOCK request
DERSIP074I EVR major subtask TCB=008CFE00 was terminated normally.
DERCMD201I zJOS is ready to accept command.
DERSIP089I Starting up all EVX components...
DERSIP076I EVX major subtask is being started.
DERSIP076I EVC major subtask is being started.
DERSIP076I EVM major subtask is being started.
DERSIP076I EVP major subtask is being started.
DERSIP076I EVR major subtask is being started.
DERSIP076I EVS major subtask is being started.
DERSIP076I EVT major subtask is being started.
DERSIP076I EVZ major subtask is being started.
DERSCD702I Incoming signal is SCHEDULE reqst
DERSCD702I Incoming signal is SCHEDULE reqst
DERSCD702I Incoming signal is SCHEDULE reqst
DERSCD702I Incoming signal is SCHEDULE reqst
DEREVX089I Starting up all TMR components...
```

Figure 4.1: RELOAD command and its response



If you want to use other suffix, use the following command:

```
.AUTO RELOAD,TAB=xx
```

or

```
F XDI,AUTO RELOAD,TAB=xx
```

Where xx is 2-digit suffix you are currently using it to manage newly updated parameters. Once loading is completed, all outstanding actions are purged, and all newly updated parameters will immediately effective and xx become your current suffix. Next time when you issue RELOAD without TAB= keyword, Sekar assumes TAB=xx. Please take a note that if reloading process is failed, you have to reissue the above command.

## Chapter 5 Integrated Automation

In previous chapter (chapter 3), we have clearly discussed how to manage Sekar parameters in zJOS-XDI parameters library. In either event or action tables, there is a parameter named system name. This name reflects to system or host name on which event is occurs, to which each action is targeted. All describe that EMS can be established on multiple interconnected z/OS machines. Yes, Sekar gives you a chance to establish integrated automation among interconnected z/OS hosts.

### 5.1. 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

#### 5.1.1 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.

## 5.1.2 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.

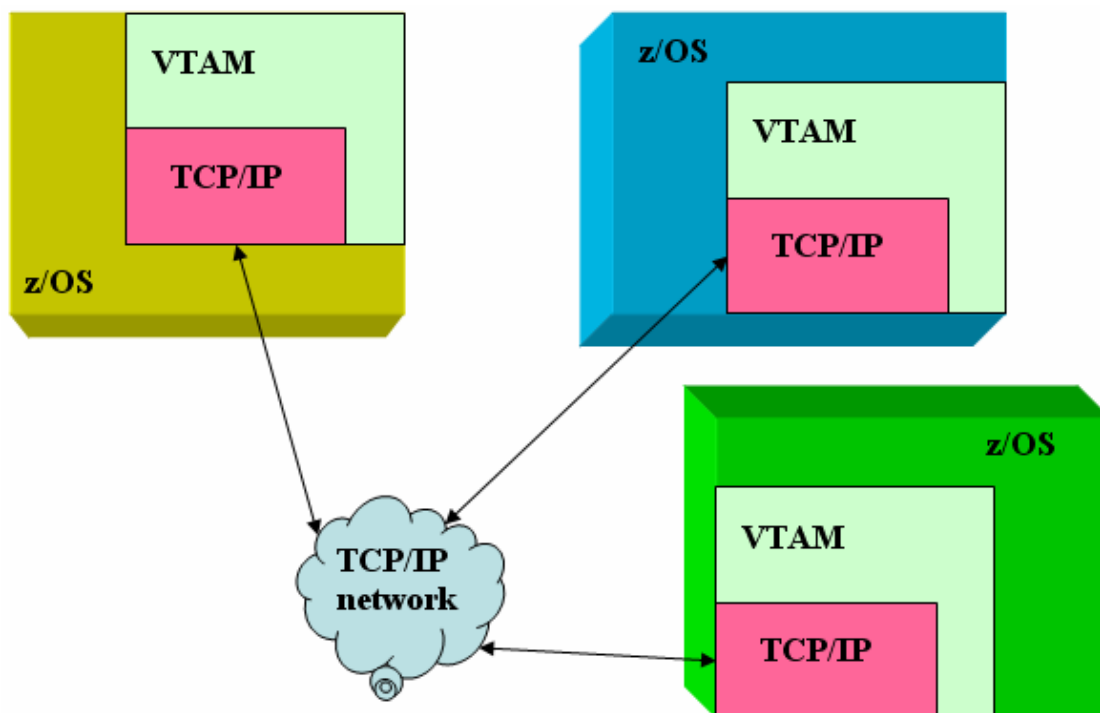


Figure 5.1: Networked-z/OS



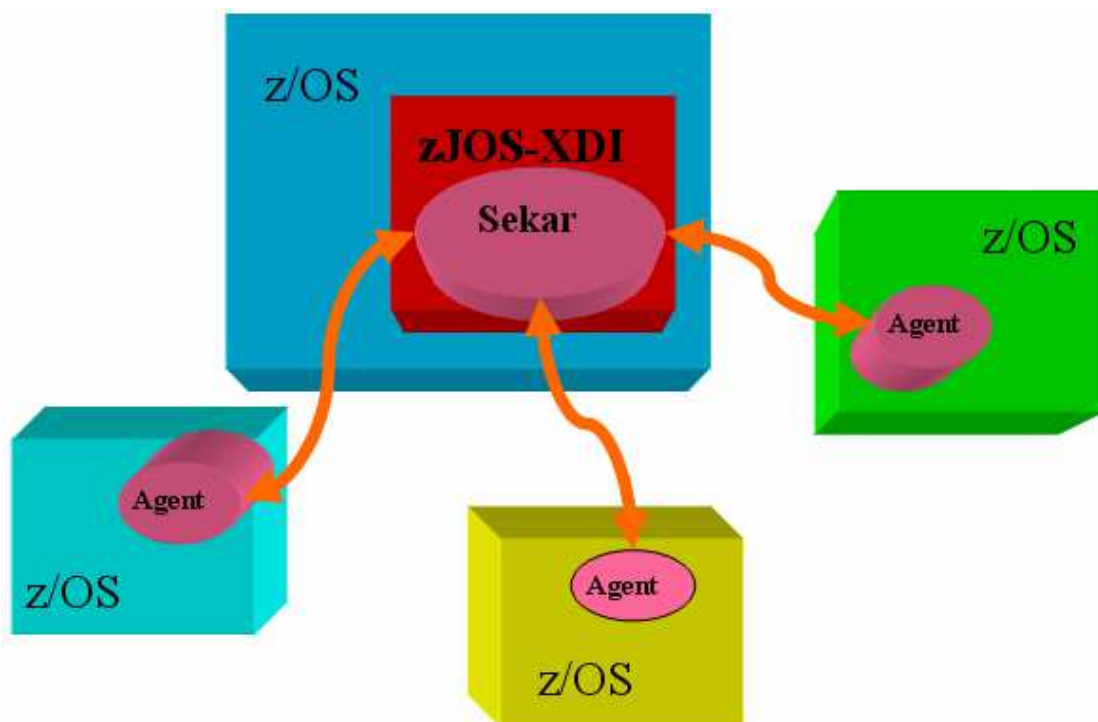
## 5.2. Sekar for Integrated zJOS Network

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

- Complete copy of licensed zJOS/Sekar package on one z/OS system host which is assigned as EMS server.
- Copy of licensed zJOS agent on each z/OS system host which is assigned as EMS 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 Sekar and zJOS socket server active with well-setup parameters on EMS server machine. Make sure Sekar parameters involve all EMS client machines.
- zJOS agent address space (XDA) which represent Sekar agent up and active on each EMS client machine.



*Figure 5.2: Sekar for integrated zJOS network*



## 5.2.1 Preparing zJOS Server

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

To implement an integrated EMS, Sekar 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.
2. Make sure zJOS agent is ready on each connected z/OS host which designated to be a member of integrated EMS.
3. Bring up zJOS server. By default, server is initially down (figure 2.1). You can bring it up manually each time zJOS address space is started, or automate it later. The command is:

**.SVR START**

or

**F XDI,SVR START**

or

**START** request in control panel as shown in figure 5.3

Action Help		zJOS-XDI Control Panel						
		Row 903 to 925 of 925						
Command	Product	State	Table	Suf	Works	Usage	Day	
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LCNSD/YR	0816	
	Puspa (SCD)	<UP> ACTIVE	loaded	02	000001	LCNSD/YR	0816	
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	0816	
<b>start</b>	Net Server	DOWN INACTIVE	unloaded	**	000000	standard	none	
Date	Time	Log						
11.279	06:37:23	Statistics:						
11.279	06:37:23	Config: SSN=zJOS Load=LPA COM=1DA6D040 WSA=00C69F90						
11.279	06:37:23	Tasks: Maj=009 EVX=01 Net=00 SCD=001 Abn=000 Prm=011						

Figure 5.3: Activating zJOS server

Action Help		zJOS-XDI Control Panel							Row 926 to 926 of 926
Command	Product	State	Table	Suf	Works	Usage	Day		
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LCNSD/YR	0816		
	Puspa (SCD)	<UP> ACTIVE	loaded	02	000001	LCNSD/YR	0816		
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	0816		
*START	Net Server	DOWN INACTIVE	unloaded	**	000000	standard	none		
Date	Time	Log							
11.279	06:43:45	START request to NetServer was sent to zJOS.							
***** Bottom of data *****									

Figure 5.3a: Immediate response when start request (fig 5.3) is entered.



Figure 5.3a shows immediate response when START request for zJOS server is entered. At the same time, control panel actually sends the request to zJOS address space as when it is entered via .SVR or F XDI,SVR command. In the console log, activation progress is displayed as shown in figure 5.3b.

```
DERSIP076I SVR major subtask is being started.
*DERSVR500I zJOS server initialization in progress...
DERSVR543I A NETCCB was built for zJOS agent on system BNIPROD
DERSVR543I A NETCCB was built for zJOS agent on system TLC2007
DERSVR544I 002 NETCCBs were built for zJOS agents
DERSVR515I zJOS#XDI pid=**none** was initialized for 100 sockets.
DERSVR522I MAINTask successfully done GETCLIENTID
DERSVR509I Socket 000 and max=099 was obtained for task
DERSVR522I MAINTask successfully done SOCKET
DERSVR510I Server host is NSILAB IP=100.100.0.11
DERSVR511I Socket 000 bound to port 07777, IP 100.100.0.11
DERSVR501I zJOS server initialization complete.
DERSVR512I Server is listening on port 07777
```

Figure 5.3b: zJOS server activation progress as displayed in console log.

The log 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. 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 PORT=nnnn parameter, then recycle zJOS address space.

Message DERSVR543I shows network connection control block (NETCCB) was built to accommodate agent connection request. Each NETCCB represents one agent connection.

Finally, when zJOS server activation has been done, you can see its status in the control panel as shown in figure 5.4. Just hit enter key awhile after start request was entered to obtain this information. Status information can also be displayed on the system console by issuing .STATUS command as shown in figure 5.4a..

Action Help		zJOS-XDI Control Panel						Row 926 to 949 of 949	
Command	Product	State	Table	Suf	Works	Usage	Day		
	Sekar (EMS)	<UP> ACT(SS1)	loaded	00	000000	LCNSD/YR	0816		
	Puspa (SCD)	<UP> ACTIVE	loaded	02	000001	LCNSD/YR	0816		
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	0816		
	Net Server	<UP> ACTIVE	unloaded	↕	000000	standard	none		
Date	Time	Log							
11.279	06:43:45	START request to NetServer was sent to zJOS.							
11.279	06:55:28	Statistics:							
11.279	06:55:28	Config: SSN=zJOS Load=LPA COM=1DA6D040 WSA=00C69F90							
11.279	06:55:28	Tasks: Maj=010 EVX=01 Net=00 SCD=001 Abn=000 Prm=011							

Figure 5.4: Status information in control panel when zJOS server active



```
.STATUS
DERCMD068I zJOS XDI status: 732
Component- Stat- -Agent-- Tbl Works -Usage-- #dayX
Sekar (EMS) <UP> ACT(SSl) IN 00000 LCNSD/YR 0816
Puspa (SCD) <UP> ACTIVE IN 00001 LCNSD/YR 0816
AutoXfer DOWN INACTIVE OUT 00000 LCNSD/YR 0816
Net-Server <UP> ACTIVE N/A 00000 standard none
zJOS XDI statistics:
Config: SSN=zJOS Load=LPA COM=1DA6D040 WSA=00C69F90
Subtasks: Major=010 EVX=001 SVR=000 SCD=001 Abn=000
Network agents: total=0002 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=N,22=N)
Queues: ARQs=00000 SQBs=00000 EOTs=00001 RMG=00000
State: NORMAL Parm: SYS=00 EMS=00 SCD=02 DEST=00
SCD: Lib=0 O=EVX M=EVX Pos=SCHEDULE-SCT EnQ=FREE
DERCMD201I zJOS is ready to accept command.
```

Figure 5.4a: Status information obtained by .STATUS command when zJOS server active

Once zJOS server active, you can see which agents are currently connected by displaying a list of generated NETCCBs. .Issue the following command:

**.LIST NETCCB**

or

**F XDI,LIST NETCCB**

or

**LIST** request in control panel as shown in figure 5.5.

Action	Help	zJOS-XDI Control Panel						Row 950 to 972 of 972
Command	Product	State	Table	Suf	Works	Usage	Day	
	Sekar (EMS)	<UP> ACT(SSl)	loaded	00	000000	LCNSD/YR	0816	
	Puspa (SCD)	<UP> ACTIVE	loaded	02	000001	LCNSD/YR	0816	
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	0816	
<b>list</b>	Net Server	<UP> ACTIVE	unloaded	**	000000	standard	none	

Figure 5.5: Entering LIST request to zJOS server

Action	Help	zJOS-XDI Control Panel						Row 973 to 974 of 974
Command	Product	State	Table	Suf	Works	Usage	Day	
	Sekar (EMS)	<UP> ACT(SSl)	loaded	00	000000	LCNSD/YR	0816	
	Puspa (SCD)	<UP> ACTIVE	loaded	02	000001	LCNSD/YR	0816	
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	0816	
<b>*LIST</b>	Net Server	<UP> ACTIVE	unloaded	**	000000	standard	none	
Date	Time	Log						
11.279	08:33:22	Agent 02361080 on BNIPROD conn=N EMS=N SCD=N type=E.						
11.279	08:33:22	Agent 020FCD50 on TLC2007 conn=N EMS=N SCD=N type=E.						
***** Bottom of data *****								

Figure 5.5a: List of NETCCB in respond to LIST request to zJOS server





Figure 5.5a shows a list of agent statuses represented by NETCCBs in respond to LIST request entered in control panel as shown in figure 5.5. Each line of the list show agent-ID in 8-digit hexadecimal, name of system on which agent runs, connection status (conn=Y if connected), availability of event tables (EMS=Y if available), availability of scheduler table (SCD=Y if available) and encoding type of agent's host (type=E for EBCDIC or type=A for ASCII).

The same information can also be gathered in system console log by issuing .LIST NETCCB command as shown in figure 55b. Each agent information is displayed in message DERCMD094I that contain exactly the same as a line prompted in control panel.

```
.LIST NETCCB
DERCMD094I Agent 02361080 on BNIPROD conn=N EMS=N SCD=N type=E.
DERCMD094I Agent 020FCD50 on TLC2007 conn=N EMS=N SCD=N type=E.
DERCMD201I zJOS is ready to accept command.
```

Figure 5.5c: List of NETCCB in respond to .LIST NETCCB command

## 5.2.2 Preparing zJOS Agent for z/OS

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

In an integrated EMS, Sekar runs only in one z/OS host, which is designated as EMS server. Other hosts are called as EMS member or client. Each EMS member needs zJOS agent which runs as Sekar partner. To have agent ready on EMS 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 EMS server, all products including agent are installed. Though, you don't need agent on EMS server. You rather, need it for EMS member. To install it in EMS member, you can easily put its copy onto EMS member as follow:

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





## Customize XDA procedure

zJOS agent for z/OS runs as XDA address space on EMS 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=12,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 EMS server host IP address. This is the most important point of which you have to customize. This keyword must be exactly EMS server host IP address.

PORT keyword specifies EMS 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 authorized library. Although most of zJOS-XDI modules reentrant and even refreshable, you may not place them in LPA nor in LNKST 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 program modules are placed and maintained together with all zJOS-XDI program modules in zJOS load library, the best configuration is when STEPLIB address to shared zJOS-XDI load library with zJOS address space in EMS server. Otherwise, you must only maintain zJOS load library in EMS server, and you have to clone it to all EMS member.

JCLLIB DD card is library or concatenated libraries of scheduled jobs, which is used only by Puspa for automatic jobs scheduling. Although you do not use Puspa in your environment, at least one dummy library must be specified to avoid JCL error problem.

## 5.3. Sekar Agent for z/OS

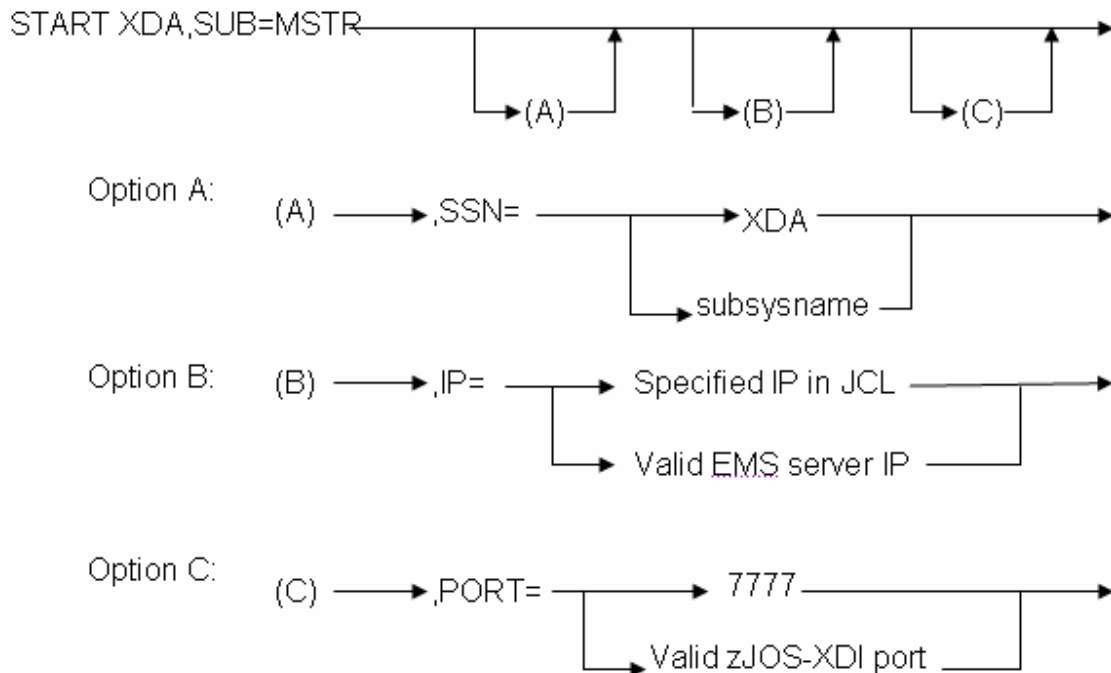
Sekar agent for z/OS is actually zJOS Agent for z/OS, which supports both Sekar (EMS) and Puspa (automatic scheduling). Once agent is well setup and up in a z/OS host, this host is then eligible to EMS server as an EMS member, as well as eligible to scheduling server as scheduler member. Refers to paragraph 5.2.2 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 socket client. System events 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 on agent address space (XDA) to be transferred to Sekar in EMS server machine. Although runs as subsystem, event listener is initialized on agent address space.

Socket client runs in agent address space to interact locally with system event listener and remotely with Sekar on EMS server. When event information is posted by event listener via ECB, socket client forward it to Sekar in EMS server via socket. Besides, socket client also executes instruction received from Sekar.

## 5.3.1 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 z/OS MVS subsystem. Default subsystem name is XDA. Use this keyword if you prefer to use different name. Valid subsystem name must be 1 to 4 alphanumeric.

**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 5.2.2.

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



```
S_XDA,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.
DERXDA489I Job status listener is being shutted down.
DERXDA490I Job status listener shutdown complete.
```

Figure 5.6: zJOS agent startup on DYAH2003

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.

Figure 5.6 shows agent startup messages when started on DYAH2003 host, where Sekar 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 agent. In common, subtask name is zJOS#nnn, where nnn is sequent number based on its occurrence.

As shown in figure 5.7, when login request accepted, server then change subtask name zJOS#001 to DYAH2003 which represent host name on which agent is running, and assign agent ID 022E9730 as shown in figure 5.6.

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.



```
DESRVR514I Contact on socket 099 accepted from port 01027 IP 100.99.125.2
DESRVR516I Socket 099 is being given to worker task.
DESRVR502I zJOS Worker zJOS#001 initialization in progress...
DESRVR517I Socket 099 is taken by zJOS#001 as 001.
DESRVR529I Worker zJOS#001 is responding to up.
DESRVR530I zJOS#001 at TCB=008D0E88 is granted to take over job.
DESRVR513I zJOS#001 has confirmed for nonblocking I/O.
DESRVR503I zJOS worker zJOS#001 initialization complete.
DESRVR518I Socket 099 is disassociated from maintask.
DESRVR545I zJOS#001 is joined to agent on DYAH2003 EBCDIC
DESRVR548I DYAH2003 received NMCCB req=ACQR obj=EMS from agent 022E9730 on DYAH2003
DESRVR551I 0002 EVB entries are sent to DYAH2003
DESRVR548I DYAH2003 received NMCCB req=ACQR obj=SCD from agent 022E9730 on DYAH2003
DESRVR549W DYAH2003 got error NMCCB from DYAH2003 reason: table not loaded yet
DESRVR548I DYAH2003 received NMCCB req=SAVE obj=EMS from agent 022E9730 on DYAH2003
DESRVR548I DYAH2003 received NMCCB req=ERRD obj=SCD from agent 022E9730 on DYAH2003
```

Figure 5.7: zJOS server accepting agent connection request from DYAH2003

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 XDA address space 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**

## 5.3.2 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**

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.

To stop agent interaction activities, issue the following command:

**-DISCONNECT**

This will cause socket client subtask logoff from server and terminate connection. Agent remains up. In case there is a networking problem, for example if TCP/IP stack 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 is for emergency only. It just 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 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.

### 5.3.3 Controlling zJOS Agent

#### Display current status information

To get agent status information, issue the following command:

**-STATUS**

Then agent current status is displayed as shown in figure 5.8. 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.

```
-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
```

*Figure 5.8: Example of agent status information*



## Display list of parameters received from server

There are 2 types of parameters which received from zJOS server, EMS and scheduler parameters. Each EMS parameter is represented by event parameter control block (EVB). Issue the following command to display all EVBs:

```
-LIST EMS or -LIST EVB
```

XDA then display response as shown in figure 5.9.

```
-LIST EVB
DERSPY494I EVB table of content:
0001 Event=MSG Key(08)=TESTONLY
0002 Event=CMD Key(05)=JAJAL
0003 Event=CMD Key(04)=DSMF
```

Figure 5.9: Agent on DYAH2003 responding agent (itself) command

Each scheduling parameter is represented by end-of-task block (EOT). Issue the following command to display all EOTs:

```
-LIST SCD or -LIST EOT
```

## Requesting EMS parameters

When agent is started, connection to server normally establish automatically. Then server will 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 EMS parameter to this agent by issuing the following command:

```
-GET EMS
```

Agent then reissue request to the server.

### 5.3.4 Remote Command

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 5.10: Issuing remote command to DYAH2003 from EMS server

Figure 5.10 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 you got respond as shown in figure 5.11.

```
DERACT635I 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
DERACT635I 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    XDIEEXEC  NSW S
VLE    VLE    VLE    NSW S  RACE   RACE   RACE    NSW S
JES2   JES2   IEFPROC  NSW S  DLE    DLE    DLE    NSW S
VTAM   VTAM   ACFVTAM  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 5.11: 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.

## 5.3.5 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 command, 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.

## 5.4. The Goal of Integrated Automation

Combined remote command and remote event capturing capabilities of Sekar and agent is the main idea to establish an integrated EMS on networked-z/OS. Integrated EMS is an establishment of EMS on networked systems which appear as a single system.

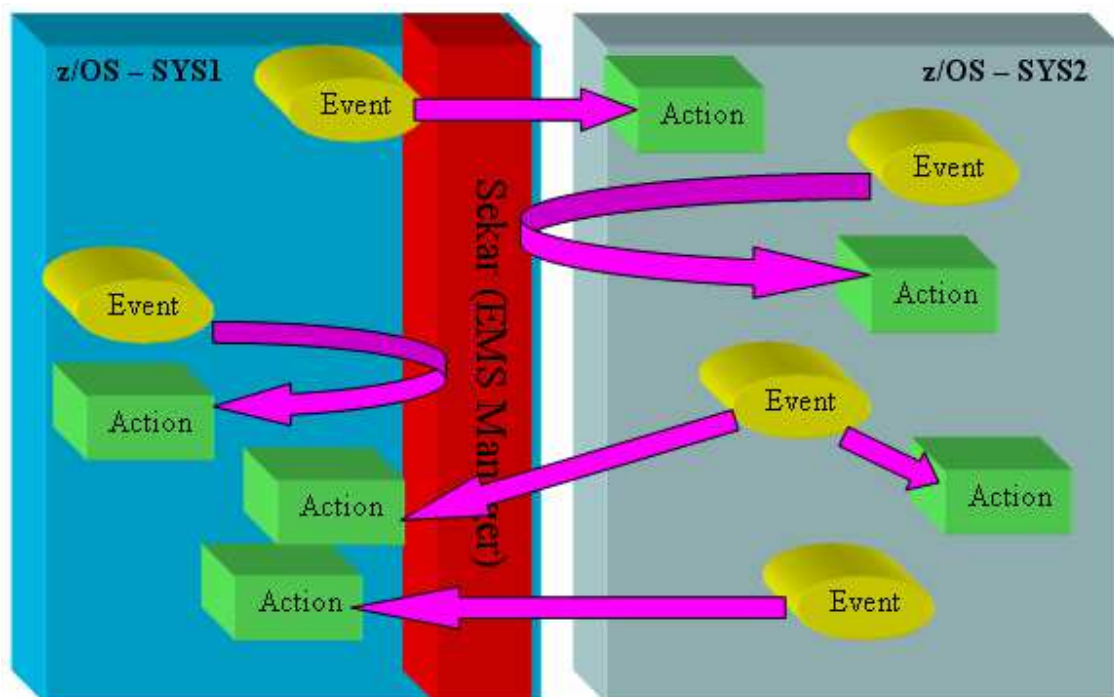


Figure 5.12: Integrated EMS on networked-z/OS



As discussed in chapter 3, you can specify system name in event entry, whereas in each associated action entry, you can specify the same or different system name. This means, event might occur in any host on the network and each associated action might also be performed in any host on the network. For example, in networked of 3 systems, SYS1, SYS2 and SYS3, event in system SYS1 can be responded with action in either system SYS1, SYS2 or SYS3. Event in system SYS2 and SYS3 can also be responded with action in either system SYS1, SYS2 or SYS3. Hence, system SYS1, SYS2 and SYS3 appear as a single system. Figure 5.12 illustrates how Sekar manages event in networked system SYS1 and SYS2.

To get clearer understanding on integrated EMS, the following example is a good illustration. Figure 5.13 shows action table for CSMF command. CSMF is not a valid command in standard z/OS environment. In this example, if CSMF is issued as a command on MFOC host, Sekar then accepts it and issue command START SMFCLEAR locally on MFOC host and remotely on DYAH2003 host.

The screenshot shows a terminal window with the title "Action Help". Below the title, it says "Action Table" and "Row 1 to 2 of 2". The main content is an "Event" definition for "CMD CSMF on system MFOC". The event is defined as "S Action Type System Action text/parameters". The table lists two actions: "1 COMMAND MFOC START SMFCLEAR" and "1 COMMAND DYAH2003 START SMFCLEAR". The table is followed by "\*\*\*\*\* Bottom of data \*\*\*\*\*". At the bottom, there are navigation instructions: "Command ==>" and "Scroll ==> CSR", with function key shortcuts: "F1=Help", "F3=Save/End", "F7=Up", "F8=Down", and "F12=Cancel".

Event	Action Type	System	Action text/parameters
1	COMMAND	MFOC	START SMFCLEAR
1	COMMAND	DYAH2003	START SMFCLEAR

Figure 5.13: Defining a local command with local and remote actions

Figure 5.14 shows another example, an action table for DSMF command type event on remote system DYAH2003. Table consist of 2 action entries, D SMF command targeted for system MFOC and D SMF command targeted for system DYAH2003.





```

Action Help

                                Action Table                                Row 1 to 2 of 2

Event .: CMD DSMF on system DYAH2003
S Action Type System Action text/parameters
* 1 COMMAND MFOC D SMF
- 1 COMMAND DYAH2003 D SMF
***** Bottom of data *****

Command ==>
F1=Help      F3=Save/End    F7=Up        F8=Down      Scroll ==> CSR
F12=Cancel
    
```

Figure 5.14: Defining a remote command with local and remote actions

Effect of the above example is shown in figure 5.15 and 5.16. When DSMF command is issued on DYAH2003 console, then appear D SMF response on DYAH2003 console as shown in figure 5.15 and on MFOC console as shown in figure 5.16.

```

DSMF
DERMGT635I Executing 005 bytes cmd> D SMF0...
IEE974I 06.28.20 SMF DATA SETS 418
      NAME                VOLSER SIZE(BLKS) %FULL STATUS
      P-DYAH2003.SMF.MAN1  NITPRM    1800    100 DUMP REQUIRED
      S-DYAH2003.SMF.MAN2  NITPRM    1800     81  ACTIVE
      S-DYAH2003.SMF.MAN3  NITPRM    1800     0  ALTERNATE
    
```

Figure 5.15: DSMF command response on DYAH2003 (EMS member)

```

DEREVX088I NACCB is posted to schedule on DYAH2003 action D SMF
IEE974I 06.25.00 SMF DATA SETS 312
      NAME                VOLSER SIZE(BLKS) %FULL STATUS
      P-MFOC.SMF.MAN1     NITPRM    1800     9  ACTIVE
      S-MFOC.SMF.MAN2     NITPRM    1800     1  DUMP REQUIRED
      S-MFOC.SMF.MAN3     NITPRM    1800     0  ALTERNATE
    
```

Figure 5.16: zJOS-XDI server responding DSMF command issued on DYAH2003

Internally, when DSMF command is issued on DYAH2003, agent then captures it and sends it to server. Sekar then looks up event table to evaluate whether DSMF is a valid command for DYAH2003. Upon completion, Sekar then execute 2 entries of associated action table. According to the action table as shown in figure 5.14, Sekar then perform remote command D SMF on system DYAH2003 and execute D SMF on local system.



## Chapter 6 Implementing Your Innovation with Sekar

With Sekar, you can implement your innovation in how to manage your system simpler. Automation is the basic result of EMS, which is main function of Sekar. Besides, Sekar encourage you to find out your best innovative idea to make up your system more friendly, which can increase its productivity. IBM have given example in JES2, that we can issue \$DSPPOOL as alternative of \$D SPOOL command which need longer time to type. We can issue 2D reply as short way of R 2,D reply command. IBM have even provided short form and acronym for most of console, JES and TSO commands as well as ISPF short-path in menu selection with x.y.z and =x option forms. These describes how important encouragement to increase the productivities.

Let assume all given shot forms, short way and shot path in the above examples are already standard feature. Sekar do more for you. You can implement your innovation using the following 2 opportunities:

- CMD type event and its associated actions
- Rule type action for any event type.

### 6.1. Innovation with Command

In Sekar EMS mechanism, command is anything invoked on console, regardless its validity according to standard system commands. As long as action table is provided (by you), any string associated with it will become a valid command. For example, DPROG initially is not a valid command. When you issue DPROG in console, system respond with message IEE305I as shown in figure 4.1.



```
DPROG
IEE305I DPROG      COMMAND INVALID
```

Figure 6.1: DPROG command and its response at initial time.

Though, you can make DPROG as a valid command by defining it in event table. In this example, DPROG is defined as a command type event as shown in figure 6.2.



```

Action  Help

Managed Event Detail

Event verb  dprog          on system  MFOC          -
Date start  2007  01  01  end  2007  12  31  (YYYY MM DD)
Time start  -  04  45  00  end  -  17  00  00  (hh mm ss)
Interval ---  -  -  -  (hh mm ss) for TOD event only

Event type  MSG option  Valid day
2 1. (MSG) Message      1. Let as is      / Sunday
  2. (CMD) Command      2. Suppress msg   / Monday
  3. (TOD) Time-of-day  3. Suppress log    / Tuesday
  4. (EOT) End-of-task  4. Suppress all    / Wednesday
  5. (EOS) End-of-step              / Thursday
  6. (EOJ) End-of-job              / Friday
                                   / Saturday
                                   / Holiday

Command ==>
F1=Help      F3=Save/End  F7=Up        F8=Down      F12=Cancel
    
```

Figure 6.2: Defining DPROG command in event table

This tells Sekar that DPROG is valid event from 5:10 to 17:00 everyday including holiday since January 1, 2007 up to December 31, 2007. Next is associating action table containing D PROG,APF and D PROG,LNK commands as shown in figure 6.3 to DPROG command event.

```

Action  Help

Action Table                                     Row 1 to 2 of 2

Event .: CMD  DPROG          on system  MFOC
S Action Type System  Action text/parameters
-  1  COMMAND  MFOC      D PROG,APF          -
-  1  COMMAND  MFOC      D PROG,LNK          -
***** Bottom of data *****

Command ==>
F1=Help      F3=Save/End  F7=Up        F8=Down      F12=Cancel

Scroll ==> CSR
    
```

Figure 6.3: Defining action table associated to DPROG command event

This means that, when DPROG command is issued during valid timeframe, Sekar then take it and respond it with issuing D PROG,APF and D PROG,LNK. After RELOADED, newly defined DPROG then effective. Figure 6.4 shows although within valid date range, DPROG still invalid when issued before 5:10 or after 17:00.

```
DPROG
IEE305I DPROG      COMMAND INVALID
```

Figure 6.4: DPROG command still invalid when issued outside timeframe

If DPROG is issued within valid timeframe, as shown in figure 6.5, Sekar then issue D PROG,APF followed with D PROG,LNK. Hence, you can feel that you have made your own command DPROG to display current content of APF and LNKLIST tables.

```
DPROG
CSV450I 05.33.50 PROG,APF DISPLAY 329
FORMAT=DYNAMIC
ENTRY VOLUME DSNNAME
 1 NIT IPL SYS1.LINKLIB
 2 NIT IPL SYS1.SWCLIB
 3 NIT TMP SYS5.XDIV212.LOADLIB
 4 NIT SAV SYS3.XDIV212.LOADLIB
 5 NIT SAV SYS7.XDIV212.LOADLIB
 6 NIT IPL SYS1.MIGLIB
 7 NIT IPL SYS1.SERBLINK
 8 NIT IPL SYS1.CSSLIB
 9 NIT IPL EQR410.SEQRBMOD
10 NIT IPL EQR410.SEQRBUTH
CSV470I 05.33.50 LNKLIST DISPLAY 328
LNKLIST SET LNKLIST00 LNKAUTH=LNKLIST
ENTRY APF VOLUME DSNNAME
 1 A NIT IPL SYS1.LINKLIB
 2 A NIT IPL SYS1.MIGLIB
 3 A NIT IPL SYS1.CSSLIB
 4 A NIT IPL SYS1.SERBLINK
 5 NIT IPL NETV510.SCNMLNK1
```

Figure 6.5: DPROG command results execution of D PROG,APF and D PROG,LNK

## 6.2. Standard XDI Rule

Standard XDI rule for Sekar is actually a rexx program. When command and/or reply action types are felt not enough to handle your specific case, you might need to write rule. Rule is executed as just an ordinary rexx program on TSO batch environment. As it is a rexx program, rule capability will be limited at rexx capabilities. To have more capable rule, you have to write privileged program instead of just rexx program, which will be discussed in next paragraph.

Rule will only receive associated event information. Neither XDI nor Sekar internal information is passed to your rule. Event information is passed by Sekar to your rule in style of command argument:

**RULENAME sysname event\_information\_text**

Where:



1. **RULENAME** is name of your rule, which is a name of member of rule library, left justified and padded with blanks.
2. **Sysname** is 1-to-8 byte name of system on which event occurred.
3. **Event\_information\_text** is a text string explaining the event. The content of the string depends on the type of event:
  - a. MSG: Complete message text. For WTOR message, reply id is placed in front of message text.
  - b. CMD: Complete command text
  - c. TOD: None
  - d. EOJ: Name of ended job
  - e. EOS: Name of job followed with name of ended job step.

Rule will run on XDIRULE address space which is TSO batch started by Sekar from XDI address space. The following JCL is given standard rule procedure. You can customize it as necessary.

```
//XDIRULE PROC M=,ARG=,V=V2,LVL=12,HLQ=SYS5
//XDITSO EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=0M,
//      PARM='%&M &ARG'
//STEPLIB DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..LOADLIB
//SYSEXEC DD DISP=SHR,DSN=&HLQ..ZJOS&V&LVL..RULELIB
//SYSTSIN DD DUMMY
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
```

When associated event occurs, Sekar start XDIRULE with the following console command:

**S XDIRULE,M=*rulename*,ARG='sysname event\_information\_text'**

As describe in the above rule procedure JCL, *rulename* must be a member name of RULELIB dataset which is accessed as SYSEXEC file.

Rule does not really need zJOS-XDI program modules, since most of modules not applicable for user program. The only module that might be needed by rule is DERCCM, a small privileged routine to pass command to console, with the following syntax:

**DERCCM *command\_text***

For example, to issue D A,L command to console from your rule/rexx program, use the command DERCCM *d a,l*. Hence, zJOS-XDI load library has to be accessed as STEPLIB file in XDIRULE procedure.



### 6.2.1 Example

The following rexx script is a simple example of rule named WTORRULE to reply “Y” to specific WTOR message with id SDDWTOR\*.

```
arg msgtxt
address TSO
txt = "" || msgtxt || ""
'SEND' txt 'USER(IBMUSER)'
sysnm = word(msgtxt,1)
rplid = word(msgtxt,2)
msgtxt = subword(msgtxt,3)
rpltxt = "R " || rplid || ",Y"
"DERCCM" rpltxt
exit
```

Rule receives whole argument text as an ordinary rex argument. The first word of text is system name, the second word is reply id and the rest is message text. This rule performs 2 actions.

1. Send whole argument text to TSO userid IBMUSER as a TSO message.
2. Reply to SDDWTOR\* message with reply text “Y” and reply id parsed from 2<sup>nd</sup> word of argument text.

To install WTORRULE rule:

1. WTORRULE must be placed as a member in RULELIB or any dataset referred as SYSEXEC file in XDIRULE procedure. Member name must be WTORRULE
2. WTORRULE must be registered as rule action for SDDWTOR\* message event as shown in figure 6.6.

Action Help		Action Table		Row 1 to 3 of 3
Event .: MSG <u>SDDWTOR*</u> . on system <u>MFOC</u> .				
S	Action Type	System	Action text/parameters	
3	<u>RULE</u>	<u>MFOC</u>	<u>WTORRULE</u> -	
1	<u>COMMAND</u>	<u>MFOC</u>	SE 'SDDWTOR REPLIED BY WTORRULE!',USER=(IBMUSER)-	
1	<u>COMMAND</u>	<u>MFOC</u>	SE 'HAVE FUN!!! HE HE HE',USER=(IBMUSER) -	
***** Bottom of data *****				
Command ==>				
Scroll ==> CSR				
F1=Help	F3=Save/End	F7=Up	F8=Down	F12=Cancel

Figure 6.6: WTORRULE in action table of SDDWTOR\* event



Upon effective, WTORRULE runs every time message prefixed with SDDWTOR\* occurs. When message "04 SDDWTOR(SYS1): Ini hanya ujicoba" as shown in figure 6.7, WTORRULE is then active and issue "R 04,Y" reply as its 2<sup>nd</sup> action.

```
@04 SDDWTOR(SYS1): Ini hanya ujicoba
$HASP100 XDIRULE ON STCINRDR
$HASP373 XDIRULE STARTED
IEE403I XDIRULE - STARTED - TIME=01.04.01
IEE600I REPLY TO 04 IS;Y
IEE404I TESTWTOR - ENDED - TIME=01.04.02
$HASP395 TESTWTOR ENDED
IEE404I XDIRULE - ENDED - TIME=01.04.03
$HASP395 XDIRULE ENDED
$HASP250 TESTWTOR PURGED -- (JOB KEY WAS C047AC3D)
IEA989I SLIP TRAP ID=X33E MATCHED. JOBNAME=*UNAVAIL, ASID=0037.
IEA989I SLIP TRAP ID=X33E MATCHED. JOBNAME=*UNAVAIL, ASID=0035.
```

Figure 6.7: WTORRULE action shown in syslog

Its first action is TSO message in IBMUSER terminal, as shown in figure 6.8. whole event information is displayed.

```
SDDWTOR REPLIED BY WTORRULE! CN(INTERNAL)
HAVE FUN!!! HE HE HE CN(INTERNAL)
+MFOC 04 SDDWTOR(SYS1): INI HANYA UJICOBA XDI
***
```

Figure 6.8: Effect of WTORRULE action in IBMUSER terminal

## 6.3. Innovation with Rule

As discussed in 6.2 above, the most important thing you should understand is that the way Sekar execute a rule is by issuing START command:

**S XDIRULE,M=rulename,ARG='sysname event\_information\_text'**

This command is the only correlation between Sekar and rule. Hence, XDIRULE procedure is not necessarily as the above given procedure. The rule is also not necessarily a rexx program. So, you can change XDIRULE procedure to your own. You can also write non-rexx program for your rule. As long as can be triggered by the above command, it will become a valid rule for Sekar. The only thing you should care is that XDIRULE procedure must be applicable for any rule name. Program which is called in EXEC card of XDIRULE must capable to call rule pointed by M=rulename keyword as describe in figure 6.9.

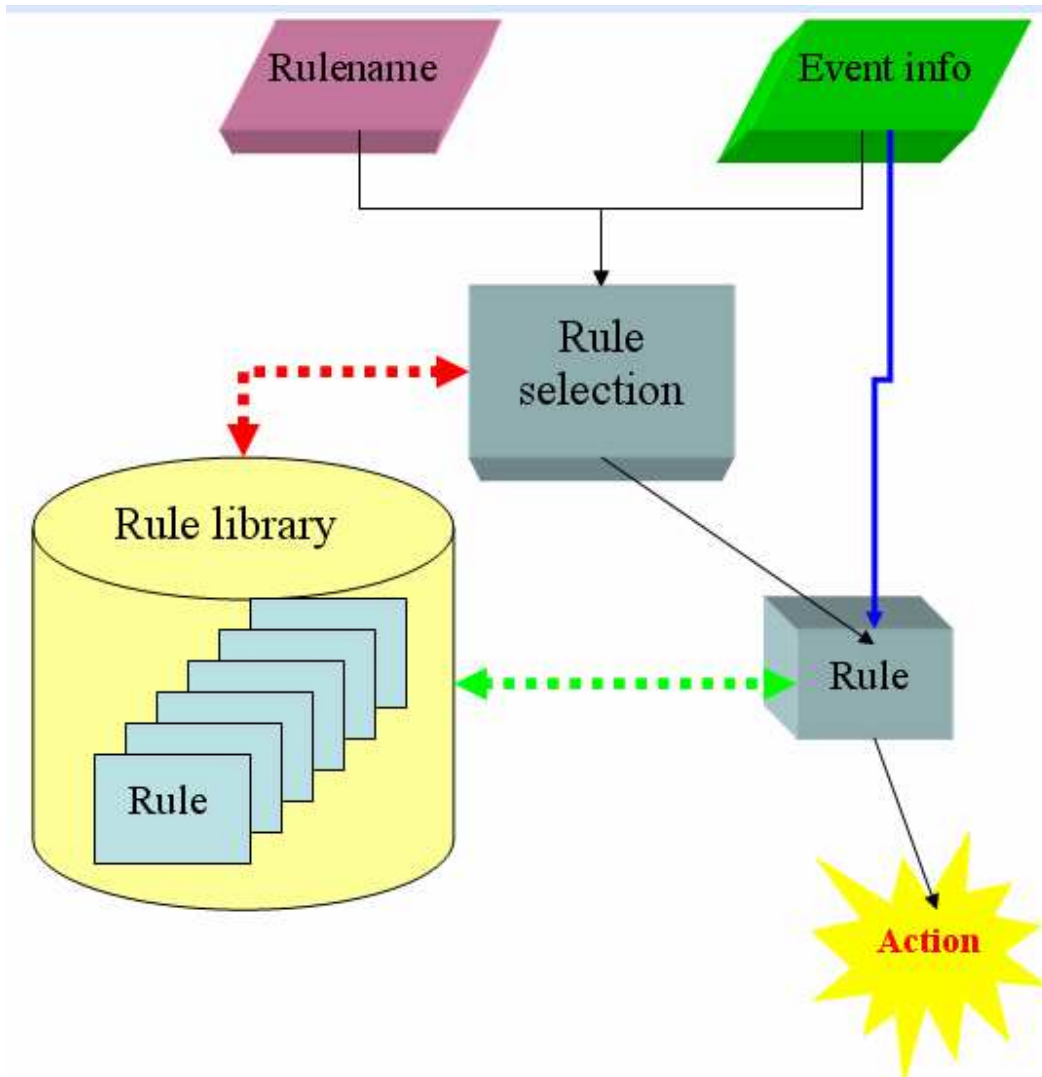


Figure 6.9: Rule processing logic flow

In given standard XDIRULE procedure, rule selection is done internally by TSO program IKJEFT01. By placing rulename and event information text in PARM= keyword, TSO assumes as a command text and executes it immediately after initialization complete. If you replace IKJEFT01 with your own program, rule selection then becomes your responsibility.

## 6.3.1 Rule Programming

A rule is actually can either be a rexx exec or program module. This gives you a chance to enhance EMS functionalities and application in your specific system environment. For simpler cases, rule can be written in either TSO standard Rexx language or any other application programming language, such as Cobol,



PL/1, C++, Pascal, Fortran and so forth. For more complex cases, which need internal information, you are recommended to use assembly instead. At this moment, discussion scope is limited only surrounding rule programming without modifying XDIRULE procedure, which means you still use TSO to find and execute your rule. The only thing you should care is search order and passing event information.

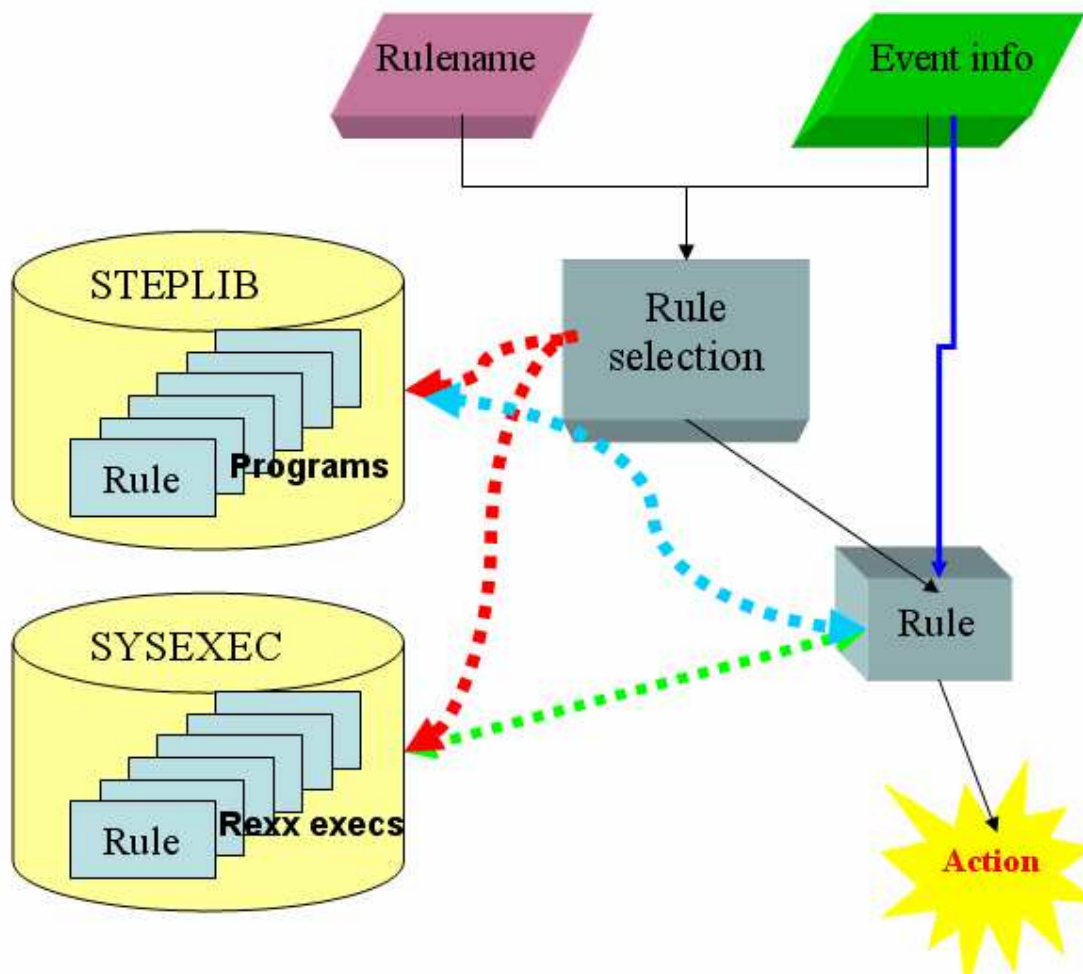


Figure 6.10: Rule rexx exec and programs

## Rule search order

Since rule is executed as a command on TSO environment, rule is searched by TSO in standard TSO search order. Program module is searched prior to rexx and clist exec. Hence if your rule is rexx and there is a program module with the same name in STEPLIB file, LPA/ELPA or LNKLIST concatenation, it will always be executed, instead of your rule, and may yields unpredictable problem.

In given XDIRULE procedure, however, rulename is invoked via `PARM=` keyword in EXEC card, which is specified as `PARM='%&M &ARG'`. The percent sign (%) is to limit TSO search scope only for around SYSEXEC and SYSPROC files. For example, if your rulename is MYRULE, JCL substitution for %&M variable results %MYRULE. TSO then search SYSEXEC and SYSPROC files only to find member named MYRULE. If your rule is a program module, which certainly not in SYSEXEC nor SYSPROC files, XDIRULE is then failed. Hence, % prefix as in given XDIRULE procedure, valid only if all rules are rexx or clist execs. You are even recommended to use percent (%) prefix to avoid your rexx exec rule being overridden by program module with the same name.

If your rule is a program module, unless required to be in LPA/ELPA or LNKLIST, strongly recommended to be placed in first concatenated dataset in STEPLIB file. This to make sure your rule is in first search order. Percent (%) prefix sign must not be used in `PARM=` keyword in EXEC card in XDIRULE procedure. You must change it to `PARM='&M &ARG'` instead.

### **Passing event information to rule**

Main and the most important input for a Sekar rule is a complete information regarding event. Except for TOD event, complete event information text consist of name of system on which event was occurred and detail of event information. For TOD event, since TOD is very common information which can be taken from anywhere including in rule routine itself, hence not necessarily to be passed, so you can ignore it.

Complete event information text is passed by IKJEFT01 to your rule as a single string in command argument style. In rexx program, you can easily take it with `arg` instruction as follow:

```
arg complete_text
parse var complete_text sysname event_info_text

or

arg sysname event_info_text
```

In such style, internally argument is assumed as a single string and pointed by register 1, with the following structure:

```
INFO_AREA    DSECT
INFO_LENGTH  DC   AL2(length)
INFO_TEXT    DC   C'complete event info text'
```

Complete event info placed in `INFO_TEXT` area, and its length in `INFO_LENGTH` field. Text is unformatted, so you have to parse to find system name, reply id (for





WTOR) and other major information. System name is the first string-word within INFO\_TEXT area. In assembly you can parse system name as follow:

```

Using INFO_AREA,R1
xr      R15,R15
icm     R15,b'0011',INFO_LENGTH
la      R2,sysname
la      R3,1'sysname
mvi     sysname,c' '
mvc     sysname+1(1'sysname-1),sysname
la      R4, INFO_TEXT
Loop_sysn equ *
cli     0(R4),c' '
be      Got_sysn
mvc     0(1,R2),0(R4)
la      R2,1(R2)
la      R4,1(R4)
bctr    R15,0
bct     R3,Loop_sysn
Got_sysn equ *
```

If you are using compiler type language, to get and parse event information is not as easy as in rexx or assembly. It depends on which language you are using, and whether command style argument is supported. You must follow the product manual document. If command style argument is not supported, you can use either rexx or assembly as an interface prior to your program. In assembly you can easily manipulate command style argument to conventional program call parameters to be passed to your rule program. But, if you prefer to use rexx, you can use one of following facilities to pass argument as conventional program call parameters to your rule program:

**Address LINKMVS 'yourpgm parm<sub>1</sub> parm<sub>2</sub> parm<sub>3</sub>'**

or

**Address LINKPGM 'yourpgm parm<sub>1</sub> parm<sub>2</sub> parm<sub>3</sub>'**

Use Address LINKMVS if you want to pass both string length and string data to your rule. Otherwise, use Address LINKPGM, and only string data is passed to your rule. Both ways will bring your rule at the same TCB level with the interface. If you want your rule run in different TCB level, use Address ATTACHMVS or Address ATTACHMVS instead. Your rule then runs as subtask or child process.

Further about rexx can be in IBM manuals z/OS MVS TSO/E Rexx User's Guide (SA22-7791) and z/OS MVS TSO/E Rexx Reference (SA22-7790).



## 6.3.2 Creating Your Own XDIRULE

Creating your own XDIRULE means replacing IKJEFT01 with your own program. Here you are assumed as advanced users, who have in-depth skill in z/OS MVS systems programming environment. In-depth skill in zJOS/Sekar does not really matter since handshaking between Sekar and XDIRULE is nothing more than just a start command string of:

**S XDIRULE,M=*rulename*,ARG=*'sysname event\_information\_text'***

In this case, Sekar is just a medium to enable certain event to trigger XDIRULE. While triggering XDIRULE, Sekar informs name of rule and event information to XDIRULE via START command arguments. Once XDIRULE task is started, it becomes an independent task on z/OS system, which can do anything.

By creating your own XDIRULE procedure, you can even use *rulename* just as input information instead of a member name. Since given *rulename* in START command string was selected based on type of captured event which has been validated against specified timeframe in your event table, *rulename* can also be used as such switch control information for your specific processing.

However, the important thing you must seriously care, once you enable your own XDIRULE procedure, all existing rules will no longer applicable. If you want them remain applicable, you have to adopt logic mechanism of IKJEFT01 in your new program.

## 6.4. zJOS Supported Rexx Functions

zJOS provides some rexx functions to support rule programming. These are:

- i. zjaxfer()
- ii. zjcal()
- iii. zjcmd() or xcommand()
- iv. zjevent()
- v. zjholday()
- vi. zjpuspa()
- vii. zjsekar()
- viii. zjserver()
- ix. zjset()
- x. zjstate()
- xi. zjwait()
- xii. zjwto()
- xiii. zjwtoor()



## 6.4.1 zjaxfer()

zjaxfer() is a rexx function to obtain state information of XDI/AutoXfer. As it is a privileged function, it needs authorization from zJOS. Your local security setting won't be able to detect its internal process.

Syntax:

```
var = zjaxfer()
```

Where:

1. var is variable name to contain function result:
  - a. State information followed with destination table id. Possible states are:
    - i. ACTIVE 00
      1. AutoXfer is currently active with destination table 00 (member XDITAB00)
    - ii. INACTIVE 05
      1. AutoXfer is currently inactive and destination table 05 (member XDITAB05) is addressed to be used when activated
  - b. 'NOTHING' → zJOS subsystem does not exist.
  - c. 'UNAUTHORIZED nnn' → command was not granted
    - i. nnn is zJOS authorization exception code

Example:

To obtain current state of AutoXfer, you can write it in rexx as follow:

```
x = zjaxfer()  
axfrstate = word(x,1)  
axfrtable = word(x,2)
```

## 6.4.2 zjcal()

zjcal() is a rexx function to confirm or obtain special calendar information. As it is a privileged function, it needs authorization from zJOS. Information is based on combined current system calendar and user specified holiday table.

Syntax:

```
var = zjcal('calcode')
```

or



```
var = zjcal()
```

Where:

1. Argument 'calcode' is text of special calendar code. If calendar code match with current calendar, the function returns 'Y'. Otherwise it returns 'N'. Supplied calendar codes are:
  - HOLI for holiday
  - IDAO for initial working day after off
  - IWWD for initial week working day
  - IMWD for initial month working day
  - FWWD for final week working day
  - FMWD for final month working day
  - EMON for end of month day
  - OFFD for off day (holiday, Saturday or Sunday)
2. If no argument specified, function returns a string text contains list of current calendar codes.

Example:

The following example is rexx routine to display special calendar currently in effect:

```
Select
  When zjcal('IDAO') = 'Y' then,
    Say "Today is initial working day after off"
  When zjcal('IWWD') = 'Y' then,
    Say "Today is initial week working day"
  When zjcal('IMWD') = 'Y' then,
    Say "Today is initial month working day"
  When zjcal('FWWD') = 'Y' then,
    Say "Today is final week working day"
  When zjcal('FMWD') = 'Y' then,
    Say "Today is final month working day"
  When zjcal('EMON') = 'Y' then,
    Say "Today is end of month day"
  When zjcal('OFFD') = 'Y' then,
    Say "Today is off day"
  Otherwise NOP
End
```



## 6.4.3 zjcmd() or xcommand()

zjcmd() is a rexx function to pass command text to console. As it is a privileged function, it needs authorization from zJOS. Nevertheless, the use of zjcmd() also depend on your local security setting.

Syntax:

```
var = zjcmd('command_text')
```

or

```
var = zjcmd(cmdvar)
```

Where:

1. 'command\_text' is text of command string, which must be enclosed with either single or double quote.
2. cmdvar is variable name to contain text of command string.
3. var is variable name to contain function result. Possible results are:
  - a. 'ISSUED' → command was issued
  - b. 'UNAUTHORIZED nnn' → command was not granted  
nnn is zJOS authorization exception code

Example:

To issue MVS command D PROG,APF, you can write it in rexx as follow:

```
x = zjcmd('D PROG,APF')
```

Alternatively, you can use variable as follow:

```
cmd = "D PROG,APF"  
x = zjcmd(cmd)  
if x = 'ISSUED' then ...  
else ...
```





## 6.4.4 zjevent()

When you want to handle an event very specifically or just for trial prior to set it up permanently in Sekar EMS table, you may do it in rexx program with your own logic. `zjevent()` is a rexx function to request Sekar for notification when a certain event as specified in arguments occurs, or when any event previously requested by using `zjset()` function occurs, and optionally perform an immediate simple action. When issued, `zjevent()` function may entering wait state condition to listen notification from Sekar regarding occurrence of previously selected event.

When issued with arguments, `zjevent()` only listens an event as specified in the arguments and perform a simple action if one specified in argument. Function is immediately woken up and terminated when the event occurs. Actually when notified by Sekar that the expected event occurs. To wait next similar event, you must iterate it. By using `zjevent()` function with arguments, your rexx program can only listen a single event at a time.

When issued without argument, `zjevent()` listens all previously requested events by using `zjset()` function. Hence, `zjevent()` without argument must be preceded by `zjset()` function. This gives you chance to your rexx program to listen multiple events at a time. When one of expected events occurs, `zjevent()` is immediately woken up and terminated. To continue wait for next event, either similar or other event, you must iterate it. As `zjset()` function does not support immediate action, no action is perform when `zjevent()` function without argument is woken up. All action must be handle in your rexx program by using `zjcmd()` function.

As it is a privileged function, `zjevent()` needs authorization from zJOS subsystem. Nevertheless, the use of `zjevent()` also depend on your local security setting.

Syntax:

```
var = zjevent('evtype','evtext',['SUPPRESS'],'action')
```

or

```
var = zjevent(evtypevar,evtextvar,optvar,actionvar)
```

or

```
var = zjevent() /* without argument */
```

Where:

1. Arguments:



- a. 'evtype' (expression) or evtypevar (variable) is type of event in 3-character abbreviation, which MSG for message event, CMD for command event, EOJ for end-of-job event or EOS for end-of-step event.
  - b. 'evtext' (expression) or evtextvar (variable) is text or string represents part of the source of event information you want to capture or trap.
  - c. 'SUPPRESS' (expression) or optvar (variable) is suppression option for MSG or CMD event only. If optvar (variable) is used, it must contain 'SUPPRESS' or nulls or blanks.
    - i. SUPPRESS for MSG event causes message to be suppressed. Hence messages no longer eligible for subsequent trapping for either zjevent() invocations or EMS table entries.
    - ii. SUPPRESS for CMD event causes command to be suppressed and not executed. Hence command no longer eligible for subsequent trapping for either zjevent() invocations or EMS table entries.
  - d. 'action' (expression) or actionvar (variable) is text to express a simple action you want to perform. There are 3 possible simple actions:
    - i. 'MSG=message text'
    - ii. 'CMD=command text'
    - iii. 'REPLY=reply text'
2. var is variable name to contain function result, which depend on event type:
- a. For MSG event from WTO message, var contains:  
'MSG message text'
  - b. For MSG event from WTOR message, var contains:  
'MSG (REPLYID=nn) message text'
  - c. For CMD event, var contains:  
'CMD command text'
  - d. For EOJ event, var contains:  
'EOJ JOB=jobname SCC=nnn MAXCC=nnn '
  - e. For EOS event, var contains:  
'EOS JOB=jobname STEP=stepname SCC=nnn UCC=nnn'
  - f. If one or more arguments were invalid or missing, var contains:  
'ERROR\_ARGUMENT'
  - g. If command was not granted, var contains:  
'UNAUTHORIZED nnn'  
nnn is zJOS authorization exception code

Example:



To capture command D APF and change it to D PROG,APF, you can write it in rexx as follow:

```
X = zjevent('CMD','D APF','SUPPRESS',  
            'CMD=d prog,apf')
```

Alternatively, you can use variable as follow:

```
cmd = "D APF"  
xcmd = "CMD=D PROG,APF"  
x = zjevent('cmd',cmd,'SUPPRESS',xcmd)  
if x = cmd then ...  
else ...
```

With option SUPPRESS, D APF command will appear as a valid command and result as if you issue D PROG,APF command.

Notes:

1. When a CMD or MSG event occurs, zjevent() is notified prior to EMS table. Therefore, you must care of suppression option of zjevent() function. If event is suppressed, event is no longer eligible for either subsequent zjevent() or EMS table entries.
2. Once zjevent() got missed, it will remain stay in wait state until similar event occurs. Unless very urgent situation, you should not cancel it. Doing so, causes an orphaned EVB control block and will remain in ECSA until next IPL.

## 6.4.5 zjholday()

zjholday() is a rexx function to obtain holiday information during current month. As it is a privileged function, it needs authorization from zJOS. Information is based on combined current system calendar and user specified holiday table.

Syntax:

```
var = zjholday(holiarg)
```

Where:

Holiarg is a text string contains valid holiday argument:

- 'NEXT' – to return number of days to reach the nearest holiday from today within this month. Zero is returned if nothing is found.



- 'DAY' – to return week day name (e.g. Sunday, Monday etc.) of the nearest holiday from today within this month. Null is returned if nothing is found.
- 'WDAY' – to return 0 to 6 week day number (e.g. 0 for Sunday, 1 for Monday etc.) of the nearest holiday from today within this month. Null is returned if nothing is found.
- 'DESC' – to return the holiday description of the nearest holiday from today within this month. Null is returned if nothing is found.
- 'FINAL' – to return number of days to reach the last holiday this month. Zero is returned if nothing is found.
- 'FDAY' – to return week day name (e.g. Sunday, Monday etc.) of the last holiday this month. Null is returned if nothing is found.
- 'FWDAY' – to return 0 to 6 week day number (e.g. 0 for Sunday, 6 for Saturday etc.) of the last holiday this month. Null is returned if nothing is found.
- 'FDESC' – to return the holiday description of the last holiday this month. Null is returned if nothing is found.

Example:

The following example is rexx routine to alert operation team:

```
x = zjholday('next')
If x <> 0 then,
    Say "Please be prepared" zjholday('desc'),
    "is coming in" x "days."
```

## 6.4.6 zjpuspa()

zjpuspa() is a rexx function to obtain state information of zJOS/Puspa. As it is a privileged function, it needs authorization from zJOS. Your local security setting won't be able to detect its internal process.

Syntax:

```
var = zjpuspa()
```

Where:

var is variable name to contain function result:



- a. State information followed with schedule table id. Possible Puspa states are:
  - i. READY 00  
Puspa is currently ready for work with schedule table 00. All internal agents up, but schedule is not being processed yet.
  - ii. ACTIVE 00  
Puspa is currently active and schedule table 00 is being processed.
  - iii. INACTIVE 05  
Puspa is currently inactive and schedule table 05 is addressed to be used when activated. All or some internal agents down.
  - iv. PASSIVE 05  
Puspa is currently passive and schedule table 05 was processed and is still addressed to be used for next process. All internal agents up, schedule was processed, but is not refreshed yet for next process.
- b. 'NOTHING' → zJOS subsystem does not exist.
- c. 'UNAUTHORIZED nnn' → command was not granted
  - v. nnn is zJOS authorization exception code

Example:

To obtain current state of Puspa, you can write it in rexx as follow:

```
x = zjpuspa()  
scdstate = word(x,1)  
scdtable = word(x,2)  
select  
  when scdstate = 'ACTIVE' then ...  
  when scdstate = 'INACTIVE' then ...  
  when scdstate = 'PASSIVE' then ...  
  otherwise ...  
end
```

## 6.4.7 zjsekar()

zjsekar() is a rexx function to obtain state information of zJOS/Sekar (EMS). As it is a privileged function, it needs authorization from zJOS. Your local security setting won't be able to detect its internal process.

Syntax:

```
var = zjsekar()
```



Where:

var is variable name to contain function result:

- a. State information followed with EMS table id. Possible Sekar states are:
  - i. ACTIVE 00  
Sekar is currently active with EMS table 00 (member XDIEMS00)
  - ii. ACT(MCS) 00  
Sekar is currently active using MCS with EMS table 00 (member XDIEMS00). This is a legacy feature from 2.1.2.
  - iii. INACTIVE 05  
Sekar is currently inactive and EMS table 05 (member XDIEMS05) is addressed to be used when activated
  - vi. ONHOLD 05  
Sekar is currently inactive while zJOS subsystem is being held (may be zJOS address space down). EMS table 05 (member XDIEMS05) is addressed to be used when activated
- b. 'NOTHING' → zJOS subsystem does not exist.
- c. 'UNAUTHORIZED nnn' → command was not granted
  - vii. nnn is zJOS authorization exception code

Example:

To obtain current state of Sekar, you can write it in rexx as follow:

```
x = zjsekar()  
emsstate = word(x,1)  
emstable = word(x,2)  
select  
  when emsstate = 'ACTIVE' then ...  
  when emsstate = 'INACTIVE' then ...  
  when emsstate = 'PASSIVE' then ...  
  otherwise ...  
end
```

## 6.4.8 zjserver()

zjserver() is a rexx function to obtain state information of zJOS Server. As it is a privileged function, it needs authorization from zJOS. Your local security setting won't be able to detect its internal process.





Syntax:

```
var = zjserver()
```

Where:

var is variable name to contain function result:

- a. State information. Possible states are:
  - i. ACTIVE
  - ii. INACTIVE
- b. 'NOTHING' → zJOS subsystem does not exist.
- c. 'UNAUTHORIZED nnn' → command was not granted
  - viii. nnn is zJOS authorization exception code

Example:

To obtain current state of AutoXfer, you can write it in rexx as follow:

```
svrstate = zjserver()  
if svrstate = 'ACTIVE' then ...  
else ...
```

## 6.4.9 zjset()

zjset() is a rexx function to request Sekar for notification when a certain event as specified in its arguments occurs. Unlike zjevent(), once request is confirmed by Sekar, zjset() is not entering to wait state until notified by Sekar. Rather, zjset() immediately finish and return to your rexx program when request is confirmed. You are responsible to handle the notification from Sekar by issuing zjevent() function without argument.

Such mechanism gives chance to your rexx program to handle multiple events or multiple types of events at a time. You may issue several zjset() for all expected events, then followed by iterated zjevent() without argument as in the following example:

```
req1 = zjset('MSG','$HASP492')  
req2 = zjset('MSG','IST020I')  
req3 = zjset('CMD','P','SUPPRESS')  
Do forever  
  event = zjevent()  
  evtype = strip(word(event,1))  
  Select  
    When evtype = 'MSG' then,  
      Do  
        msgid = strip(word(event,2))  
        Select  
          When msgid = '$HASP492' then,
```



```
Do
    action1 = zjcmd('START VTAM')
    action2 = zjcmd('START SDSF')
End
When msgid = 'IST020I' then,
    Do
        action1 = zjcmd('START TSO')
        action2 = zjcmd('START CICSPROD')
    End
    Otherwise NOP
End
End
When evtype = 'CMD' then,
    Do
        cmdverb = strip(word(event,2))
        cmdtarg = strip(word(event,3))
        If cmdverb = 'P' then,
            If cmdtarg = 'VTAM' then,
                action3 = zjcmd('Z NET,QUICK')
            Else,
                action3 = zjcmd('STOP ' || cmdtarg)
            End
        End
    Otherwise NOP
End
```

The above example shows you how rexx program can handle \$HASP392 and IST020I messages and STOP command. When \$HASP392 message occurs, this indicates JES2 initialization complete, then start VTAM and SDSF. When IST020I message occurs, this indicates VTAM initialization complete, then start TSO and CICSPROD. Such idea is very common in EMS.

One thing you should keep in mind is the way this example handle P command. P is a short form of STOP command verb. When P command issuance occurs, suppress it to avoid execution. Then, check the command argument, which is a name of workload or job to be brought down. If the targeted job is VTAM, then issue Z NET,QUICK. Else, issue STOP for the same target. This will affect as if P command applicable for VTAM, which is actually not. Although STOP and P are actually executed by the same processor, STOP verb won't be trapped since Sekar only evaluate the command text.

Combined zjset() and zjevent() functions is the main design of zJOS rexx support for Sekar. Most of EMS mechanism you have implemented in Sekar EMS table can be handled by a single rexx program which fully follows your idea, except for remote events. Currently, zJOS rexx functions package does not support remote event exchange.

As it is a privileged function, it needs authorization from zJOS. Nevertheless, the use of zjset() also depend on your local security setting.

Syntax:



```
var = zjset('evtype','evtext',['SUPPRESS'])
```

or

```
var = zjset(evtypevar,evtextvar,optvar)
```

Where:

1. Arguments:

- a. 'evtype' (expression) or `evtypevar` (variable) is type of event in 3-character abbreviation, which MSG for message event, CMD for command event, EOJ for end-of-job event or EOS for end-of-step event.
- b. 'evtext' (expression) or `evtextvar` (variable) is text or string represents part of the source of event information you want to capture or trap.
- c. 'SUPPRESS' (expression) or `optvar` (variable) is suppression option for MSG or CMD event only. If `optvar` (variable) is used, it must contain 'SUPPRESS' or nulls or blanks.
  - i. SUPPRESS for MSG event causes message to be suppressed. Hence messages no longer eligible for subsequent trapping.
  - ii. SUPPRESS for CMD event causes command to be suppressed and not executed. Hence command no longer eligible for subsequent trapping.

2. `var` is variable name to contain function result:

- a. If request was confirmed, `var` contains:
  - iii. 'SET'
- b. If one or more arguments were invalid or missing, `var` contains:
  - iv. 'ERROR\_ARGUMENT'
- c. If authorization was not granted, `var` contains:
  - v. 'UNAUTHORIZED nnn'
    1. nnn is zJOS authorization exception code

Example:

See the above example.

Note:

Logical path length between `zjset()` and `zjevent()` functions issuance could be vary, depend on your rexx program logic. If the path length is quite significant, event might occur prior to `zjevent()` function issuance. In such case, `zjevent()` will immediately return to your rexx program, since event



information has already given in your program area. There is no indicator telling you whether event occurs earlier than `zjevent()` function issuance. You, therefore, should care of path length if time precision is required.

## 6.4.10 **zjstate()**

`zjstate()` is a rexx function to obtain state information of any local workload. You can use `zjstate()` function to detect whether a certain job is up or down. As it is a privileged function, it needs authorization from zJOS. Nevertheless, the use of `zjstate()` also depend on your local security setting.

Syntax:

```
var = zjstate('workload_name')
```

or

```
var = zjstate(namevar)
```

Where:

'workload\_name' is text of name of workload (jobname), which must be enclosed with either single or double quote.

namevar is variable name to contain text of workload name (jobname).

var is variable name to contain function result.

- a. If workload up, var variable will contain 5 words of:  
    'UP type ASCB=xxxxxxxx ASID=xxxx JOBID=xxxxxxxx'  
    Type is either STC, JOB or TSU
- b. If workload down, var contains:  
    'DOWN'
- c. If argument invalid (no argument or argument more than 8-byte)  
    'ERROR\_ARGUMENT'
- d. If authorization not granted by zJOS  
    'UNAUTHORIZED nnn'  
    nnn is zJOS authorization exception code

Example:

To obtain state information of VTAM, you can write in rexx:

```
x = zjstate('VTAM')  
vtamstate = word(x,1)
```



Alternatively, you can use variable as follow:

```
job = "VTAM"  
x = zjstate(job)  
vtamstate = word(x,1)
```

Upon completion, if VTAM is up, x may contains:

```
"UP STC ASCB=00F8A000 ASID=0021 JOBID=STC09703"
```

You can get all information by parsing the result (x) using rexx standard parsing method as follow:

```
If vtamstate = 'UP' then,  
  Parse var x,  
    'UP' jobtype 'ASCB=' ascb 'ASID=' asid,  
    'JOBID=' jobid
```

## 6.4.11 zjwait()

zjwait() is a rexx function to enter to wait state based on TOD clock or interval. Entering wait state is not a privileged process, hence zjwait() function need not authorization from zJOS. Besides, this TOD processing internally is nothing to do with zJOS. It is completely done within your address space. While waiting, your rexx program is suspended until the time is expired.

Syntax:

```
var = zjwait('timeval')
```

or

```
var = zjwait(timevar)
```

Where:

'timeval' is time value, which must be enclosed with either single or double quote. Time value must be in the following format:

- TOD value → wait until specified TOD. Format is:  
HH:MM:SS
- Interval value → wait for specified interval. Formats are:  
+HH:MM:SS  
+MM:SS  
+SS



`timevar` is variable name to contain time value.

`var` is variable name to contain function result.

- a. If wait state was entered:  
    'EXPIRED FOR nnnnnn SECS'
- b. If wait state was not entered (for example: already late):  
    'EXPIRED FOR -nnnnnn SECS'
- c. If CPU clock was error:  
    'ERROR\_CLOCK RC=nnnn'
- d. If argument invalid (no argument or argument more than 9-byte)  
    'ERROR\_ARGUMENT'

Example:

1. To wait until 23:30:00:

```
x = zjwait('23:30:00')
y = zjwto('Now is 23:30:00')
```

2. To wait for 5 minutes 45 seconds and using variable:

```
interval = '+05:45'
x = zjwait(interval)
```

## 6.4.12 **zjwto()**

`zjwto()` is a rexx function to issue WTO message to console. Issuing WTO is not a privileged process, hence it need not authorization from zJOS. Besides, this WTO processing internally is nothing to do with zJOS subsystem. It is completely done within your address space.

Syntax:

```
var = zjwto('message_text')
```

or

```
var = zjwto(msgvar)
```

Where:

'message\_text' is a text of message to be issued, which must be enclosed with either single or double quote.

`msgvar` is variable name to contain message text.





`var` is variable name to contain function result.

- a. If WTO was successfully issued:  
    'ISSUED'
- b. If WTO was unsuccessful:  
    'ABORTED'
- c. If argument invalid (no argument or argument more than 126-byte length)  
    'ERROR\_ARGUMENT'

Example:

To issue "Good morning!" to system console:

```
x = zjwto(Good morning!')
```

### 6.4.13 **zjwto()**

`zjwto()` is a rexx function to issue WTOR message to console. Issuing WTOR is not a privileged process, hence it need not authorization from zJOS. Besides, WTOR processing internally is nothing to do with zJOS subsystem. It is completely done within your address space.

As it is WTOR, reply is required to finish the function. While waiting for reply, your rexx program is suspended until your message is replied. Reply text can be any 1 to 54 characters string

Syntax:

```
var = zjwto('message_text')
```

or

```
var = zjwto(msgvar)
```

Where:

'message\_text' is a text of message to be issued, which must be enclosed with either single or double quote.

`msgvar` is variable name to contain message text.

`var` is variable name to contain function result.

- a. If WTOR was successfully issued:  
    'REPLY=replytext'



- b. If WTOR was unsuccessful:  
    'ABORTED RC=nnn'
- c. If argument invalid (no argument or argument more than 126-byte length)  
    'ERROR\_ARGUMENT'

Example:

To issue "Good morning!" to system console and require reply:

```
x = zjwtr(Good morning!')
/* check reply text */
If substr(x,1,6) = "REPLY=" then,
    Parse var x "REPLY=" reply
```

## 6.5. Using zJOS Rexx Functions

If you are choosing rexx for your innovative rules, zJOS supported rexx functions might help you exploring more your automation system. In standard zJOS rule, except for zjevent() function, the use of these functions does not need specific requirements. Just code them in your rule program.

### 6.5.1 Non-rule Rexx Programming

The zJOS supported rexx functions are not only for rule programming. You may use them in your ordinary rexx programs. Some zjxxxx() functions which do not need authorization, available even when zJOS address space not up. You can either call them in TSO (batch or terminal session) or native MVS rexx job. The functions, however, are in rexx local functions package which are placed in zJOS load library. Hence, you have to include zJOS library in your STEPLIB DD.

In TSO terminal session, you must include zJOS load library in STEPLIB DD of your logon procedure. In TSO batch job, you must include zJOS load library in STEPLIB DD of your job JCL as follow:

```
//JTSOREXX JOB ...
//TSO      EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=0M
//STEPLIB DD DISP=SHR,DSN=your.zJOS.load.library
//SYSEXEC DD DISP=SHR,DSN=your.rexx.library
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN DD *
%your_rexx_module_name arguments
```



/\*

In native MVS rexx job, zJOS load library must also be included in STEPLIB DD of your job JCL. Below is an example of native rexx batch job JCL.

```
//JMVSREXX JOB ...
//REXX      EXEC PGM=IRXJCL,REGION=0M,
//      PARM=' your_rexx_module_name arguments'
//STEPLIB DD DISP=SHR,DSN=your.zJOS.load.library
//SYSEXEC DD DISP=SHR,DSN=your.rexx.library
//SYSTSPRT DD SYSOUT=A
//SYSPRINT DD SYSOUT=A
//SYSTSIN DD DUMMY
```

## 6.5.2 Using zjset() and zjevent() in Your Program

As explained in 6.4.3, zjevent() function without argument can trap occurrence of any event previously requested by using zjset() function. In 6.4.7 explains that you can request Sekar to collect event occurrence for your rexx program using zjset() function. Both functions support message, command, end-of-step (EOS) and end-of-job (EOJ) events. When you invoke combined zjset() and zjevent() in your rexx program, you will feel you as if have a mini-Sekar. Your rexx able to manage most of local events as if you use Sekar EMS table. If you need multiple actions, you may do it outside. For example:

```
set1    = zjset('MSG','msg_string')
set2    = zjset('CMD','cmd_string','SUPPRESS')
set3    = zjset('EOS','jobname.stepname')
set4    = zjset('EOJ','jobname')
Do forever
  event  = zjevent()
  evtype = strip(word(event,1))
  Select
    When evtype = 'MSG' then call MSG_handler
    When evtype = 'CMD' then call CMD_handler
    When evtype = 'EOS' then call EOS_handler
    When evtype = 'EOJ' then call EOJ_handler
    Otherwise NOP
  End
End
Exit

MSG_handler:
/* routine to handle message events */
Return

CMD_handler:
/* routine to handle command events */
Return
```



```
EOS_handler:
/* routine to handle end-of-step events */
Return

EOJ_handler:
/* routine to handle end-of-job events */
Return
```

Since its function and characteristic almost close with EMS main function, hence, you should not use it in standard Sekar rule. Because, once issued, it will entering wait state until the event it waiting for is occurred.

:

## Chapter 7 Using zJOS Control Panel

zJOS control panel is ISPF panel driven by TSC module which only applicable within TSO/ISPF online session. Since TSC is an authorized/privileged module which capable to explore all zJOS internals and runs in supervisor state with key zero, you should limit it from common user access. It's supposed for zJOS administration use only. Figure 2.2 in chapter 2 shows complete appearance of zJOS control panel. Once zJOS was completely installed and setup using XDI standard procedure, "XDI" verb should be registered in your current ISPF system command table. Hence you should be able to issue "XDI" command anywhere within TSO/ISPF environment. Upon issuance of "XDI" verb, a series of actions are taken to invoke zJOS control panel.

Action		Help		zJOS-XDI Control Panel				Row 975 to 997 of 997	
Command	Product	State	Table	Suf	Works	Usage	Day		
	Sekar (EMS)	<UP> ACT(SS)	loaded	00	000000	LCNSD/YR	0816		
	Puspa (SCD)	<UP> ACTIVE	loaded	02	000001	LCNSD/YR	0816		
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	LCNSD/YR	0816		
	Net Server	<UP> ACTIVE	unloaded	**	000000	standard	none		
Date	Time	Log							
11.279	10:44:35	Statistics:							
11.279	10:44:35	Config: SSN=ZJOS Load=LPA COM=1DA6D040 WSA=00C69F90							
11.279	10:44:35	Tasks: Maj=010 EVX=01 Net=00 SCD=001 Abn=000 Prm=011							
11.279	10:44:35	Network agents: total=0002 active=0000 local=N/A							
11.279	10:44:35	Network traffic: Snd=00000000 Rcv=00000000 Que=00000							
11.279	10:44:35	JES I/F: Up=Y PIT=Y Conn=Y Idr=Y FR(5=N,12=N,22=N)							
11.279	10:44:35	Q's: ARQ=000 SQB=000 EOT=001 RMG=000 RML=010 QTR=000							
11.279	10:44:35	State: NORMAL Parm: SYS=00 EMS=00 SCD=02 DEST=00							
11.279	10:44:35	SCD: Lib=0 O=EVX M=EVX Pos=SCHEDULE-SCT EnQ=FREE							
11.279	10:44:35	SCD Free-pool: SCT=001976 TRG=0058898 EOT=0059417							
11.279	10:44:35	SCD Used-pool: SCT=001024 TRG=0001102 EOT=0000583							
11.279	10:44:35	SCD Curr-pool: SCT=000389 TRG=0000439 EOT=0000583							
11.279	10:44:35	SCD: Exec=000313 Wait=000000 Void=000071 Rest=000076							
11.279	10:44:35	SCD: Cur=000 Max=001 S=11278/01:24:17 Elaps=22:35:43							
11.279	10:44:35	SCD: TRV=00000 Max peak:(SCD=0080 RMG=0002 EVX=0001)							
11.279	10:44:35	Exits: EOI=(Up,ACTIV) WTO=(Up,ACTIV) DSM=(Up,ACTIV)							
11.279	10:44:35	MSG:(RMG=Y, EVX=Y, SCD=Y) Trace:(SSI=N, XDI=N) Timer=N							
11.279	10:44:35	Parmlib=NSI.ZJOS219.PARMLIB							
11.279	10:44:35	LDS=ZJOS.PUSPA.LDS,Vol=Z19SHR							
11.279	10:44:35	Current schedule table requires structure reformatting.							
11.279	10:44:35	Your (DERU ) authorities: Oper=Y Setting=Y Update=Y.							
11.279	10:44:35	Genlevel=20100101 CPUid=00AAF4/2098 System=NSILAB /SYS1							

Figure 7.1: zJOS status displayed in control panel

The heading of zJOS control panel is a list of product status, as shown in figure 7.1, which is updated each time enter-key is stroked. The body of panel is a list of logs which is a scrollable region to contain respond messages against request you have just entered. By default, if no message is logged, logs will contain most recent internal statistics. Hence, when you just hit enter-key on the zJOS control panel, complete status information is displayed.

Although zJOS control panel runs in privileged mode, all ISPF and PDF features not impacted. The only thing you should understand is that zJOS control panel



does not support split screen. Means, when zJOS control panel is invoked while ISPF in split screen mode, or you split the screen while zJOS control panel is in session, you won't be able to invoke zJOS control panel again before previously invoked zJOS control panel is terminated. When you try, you will get message DERISC801E.

Since control panel accesses zJOS control blocks, the panel won't available until zJOS up. However, once zJOS up, control panel will available even if zJOS is brought down. Because, once zJOS up, it builds all data area and control blocks in ECSA and leave them remain when it down. When zJOS is started back up, it reclaims all data area and control blocks it has been built in previous session.

## 7.1. Starting and Stopping zJOS

The zJOS control panel provides facilities to start and shutdown zJOS address space. These facilities are listed in action-bar menu of zJOS control panel. Note that this facilities not applicable before zJOS was first time started since IPL. Once zJOS up for the first time, these facilities then available for use. Each particular facility will follow the situation. When zJOS is up, only shutdown facility is available. When zJOS is down, only startup facility is available.

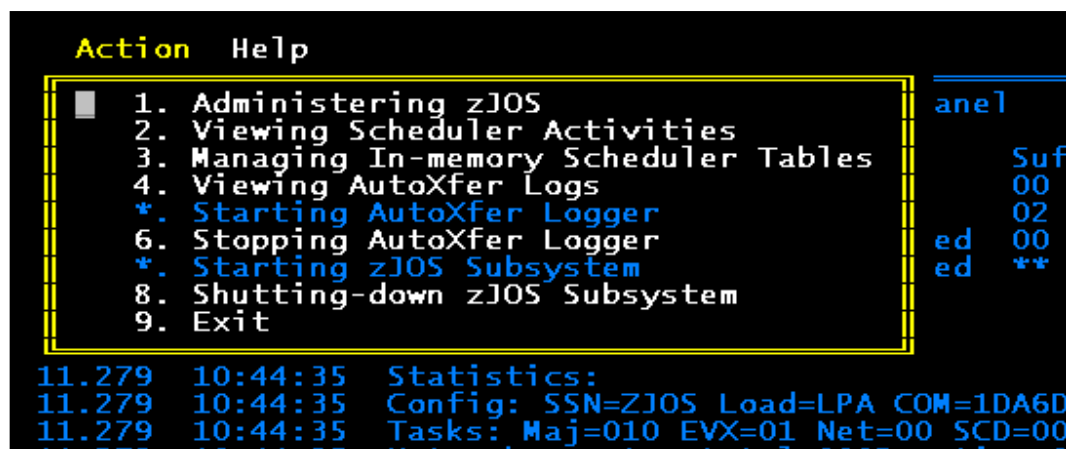


Figure 7.2: Action-bar menu when zJOS up

Figure 7.2 shows action-bar menu while zJOS up. Only available facilities are highlighted and numbered. Darkened unnumbered options indicate that related facilities are unavailable at the moment.

To shutdown zJOS, select option 8 and press enter-key, or place cursor on the position of option 8 and press enter-key. Then confirmation window is popped up as shown in figure 7.3, asking whether you sure to bring zJOS down. To abort this confirmation, type "N" and press enter-key, or just press either F3 or F12.





```
Log
35  zJOS-XDI 2.1.9
35  zJOS Shutdown Confirmation
35  You are about to shutdown zJOS
35  Are you sure? . . N
35  Cmd ==>
35  SCD Used-pool: SCT=001024 TRG=0001102
35  SCD Curr-pool: SCT=000389 TRG=0000439
35  SCD: Exec=000313 Wait=000000 Void=000071
```

Figure 7.3: Confirmation window is popped up when zJOS shutdown is selected

To confirm, type “Y” and press enter-key. This will cause the same effect as when you issue command **MODIFY XDI,SHUTDOWN** on z/OS console. zJOS then bring all internal and external subtasks, close all accessed dataset, cleanup DIV and dataspace and all necessary interfaces, hold all its subsystem functions and then terminate and lets system delete its address space.

Cleaning up DIV, dataspace and all interfaces take some amount of time. You should not disturb the panel until zJOS main task is terminated. Otherwise, it will take longer time to respond the panel. zJOS main task termination is indicated by the following message:

```
DERRMG717I zJOS is down. Sekar, Puspa and AutoXfer are not available.
```

which is issued by zJOS resource manager when address space is purged.

When zJOS is down, status information is displayed on control panel as shown in figure 7.4. All product states are down except for Sekar, stated as IDLE. This because Sekar major task runs on zJOS subsystem which always exist once it was generated. IDLE indicates that subsystem still exists and active, but, all its function routines are held.

Anything issued while zJOS is down will be aborted. As shown in figure 7.4, in log appears message “**Request is aborted! zJOS subsystem does not exist.**”. It doesn’t mean that zJOS subsystem is gone. Rather, control panel unable to contact zJOS subsystem, because all its subsystem functions are held. This is neither a product error nor a bug. This is actually set by design to simulate as it were disappear. Once zJOS is brought back up, all held functions are released and contactable.



```

Action Help
zJOS-XDI Control Panel Row 1,019 to 1,037 of 1,037

Command Product State Table Suf Works Usage Day
Sekar (EMS) IDLE PASSIVE unloaded 00 000000 LCNSD/YR 0816
Puspa (SCD) DOWN NONE loaded 02 000001 LCNSD/YR 0816
AutoXfer DOWN NONE unloaded 00 000000 LCNSD/YR 0816
Net Server DOWN NONE unloaded ** 000000 standard none

Date Time Log
11.279 11:02:55 Request is aborted! zJOS subsystem does not exist.
11.279 11:02:55
11.279 11:02:55 Statistics:
11.279 11:02:55 Config: SSN=zJOS Load=LPA COM=1DA6D040 WSA=00C69F90
11.279 11:02:55 Tasks: Maj=000 EVX=00 Net=00 SCD=000 Abn=000 Prm=011
11.279 11:02:55 Network agents: total=0000 active=0000 local=N/A
11.279 11:02:55 Network traffic: Snd=000000000 Rcv=000000000 Que=000000
11.279 11:02:55 JES I/F: Up=Y PIT=Y Conn=N Indr=N FR(5=N,12=N,22=N)
11.279 11:02:55 Q's: ARQ=000 SQB=000 EOT=001 RMG=000 RML=000 QTR=000
11.279 11:02:55 State: NORMAL Parm: SYS=00 EMS=00 SCD=02 DEST=00
11.279 11:02:55 SCD: Lib=N O=EVX M=EVX Pos=MAIN-SHUTDN EnQ=FREE
11.279 11:02:55 SCD Free-pool: SCT=001976 TRG=0058898 EOT=0059417
11.279 11:02:55 SCD Used-pool: SCT=001024 TRG=0001102 EOT=0000583
11.279 11:02:55 SCD Curr-pool: SCT=000389 TRG=0000439 EOT=0000583
11.279 11:02:55 Exits: EOI=(-,ACTIV) WTO=(-,ACTIV) DSM=(-,inact)
11.279 11:02:55 MSG: (RMG=Y, EVX=Y, SCD=Y) Trace: (SSI=N, XDI=N) Timer=N
11.279 11:02:55 Parmlib=NSI.ZJOS219.PARMLIB
11.279 11:02:55 Your (DERU ) authorities: Oper=Y Setting=Y Update=Y
11.279 11:02:55 Genlevel=20100101 CPUid=00AAF4/2098 System=NSILAB /SYS1
***** Bottom of data *****

```

Figure 7.4: Appearance of control panel when zJOS is down

To bring zJOS back up, go to action-bar again. Now, the appearance of action-bar menu is changed as shown in figure 7.5. Most of options are darkened and unnumbered, except for option 1, 7 and 9.

```

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

11.279 11:02:55
11.279 11:02:55 Statistics:
11.279 11:02:55 Config: SSN=zJOS Load=LPA COM=1DA6D0
11.279 11:02:55 Tasks: Maj=000 EVX=00 Net=00 SCD=000

```

Figure 7.5: Action-bar menu when zJOS down

To bring zJOS back up, select option 7 by typing “7” in option field or position the cursor in option 7 row, then press enter-key. Then startup information window is popped up as shown in figure 7.6.



```
Log
02:55 zJOS-XDI 2.1.9 exist.
02:55 zJOS Start Options
02:55 Procedure name . . XDI 00C69F90
02:55 Subsystem name . . zJOS 0 Prm=00
02:55 Parameter suffix 00 1=N/A
02:55 Start option Que=0000
02:55 2 1. Warm 2=N, 22=N
02:55 2. Cold (reload parmlib) 0 QTR=00
02:55 3. Refresh memory DEST=00
02:55 Command ==> EnQ=FRF
02:55 Exits: E0J=(--,ACTIV) WTO=(--,ACTIV) DSM=(--,inacti
02:55 MSG=(PMG-Y,ENV-Y,SCD-Y) T=000058
02:55 T=000058
```

Figure 7.6: Startup information window is popped up when zJOS start is selected

You are requested to fill up all necessary zJOS startup information, include name of procedure, subsystem, selected parameter suffix and start option. Popped up window provides all default values based on recently used information. You may change them as necessary. When complete, then press enter-key.

Procedure name must be a valid name of member of current PROCLIB dataset (normally SYS1.PROCLIB) which is assigned to contain zJOS started task procedure JCL. You are strongly recommended to use "XDI", since it is product default.

Subsystem name is a name you have already assigned to zJOS subsystem at the first time started. You should reuse that name. Otherwise, the previous name becomes orphaned subsystem which could potentially disturb z/OS internal logic mechanism. If you think necessary to change the subsystem name, you should consult with XDI support personnel to make sure all are on the right track.

Parameter suffix is 2-digit numeric xx to select zJOS system parameter member XDISYSxx. Make sure the member you selected exist. Otherwise zJOS startup will be aborted.

Start option is the way zJOS startup is to be performed. Normal way is WARM, and is recommended. zJOS just bring all processes up and ready. Parameters are just checked, instead of reloaded. If you want zJOS to reload all parameters, for example when some were updated, then you should select COLD. Take a note that you should never use option 3 (refresh memory) unless recommended by XDI support personnel. This option is for maintenance purpose only. This is to purge all zJOS modules in LPA, then reload them back from zJOS LOADLIB dataset.



When complete, then press enter-key. zJOS startup will be performed soon, but, it will take some amount of time. You should disturb the panel while startup is in progress.

Unless you fill wrong information, panel will send the correct startup command to z/OS. Hence, to avoid mistake in typing command which is potentially causing serious problem, you are strongly recommended to always use control panel to start zJOS.

## 7.2. Issuing Sekar Command

All zJOS console commands (including Sekar) are applicable on zJOS control panel. The panel provides a command slot for each zJOS product as shown in figure 7.7.

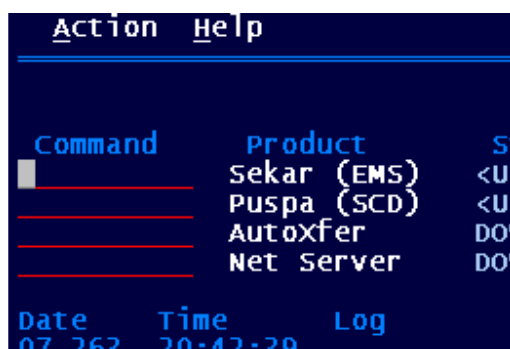


Figure 7.7: Command slots area of control panel

To issue Sekar command, you don't need to enter complete command text. Instead, just enter valid request in Sekar command slot and press enter-key. For example, to reload Sekar parameters (EMS table), in console you need to issue either `.AUTO RELOAD` or `F XDI,AUTO RELOAD`.

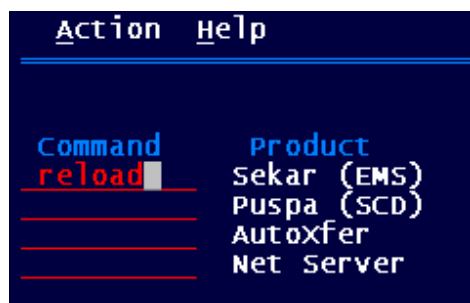


Figure 7.8: Entering RELOAD request to Sekar

The above commands actually ask console to send RELOAD request to Sekar in zJOS address space named XDI. Using control panel, you just need to enter



**RELOAD** in Sekar command slot and press enter-key as shown in figure 7.8. The panel will internally contact Sekar via cross-memory service to do the request.

Action	Help	zJOS-XDI Control Panel				
Command	Product	State	Table	Suf	works	
	Sekar (EMS)	<UP> ACT(SS)	loaded	99	000000	
	Puspa (SCD)	<UP> READY	loaded	00	000000	
	AutoXfer	DOWN INACTIVE	unloaded	00	000000	
	Net Server	DOWN INACTIVE	N/A	**	000000	

Figure 7.9: Request to Sekar to reload different EMS table

To reload different EMS table, you just need to enter (overtyping) 2-digit suffix of XDIEMSxx member you want to reload in column Suf and press enter-key. You don't even need to enter anything in command slot. Figure 7.9 illustrates how to request Sekar to load XDIEMS99 replacing current EMS table. This equivalent as issuing either **.AUTO RELOAD TAB=99** or **F XDI,AUTO RELOAD,TAB=99**.

Reload or replace (load different suffix) EMS table request will always be accepted by Sekar. All timer tasks are then brought down prior to reload EMS table. All current event listeners are also brought down as well. During reload process, all Sekar functions are disabled. Upon completion of reload process, all those tasks are brought back up with newly loaded parameters. No outstanding process is carried over afterward. Hence you should really understand the impact.

Sekar doesn't have many commands to control, since it is an automation expert. Other Sekar commands are START and STOP request. However, as mentioned many times in earlier chapters, you should not stop Sekar, unless for very certain case which is recommended by XDI support personnel.

Be aware that control panel is only for zJOS administrator. Allowing unauthorized people accessing this panel will potentially cause a serious problem.

## 7.3. Obtaining Helps

Some help panels and messages are provided to give you direct assistance while you are facing zJOS control panel. In most cases, you need to press F1 to obtain help information. Some unprotected fields, however, are sensitive against F1 interruption. Hence you have to understand where you place the cursor prior to pressing F1 to obtain help information as you expect.



## 7.3.1 Common Help Information

Common help information is a simple description which involves the whole parts of the current panel or even the component. Such help is provided as a panel. You can obtain common help information either from help-bar menu (as shown in figure 7.10) or by hitting F1 with cursor placed outside sensitive fields. Option 1 of help-bar menu is to obtain brief information or tutorial regarding zJOS solution. This explains you information about Sekar, Puspa and AutoXfer briefly. To navigate (scroll) the panel, you have to use F7 and F8. To exit and return to the control panel, you can either use enter-key, F3 or F12.

```

Action  Help
-----
1. zJOS-XDI Tutorial
2. About zJOS-XDI(R)

Control Panel      Row 242 to 255
-----
Comman  Table      Suf  Works  Usage  D
Sekar (EMS)  <UP> ACT(SS) loaded  00  000004 LICENSED n
Puspa (SCD)  <UP> ACTIVE  loaded  00  000013 **DEMO** .
AutoXfer     DOWN INACTIVE unloaded  00  000000 **DEMO** .
Net Server   DOWN INACTIVE  N/A    **  000000 standard n

Date      Time      Log
07.264    02:15:47  Request to Puspa (scheduler) was completely done.
07.264    02:16:01
07.264    02:16:01  Statistics:
07.264    02:16:01  Config: SSN=XDI Load=LPA COM=0802A3A0 WSA=00C42F90
07.264    02:16:01  Subtasks: Major=009 EVX=000 SVR=000 SCD=001 Abn=000
07.264    02:16:01  Network agents: total=0000 active=0000 local=N/A
07.264    02:16:01  Network traffic: snd=00000000 rcv=00000000 que=000000
07.264    02:16:01  JES I/F: Up=Y PIT=Y Conn=Y Irdr=Y FR(5=N,12=Y,22=N)
07.264    02:16:01  Queues: ARQS=00001 SQBS=00000 EOTS=00001 RMG=00000
07.264    02:16:01  State: NORMAL Parm: SYS=00 EMS=00 SCD=00 DEST=00
07.264    02:16:01  SCD: Lib=0 O=EVXMS M=EVXMS Pos=SCHEDULE-SCT EnQ=FREE
07.264    02:16:01  SCD Free-pool: SCT=009588 TRG=0199479 EOT=0499963
07.264    02:16:01  SCD used-pool: SCT=000412 TRG=0000521 EOT=0000037
07.264    02:16:01  SCD Curr-pool: SCT=000024 TRG=0000047 EOT=0000037
***** Bottom of data *****

```

Figure 7.10: Help-bar menu

Other common help is zJOS console help, which explaining you how to operate zJOS control panel. This help is obtained by pressing F1 with cursor position outside F1-sensitive fields. Figure 7.11 shows appearance of this help panel window. Since this help screen is larger than its window size, so the window is scrollable. To navigate the scrolling, however, you can only use enter-key, instead of F7/F8. Scrolling is wrap-around and only one-way direction, which is forward direction. To exit and return to the control panel, you can either use enter-key, F3 or F12.



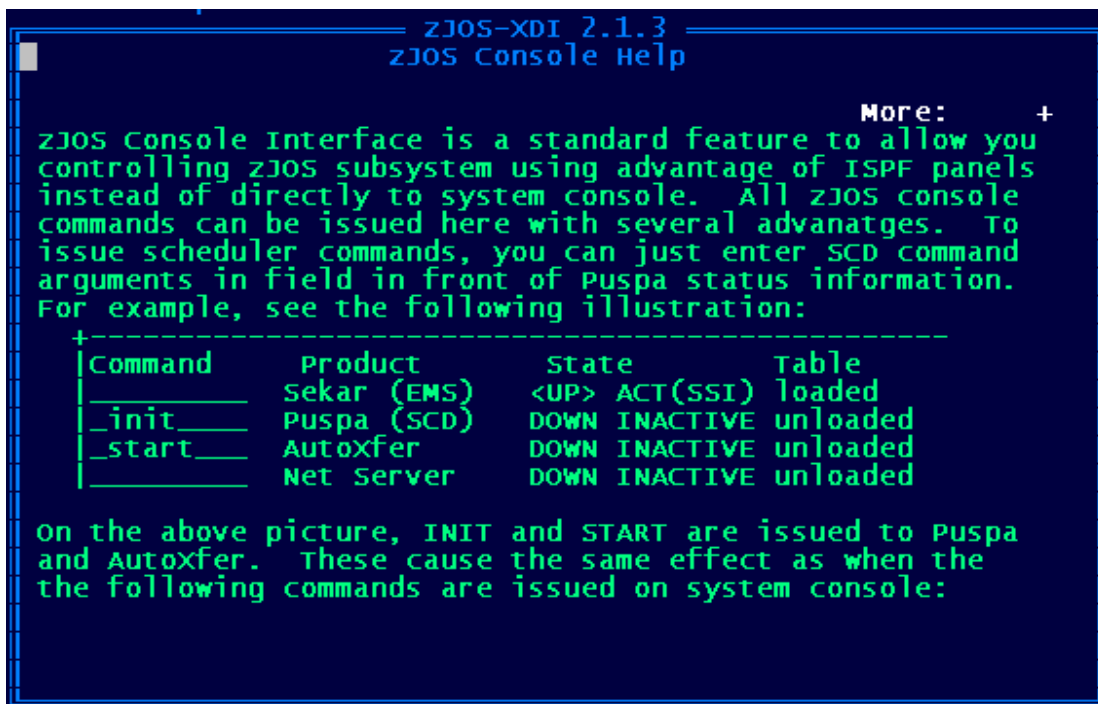


Figure 7.11: Help panel

## 7.3.2 Field-specific Help Information

There are 2 field-specific help groups, which are command slots area (command column) and parameter-suffix area (suf column). Help information only explains selected field specifically. To obtain such help information, place the cursor on certain field you expect to be explained, then hit F1 key.

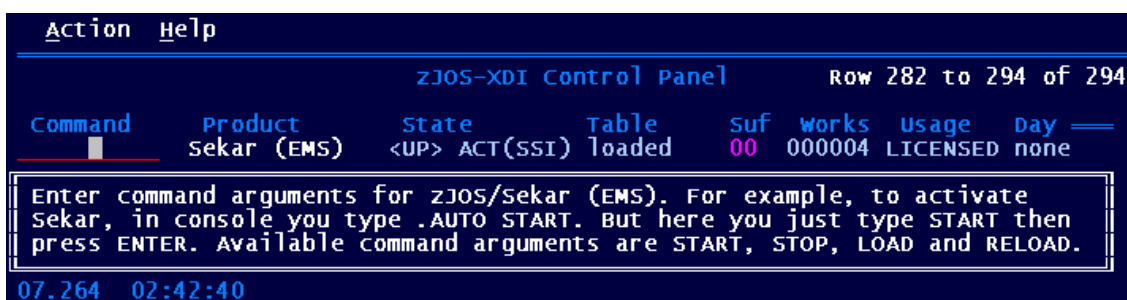


Figure 7.12: Help message for Sekar command slot

Help information is displayed using ISPF message instead of panel, and popped up in relative position to the addressed field. Figure 7.12 shows help message for Sekar command slot. Figure 7.13 shows help message for Sekar EMS table suffix.



Action Help		zJOS-XDI Control Panel				Row 295 to 307 of 307	
Command	Product	State	Table	Suf	works	Usage	Day
	Sekar (EMS)	<UP> ACT(SSI)	loaded	00	000004	LICENSED	none
	Puspa (SCD)	<UP> ACTIVE	loaded				
	AutoXfer	DOWN INACTIVE	unloade				
	Net Server	DOWN INACTIVE	N/A				
Date	Time	Log					
07.264	03:35:37	Statistics:					
07.264	03:35:37	Config: SSN=XDI Load=LPA COM					
07.264	03:35:37	Subtasks: Major=009 EVX=000 S					
07.264	03:35:37	Network agents: total=0000 active=0000 local=N/A					

Shown is a suffix of currently active EMS table. To alter it, just overtype it then press ENTER. To reload the same table, enter RELOAD in command field.

Figure 7.13: Help message for Sekar EMS table suffix



## Chapter 8 Commands and Messages Reference

### 8.1. Sekar 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)

#### 8.1.1 Entering Command via zJOS Subsystem

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

```
prefixAUTO request
```

or for more specific is:

```
.AUTO request
```

or

```
XDI AUTO request
```

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

#### 8.1.2 Entering Command via MODIFY

:

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

```
MODIFY XDI,AUTO request
```

or



**F XDI,AUTO request**

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

## 8.1.3 Entering Command via zJOS Control Panel

The second alternative to issue Sekar command is via control panel. This facility is available on TSO/E terminal while in ISPF session as discussed in chapter 7. To issue command, just enter the request on Sekar command slot as shown in figure 8.1. See chapter 7 for further explanation.

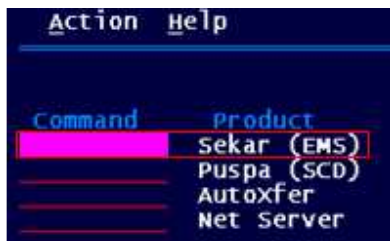


Figure 8.1: Sekar command slot on zJOS control panel

## 8.2. Sekar Commands Reference

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

### 8.2.1 LOAD request

Syntax on console:

**LOAD**

Syntax on control panel:

**LOAD**

Function:

Load EMS table onto memory.

### 8.2.2 RELOAD request

Syntax on console:

**RELOAD**

or

**RELOAD TAB=nn**

Where nn is 2-digit suffix to address EMS table XDIEMSnn member of zJOS PARMLIB

Syntax on control panel:

**RELOAD**

Function:

Reload EMS table onto memory to replace existing loaded table.

Note

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

### 8.2.3 START request

Syntax on console:

**SSI**

Syntax on control panel:

**START** or **S**

Function:

Activate EMS, which is actually activating zJOS subsystem.

Notes:

1. On console, this can only be passed via MODIFY command
2. Normally EMS is automatically activated when zJOS is brought up.



## 8.2.4 STOP request

Syntax on console:

**STOP**

Syntax on control panel:

**STOP**

Function:

Inactivate EMS, which is actually deactivating zJOS subsystem.

Note:

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

## 8.3. 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.

## 8.4. zJOS System Commands Reference

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





## 8.4.1 ASCB request

Syntax  
**ASCB**

Function:  
List all ASCBs and each with detail information.

## 8.4.2 HELP request

Syntax  
**-HELP**

Function:  
Display zJOS command reference summary on console.

## 8.4.3 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.

## 8.4.4 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:

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

## 8.4.5 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:

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

## 8.4.6 SHUTDOWN request

Syntax

**SHUTDOWN**

Function:

Bring zJOS address space down.

## 8.4.7 START request

Syntax

**START**

Function:

Bring zJOS address space up.

Notes:

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



## 8.4.8 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. .

## 8.5. 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.

## 8.6. 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.

### 8.6.1 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.

## 8.6.2 DISCONNECT request

Syntax

**-DISCONNECT**

Function:

Requesting disconnection (sign-off) to zJOS server.

## 8.6.3 DROP request

Syntax

**-DROP**

Function:

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

## 8.6.4 GET request

Syntax

**-GET component\_name**

Where **component\_name** is either EMS or SCD.

Function:

Requesting server to send EMS or scheduling parameters.

## 8.6.5 HELP request

Syntax

**-HELP**

Function:

Display zJOS Agent command reference summary on console.



## 8.6.6 LIST request

Syntax

**-LIST component\_name**

Where **component\_name** is either EMS or SCD.

Function:

List down EMS or scheduling parameters received from server.

## 8.6.7 START request

Syntax

**-START**

Function:

Bring up agent address space.

Note:

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

## 8.6.8 STOP request

Syntax

**-STOP**

Function:

Bring down agent address space.

## 8.7. Sekar Messages

All Sekar 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.

1. I – informational message
2. W – warning
3. E – error message
4. A – needs user action
5. 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.





## Notes