IBM Workload Scheduler
Version 8.6 (Revised March 2016)

Scheduling with the agent for z/OS

IBM Workload Scheduler
Version 8.6 (Revised March 2016)

Scheduling with the agent for z/OS

Note
Before using this information and the product it supports, read the information in "Notices" on page 199.

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

© Copyright IBM Corporation 2011, 2016.
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Figures v

Tables vii

About this publication ix
What is new in this release ix
Who should read this publication ix
Accessibility ix
Technical training ix
Support information ix

Chapter 1. Overview 1

Chapter 2. Installing and configuring . . 3
Load the agent for z/OS software 4
Run the EELINST installation aid 6
Add SMF and JES event tracking exits 11
SMF exits 12
JES2 exits 12
JES3 exits 13
Invoking the EELEXIT macro 14
Update SYS1.PARMLIB 16
Defining subsystems 16
Authorizing the load-module library 18
Updating SMF parameters 19
Updating z/OS dump options 20
Updating the z/OS link-library definition . . . 20
Starting the product automatically 21
Update RACF for the agent for z/OS started task . 21
Set up the SSL environment 22
Update SYS1.PROCLIB 23
Complete the installation 24
Starting the agent and checking the connection 25
Ensuring that all installation tasks are complete 25
Checking the message log 25
Verifying tracking events 26
Performing problem determination for tracking events 27
Recommendations for allocating the job library data set (EELJBLIB) 29
Customization parameters 29
Specifying runtime options for the event writer 30
Specifying the exit policy for the agent 30
Defining HTTP connection options 31
Specifying generic runtime options for the agent 35
Configuring the agent for z/OS exits 39
Configuring exit EELUX000 (start/stop) 39
Configuring exit EELUX002 (job-library-read) . . 40
Configuring exit EELUX004 (event filtering) . . 43
Running the agent in a sysplex environment . . . 44

Chapter 3. Using 47
Computer and workstation names of the agent . . 47
Listing the agents for z/OS 48
Defining jobs 48
Defining in the Dynamic Workload Console . . 49
Defining in composer 53
Defining the JCL 56
HFS or ZFS file triggering 57
Syntax 57
Arguments 57
Example 59
Submitting jobs 59
Using variables in your jobs 63
Variables resolved by Tivoli Workload Scheduler 63
Variables resolved by the agent for z/OS . . . 69
Managing job instances 89
Tracking jobs 89
Controlling how the event writer records job completion codes for specific jobs 93
Viewing job logs 94
Using system commands to control the agent . . . 94
Switching domain managers 96

Chapter 4. Troubleshooting and reference 99
Understanding resynchronization messages 99
Component versions must be aligned for the full current functionality 101
Saturation of DB2 transaction log halts processing of jobs 101
Data areas 102
Messages 197

Notices 199
Trademarks 201
Terms and conditions for product documentation 201

Index 203

© Copyright IBM Corp. 2011, 2016

Figures

1. The role of the agent for z/OS. 1
2. The agent for z/OS in a SYSPLEX configuration. 46
3. The General page of a new agent for z/OS definition 51
4. JCL is by reference. 52
5. JCL is by definition. 53
6. The route followed by a job within the agent for z/OS. 60
7. The route followed by a status event within the agent for z/OS as it is returned by JES on its way to IBM Workload Scheduler. 62

© Copyright IBM Corp. 2011, 2016

Tables

1. Checklist for installing the agent for z/OS 3
2. Sample library members for the agent for z/OS data sets 5
3. Agent for z/OS libraries loaded by SMP/E 5
4. Sample jobs created by the installation aid 8
5. Data sets used with the agent for z/OS. . . . 8
6. List of sample exits for event tracking. . . . 12
7. Examples of MAXECSA storage values 18
8. Required data sets for the agent for z/OS 23
9. Optional data sets for the agent for z/OS 24
10. Events generated by the agent for z/OS. 26
11. Types of missing event and relative problem determination actions 28
12. Supported variables in JSDL definitions 63
13. Properties for dynamic jobs on IBM Workload Scheduler agent for z/OS 65
14. Symbols that mark the end of variables. 72
15. Predefined job stream-related variables 73
16. Predefined job-related variables 75
17. Predefined date-related variables 75
18. Predefined dynamic-format variables 76
19. Dynamic-format substitution results 79
20. Job events and statuses as mapped by the involved components 90
21. Error codes returned after a job is submitted. 91

© Copyright IBM Corp. 2011, 2016

About this publication

This publication describes how to install, configure, use, and troubleshoot the IBM Workload Scheduler distributed - Agent for z/OS.

What is new in this release

Learn what is new in this release.

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

For information about the APARs that this release addresses, see the Tivoli Workload Scheduler Release Notes at http://www-01.ibm.com/support/docview.wss?rs=672&uid=swg27038323 and the Dynamic Workload Console Release Notes at http://www-01.ibm.com/support/docview.wss?rs=672&uid=swg27038328.

Who should read this publication

Learn the audience of this publication.

This publication is intended for users who want to use the agent for z/OS to schedule work from IBM Workload Scheduler on the JES2 or JES3 subsystem of z/OS.

Accessibility

Accessibility features help users with a physical disability, such as restricted mobility or limited vision, to use software products successfully.

With this product, you can use assistive technologies to hear and navigate the interface. You can also use the keyboard instead of the mouse to operate all features of the graphical user interface.

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

Technical training

Cloud & Smarter Infrastructure provides technical training.

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

Support information

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

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

© Copyright IBM Corp. 2011, 2016

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

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

Chapter 1. Overview

Use the agent for z/OS to schedule work from IBM Workload Scheduler on the JES2 or JES3 subsystem of z/OS.

You install and configure the agent for z/OS in the z/OS system. As soon as it is configured, the agent automatically links with the dynamic workload broker component of IBM Workload Scheduler (defined during the configuration process) through the HTTP or HTTPS protocols.

The agent requires the installation of IBM Workload Scheduler version 8.6, or later, in your environment.

With the agent for z/OS you can define jobs and schedules on IBM Workload Scheduler and submit a subset of the jobs to a z/OS system. You run the planning tasks on IBM Workload Scheduler while the execution is demanded to the z/OS system.

As the next figure shows, the agent for z/OS acts as a proxy between dynamic workload broker, which is the IBM Workload Scheduler component that actually submits workload, and JES, which is the component in the z/OS® system that executes the workload. The agent passes the workload from IBM Workload Scheduler to JES, and returns all updates about its execution back to IBM Workload Scheduler in the form of events.

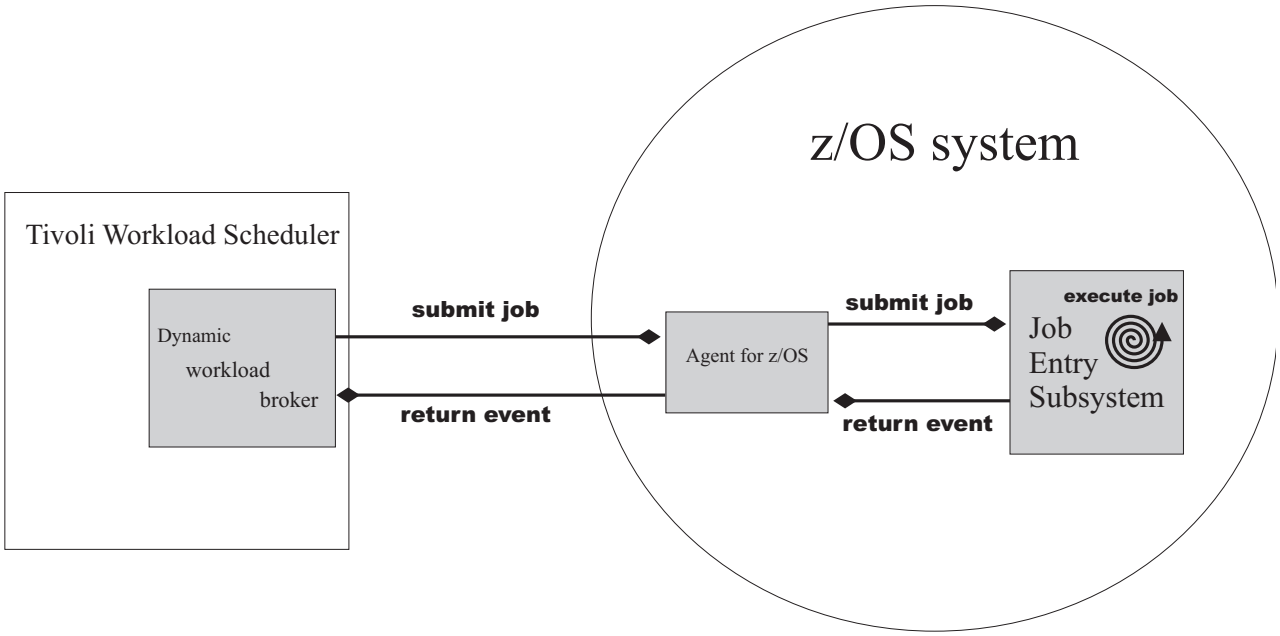


Figure 1. The role of the agent for z/OS.

The agent for z/OS represents a lightweight end-to-end scheduling solution that enables users to define and manage workload that is to be processed by JES entirely from IBM Workload Scheduler, without any need to define anything on the z/OS side.

© Copyright IBM Corp. 2011, 2016

The agent exploits the HTTP submission protocol to receive job submission requests and to asynchronously send back job status notifications. When configuring the agent, the installer is requested to specify the hostname of the dynamic workload broker to which the agent is to connect.

The jobs submitted to an agent for z/OS are similar to other IBM Workload Scheduler dynamic jobs. The job definition uses the XML syntax of the Job Submission Description Language (JSDL), where the application name is `JCL` and includes one of the following:

- The entire JCL definition of the task to be run by JES. In this case, the JCL is part of the job definition stored in the IBM Workload Scheduler database and is sent to the agent with the rest of the job at submission time. This is called submission `by definition`.
- The names of the data set and of the file containing the JCL definition in the z/OS system where the agent runs. When the job is submitted, the agent uses this information to track the JCL definition in the z/OS system. This is called submission `by reference`.

You can use the following interfaces to define these jobs:

- The composer or the dynamic workload broker command lines
- Workload Designer of the Dynamic Workload Console
- The dynamic workload broker graphical user interface

The agent supports the scheduling of jobs, but not of started tasks, on the z/OS system.

JCL tailoring (variable substitution in z/OS terms) is carried out before submitting the JCL to JES using the variable values specified in the variables tables in IBM Workload Scheduler.

You manage agent for z/OS jobs as you manage other IBM Workload Scheduler jobs. Use the Dynamic Workload Console to see them in graphical or tabular views, or in host-lists. You can also see the related job logs and carry out most actions (such as `rerun`, `cancel`, but not `kill`) typically available for other jobs.

From the Dynamic Workload Console or the IBM Workload Scheduler command line you can view the job log for as long as the job output is kept in the JES spool.

Compatibility issues between the agent for z/OS and the IBM Workload Scheduler for z/OS trackers

The agent for z/OS and the IBM Workload Scheduler for z/OS trackers can coexist as long as you run the standard IBM Workload Scheduler for z/OS tracker exits. If you are concerned about such coexistence, you must not therefore install any of the sample exits provided with the agent.

Coexistence with the Tivoli® Workload Scheduler for Applications for z/OS gateway

The agent for z/OS and the gateway can coexist on the same system.

Chapter 2. Installing and configuring

This chapter describes the tasks you need to complete to download, install, and configure the agent for z/OS software on the target z/OS system.

You install and configure the agent for z/OS in the z/OS system. As soon as it is configured, the agent automatically links with the dynamic workload broker component of IBM Workload Scheduler (defined during the configuration process) through the HTTP or HTTPS protocols.

Installation check-list

The following table summarizes the installation and setup tasks for the agent for z/OS:

Table 1. Checklist for installing the agent for z/OS

Task	Description
1	"Load the agent for z/OS software" on page 4 Run the following steps: 1. Run SMP/E to receive the agent for z/OS software. 2. Apply the agent for z/OS maintenance.
2	"Run the EELINST installation aid" on page 6 Run `EELINST` for every instance of the agent for z/OS that you want to create as soon as the agent for z/OS software is loaded. It helps you to: • Create the sample job JCL to generate tailored samples from the `EELINST` dialog. • Allocate data sets for the agent. • Add SSL certificates. • Define initialization statements creating members in the parameter library (which is identified by the EELPARM DD statement in the agent for z/OS started task).
3	Add SMF and JES event tracking exits **Note:** Run this step if you do not already have an IBM Workload Scheduler for z/OS tracker running in the system. If you have it, you are required to apply the exits related to the product latest version. For example, if you are running IBM Workload Scheduler for z/OS tracker V9.2, you must apply the exits related to IBM Workload Scheduler for z/OS V9.2. However, if you are running in a JES3 environment, you must always apply the exit IATUX09, which is provided with the agent for z/OS.
4	Update SYS1.PARMLIB Run the following tasks if they apply to your installation: • Define the agent for z/OS subsystem (`IEFSSN`*nn*). • Authorize the agent for z/OS load module library (`IEAAPF`*nn* or PROg*nn*). • Update dump-content definitions. • Update the z/OS link-library definition (`LNKLST`*nn*). • Update SMF parameters (`SMFPRM`*nn*). Run this step only if you do not already have an IBM Workload Scheduler for z/OS tracker running in the system. • Choose whether to start the agent automatically (`COMMND`*nn*).

© Copyright IBM Corp. 2011, 2016

Table 1. Checklist for installing the agent for z/OS (continued)

Task	Description
5	"Update RACF for the agent for z/OS started task" on page 21
6	Set up the SSL environment Do the following: 1. Create as many private keys, certificates, and trusted certification authority (CA) chains as you plan to use in your network. 2. Specify the SSL keywords in the `HTTPOPTS` initialization statement.
7	Update SYS1.PROCLIB Create a JCL procedure for the address space.
8	Complete the installation 1. IPL the system where you have installed the agent. 2. Verify the installation.

Load the agent for z/OS software

The first installation step is to use SMP/E to download the agent for z/OS software from the distribution tape.

To load the agent for z/OS software on your z/OS system, process the software distribution tape using the facilities of System Modification Program Extended (SMP/E). This creates or updates the necessary software libraries on your system.

To download the agent software, you can use either:

- The SMP/E dialogs.
- The sample jobs that are provided in the distribution tape. Use them with the SMP/E RECEIVE, APPLY, and ACCEPT commands. The SMP/E jobs assume that all DDDEF entries that are required for SMP/E execution have been defined in appropriate zones.

The next sections list and describe the samples provided to run the RECEIVE, APPLY, and ACCEPT commands. See the Program Directory for further details.

Setting up the environment

You can use the sample library members EELALSMP and EELALLOC to create and initialize the SMP/E environment and the agent for z/OS product libraries that are needed to support the installation and continuing maintenance of the agent.

The EELALSMP job initializes an SMP/E CSI, adding a global zone, and the agent for z/OS FMID.

The EELALLOC job allocates all Tivoli target and distribution libraries.

Using the samples for RECEIVE, APPLY, and ACCEPT processing

The following table lists the sample library members that are included in the distribution tape for your use:

Table 2. Sample library members for the agent for z/OS data sets

Sample member	Function
EELACPTE	Use to run SMP/E ACCEPT processing for the agent for z/OS data sets
EELALLOC	Use to allocate all the agent for z/OS data sets
EELAPPLE	Use to run SMP/E APPLY processing for the agent for z/OS data sets
EELDDDEF	Use to set up DD definitions for all agent for z/OS data sets
EELRECVE	Use to run SMP/E RECEIVE processing for the agent for z/OS data sets

You might need to change the distribution library and zone name to reflect those defined in the agent for z/OS CSI.

See the program directory for further details.

Results

The following table describes the distribution and target libraries that are created or updated by SMP/E.

Table 3. Agent for z/OS libraries loaded by SMP/E

SMP/E DD name		Description
Distribution	**Target**	
AEELCLIB	SEELCLIB	CLISTs
AEELDATA	SEELDATA	Default SSL certificates
AEELMAC0	SEELMAC0	Assembler macros
AEELMISC	SEELMISC	License information
AEELMOD0 (object)	SEELLMD0 (load)	Agent for z/OS modules
AEELMSG0	SEELMSG0	Messages
AEELPNL0	SEELPNL0	Panels for the EELINST installation aid
AEELSAMP	SEELSAMP	Sample exits, programs, and JCL

You should place the load modules in a separate library. Create the library before you run the SMP/E jobs.

Alternatively, you can place the new load modules in one of your existing load-module libraries, for example SYS1.LINKLIB. The remaining data sets loaded by SMP/E are new data sets that you must create before running the SMP/E jobs. The agent for z/OS program directory contains the JCL and instructions for loading the software.

After you have loaded the agent for z/OS software, apply any recommended maintenance described in the PSP bucket.

Run the EELINST installation aid

EELINST is a CLIST-driven ISPF dialog that helps you setup an agent for z/OS instance. Set EELINST up as soon as the agent for z/OS software is installed.

EELINST helps you with the installation by:

- Building the batch-job JCLs which are tailored to your requirements and that you can use for a complete installation.
- Creating the data sets used for your agent for z/OS instance.
- Defining initialization statements in the parameter library (EELPARM).
- Loading the default SSL certificates on the RACF® keystore database.
- Listing the remaining manual steps required to complete the installation.

Setting up the EELINST installation aid

EELINST reads skeleton JCLs from the SEELSAMP library, tailors the JCLs, and then writes the tailored JCLs to an output library that you specify. The components of EELINST reside in these libraries:

SEELCLIB
CLIST to drive the dialog

SEELPNL0
EELINST panels

SEELSAMP
Sample JCL

To be able to run EELINST, allocate these libraries to the DD statements in your TSO session:

- SEELCLIB to SYSPROC
- SEELPNL0 to ISPPLIB
- SEELSAMP to ISPSLIB

Running EELINST

To invoke EELINST, enter the EELINST TSO command from an ISPF environment. This panel is displayed:

```
                        Welcome to the Tivoli Workload Scheduler distributed
                                     Agent for z/OS Installation


The installation program can assist you with the following operations:


⌂ Building a batch-job JCL that is tailored to your requirements and that can be
  used for a complete installation.

⌂ Creating the data sets used for your Agent for z/OS address space.

⌂ Defining initialization statements in the parameters library (EELPARM).

⌂ Loading the default SSL certificates on the RACF keystore database.

⌂ Providing a list of all the other manual steps you must apply on the system to
  complete the installation.



Press ENTER to continue the installation. Press PF3 to quit.
```

When you press ENTER, the following panel is displayed, with some of the fields pre-filled with default values:

```
                             Create customized sample jobs                      1/5


Job statement information:
 //ZAGEINST JOB JOBNN_____________________________________________________
  _______________________________________________________________________
  _______________________________________________________________________
  _______________________________________________________________________

  Product libraries (Steplib is optional):
  Steplib library    ===> _______________________________________________
  Message library    ===> ZAGE.INST.SEELMSG0_____________________________
  Data library name  ===> ZAGE.INST.SEELDATA_____________________________

  Agent for z/OS subsystem information:
  Datasets prefix    ===> TWSSSD_____________________
  Subsystem name     ===> ZAGE          Subsystem name
  Unit name          ===> 3390___       Default unit name
  Volume serial      ===> EELVOL_       Default volume serial

  Input  library: specified in the logon procedure (ISPSLIB)
  Output library:
  Output dsn name    ===> _______________________________________________

  SYSOUT class       ===> *             SYSOUT class for reports

  Press ENTER to continue, PF12 for previous panel, PF3 to quit.
```

The length of the subsystem name can be at most 4 characters.

The length of the data set prefix can be at most 26 characters.

When you press ENTER, the following sample jobs are created in the *Output dsn name* data set:

Table 4. Sample jobs created by the installation aid

Sample name	Description
EELAGT	Generates an agent for z/OS started task procedure sample.
EELAGTP	Generates default parameters for the agent for z/OS started task procedure sample.
EELALLDS	Allocates the data sets used for the agent for z/OS started task.
EELFLWAS	Calls the filewatch utility used to check HFS or ZFS files.
EELJER2V	Restores the JES2 exits usermod.
EELJER3O	Restores the EELUX091 JES3 usermod.
EELJER3U	Restores the EELUX091 and EELUX191 as JES3 usermods.
EELJES2	Assembles and link-edits the JES2 EXIT7.
EELJES2U	Installs the JES2 EXIT7 usermod.
EELJES2V	Installs the JES2 EXIT51 usermod.
EELJES21	Assembles and link-edits the JES2 EXIT51.
EELJES3	Assembles and link-edits the JES3 exits.
EELJES3O	Installs the EELUX091 JES3 usermod.
EELJES3U	Installs the JES3 usermod.
EELRCERT	Copies default certificates for SSL communication (EELCERCL member in library MISC) to RACF.
EELRETWT	Sample program to simulate abends, return codes, and waits.
EELRMD	A readme listing the successive manual steps.
EELSMF	Assembles and installs the SMF exits.

Then, the following panel is displayed:

```
                     Create the data sets used for the Started task (EELALLDS)         2/5

The data sets in EELALLDS will be named:
DATASET PREFIX.SUBSYSTEM NAME.NAME

SUBMIT/EDIT JOB    ===> E           S to submit JOB, E to edit
                                    N to skip to the next panel


Press ENTER to continue the installation. Press PF3 to quit.
```

After submitting `EELALLDS`, all the data sets for the new started task are available. They are:

Table 5. Data sets used with the agent for z/OS.

Data set allocated by EELALLDS	DD name	Dataset description
-	EELBRDS	Internal reader.

Table 5. Data sets used with the agent for z/OS. (continued)

Data set allocated by EELALLDS	DD name	Dataset description
dataset_prefix.subsystem_name.EELDUMP *dataset_prefix.subsystem_name*.SYSDUMP	EELDUMP SYSDUMP	Diagnostic data sets where the agent for z/OS writes debugging information when validity checking discovers internal error conditions.
dataset_prefix.subsystem_name.EV *dataset_prefix.subsystem_name*.HT *dataset_prefix.subsystem_name*.HTREF	EELEVDS EELHTDS EELHTREF	Event data sets containing records that describe events created by the job tracking functions of the agent for z/OS. EELHTDS records events coming from dynamic workload broker, while EELEVDS records events created by JES and SMF that are to be transmitted to dynamic workload broker. EELHTREF is a service data set used for jobs submitted `by reference` or where the JCL requires variable resolution.
-	EELJBLIB	Job library data set for the JCLs submitted `by reference`. When the data set name is not specified in an agent for z/OS job definition, the agent searches EELJBLIB for the member name with which the JCL was saved.
-	EELMLIB	Messages library.
dataset_prefix.subsystem_name.MLOG	EELMLOG	Message log.
dataset_prefix.subsystem_name.PARM	EELPARM	Parameters library containing initialization statements that define runtime options for the agent for z/OS subsystem.

A sample copy of the agent started task (for example, `AGT1`) is inserted in the output data set.

Check the job output to make sure this step has completed before you continue with the following steps.

The next step inserts the default initialization statements in the parameter library used by the agent started task.

```
                  Define initialization statements in the parameters library        3/5


INITIALIZATION PARAMETERS ===> AGT1P___     Member in the EELPARM library
CREATE/EDIT PARAMETERS    ===> E            C to create, E to edit

Press ENTER to continue the installation. Press PF3 to quit.
```

The length of the member name in the EELPARM library can be at most 8 characters.

Select E to define the initialization statements with valid values at this time, or C to just create the statements and leave their functioning definition for a later time (this will require that you restart the agent). See "Customization parameters" on page 29 for additional information on the initialization statements.

The next panel gives you the option to import the default SSL certificates into the RACF keystore data set.

```
                                                                          More:     +
                  Run the EELRCERT job to import the default certificates to RACF      4/5


  To use them, configure the HTTPOPTS initialization statement to activate the SSL
  communication, setting parms TDWBSSL and/or SSL and add the following configurations:
  SSLKEYRINGTYPE(SAF)
  SSLKEYRING(EELRING)

  Note: If you already run default certificates with Tivoli Workload
        Scheduler for z/OS, skip this step and configure accordingly.

 SUBMIT/EDIT JOB    ===> N           S to submit JOB, E to edit
                                     N  to skip to the next panel
```

If in the same system you run also IBM Workload Scheduler for z/OS and the related default SSL certificates, skip this step but configure the SSLKEYRING parameter with the value already used for that scheduler. If you fail to do this, the submission of EELCERT has no effect and returns an RC4 error code.

At the end, a panel that lists all the remaining steps you have to manually run on your z/OS environment to complete the installation is displayed:

```
          List of additional manual steps required to complete the installation       5/5

o Add SMF and JES event tracking exits.
  Complete this step only if you do not already have an IBM Workload Scheduler for z/OS
  tracker active on the z/OS system. Otherwise, see the Scheduling with the Agent for z/OS
  manual.

o Update SYS1.PARMLIB.
  - Define the Agent for z/OS subsystem (IEFSSNnn).
  - Authorize the Agent for z/OS load module library (IEAAPFnn or PROgnn).
  - Update dump-content definitions.
  - Update the z/OS link-library definition (LNKLSTnn).
  - Update SMF parameters (SMFPRMnn). Apply this change only if you do not already run
    a Tivoli Workload Scheduler for z/OS tracker on the z/OS system.
  - Choose whether to start the Agent for z/OS automatically (COMMNDnn).

o Set up the RACF environment.

o Set up the SSL environment.
  - Create as many private keys, certificates, and trusted certification authority
    (CA) chains as you plan to use in your network.
  - Configure the scheduler, by specifying the HTTPOPTS statement for each
    component of your network.

o Update SYS1.PROCLIB.
  - Create a JCL procedure for the z/OS address space.

o Activate TCP/IP connections.
  - Add TCP/IP network definitions. Define an IP address for the Agent for z/OS.
  - Add TCP/IP initialization options. Include initialization statement options in
   the parameters library for all the Agent for z/OS started tasks.

o Complete the installation.
  - IPL each system where you have installed the Agent for z/OS.
  - Verify the installation.

This information is also listed in the §EELRMD member of
the output data set

Press ENTER to complete, PF12 to return to the previous panel, PF3 to quit.
```

The steps are described in the following sections.

Add SMF and JES event tracking exits

The agent for z/OS tracks the progress of jobs through the z/OS system by using JES2, JES3, and SMF exit points. Add these exits on the z/OS system where you installed the agent, unless you already run a tracker of a supported IBM Workload Scheduler for z/OS version (in this case, see “JES2 exits” on page 12 and “JES3 exits” on page 13).

Several sample event-tracking exits, that simplify the installation of event tracking, are available in the SEELSAMP sample library. To assemble and install the exits, you can use the sample JCL provided to install the exits as SMP/E usermods, or you can assemble and link-edit the exits yourself. For JES exits, apply usermods in the CSI where JES is included: this is the best method. It has the advantage that SMP automatically reassembles the exits if maintenance is applied to the JES control blocks that the agent for z/OS depends upon.

The sample exits all use the EELEXIT macro to create event-generating code. See “Invoking the EELEXIT macro” on page 14 for more information.

The following table describes the samples that you can use to generate and install the exits. The sample exit, skeleton JCL, and usermod entries identify the members in the SEELSAMP library.

Table 6. List of sample exits for event tracking

Exit name	Exit type	Sample exit	Sample JCL/ usermod	Event supported
IEFACTRT	SMF	EELACTR1	EELSMF	Job and step completion
IEFUJI	SMF	EELUJI1	EELSMF	Job start
IEFU83	SMF	EELU831	EELSMF	Job Print end
EXIT7	JES2	EELXIT74	EELJES2/ EELJES2U	JCT I/O exit for JES2, purge
EXIT51	JES2	EELXIT51	EELJES21/ EELJES2V	JES2 QMOD phase change exit, z/OS V1.7 or later
IATUX09	JES3	EELUX091	EELJES3/ EELJES3O/ EELJES3U	On job queue
IATUX19	JES3	EELUX191	EELJES3/ EELJES3U	Output processing complete

SMF exits

This section provides details about adding the SMF event tracking exits used by the agent for z/OS.

You must tailor the sample JCL to the requirements of your installation. You can copy any of the members from the SEELSAMP library to one of your own libraries and manually tailor the JCL.

If you are unfamiliar with how to activate SMF exits, see "Updating SMF parameters" on page 19 and the documentation for SMF.

JES2 exits

This section provides details about adding the JES2 event tracking exits used by the agent for z/OS. However, if you have the agent for z/OS and IBM Workload Scheduler tracker running on the same system, the JES2 exits are already installed.

The EELSAMP sample library contains a number of members that you can use to assemble and link-edit JES exits. EELJES2 and EELJES21 provide sample JCL to assemble and link-edit the JES2 exits. However, you are encouraged to use members EELJES2U and EELJES2V. These samples provide the JCL to install the JES2 exits as SMP/E usermods. The usermods are defined so that both the JES and the agent for z/OS target zones are informed of the dependencies. This ensures that future maintenance to either component (JES2 or the agent for z/OS) will be handled correctly.

The sample EELJER2V is provided to reject and restore the JES2 exits as SMP/E usermods, if needed.

The load modules of the JES2 exits, which are EXIT7 and EXIT51, are called TWSEXIT7 and TWSXIT51, and their entry points are called TWSENTR7 and TWSENT51, respectively.

The sample library member EELXIT74 contains the assembler source code of a JES2 JCT I/O exit, JESEXIT7. EELXIT74 is used for JES2. The agent for z/OS uses JESEXIT7 to detect new jobs on the internal reader and also to detect output group purge.

The sample library member EELXIT51 contains the assembler source code of the JES2 QMOD Phase Change exit, JES2 EXIT51. The agent for z/OS uses JES2 EXIT51 to detect job errors occurring during the JES2 input phase.

Include these records in the JES2 initialization member:

JES2 Initialization Statements

Add the following records to the JES2 initialization member:

```
Load agent for z/OS exit mod */
EXIT(7) ROUTINES=TWSENTR7,STATUS=ENABLED /*
Define EXIT7 entry point */
```

And also:

```
LOAD(TWSXIT51) /*
Load agent for z/OS exit mod */
EXIT(51) ROUTINES=TWSENT51,STATUS=ENABLED /*
Define EXIT51 entry point */
```

To dynamically install the JES2 exits, use these commands when the modules are available in the LNKLST:

```
$ADD LOADMOD(TWSEXIT7),STORAGE=PVT
$T EXIT(7),ROUTINES=TWSENTR7,
 STATUS=ENABLED
$ADD LOADMOD(TWSXIT51),STORAGE=PVT
$T EXIT(51),ROUTINES=TWSENT51,
 STATUS=ENABLED
```

To put a new version of an exit (that was previously installed) in place, use these commands when the modules are available in the LNKLST:

```
$TLOADMOD(TWSEXIT7),REFRESH
$TLOADMOD(TWSXIT51),REFRESH
```

For more information about JES2 initialization statements, see *JES2 Initialization and Tuning Reference*.

JES3 exits

This section provides details about adding the JES3 event tracking exits used by the agent for z/OS.

If you are running in a JES3 environment ensure that you have applied APAR PI47613, which supports JES3 for the agent for z/OS. If you also have an IBM Workload Scheduler for z/OS tracker running in the same system, ensure that you applied APAR PI57699 to grant JES3 compatibility on the tracker.

Note: The exit IATUX29 is provided and used only by the IBM Workload Scheduler for z/OS tracker. The exit IATUX09 is provided and used only by the agent for z/OS. The exit IATUX19 is provided and used by both the tracker and the agent for z/OS.

The EELSAMP sample library contains a number of members that you can use to assemble and link-edit JES exits. EELJES3 provide sample JCL to assemble and link-edit the JES3 exits. However, you are encouraged to use members EELJES3O or EELJES3U. This samples provide the JCL to install the JES3 exits as SMP/E usermods. The usermods are defined so that both the JES and the agent for z/OS target zones are informed of the dependencies. This ensures that future maintenance to either component (JES3 or the agent for z/OS) will be handled correctly.

The sample EELJES3U is provided to receive and apply both the IATUX09 and IATUX19 usermods. The sample EELJES3O is provided to receive and apply only the IATUX09 exit usermod. The sample EELJER3U is provided to reject and restore both the IATUX09 and IATUX19 usermods, if needed. The sample EELJER3O is provided to reject and restore only the IATUX09 exit usermod, if needed.

To activate the exits for a JES3 system, you can link them to a library that is concatenated ahead of SYS1.JES3LIB. Alternatively, you can replace the existing exits in SYS1.JES3LIB with the Agent for z/OS-supplied IATUX09 and IATUX19 exits. For more information, see *JES3 Initialization and Tuning Reference*.

If you get RC=4 and the warning `ASMA303W Multiple address resolutions may result` when you assemble IATUX19 running the JES3 usermod samples, you can ignore the message. If version ASMA90 of the compiler reports errors, and the RMODE=ANY statement is defined, remove the RMODE=ANY statement from the sample exit.

Invoking the EELEXIT macro

The sample event tracking exits shipped with the agent for z/OS are written in assembler language. The event tracking code in these exits is generated by an assembler macro called EELEXIT. The following sections describe how to invoke the EELEXIT macro.

Invoking EELEXIT in SMF exits

EELEXIT establishes its own addressability in SMF exits. It saves and restores all used registers. To do this, it expects Register 13 to point to a standard z/OS save area.

There are two ways to invoke the EELEXIT macro in an SMF exit:

- Invoke EELEXIT with all registers unchanged since the exit was called (except Register 15).
- Save all registers on entry to the exit and then invoke EELEXIT by specifying the address of the initial save area.

In both cases, EELEXIT must be invoked in Supervisor state, PSW key 0.

Invoking EELEXIT in JES exits

In JES exits, EELEXIT must be invoked in Supervisor state, PSW key 1. EELEXIT expects code addressability to be already established. It also expects registers to be set up as follows:

- **EXIT7**

R0 JCT read/write indicator (JES2 SP Version 3 and earlier); address of a parameter list mapped by the JES2 $XPL macro (JES2 SP Version 4 and later).

R1 Address of the JCT being read or written.

R13 Address of the current PCE.

- **EXIT51**

  R1 Address of a parameter list mapped by the JES2 $XPL macro (JES2 with z/OS v1.7, or later).

- **IATUX09**

  R11 Address of the current FCT entry.

  R11 Address of the TVTABLE entry.

  R13 Address of the input-service data area for the current function.

- **IATUX19**

  R8 Address of the current JDS entry.

  R9 Address of the current RESQUEUE entry.

  R11 Address of the current FCT entry.

  R12 Address of the current TVTABLE entry.

Note that these register conventions are already set up when the exit is called. You must invoke EELEXIT while these registers are unchanged.

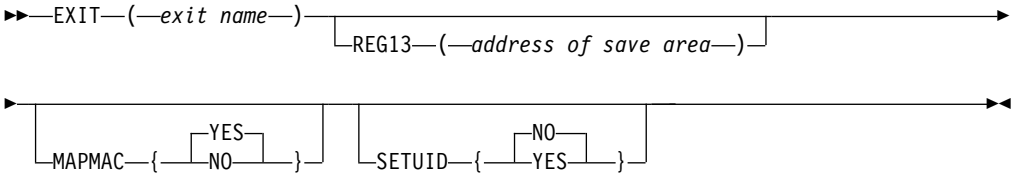
If a shipped JES exit sample (or the EELEXIT macro) has been user–modified, make sure that it does not prevent or filter the tracking of the agent for z/OS itself.

See the NOTES section of the EELEXIT prolog for information about the register contents that are destroyed by EELEXIT in JES exits.

Macro invocation syntax for EELEXIT

EELEXIT produces event tracking exit code by generating assembler code to perform in an SMF or JES exit.

Syntax



Parameters

EXIT = *exit name*
A required keyword defining the name of the exit in which the macro is used. The following names can be specified: IEFACTRT, IEFUJI, EXIT7, and EXIT51. For exits IEFACTRT and IEFUJI, a warning message is issued if the name of the current CSECT differs from the name specified by the EXIT keyword.

REG13 = *address of save area*
An optional keyword defining the address of the current-register save area

when the SMF or JES2 exit was called. The default for this keyword depends on the name specified by the EXIT keyword. If the current exit is EXIT7, the default is PCELPSV. In all other cases, the default is the second fullword in the current save area (if the current save area is properly chained, and the previous save area contains the registers at entry to the exit).

If the default does not apply, the REG13 keyword must be specified. Its value must be a fullword pointing to the save area that was used to store all the registers when the exit was entered.

MAPMAC = {*YES*|*NO*}
An optional keyword specifying whether the macro should generate the required assembler mapping macros. The default is to generate these mapping macros. The following mapping macros are required by EELEXIT code: CVT, IEFJESCT, IEFJSSOB, and IEFJSSIB. The IEFACTRT exit also requires the IEFJMR macro.

SETUID = {*YES*|*NO*}
An optional keyword specifying whether the macro should generate code to place the current user ID in the JMRUSEID field when the IEFUJI exit is taken. Specify YES to generate this code. If you specify NO, which is the default, the JMRUSEID field is not updated. You are recommended to specify YES if you use the current user ID to filter data set close events. You need these mapping macros when you specify YES: IHAPSA, IHAASCB, IHAASXB, and IHAACEE.

Return codes

The following return codes can be generated at assembly time:
4 Input invalid, check for warning messages.
12 Unsupported exit specified for the EXIT keyword.

Messages

The following messages can be generated at assembly time:

- WARNING: EXIT NAME DIFFERS FROM CURRENT CSECT NAME
- WARNING: MAPMAC VALUE MAPMAC IS NOT RECOGNIZED
- EXIT NAME EXIT IS NOT SUPPORTED

Update SYS1.PARMLIB

The following sections describe the updates to SYS1.PARMLIB necessary for your environment.

Defining subsystems

You must define the name of every new agent for z/OS subsystem in the active subsystem-name-table member of SYS1.PARMLIB.

Consider the following when you define the subsystem name:

- The subsystem and the started task names for the agent for z/OS must be the same.
- Agent for z/OS instances connected to the same dynamic workload broker cannot have the same system-STC name identification.
- Because subsystem names on a given LPAR must be unique, and because all agent for z/OS started tasks must have the same name as their associated

subsystems, all started tasks on any given LPAR must have unique names (that is, every agent for z/OS instance inside a z/OS image must have a unique Subsystem/STC name).

To define the subsystems, update the active IEFSSNnn member in SYS1.PARMLIB. Include records as in the following example:

```
Subsystem definition record
SUBSYS SUBNAME(subsystem name) INITRTN(module name) INITPARM ('maxecsa,suffix')
```

where:

subsystem name
The name assigned to an agent for z/OS subsystem. The name must be from 2 to 4 characters. All the subsystem names, as defined in the SYS1.PARMLIB member IEFSSNnn, must be unique within a GRS complex. Also, the subsystem names must be unique within your SYSPLEX, both local and remote systems. The started task name used for an agent for z/OS address space must exactly match the name of the associated subsystem.

module name
The name of the subsystem initialization module, EELINITJ.

maxecsa
Defines the maximum amount of extended common service area (ECSA) that is used to queue job tracking events. The value is expressed in kilobytes (1 KB equals 1024 bytes). The default is 4, which means that a maximum of 4 KB (4096 bytes) of ECSA storage is needed to queue job tracking events. The maximum value allowed for MAXECSA is 2816.

suffix The module name suffix for the EELSSCM module that EELINITJ loads into common storage. EELSSCM is the subsystem communication module. The suffix must be a single character. Because the name of the module shipped with the agent for z/OS is EELINITJ, specify J as the suffix value. If you do not provide a suffix, EELINITJ attempts to load module name EELSSCMJ. You can also specify a subsystem communication module name in the SSCMNAME keyword of the TWSOPTS initialization statement to load an updated version of the module before a scheduled IPL.

"Updating the z/OS link-library definition" on page 20 provides more information about EELSSCM modules.

The next example illustrates a record you can include in the SYS1.PARMLIB IEFSSN*nn* member:

```
/*Subsystem definition example*/
SUBSYS SUBNAME(ZAG3) INITRTN(eelinitj) INITPARM ('100,J')
```

The record defines an agent for z/OS subsystem called ZAG3. Its initialization module is EELINITJ. The amount of ECSA that is allocated, 101104 bytes, is enough for 1136 job tracking events. Because suffix value J is specified, EELINITJ loads module EELSSCMJ.

Calculating MAXECSA values

The agent for z/OS allocates ECSA storage for job tracking events in blocks of 1424 bytes. Each block is equivalent to 16 events. Every job creates a minimum of six events.Table 7 on page 18 gives examples of the storage needed for, the storage actually allocated, and the events accommodated for several MAXECSA values.

If you want to calculate values that are not shown in the table for a given MAXECSA value, use this method:

- Space requested = MAXECSA * 1024
- Blocks = space requested / 1424 (round down to a whole number)
- Space allocated = blocks * 1424
- Events accommodated = blocks * 16

Table 7. Examples of MAXECSA storage values

MAXECSA value	Amount of MAXECSA space requested	Blocks of ECSA space allocated (bytes)	Number of events accommodated
0	0	0 (0)	0
4	4096	2 (2848)	32
8	8192	5 (7120)	80
16	16384	11 (15664)	176
36	36864	25 (35600)	400
72	73728	51 (72624)	816
100	102400	71 (101104)	1136
200	204800	143 (203632)	2288
400	409600	287 (408688)	4592
500	512000	359 (511216)	5744

Important:

- Allocate enough ECSA storage so that job tracking events are not lost when the event writer subtask of the agent for z/OS is not active. When the event writer is active, the number of queued events in ECSA is almost always 0. Allocate enough ECSA for the maximum amount of time you expect the event writer to be inactive.

  For example, after the IPL of a z/OS system, job tracking events can occur before the agent for z/OS address space has become active. If you expect a maximum of 50 events to occur during this time, you should set a MAXECSA value of 8, as shown in the table. When the event writer becomes active, the queued events are processed and removed from ECSA.

  If events are lost, message EELZ035E is written in the message log.
- All ECSA storage is allocated above the 16MB line.

Authorizing the load-module library

This section explains how to activate the load-module library for the agent for z/OS.

You must update the active authorized-program-facility member (IEAAPF*nn*, or PROG*nn*) to authorize the load-module library. Each record, except the last, ends with a comma. For the following example, assume that you have installed the agent for z/OS load modules in the data set TWS.SEELLMD0 and that this data set is on volume ABC123. To authorize this library, insert this record before the last entry in the IEAAPF*nn*:

```
TWS.SEELLMD0     ABC123,
```

or update the PROG*nn* member.

Note that libraries that are defined in the IEAAPF*nn* or PROG*nn* member are authorized only if they remain on the volume specified. If DFHSM is used in your system, change DFHSM parameters so that the new authorized library is not migrated by DFHSM.

Updating SMF parameters

Updating SMF parameters is necessary to activate the exits used by the agent for z/OS for event tracking.

The SMFPRM*nn* member defines parameters for the System Management Facilities (SMF). You must verify that the active SMF parameter member, SMFPRM*nn*, specifies that all SMF exits used by the agent for z/OS for event tracking are activated, and that the required SMF records are being collected. If this is not the case, you must update the active SMF parameter member. Event tracking requires these SMF exits:

IEFACTRT
: Job-end and step-end exit.

IEFUJI
: Job initiation exit.

IEFU83
: Record write exit. Used for job-print-end.

The agent for z/OS uses the following SMF record types:

26 For all job tracking
30 For all job tracking

Active exits are defined by the EXITS parameter of the SYS and SUBSYS keywords. An example of these keywords is:

```
/*SYS and SUBSYS keywords*/
 SYS(TYPE(6,26,30,60,62),EXITS(IEFACTRT,IEFUJI,IEFU83))
 SUBSYS(STC,EXITS(IEFUJI,IEFACTRT,IEFU83))
 SUBSYS(JESn,EXITS(IEFUJI,IEFACTRT,IEFU83))
```

Important:

- JES*n* is either JES2 or JES3. This parameter does not refer to JES itself, but to batch jobs handled by JES. So do not suppress exit invocation. Ensure that you do not specify TYPE6=NO and TYPE26=NO on the JOBCLASS and STCCLASS statements of the JES2 initialization parameters.
- You might find it useful during installation to code two SMFPRM*nn* members, one with the exits active and the other with the exits inactive. You can then use the SET SMF=*nn* z/OS command to switch your current SMF parameters to the new member. By switching back, using the SET SMF=*nn* command, you avoid the need to re-IPL, if you encounter a problem.
- Exits for SUBSYS STC are required only if you run also IBM Workload Scheduler for z/OS in the same system. If you run only the agent for z/OS, the following line is of no use:

  ```
  SUBSYS(STC,EXITS(IEFUJI,IEFACTRT,IEFU83))
  ```

Use the PROG*nn* parmlib member to specify installation exits and control their use. Using PROG*nn*, you can associate multiple exit routines with installation exits at IPL, or while the system is running. Consider using PROG*nn* in addition to SMFPRM*nn* to specify exits, whether or not you want to take advantage of these functions.

The following example shows how you can specify SMF exits in a PROG*xx* parmlib member. If you specify this in SMFPRM*nn*:

```
SYS(...EXITS(IEFACTRT,IEFUJI,IEFU83))
```

you would add this to get the equivalent processing in PROG*nn*:

```
EXIT ADD EXITNAME(SYS.IEFACTRT) MODNAME(IEFACTRT)
EXIT ADD EXITNAME(SYS.IEFUJI) MODNAME(IEFUJI)
EXIT ADD EXITNAME(SYS.IEFU83) MODNAME(IEFU83)
```

When you associate new exit routines with SMF exits through PROG*nn* or the SETPROG command, you must use the following naming conventions:

- For exits listed on the EXITS keyword of the SYS statement in SMFPRM*nn*, each exit will have the name SYS.*xxxx* (where *xxxx* is one of the exits listed).
- For exits listed on the EXITS keyword of the SUBSYS statement of SMFPRM*nn*, each exit will have the name SYS*zzzzz*.*xxxx* (where *zzzz* is the name of the subsystem and *xxxx* is one of the exits listed).

If you define two members in SYS1.PARMLIB with two different names, for example, PROG03 in which there is the statement `EXIT ADD EXITNAME(SYS.1 EFACTRT) MODNAME(EQQACTR1)`, you can switch to the version EQQACTR1, without the need to re-IPL, by issuing the command: `/SET PROG=03`.

For information about using PROG*nn* to control the use of exits and exit routines, see *IBM z/OS MVS Initialization and Tuning Reference*.

Updating z/OS dump options

This section describes how to update the z/OS dump options for the agent for z/OS.

The sample JCL procedure for an agent for z/OS address space includes a DD statement and a dump data set is allocated by the EELRETWT sample created by EELINST. SYSMDUMP is the dump format preferred by the service organization.

Ensure that the dump options for SYSMDUMP include RGN, LSQA, TRT, CSA, and GRSQ on systems where an agent for z/OS address space will run. To display the current SYSMDUMP options, issue the z/OS command DISPLAY DUMP,OPTIONS. You can use the CHNGDUMP command to alter the SYSMDUMP options. Note that this will only change the parameters until the next IPL is performed.

To dump an agent for z/OS address space using the z/OS DUMP command, the SDUMP options should specify RGN, LSQA, TRT, CSA, and GRSQ. Consider defining these options as your system default.

Updating the z/OS link-library definition

This section documents what you should do to update the z/OS link-library definition if you installed the agent for z/OS in a separate load-module library.

If you installed the agent for z/OS in a separate load-module library, you should define this library in the active LNKLST*nn* member.

If you installed load modules in the data set TWS.SEELLMD0 and this data set is cataloged in the master catalog, insert this record before the last entry in the LNKLST*nn* member to add this library to the link library concatenation:

```
Adding LINKLIB
TWS.SEELLMD0
```

If you choose not to define the IBM Workload Scheduler for z/OS load-module library in the LNKLST*nn* member, you *must*:

- Copy the tracker modules, EQQINITM and EQQSSCMM, to a library in the z/OS link-library concatenation. EQQINITM is used by the master-scheduler-initialization function when the z/OS system is being IPLed. EQQINITM then loads EQQSSCMM into common storage. EQQSSCMM is about 23KB and is placed above the 16MB line. Remember to copy the modules again whenever they are updated by IBM Workload Scheduler for z/OS maintenance. This is especially important for the EQQSSCMM module, which must be at the same update level as the rest of the IBM Workload Scheduler for z/OS code.
- Define the IBM Workload Scheduler for z/OS load-module library on a STEPLIB DD statement in the started-task JCL.

Starting the product automatically

The COMMND*nn* member of SYS1.PARMLIB lists z/OS commands automatically issued during system initialization. To avoid delays in starting the agent for z/OS when the z/OS system is started, consider including the names of the agent for z/OS started task in this member. For information about how to include start commands for an address space, see the *IBM z/OS MVS Initialization and Tuning Reference*.

Update RACF for the agent for z/OS started task

This section describes how to define the agent for z/OS to your security system.

If your installation protects data and resources from unauthorized use, you must define the agent for z/OS to your security system. This section assumes that the Resource Access Control Facility (RACF) is installed and active on your z/OS system. It describes the activities you must perform to define and enable the security environment for the agent for z/OS.

RACF controls the interaction between users and resources. You define resources and the level of access allowed by users to these resources in RACF profiles. A user is an alphanumeric user ID that RACF associates with the user.

The agent for z/OS needs access to z/OS resources for the work it schedules. The user ID associated with the agent can be obtained from:

- The agent for z/OS address space that accesses data sets used by the work it schedules, and that submits work and issues JES commands.
- The `USER` parameter on the JOB card of a batch job to be submitted.

Controlling the user ID of the address space

Since the agent for z/OS runs as a started task, you must associate the cataloged procedure name with a suitably authorized RACF user. The user ID must be defined in the STARTED resource class.

Controlling the user ID of submitted jobs

The agent for z/OS can submit to JES two types of jobs:

- Normal production jobs, which are submitted from a IBM Workload Scheduler plan.
- Ad-hoc jobs, which you can submit directly using the Dynamic Workload Console or conman.

The agent submits production and ad-hoc jobs to the internal reader when all prerequisites are fulfilled. You can determine the authority given to a job in the following ways:

- You can submit work with the authority of the agent for z/OS address space. The job is given the same authority as the agent for z/OS.
- You can include a password in the JCL to propagate the authority of a particular user.

Protecting data sets

For basic security of data, you should restrict access to the following product data sets:

- The internal reader (EELBRDS)
- The diagnostic data sets (EELDUMP and SYSDUMP)
- The event data sets (EELEVDS and EELHTDS)
- The service data set (EELHTREF)
- The message library (EELMLIB)
- The message log (EELMLOG)
- The parameter library (EELPARM)

Moreover, software support people must be able to debug problems and reorganize files. You might give them alter access to all the product data sets.

Set up the SSL environment

This section describes how to set up SSL protection for the connection between your agent for z/OS and IBM Workload Scheduler.

To provide SSL security for the HTTP connection between the agent for z/OS and the dynamic workload broker of IBM Workload Scheduler, in the HTTPOPTS initialization statement:

- Set the SSL and/or TDWBSSL keywords to `Yes`
- Provide values for the SSL-related keywords
- Select SSL-enabled ports for the two connecting counterparts in the PORTNUMBER (for the agent) and TDWBPORTNUMBER (for dynamic workload broker) keywords

Using security certificates

When you install the agent, the following default security certificates are automatically stored in the SEELDATA library:

EELCERCL
The security certificate for the HTTP client (the dynamic workload broker).

EELCERSR
The security certificate for the HTTP server (the agent for z/OS).

Unless you already did so while running the EELINST installation aid (panel 4/5), or unless you already use SSL with IBM Workload Scheduler for z/OS, you must choose between using these default certificates or creating your own. In both cases, you need to manually import them into your security system. If you are using RACF, you are provided with the EELRCERT sample job that imports the certificates. To run this job, ensure that you use the same user ID that RACF associates with the agent for z/OS started task.

The EELRCERT job:

- Copies the EELCERCL and the EELCERSR certificates to temporary sequential data sets.
- Imports EELCERCL and EELCERSR to RACF.
- Deletes the temporary sequential data sets.
- Creates the SAF key ring that is used to connect the imported certificates.
- Updates the RACF database with the new certificates and key ring.

Update SYS1.PROCLIB

This section describes how to define a JCL procedure for the agent for z/OS address space.

You must define a JCL procedure or batch job for the agent for z/OS address space.

To help you do this, the EELINST installation aid generates the following members in the output library that you specified in the `Create customized sample jobs` dialog:

EELAGT
Sample started task procedure for the agent.

EELAGTP
Sample started task parameters for the agent.

These members contain started task JCL that is tailored with the values you entered in the dialog. Tailor these members further, according to the data sets you require. Alternatively, you can copy a member from the SEELSAMP library to one of your own libraries, and tailor it manually.

If you create a new library for your agent for z/OS started-task procedures, remember to specify the library in the JES PROCLIB concatenation. Then you must restart JES to include the new library.

Required data sets

Include the following required datasets in your JCL procedure:

Table 8. Required data sets for the agent for z/OS

DD Name	Defines
EELBRDS	A JES internal-reader.
EELEVDS	Event data set for the submit checkpointing function and for the event writer task.
EELHTDS	Event data set for storing events originated by dynamic workload broker.

Table 8. Required data sets for the agent for z/OS (continued)

DD Name	Defines
EELHTREF	Service dataset used for processing JCLs *by reference* and variable substitution.
EELMLIB	Message library.
EELMLOG	Output message log.
EELPARM	Parameter library.

Optional data sets

The following table shows the data sets that you can optionally include in your JCL procedures. Specify these data sets only if you want to use the function with which they are associated.

Table 9. Optional data sets for the agent for z/OS

DD Name	Defines
EELDUMP	Diagnostic dump output.
STDENV	Contains environment variables. The STDENV DD name can point to a sequential DS or a PDS member (for example, a member of the PARMLIB) in which you can define environment variables to initialize Language Environment®. STDENV must have a F or FB format with a record length equal or greater than 80. In this data set/member you can put your environment variables specifying VARNAME=value. On each row you can specify only 1 variable, characters after column 71 are ignored. If you need more than 71 characters, you can add any character in column 72 and continue on the next row (the character in column 72 is ignored).
STEPLIB	Load-module library.
SYSMDUMP	Dump data set.

Complete the installation

This section describes the final steps you need to follow to complete and verify the installation of the agent.

When you have completed the installation tasks for the agent:

1. IPL each system where you have installed the agent.
2. Verify the installation.

To verify the agent, run these tasks:

1. Ensure that you have completed all the necessary installation tasks.
2. Start the agent and check the connection with dynamic workload broker.
3. Check the message log (EELMLOG).
4. Verify that tracking events are created in the event data set (EELEVDS).

5. Perform problem determination for tracking events if events are missing from the event data set.

Starting the agent and checking the connection

The first step to verify that the installation was successful is to start the agent and check that it connects with the specified dynamic workload broker of IBM Workload Scheduler.

To start the agent for z/OS, use the z/OS START command using the subsystem or started task name you defined for the agent in the EELINST panels.

As the agent starts and successfully connects with dynamic workload broker, the `Agent open for ebusiness` message is displayed on the z/OS console.

The first time that the agent starts and connects successfully with the dynamic workload broker of which you provided hostname and port in the HTTOPTS initialization statement, it is automatically defined in the IBM Workload Scheduler database with workstation name:

subsystem name-system name

where:

subsystem name
Is the name of the z/OS started task that starts the agent.

system name
Is the name of the z/OS system.

You can now use this workstation name to design, submit, and monitor workload for z/OS. For more details, see "Computer and workstation names of the agent" on page 47.

Ensuring that all installation tasks are complete

Ensuring that all installation tasks are complete.

Ensure that you have performed all the installation tasks that are needed for your agent for z/OS to run properly. That is, you should have:

- Followed the appropriate procedures for the agent for z/OS subsystem that you are installing.
- Installed the required JES and SMF exits, and verified that they are active.
- Created a JCL procedure for the tracker.
- Allocated required data sets.
- Given the security access for the subsystem to access the data sets.
- Specified the initialization statements in the parameter library (EELPARM).

Checking the message log

This section describes how to verify the message log.

After starting the agent, check the message log:

- Check that the return code for all initialization options is 0 (message EELZ016I).
- Ensure that all required subtasks are active.
  - The data-router and submit tasks are always started. You should see these messages:

```
EELZ005I SUBTASK DATA ROUTER IS BEING STARTED
EELF001I DATA ROUTER TASK INITIALIZATION IS COMPLETE

EELZ005I SUBTASK JOB SUBMIT IS BEING STARTED
EELSU01I THE SUBMIT TASK HAS STARTED
```

  - Also, verify that the agent has started an event writer. You should see these messages:

```
EELZ005I SUBTASK EVENT WRITER IS BEING STARTED
EELW065I EVENT WRITER STARTED
```

- Examine error messages.

  Important: The first time the event writer is started, it formats the event data set. Ignore the SD37 abend code that is issued during the formatting process.
- Check that your log is complete. To do so, issue a dummy MODIFY command like this: `F ssname,xx`. Message EELZ049E is written to the log when the command is processed. If this message is the last entry in the log, it means that the log works properly.

Verifying tracking events

This section describes how to check that the agent is collecting tracking event information and writing it to the event data set (EELEVDS).

Job tracking works correctly only if the agent for z/OS receives information about all the status changes of the jobs it submitted. Job tracking gets this information from SMF and JES exits. These exits gather the necessary information, and an exit record is added to the event writer queue of the agent via ECSA buffers. The event writer queue is active also when the agent is not active.

The event writer

The event writer removes the event from its queue and creates an event record that is written to an event data set. The event writer also forwards the event if it has been started with an event reader function.

The event data set

The event data set is needed to even out any difference in the rate that events are being generated and processed, and to prevent events from being lost if the agent for z/OS must be restarted.

The first byte in an exit record is A if the event is created on a JES2 system, or B if the event is created on a JES3 system. This byte is found in position 21 of a standard event record, or position 47 of a continuation (type N) event. Bytes 2 and 3 in the exit record define the event type. Table 10 shows the event types that are generated by the agent for z/OS.

Table 10. Events generated by the agent for z/OS.

Event type	Description	Generated by...
KJ1	Job submission event. A job has been submitted to JES by the agent for z/OS.	Agent
A1	Reader event. A job has entered the JES2 system.	JES2 exits EXIT7 and EXIT51

Table 10. Events generated by the agent for z/OS. (continued)

Event type	Description	Generated by...
B1	Reader event. A job has entered the JES3 system.	JES3 exit IATUX09
A2 or B2	Job-start event. A job has started to execute.	SMF exit IEFUJI
A3J or B3J	Job-end event. A job has finished executing.	SMF exit IEFACTRT
A3P	Job-termination event. A job has been added to the JES2 output queues.	JES2 exit EXIT7
B3P	Job-termination event. A job has been added to the JES3 output queues.	JES3 exit IATUX19
A5	Purge event. All output for a job has been purged from the JES2 system.	JES2 exit EXIT7
B5	Purge event. All output for a job has been purged from the JES3 system.	SMF exit IEFU83

If any of these event types are not being created in the event data set (EELEVDS) after the first submission, a problem must be corrected before the agent for z/OS is started in production mode.

Perform these actions to verify that events are being created on your system:

1. Run a job from conman or the Dynamic Workload Console:
   a. Submit a job like the following, ensuring that the output is written to a non-held output class:

      Test job

      ```
      //VERIFY1  JOB   STATEMENT PARAMETERS
      //VERIFY   EXEC  PGM=IEBGENER
      //*
      //SYSPRINT DD DUMMY
      //SYSUT2   DD SYSOUT=A
      //SYSIN    DD DUMMY
      //SYSUT1   DD *
          SAMPLE TEST OUTPUT STATEMENT 1
      //*
      ```

   b. Verify that the job has executed, printed, and purged.
   c. Browse the EELEVDS data set using the ISPF/PDF browse facility. You will find the following events in the event data set:
      - Type KJ1 event
      - Type A1 event (for JES2) or type B1 event (for JES3)
      - Type A2 event (for JES2) or type B2 event (for JES3)
      - Type A3J event (for JES2) or type B3J event (for JES3)
      - Type A3P event (for JES2) or type B3P event (for JES3)

Performing problem determination for tracking events

This section describes how to run problem determination for tracking events if events are missing from the event data set.

Problem determination depends on which event is missing. In the following table, the first column refers to the event type that is missing, and the second column tells you what action to perform.

Table 11. Types of missing event and relative problem determination actions

Type	Problem determination actions
All	1. Verify in the EELMLOG data set that the event writer has started successfully. 2. Verify that the definition of the EELEVDS ddname in the agent for z/OS started-task procedure is correct (that is, events are written to the correct data set). 3. Verify that the required exits have been installed. 4. Verify that the IEFSSN*nn* member of SYS1.PARMLIB has been updated correctly, and that an IPL of the z/OS system has been performed since the update.
KJ1	Verify that the agent for z/OS subsystem was correctly defined.
A1	If event A3P is also missing: 1. Verify that the agent for z/OS version of the JES2 exits 7 and 51 routines have been correctly installed. Use the JES commands `$T EXIT(7)` and `$T EXIT(51)` or `$DMODULE(TWSEXIT7)` and `$DMODULE(TWSXIT51)`. 2. Verify that the JES2 initialization data set contains a LOAD statement and an EXIT7 statement for the agent for z/OS version of the JES2 exit 7 (TWSEXIT7). 3. Verify that the exit has been added to a load module library reachable by JES2 and that JES2 has been restarted since this was done. If event A3P is present in the event data set, call an IBM® service representative for programming assistance.
B1	1. Verify that the agent for z/OS version of the JES3 exit IATUX09 routine has been correctly installed. 2. Verify that the exit has been added to a load-module library that JES3 can access. Verify that JES3 has been started.
A2 or B2	1. Verify that the job for which no type 2 event was created has started to execute. A type 2 event will not be created for a job that is flushed from the system because of JCL errors. 2. Verify that the IEFUJI exit has been correctly installed: a. Verify that the SMF parameter member SMFPRM*nn* in the SYS1.PARMLIB data set specifies that the IEFUJI exit should be called. b. Verify that the IEFUJI exit has not been disabled by an operator command. c. Verify that the correct version of IEFUJI is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, z/OS uses the first IEFUJI module found. d. Verify that the library containing this module was updated by the agent for z/OS version of IEFUJI and that z/OS has been IPLed since the change was made.

Table 11. Types of missing event and relative problem determination actions (continued)

Type	Problem determination actions
A3J or B3J	1. Verify that the IEFACTRT exit has been correctly installed. 2. Verify that the SMF parameter member SMFPRM*nn* in the SYS1.PARMLIB data set specifies that the IEFACTRT exit should be called. 3. Verify that the IEFACTRT exit has not been disabled by an operator command. 4. Verify that the correct version of IEFACTRT is active. If SYS1.PARMLIB defines LPALIB as a concatenation of several libraries, z/OS uses the first IEFACTRT module found. 5. Verify that this library was updated by the agent for z/OS version of IEFACTRT and that z/OS has been IPLed since the change was made.
A3P	If A1 events are also missing, follow the procedures described for A1 events. If A1 events are not missing, call an IBM service representative for programming assistance.
B3P	1. Verify that the agent for z/OS version of the JES3 exit IATUX19 routine has been correctly installed. 2. Verify that the exit has been added to a load-module library that JES3 can access. 3. Verify that JES3 has been restarted.
A5	1. Verify that JES2 has purged the job for which no A5 event was created. 2. Ensure that you have not specified TYPE26=NO on the JOBCLASS and STCCLASS statements of the JES2 initialization parameters. 3. If A1 events are also missing, follow the procedures described for A1 events. 4. If A1 events are not missing, call an IBM service representative for programming assistance.
B5	Verify that JES3 has purged the job for which no B5 event was created.

Recommendations for allocating the job library data set (EELJBLIB)

The job library data set contains the JCL for the jobs that the agent for z/OS will submit. It is required by the agent. The data set is first allocated when you run the EELINST installation aid.

If you subsequently need to allocate more job library data sets, give your preference to PDSE data sets.

If you must allocate PDS data sets, allocate the job library data set with a only primary space allocation. If a secondary allocation is defined and the library goes into an extent when the agent is active, you must stop and restart the agent. Also, do not compress members in this PDS. For example, do not use the ISPF `PACK ON` command, because the agent does not use ISPF services to read it.

Customization parameters

The parameters necessary to customize the agent for z/OS are provided in the form of initialization statements. This section documents all the initialization statements required by the agent.

The initialization statements required to run your agent for z/OS started task are created with working default values when you run the **Define initialization statements in the parameters library** step of the EELINST installation aid. You have the option at that time to leave the default values, and change them later, or to change them directly before creating the related member in the parameters library. To change their values at a later time, you have to edit the PARMLIB data set of the agent, and restart the agent afterwards.

The agent requires the following initialization statements:

EWTROPTS
Defines runtime options for the event writer task.

EXITS Defines the exit policy for the agent.

HTTPOPTS
Defines options to connect with the dynamic workload broker.

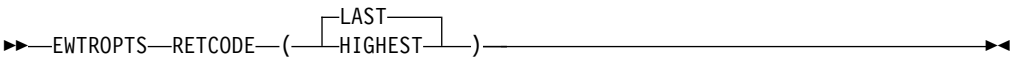
TWSOPTS
Defines generic runtime options for the agent.

Specifying runtime options for the event writer

This section describes the EWTROPTS initialization statement.

Use the EWTROPTS initialization statement to specify runtime options for the event writer task.

EWTROPTS



Parameters

RETCODE = (LAST|HIGHEST)
Defines how the event writer creates a return code for the job-end (A3J) event record. If you specify HIGHEST, the event writer creates an event record with the highest return code of all the performed steps. If you specify LAST, the event writer creates an event record with the return code of the last performed step.

The default is LAST.

Important: On z/OS 1.13 and later, the JOBRC parameter can be added in the JCL JOB card statements. If JOBRC is specified in the JCL JOB card with the MAXRC or LASTRC values, the job completion code determined by RETCODE is overriden by the JOBRC value. If JOBRC is specified with the STEP value, it is ignored by the agent for z/OS, and the job completion code logged in the event record is the one determined by the RETCODE value.

The keyword values are valid until you specify a different value and restart the agent.

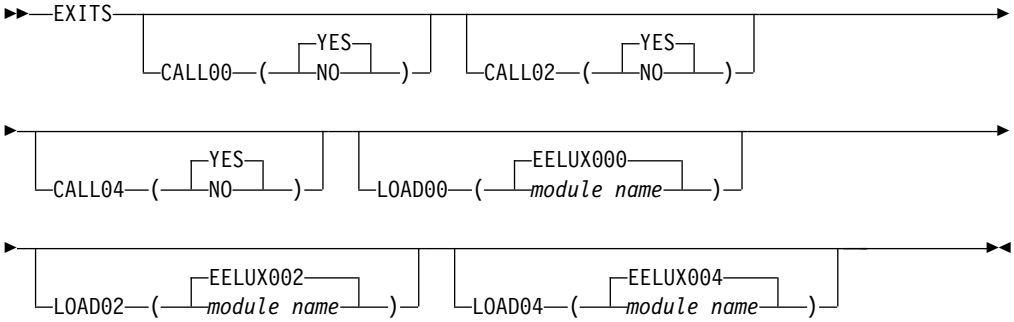
Specifying the exit policy for the agent

This section describes the EXITS initialization statement.

This statement defines exit options for the agent for z/OS. It applies to the EELUX000, EELUX002, and EELUX004 exit programs used by the agent. You can use the EXITS statement to stop the agent from attempting to load a particular exit or to change the default name of the load module.

For more information about these exit programs, see "Configuring exit EELUX000 (start/stop)" on page 39, "Configuring exit EELUX002 (job-library-read)" on page 40, and "Configuring exit EELUX004 (event filtering)" on page 43.

EXITS



Parameters

CALL00 = (YES|NO)
Specifies whether exit EELUX000 should be loaded. The exit name is either EELUX000 or its alternative as specified by the LOAD00 keyword.

The default is YES.

CALL02 = (YES|NO)
Specifies whether exit EELUX002 should be loaded. The exit name is either EELUX002 or its alternative as specified by the LOAD02 keyword.

The default is YES.

CALL04 = (YES|NO)
Specifies whether exit EELUX004 should be loaded. The exit name is either EELUX004 or its alternative as specified by the LOAD04 keyword.

The default is YES.

LOAD00 = (EELUX000|*module name*)
Specifies an alternative load module, which is called instead of the default exit named EELUX000.

LOAD02 = (EELUX002|*module name*)
Specifies an alternative load module, which is called instead of the default exit named EELUX002.

LOAD04 = (EELUX004|*module name*)
Specifies an alternative load module, which is called instead of the default exit named EELUX004.

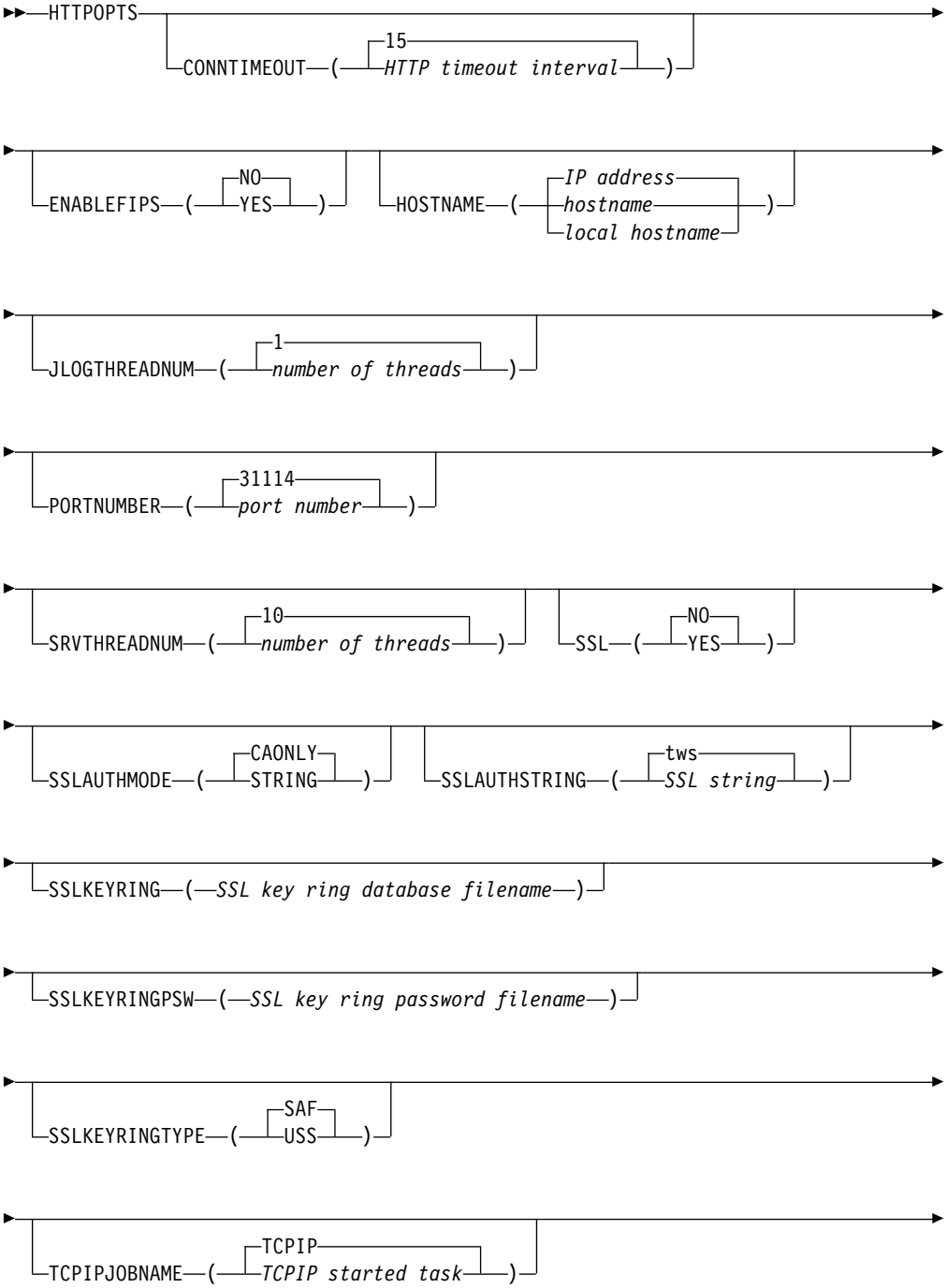
Defining HTTP connection options

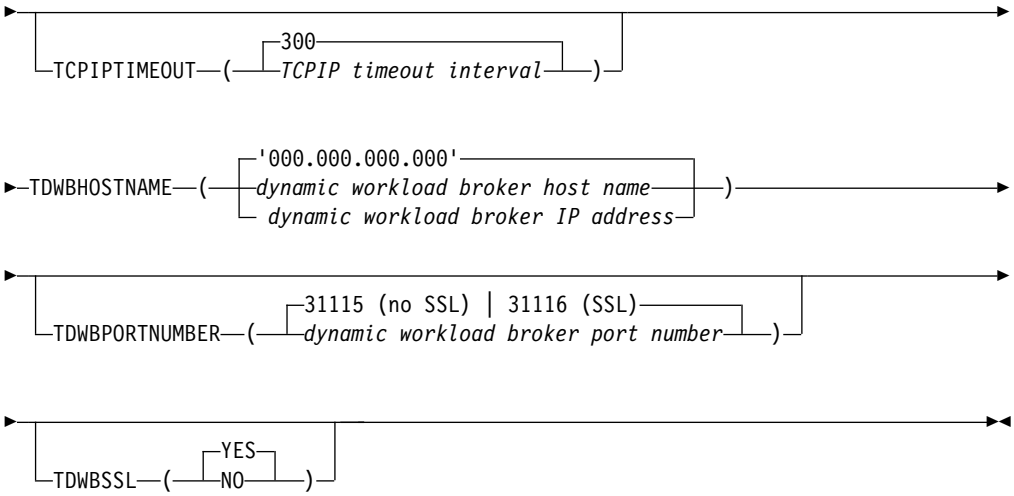
This section describes the HTTPOPTS initialization statement.

This statement defines the connection details between the agent and dynamic workload broker. Use it to specify:

- The hostname and port of the agent
- The hostname and port of the connecting counterpart
- SSL security options

HTTPOPTS





Parameters

CONNTIMEOUT = (*timeout interval*|**15**)
The number of seconds that an HTTP connection waits before a timeout occurs. Valid values are from 1 to 10000. The default is 15 seconds.

ENABLEFIPS(NO|YES)
Indicates whether the SSL communication must comply with FIPS standards. Specify YES to have a FIPS compliant SSL communication. This keyword is ignored if the SSL communication is not enabled. The default is NO.

For more information about how you activate the support for FIPS standard, see*IBM Workload Scheduler for z/OS: Planning and Installation*.

HOSTNAME = (*hostname* | *IP address*)
The local host name or IP address of the agent for z/OS used to communicate with dynamic workload broker. It can be up to 52 alphanumeric characters. The host name or IP address can be in IPV4 or IPV6 format. Enclose this value in single quotation marks. The default is the IP address returned by TCP/IP.

JLOGTHREADNUM = (*number of threads*|**1**)
The number of threads used by the HTTP server task to manage the requests concerning the job log. Valid values are from 1 to 100. The default is 1.

PORTNUMBER = (*port*|**31114**)
The port number on the agent for z/OS used to communicate with dynamic workload broker. Valid values range from 0 to 65535. The default is 31114.

SSL = (*Yes*|**No**)
Specifies if SSL is configured on PORTNUMBER to protect inbound requests. Set to `Yes` if you are using SSL to protect the agent for z/OS port. Set to `No` otherwise. The default is `No`. If SSL is on, the `SSLKEYRING` parameter is mandatory.

SSLAUTHMODE = (*STRING*|**CAONLY**)
The SSL authentication type. Valid values are:

CAONLY
The scheduler checks the validity of the certificate by verifying that a recognized Certification Authority has issued the peer certificate. The information contained in the certificate is not checked.

STRING
The scheduler checks the validity of the certificate as described in the CAONLY option. It also verifies that the Common Name (CN) of the Certificate Subject matches the string specified in the SSLAUTHSTRING parameter.

The default is CAONLY.

SSLAUTHSTRING = (*SSL string*|**tws**)
The SSL string used to verify the validity of the certificate when you set SSLAUTHMODE to STRING. The string can be up to 64 characters. The default is tws.

SSLKEYRING = (*SSL key ring database filename*)

If SSLKEYRINGTYPE is SAF (System Authorization Facility), this parameter specifies the SAF key ring used to connect the security certificates.

If SSLKEYRINGTYPE is USS (Unix System Services), this parameter specifies the database containing keys and certificates. It consists of an SSL working directory name and file name, in the format:

```
SSLworkdir/TWS.kbd
```

The parameter is case-sensitive.

SSLKEYRINGPSW = (*SSL key ring password filename*)
This parameter is required when you run SSL security and SSLKEYRINGTYPE is USS. It specifies the file containing the key password. It consists of an SSL working directory name and file name, in the format:

```
SSLworkdir/TWS.sth
```

Failure to provide an existing and correct filename results in an error message and prevents the agent from starting. The parameter is case-sensitive.

SSLKEYRINGTYPE = (*USS* | *SAF*)
Specifies if the key ring file is a key database USS file or a SAF key ring. If the type is SAF, you can use the RACF command to manage SSL connections.

Important: If the type is USS, you must provide an SSL key ring password filename for SSLKEYRINGPSW. Failure to do this will prevent the agent from starting.

SRVTHREADNUM = (*number of threads*|**10**)
The number of threads that can be used by the HTTP server task to process more requests sent by dynamic workload broker at the same time. Valid values range from 2 to 100. The default is 10.

TCPIPJOBNAME = (*TCPIP started task*|**TCPIP**)
The name of the TCPIP started task running on the z/OS system. The default name is TCPIP.

TCPIPTIMEOUT = (*TCPIP timeout interval*|**300**)
The number of seconds that an HTTP request waits for response before a timeout occurs. Valid values are from 1 to 10000. the default is 300.

TDWBHOSTNAME = (*dynamic workload broker host name*|*dynamic workload broker IP address*|**'000.000.000.000'**)
The local host name or IP address of the dynamic workload broker to which the agent for z/OS is to establish an HTTP connection. It can be up to 52

alphanumeric characters. The host name or IP address can be in IPV4 or IPV6 format. Enclose this value in single quotation marks. The parameter is mandatory.

TDWBPORTNUMBER = (*port*|**31115**|**31116**)
The port number of the dynamic workload broker to which the agent for z/OS is to establish the HTTP connection. Defaults are 31115 for non-SSL connections and 31116 for SSL connections.

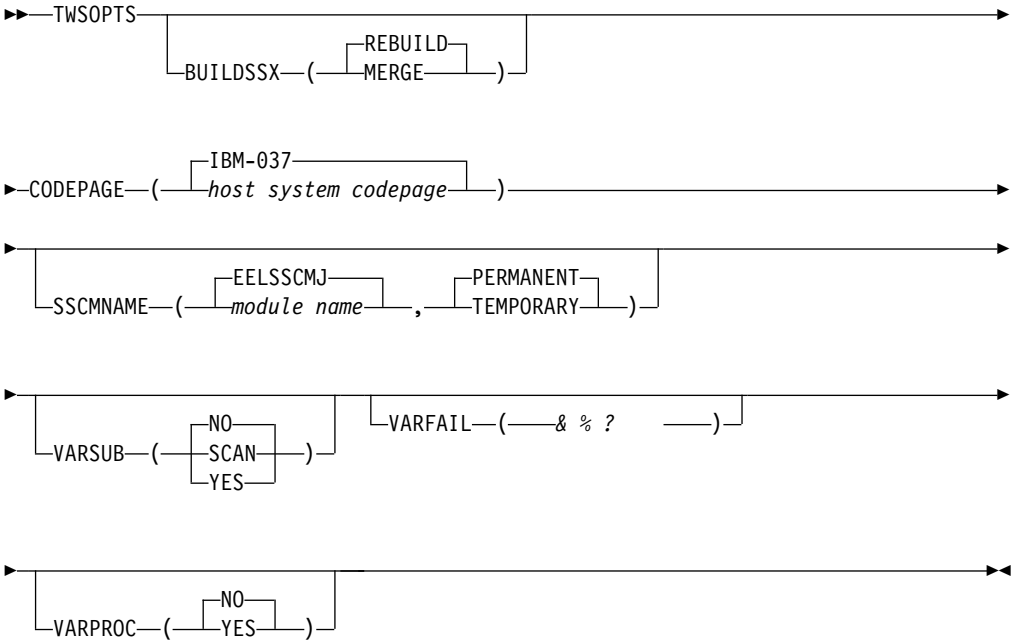
TDWBSSL = (**Yes**|**No**)
Specifies if the dynamic workload broker port defined by TDWBPORTNUMBER is protected by SSL. The default is Yes.

Specifying generic runtime options for the agent

This section describes the TWSOPTS initialization statement.

This statement defines runtime options for the agent for z/OS.

TWSOPTS



Parameters

BUILDSSX = (**MERGE**|**REBUILD**)
Defines if the subsystem communication vector table (CVT) extension for the agent for z/OS, the SSX, should be rebuilt at a new level when the address space is started. The SSX is created at subsystem initialization by the EELINITJ module. If the EELINITJ module has since been updated, by maintenance or because you are installing a new release or modification level of the agent for z/OS, use the BUILDSSX keyword to avoid a z/OS IPL.

Specify MERGE when operational data, such as the event writer queue, should be copied to the new SSX. This ensures that the new event writer queue is primed with events queued to the old SSX block. Use this option when starting an agent for z/OS address space after installing maintenance updates.

Specify REBUILD when you are migrating to, or falling back from, a new release or modification level of the agent for z/OS. The event writer queue from the old SSX will not be referenced in the new SSX. Ensure you also identify the new subsystem communication module name by using the SSCMNAME keyword.

The default is REBUILD.

Important:

- The PTF coverletter ++HOLD section identifies the service updates that require the SSX be rebuilt.
- MERGE cannot be used when the old and new SSX blocks are built for different FMIDs. Do not use MERGE when migrating to, or falling back from, a new release or modification level of the agent for z/OS.
- If you specify BUILDSSX(REBUILD) to migrate to, or fallback from, a new release or modification level of the agent for z/OS, ensure you also specify the SSCMNAME keyword.
- The BUILDSSX parameter should be removed after the next IPL of the z/OS system as it is no longer required.

CODEPAGE = (*host system codepage*|**IBM–037**)

The name of the host code page. This value is required because it is used by the monitoring task to convert the monitoring data to be sent to the monitoring agent. Provide a codepage from the following list of IBM–*nnn* values, where *nnn* is the EBCDIC code page used for your z/OS system:

IBM–037
US, Portugal, Canada (French). This is the default.

IBM–273
Germany

IBM–274
Belgium

IBM–277
Denmark - Norway

IBM–278
Sweden - Finland

IBM–280
Italy

IBM–284
Spain - Latin America

IBM–285
UK

IBM–297
France

IBM–424
Israel

IBM–500
International

IBM–838
Thai

IBM–933
Korea

IBM–935
China

IBM–937
Taiwan

IBM–939
Japan Extended

IBM–970
Latin 2

IBM–971
Iceland

IBM–975
Greece

IBM–1025
Cyrillic

IBM–1026
Latin 5 (Turkey)

IBM–1047
Open Systems

IBM–1112
Baltic

IBM–1122
Estonia

IBM–1388
China

The following is a list of the EBCDIC code pages for EURO support:

IBM–1140
Finland, Sweden

IBM–1141
Austria, Germany

IBM–1142
Denmark, Norway

IBM–1143
USA

IBM–1144
Italy

IBM–1145
Spain, spanish-speaking Latin America

IBM–1146
UK

IBM–1147
France

IBM–1148
Belgium, Switzerland

IBM–1149
Iceland

SSCMNAME = (*module name*|**EELSSCMJ**,**PERMANENT**|**TEMPORARY**)
The first keyword value defines the name of the subsystem communication module to be used instead of EELSSCMJ that was loaded at IPL. The second keyword value specifies how long the module should replace the one loaded at IPL. Use this keyword to load an updated version of the module before a scheduled IPL. The module you specify must reside in an APF-authorized library defined by either the STEPLIB ddname or LNKLST*nn* concatenation. If SSCMNAME is not specified or specifies a module that cannot be located in an authorized library, the agent for z/OS events will continue to be generated by the EELSSCMJ module loaded at IPL.

Specify PERMANENT as the second keyword value to replace the subsystem communication module loaded at IPL with the module identified in the first keyword value. In this case the module specified must reside in an APF-authorized library defined by the STEPLIB ddname. This is the default.

When TEMPORARY is specified or defaulted as the second keyword value, the module you specify will generate job tracking events only while the agent for z/OS address space is active. When the address space is stopped, events will continue to be generated by the EELSSCMJ module loaded at IPL.

Important:

- The PTF cover letter ++HOLD section identifies service updates that require a new subsystem communication module to be loaded.
- Ensure you specify this keyword when the BUILDSSX(REBUILD) option is used to migrate to, or fallback from, a new release or modification level of the agent for z/OS.
- The SSCMNAME keyword should be removed after the next IPL as it is no longer required.

VARSUB(YES|NO|SCAN)
This keyword specifies whether JCL variable substitution should be performed. YES means that variable scanning will be performed for all jobs. NO means that variable scanning will not occur. SCAN instructs the agent to search the JCL for variables only if the //*%OPC SCAN directive is found in the JCL.

VARFAIL(&, %, ?)
This keyword specifies whether or not unresolved variables in the JCL would cause a JCL error. You can use from one to three of the following characters, in any order, to bypass substitution failure (&, %, ?).

If, for example, VARFAIL(&) is specified, the agent will not consider the failure of a substitution of variables beginning with an & to be an error. Any combination of the three types is allowed, for example, VARFAIL(&, %) or VARFAIL(?), but at least one value must be specified while any repetition of characters will be rejected.

If VARFAIL is not specified, then all the lack of substitution of variables will be treated as errors, as previously.

VARPROC(YES|NO)
This keyword specifies whether or not online procedures should consider variable substitution. If VARPROC(YES) is specified, variables in online procedures will be resolved.

The default is NO.

Configuring the agent for z/OS exits

Exits EELUX000, EELUX002, and EELUX004 are called by the agent for z/OS. Your own programs can use the information passed by the exits to perform a variety of functions.

Each exit is loaded if the exit module exists, if the exit has not been disabled, and if the exit has not been replaced by another exit name in the EXITS initialization statement.

Exits are invoked using standard linkage conventions. When the exit is entered, register 1 points to a parameter list. Each address in this list points to a parameter that is passed to the exit.

The exits are entered with the RACF authority of the agent for z/OS subsystem.

Configuring exit EELUX000 (start/stop)

This section describes the agent for z/OS start/stop exit (EELUX000)

EELUX000 is called when the agent for z/OS is starting and when it is ending normally. You can use this exit to allocate resources when the agent is started and to release them when it is stopped. This avoids the extra overheads involved in allocating and then releasing resources each time they are used.

The sample library SEELSAMP that was created during installation contains the EELUX000 exit, which is a sample of start/stop exits.

Installing the exit

The load module implementing the start/stop exit must be link-edited into an APF-authorized library in the LNKLST concatenation or defined by the STEPLIB DD statement in the agent for z/OS JCL procedure. If the load module performs any input or output operations it must be link-edited with RMODE(24) according to normal z/OS restrictions. Or it can be link-edited with RMODE(ANY).

The agent for z/OS invokes the exit in AMODE 31; the AMODE parameter specified at link-edit time has no effect.

Interface to the exit

The start/stop exit is invoked in task mode, problem state, and key 8 and the job-step task is APF-authorized. The active task runs with the same access authority as the job-step task. The exit must restore this state before returning to its caller.

Control is passed to the exit using the BAL instruction. The exit must return to its caller using the address and addressing mode passed to it in general register 14.

The exit is entered in AMODE 31 but must switch to AMODE 24 before performing any input or output operations, and then switch back to AMODE 31 before returning to the caller.

When the exit is entered, register 1 contains the address of the parameter list. Each address in this list is used to locate the parameter value. These parameters are passed to the exit:

EELUX000 parameters			
ACTION	DS	CL8	(Start/stop action)
MCAUSERF	DS	A	(User field)

ACTION has the value START when the exit is called during the startup of the agent. MCAUSERF is zero for this initial call. Normally, this exit will perform exit initialization functions for the start call when you start the agent. If the exit needs to allocate storage that is used while the agent is active, you should update MCAUSERF to address this storage.

ACTION has the value STOP when the exit is called during termination of the agent. Normally, this exit performs exit termination functions for the stop call when you stop the agent. If MCAUSERF is updated by the start call, the same value is passed to the exit for the stop call.

Configuring exit EELUX002 (job-library-read)

Exit EELUX002 is invoked when a job by reference is selected for processing but the job definition does not include the name of the data set where the JCL is stored. By default, in this case the agent for z/OS searches the concatenation of data sets assigned to the EELJBLIB ddname in the agent's JCL procedure. But if you want the agent to search other data sets, use EELUX002 to perform this function.

Also consider using EELUX002 to enhance performance if you have many large partitioned data sets (PDS) concatenated to EELJBLIB. To find a member in the last data set of the concatenation, the agent must read the directory of all preceding PDSs, which can present a significant overhead. Consider defining a PDS and a corresponding ddname for each computer workstation.

The SEELSAMP member EELUX002 contains a sample job-library-read exit. This sample searches a ddname named MYJOBLIB before searching EELJBLIB.

Installing the exit

The load module implementing the job-library-read exit must be link-edited into an APF-authorized library in the LNKLST concatenation or defined by the STEPLIB DD statement in the agent for z/OS JCL procedure.

Interface to the exit

The job-library-read exit is invoked in task mode, problem state, and key 8 and the job-step task is APF-authorized. The active task runs with the same access authority as the job-step task. The exit must restore this state before returning to its caller.

Control is passed to the exit using the BAL instruction. The exit must return to its caller using the address and addressing mode passed to it in general register 14.

If the exit abends, it is flagged as *not executable*; the agent for z/OS does not try to call the exit again.

When the exit is entered, register 1 contains the address of the parameter list. Each address in this list is used to locate the parameter value. These parameters are passed to the exit:

EELUX002 parameters

```
TYPE      DS   CL1   (Constant = J)
FUNC      DS   CL1   (Constant = G)
JOBNAME   DS   CL8   (Job name)
IOAREA    DS   A     (Address of I/O area)
IOAREAL   DS   F     (Size of I/O area)
RETCODE   DS   X     (Return code)
DATAL     DS   F     (Amount of data returned)
ERRDATA   DS   CL78  (Error message returned)
ADID      DS   CL16  (Name of current application)
USRAREA   DS   A     (User field, 0 at first call)
JCLUSER   DS   CL8   (Last user updating this job)
OPNUM     DS   F     (Operation number)
IATIME    DS   CL10  (Occurrence input-arrival time, YYMMDDHHMM)
VAROCCP   DS   A     (Address of occurrence data if operation is in CP)
VAROPRP   DS   A     (Address of operation data if operation is in CP)
VARWSP    DS   A     (Address of workstation data if operation is in CP)
MCAUSERF  DS   A     (Address set by the user in the EELUX000 exit)
OCCPTR    DS   A     (Address of occurrence data)
OPRPTR    DS   A     (Address of operation data)
WSPTR     DS   A     (Address of workstation data)
AUTHGROU  DS   CL8   (Authority group)
MEMPRO    DS   CL1   (Indicator of memory problems)
TASKPTR   DS   A     (Address of TCB of caller task)
XINFO     DS   A     (Extended information address)
XJNAMLEN  DS   F     (Extended job name length)
USRFNR    DS   F         (Number of user fields)
USRFAREA  DS   A         (User fields area address)
```

JOBNAME
The name of the job that is to be submitted.

IOAREA
The address of a buffer that is allocated by the agent for z/OS, where JCL records for the current job must be placed.

IOAREAL
The amount of space, in bytes, in the IOAREA buffer

RETCODE
Is set by the exit. These values are valid:

0 Normal return.

4 End of data reached for the current job.

16 The job could not be found in any input data set.

20 There is no JCL to be returned by the exit. The agent for z/OS attempts to retrieve the JCL from EELJBLIB.

44 Not enough space. The amount of free space in the IOAREA buffer (as determined by IOAREAL) is not enough to contain the next block of data.

241 I/O error has occurred.

242 An open error has occurred. One or more input data sets could not be opened.

The exit is called again to continue processing the same job when a return code 0 or 44 is returned. All other return codes end processing of the current job.

DATAL
The amount of data returned by the exit when the return code is 0 or 4.

ERRDATA
A user message area where you can describe a problem found in the exit. The text is issued in message EELJ020 if return code 242 is set by the exit or in message EELJ024 if return code 241 is set.

Note: If you modify the message library entry for EELJ020 or EELJ024 to generate a WTO, you must ensure that no more than 70 characters of message text are defined for each line. Reorganize the text, if required. Also, ensure that ERRDATA itself does not exceed 70 characters.

ADID The name of the current application.

USRAREA
Is zero the first time the exit is called to retrieve a job. The exit can set this parameter to any value. The agent for z/OS does not use or update this parameter.

The exit should use the USRAREA parameter whenever it returns a return code 0 or 44. Normally, the USRAREA parameter is used to contain the address of a work area that the exit has performed a GETMAIN on. This work area should contain enough information to enable the exit to continue processing the same job.

JCLUSER
Is zero the first time the exit is called. The exit should set this parameter to the name of the z/OS user that is used for authority checking when the JCL contains automatic recovery statements.

OPNUM
The operation number of the operation representing this job.

IATIME
The input-arrival time of the application occurrence that this job belongs to.

MCAUSERF
A user field that lets you allocate resources in the start/stop exit, EELUX000, that this exit can later use. For example, it is possible to open files for JCL retrieval in the start type call to EELUX000 instead of opening them each time EELUX002 is called. The agent does not use or update this field. The MCAUSERF field is valid when the agent is active.

AUTHGROU
The name of the authority group.

MEMPRO
The indicator of memory problems.

TASKPTR
The address of the task control block of the caller task.

If the exit needs to access its own files, these files must be opened on the first call for a job (USRAREA value=0) and closed in either of the following ways:

- Before returning control to the agent for the last time (before return code 4 is set)

- When an error occurs that does not allow the agent to acquire further memory, the agent informs the exit by setting mempro to:

  X'04' If the limit of 608 000 bytes is reached (EELJ025 issued)

  X'08' If there is not enough storage available (EELJ021 issued)

When the EELUX002 exit is called to retrieve a job for the first time, the I/O area is 32 000 bytes. If the exit has retrieved the entire job and it fits in the buffer space available, the exit can update the I/O area, return the amount of data in the job, and set a return code 4.

If the exit has not retrieved the entire job, it can update the I/O area, return the amount of data in the job, and set a return code 0 to indicate that there is more data to be returned. The next time the exit is called, the address and the size of the I/O area will be updated because the I/O area is partly used by data from an earlier call. The exit should continue this process until there is no more data to return and then set a return code 4 to indicate that the entire job has been retrieved.

Because the available space in the buffer is reduced for each call, it is possible that the exit must set a return code 44 to indicate that the amount of free space is not enough. When return code 44 is returned, the exit is called again with a job name of eight equal signs (========). This is a reset call. The exit then prepares to process the job from the beginning.

No data can be returned on the reset call. When the exit is called again after the reset call, the I/O area is 32 000 bytes larger than before. This process of returning a “not-enough-space” condition can be repeated up to 19 times for a job. This means that the maximum buffer size that can be requested by the EELUX002 exit is 608 000 bytes. This corresponds to a job of 7599 card images. When the 608 000 byte limit is reached, the agent issues message EELJ025, and the exit is called a 20th time if MEMPRO is set to 4.

The exit can also get more buffer space by using all available space in the current buffer. When this happens and return code 0 is set, the exit is called again with 32 000 bytes free in the buffer. The reset call is not used in this case; the exit should continue processing the current job normally. Extending the buffer in this manner can be continued to a maximum buffer size of 608 000 bytes.

Configuring exit EELUX004 (event filtering)

This section describes the event filtering exit (EELUX004)

EELUX004 is called when the agent for z/OS event writer is about to write an event to the event data set. In this exit, you can choose to discard events created by JES and SMF exits.

This exit is commonly used to filter the events created by nonproduction work. If you run a significant number of test jobs and other work, and your job naming standards let you do so, consider using EELUX004 to filter the nonproduction work. The sample library SEELSAMP, that was created during installation, contains the EELUX004 exit, which is a sample of event filtering exits.

Installing the exit

The load module implementing the event filtering exit must be link-edited into an APF-authorized library in the LNKLST concatenation or defined by the STEPLIB DD statement in the agent for z/OS JCL procedure. The load module should be link-edited with RMODE(24) according to normal z/OS restrictions.

The agent for z/OS invokes the exit in AMODE 24; the AMODE parameter specified at link-edit time has no effect.

Interface to the exit

The event filtering exit is invoked in task mode, problem state, and key 8 and the job-step task is APF-authorized. The active task runs with the same access authority as the job-step task. The exit must restore this state before returning to its caller.

Control is passed to the exit using the BAL instruction. The exit must return to its caller using the address and addressing mode passed to it in general register 14.

If the exit abends, it is flagged as *not executable*; the agent does not try to call the exit again.

When the exit is entered, register 1 contains the address of the parameter list. Each address in this list is used to locate the parameter value. These parameters are passed to the exit:

EELUX004 parameters

```
JOBNAME   DS   CL8    (Name of current job)
RETCODE   DS   F      (Return code)
EXR       DS   CL80   (Exit record)
```

where:

JOBNAME
The name of the job for which a job tracking event has been recognized and for which an event record is about to be written to the event data set.

RETCODE
Is set by the exit. The following values are recognized by the job completion checker:

0 Normal return. The event writer continues normal processing; the event is written to the event data set.

8 This is not a scheduler event. The event is not written to the event data set. However, if the event is a reader event (type 1) and the job was held by a scheduler job tracking exit, the job is released from hold by the event writer.

EXR The exit record describing the job tracking event. This record is built by the SMF or JES exit that recognized the event. The job number offset, EXRJOBID, in the exit record contains JOB as the first three characters if the event is created for a job.

Running the agent in a sysplex environment

This section documents how you can run the agent for z/OS in a z/OS sysplex.

You can run the agent for z/OS in a z/OS sysplex with the following recommendations:

- There should be at least one agent for z/OS instance installed in each image of the sysplex.
- The z/OS sysplex should be connected to one dynamic workload broker only, although the same dynamic workload broker can be connected to more than one sysplex or system.

The next figure gives a representation of four systems running in a sysplex environment, connected using cross-system coupling facility (XCF) communication links, and linked to a dynamic workload broker instance.

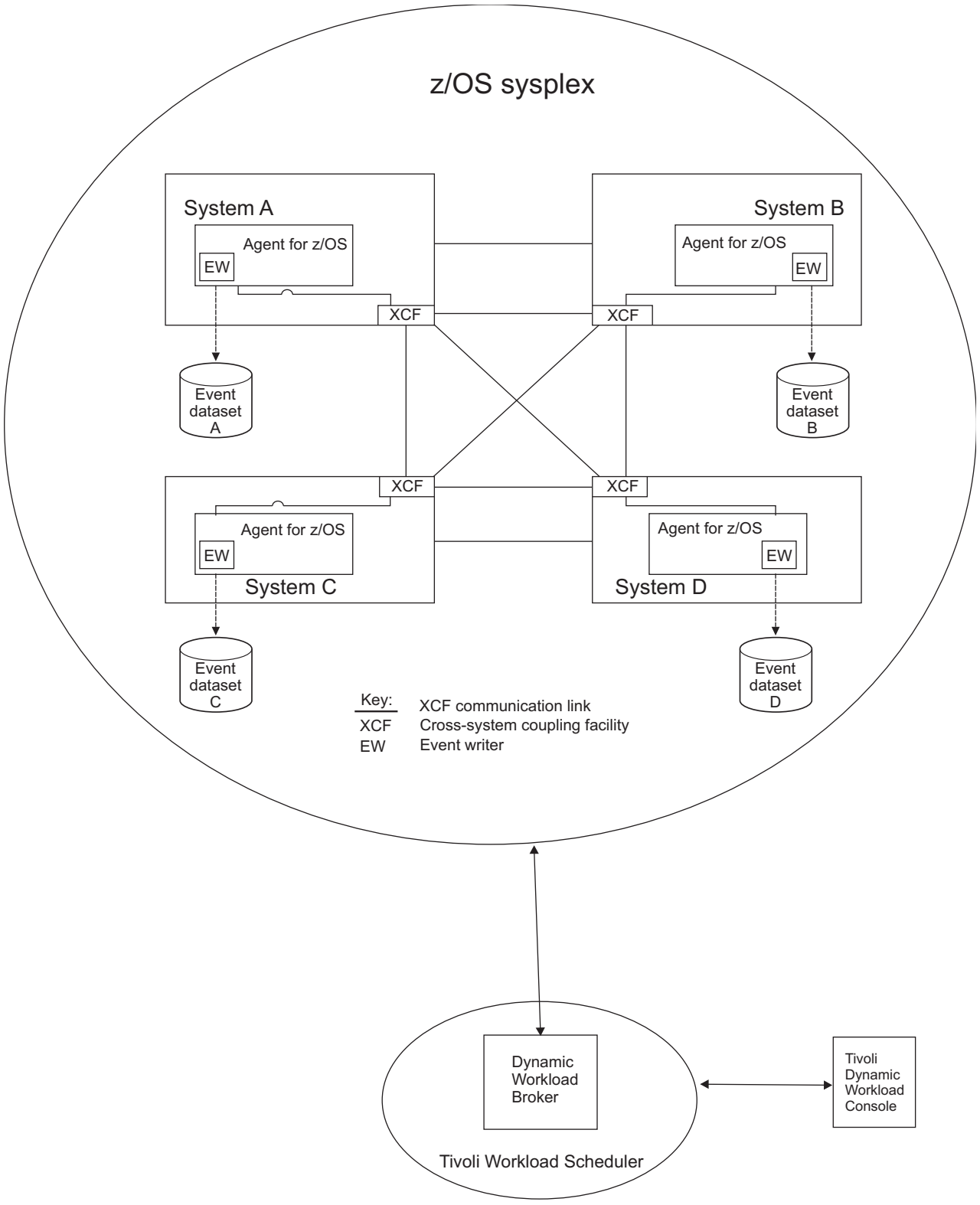


Figure 2. The agent for z/OS in a SYSPLEX configuration.

Routing a job to another system through NJE (Network Job Entry) is not supported with the agent for z/OS, because the possible variation of its job ID and JES reader date and time make tracking unreliable.

Chapter 3. Using

Computer and workstation names of the agent

The first time the agent for z/OS connects with IBM Workload Scheduler, it is automatically given a computer name.

The first time the agent for z/OS connects with the dynamic workload broker component of IBM Workload Scheduler, it is assigned a computer name which is used to identify the workstation in IBM Workload Scheduler.

The assigned computer name is:

subsystem name_system name

where:

subsystem name
is the name of the z/OS started task that starts the agent.

system name
is the name of the z/OS system

The subsystem and system names are joined with the underscore (_) character.

For example, `TDA1_ZOS10B1` is the computer name of an agent that is started by the `TDA1` started task in system `ZOS10B1`.

If either system or subsystem names contain any of the special characters represented by X'5B', X'7B', or X'7C' ($, #, or @, respectively, in the US EBCDIC 037 codepage, but possibly displayed as other special characters in other EBCDIC codepages), the special characters are replaced with underscores (_) when the computer name is composed. For example, if the subsystem name is `ZAG5` and the z/OS system name is `ZSY$1` (in the IBM 037 codepage), the computer name assigned to this particular agent in IBM Workload Scheduler is `ZAG5_ZSY_1`.

The name is stored in the databases of IBM Workload Scheduler and of dynamic workload broker. It is used to identify an agent for z/OS in the following interfaces:

- As the workstation name in the composer and conman command lines and in the Dynamic Workload Console windows
- As the computer name in the command line and graphical user interfaces of dynamic workload broker

This name is the visible part of a longer identification label assigned to the agent upon installation. An internal Id is kept to track the agent in the z/OS system and in IBM Workload Scheduler.

You can change the workstation name using the `composer rename` command or Dynamic Workload Console and use the new name to address the agent throughout the IBM Workload Scheduler interfaces. This does not apply to the dynamic workload broker interfaces where the computer name is used instead.

© Copyright IBM Corp. 2011, 2016

Listing the agents for z/OS

In the Dynamic Workload Console and in composer, the agents for z/OS are listed as workstations of type `agent`.

Follow the normal processes to list your agents:

- In the Dynamic Workload Console, select **Scheduling Environment→Design→List Workstations**
- In composer, run the `composer list ws` command

To determine which of your workstations is a agent for z/OS, or to filter your list, look at the operating system type which in their case is listed as:

- `z/OS` in the Dynamic Workload Console
- `Z` in composer

In composer, to see the `OS type` column in the output of the **`list`** command, remember to set:

```
MAESTROCOLUMNS=120
```

before running the command.

For example, the list of workstations obtained by running:

```
composer list ws=@
```

on this domain, shows that workstations `MAT229` and `MAT229_1` are agents for z/OS.

```
WorkstationName   Type    Domain   OSType Ignored Updated By       Updated On  Locked By Locked On
---------------  ------  --------  ------ ------- ----------       ----------- --------- ---------
RAL15062         manager MASTERDM    W             masterad         09/28/2010     -          -
RAL15062_1       agent      -        W            ResourceAdvisorA  10/04/2010     -          -
RAL15062_DWB     broker  MASTERDM    O             masterad         09/28/2010     -          -
MAT229           agent      -        Z            ResourceAdvisorA  10/01/2010     -          -
MAT229_1         agent      -        Z            ResourceAdvisorA  10/01/2010     -          -
AWSBIA291I Total objects: 5
```

Defining jobs

Like you do for all other IBM Workload Scheduler jobs, you can define jobs for an agent for z/OS from either the composer command line or from the Dynamic Workload Console.

Because all agent for z/OS jobs are submitted through dynamic workload broker, the task section of their job definitions is laid out in the JSDL XML language. If you define the job through the Dynamic Workload Console, the information you provide in the input panels is made into JSDL automatically.

The following characteristics qualify a job as an agent for z/OS job:

- The executing workstation is a agent for z/OS.
- In the JSDL definition in composer, the `application` keyword has name `jcl`. In the Dynamic Workload Console, the job definition type is `z/OS`.
- The job definition in JSDL either includes the JCL that is to be run by JES*n* or points to its location (data set and member names) in the z/OS system. In the first case, the JCL is said to be submitted `by definition`, otherwise it is said to be submitted `by reference`.

While the JCLs specified `by definition` are part of the agent for z/OS jobs and as such are stored in the IBM Workload Scheduler database, the JCLs specified by

reference are stored in data sets in the z/OS system hosting the agent and are retrieved at submission time using the coordinates provided in the corresponding jobs.

Defining in the Dynamic Workload Console

The easiest way to define an agent for z/OS job is with the Dynamic Workload Console.

To create a job definition:

1. From the navigation toolbar, click **Administration>Workload Design>Manage Workload Definitions**.
2. Select an engine name, enter your credentials if required, and click **Go**.
3. In the Working List toolbar of the pop-up window that opens, click **New>Job Definition>Native>z/OS**.

   The Properties window for the job opens.
4. Under the General tab:
   a. Enter:
      - The job name.
      - The name of the agent for z/OS workstation.
      - Select the **Variable resolution at runtime** checkbox, if the JCL (either by reference or by definition) includes variables that need to be resolved by the agent before it is passed to JES.
      - Optionally, specify output conditions. Define which return codes qualify the job as having completed successfully and other output conditions that do not result in SUCC status but that determine which successor job should run next.

        Successful output conditions
        A condition that when satisfied signifies that the predecessor job completed successfully. The job status is set to SUCC. Successful output conditions can be expressed as return codes, job status, output variables or based on job log content.

        Condition Name
        Specify a name that identifies the successful condition that must be met by the predecessor job before a successor job can run.

        Condition Value
        Specify the value of the condition that signifies a successful outcome for the predecessor job.

        For example, a successful output condition might be: **Condition Name** STATUS_OK and **Condition Value** RC=0

        Other conditions
        Output conditions that do not result in SUCC status and determine whether a successor job runs or not. Conditions can be expressed as return codes, job status, output variables or based on job log content.

        Condition Name
        Specify a name that identifies the condition that must be met by the predecessor job before a successor job can run.

Condition Value
Specify the value of the condition that must be met by the predecessor job before a successor job can run.

For example, you might want to create a condition that signifies that the predecessor job has completed with errors. You can define your output condition as follows: **Condition Name** `STATUS_ERR1` and **Condition Value** `RC=2`

The format of **Condition Value** for both successful output conditions and other conditions is as follows: `(RC <operator> <operand>)` where:

RC The instruction keyword

Operator
The comparison operator. Allowed operators are comparison operators (=, != or <>, >, >=, <, <=) that can be combined with logical operators (AND, OR, NOT).

Operand
Any integer between -2147483647 and 2147483647.

For example, you can enter the following expressions:

Successful output conditions:

- (RC<=3) to qualify a job as successful when the job ends with a return code less than or equal to 3.
- NOT ((RC=0) AND (RC=1)) to qualify a job successful when the job ends with a return code different from 0 and 1.
- (RC=2) OR (RC=4) to qualify a job successful when the job ends with a return code equal to 2 or equal to 4.
- (RC<7) AND (RC!= 5) to qualify a job successful when the job ends with a return code less than 7 and not equal to 5.
-

Other conditions
(RC=1) for a condition named STATUS_ERR.

(RC=4 OR RC=9) for a condition named FIRST_PATH

(RC <>5) OR (RC > 2) for a condition named SECOND_FLOW

In the **Condition Value** field for both successful conditions and other ouput conditions, you can also express the output condition using variables other than the return code. For example, you can specify three different output conditions as follows:

- **Condition Name**: `STATUS_ERR` **Condition Value**: `RC=0`
- **Condition Name**: `STATUS_ERR1` **Condition Value**: `RC=${varname}`
- **Condition Name**: `STATUS_ERR2` **Condition Value**: `RC=${LOG.CONTENT}`
- You can set a success or other output condition for the job by analyzing the job output. To analyze the job output, you must check the `this.stdlist` variable.

  For example, you enter the following expression:

  ```
  contains(${this.stdlist},"error")
  ```

  if you want to qualify a job as unsuccessful when the word "error" is contained in the job output.

– For a file transfer job specifically, you can set a success or unsuccess condition for the job by analyzing the job properties.

  For example, you enter the following expression:

  ```
  ${this.File.1.Size}>0
  ```

  if you want to qualify a file transfer job as successful when the size of the transferred file is greater than zero.

– For a file transfer job specifically, you can set a success or unsuccess condition for the job by analyzing the job properties or the job output of another job in the same job stream.

  For example, you enter the following expression:

  ```
  ${this.NumberOfTransferredFiles}=${job.DOWNLOAD.NumberOfTransferredFiles}
  ```

  if you want to qualify a file transfer job as successful when the number of uploaded files in the job is the same as the number of downloaded files in another job, named DOWNLOAD, in the same job stream.

– All Xpath (XML Path Language) functions and expressions are supported, for the above conditions, in the **Condition Value** field:
  - String comparisons (contains, starts-with, matches, and so on)
  - String manipulations (concat, substring, uppercase, and so on)
  - Numeric comparison (=, !=, >, and so on)
  - Functions on numeric values (abs, floor, round, and so on)
  - Operators on numeric values (add, sum, div, and so on)
  - Boolean operators

The next figure shows the General page with the definition of a job of type z/OS.

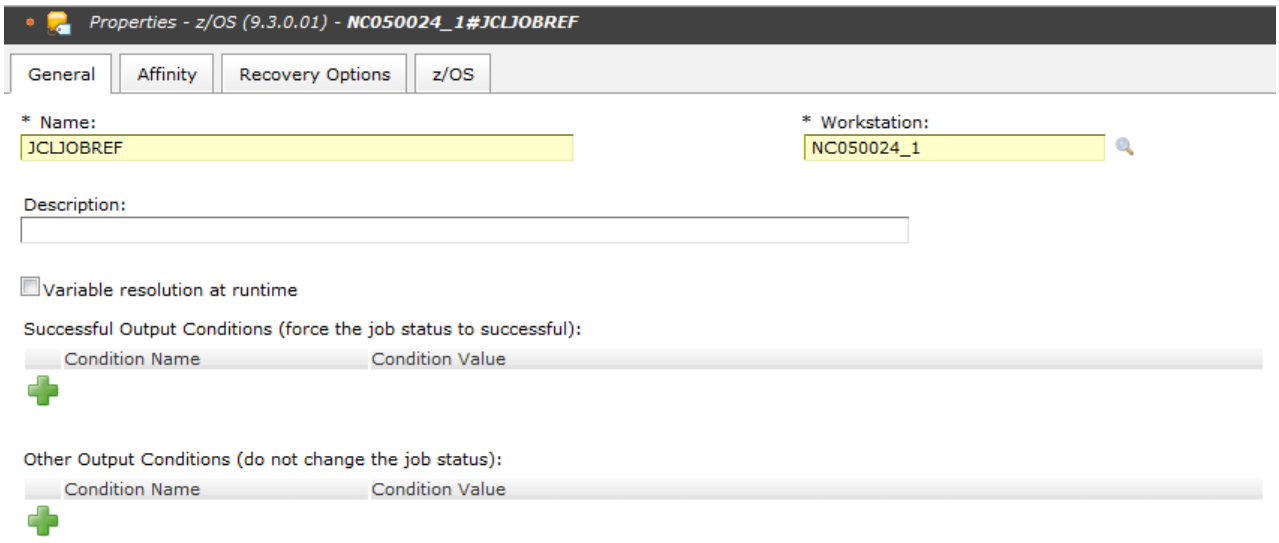


Figure 3. The General page of a new agent for z/OS definition

5. Optionally, enter additional choices in the Affinity and Recovery options pages.

Defining the JCL by reference

If you are defining a job that only needs to point to the location of the JCL you intend to submit to JES, in the z/OS page, select the **by reference** button and enter:

- The name of the data set where the JCL is stored. This name can be up to 44 characters long and is optional. If the data set name is not specified, the agent for z/OS will search for the member name in the data set concatenation library declared for the agent at installation time.
- The name of the JCL member in the data set. This name can be up to 8 characters long and is required.

The next figure shows the z/OS page where the **by reference** option is selected:

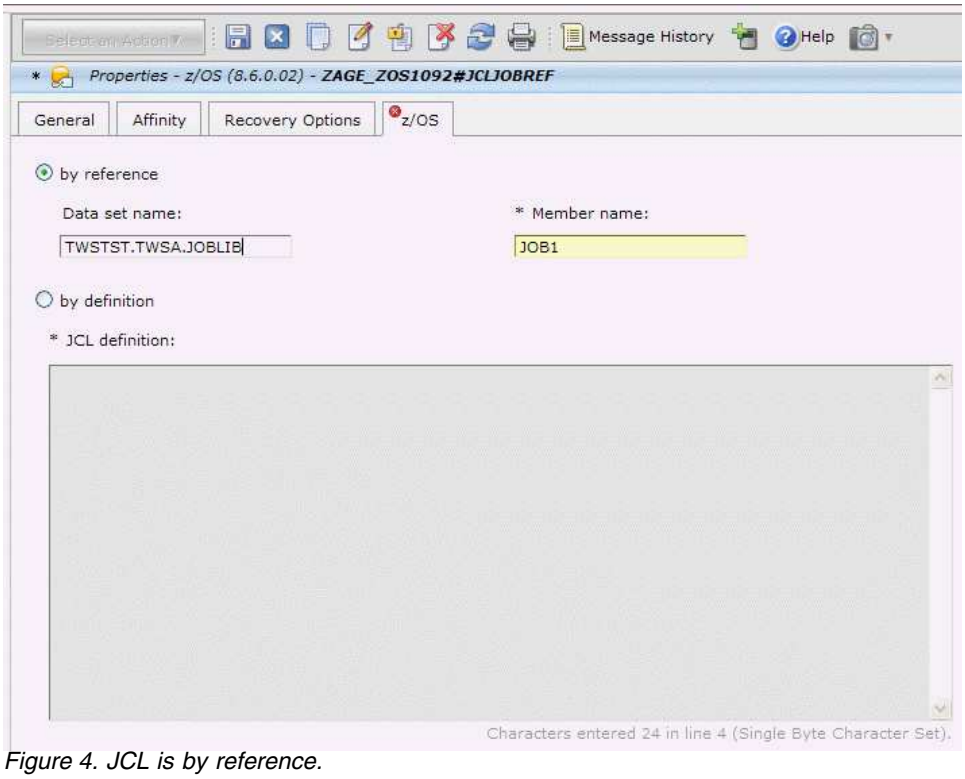


Figure 4. JCL is by reference.

Defining the JCL by definition

To add the entire JCL to the job definition, in the z/OS page select the **by definition** button and write the JCL statement in the **JCL definition** box.

The next figure shows the z/OS page where the **by definition** option is selected:

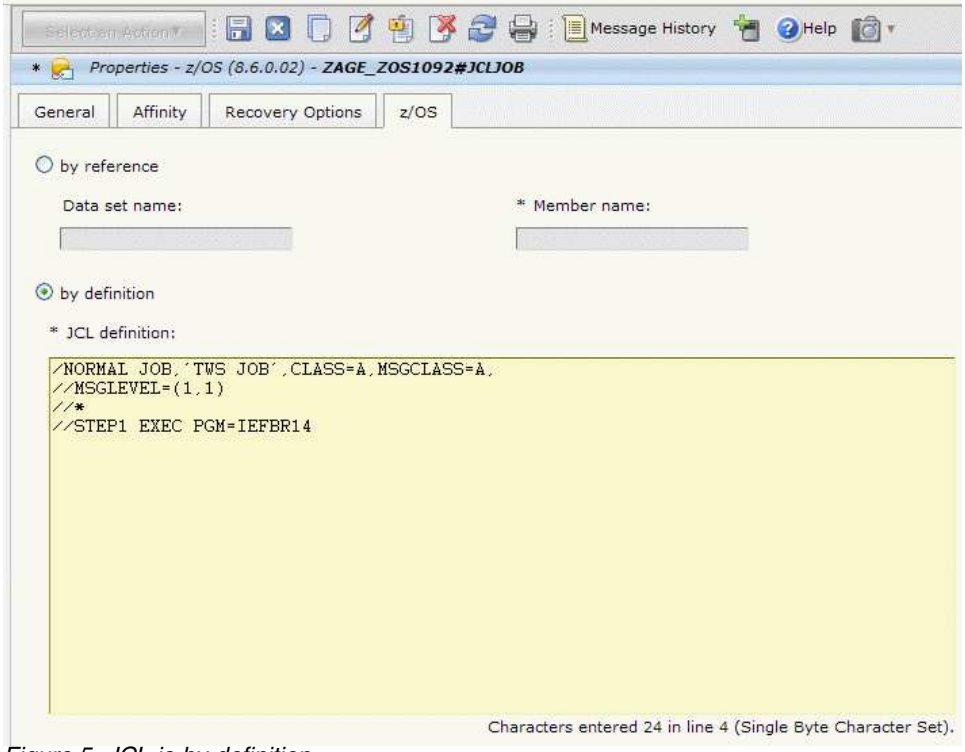


Figure 5. JCL is by definition.

Defining in composer

You can also use the composer command line to define jobs for the agent for z/OS.

The JSDL tagging changes depending on whether in the job the JCL destined for JES is defined `by reference` or `by definition`.

Defining jobs that point to the JCL location in z/OS

The following example shows the definition of a job named `JCLJOBREF`.

The workstation is an agent for z/OS named `ZAGE_ZOS1092`.

The JSDL coding points to the JCL location in the z/OS system.

```
ZAGE_ZOS1092#JCLJOBREF
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jmgr:submitJobFromJSDL xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:jmgr="http://www.ibm.com/xmlns/prod/scheduling/1.0/JobManager"
 xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
 xmlns:jsdljcl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljcl"
 xmlns:sdo="http://www.eclipse.org/emf/2003/SDO">
  <jmgr:JobDefinitionDocument>
   <jsdl:jobDefinition name="JCL">
    <jsdl:application name="jcl">
     <jsdljcl:jcl xsi:type="sdo:EDataObjectAnyType">
      <jsdljcl:JCLParameters xsi:type="sdo:EDataObjectAnyType">
       <jsdljcl:jcl xsi:type="sdo:EDataObjectAnyType">
        <jsdljcl:byRefOrByDef xsi:type="sdo:EDataObjectAnyType">
         <jsdljcl:byReference xsi:type="sdo:EDataObjectAnyType">
          <jsdljcl:dataset xsi:type="sdo:EDataObjectAnyType">TWSTST.TWSA.JOBLIB</jsdljcl:dataset>
          <jsdljcl:member xsi:type="sdo:EDataObjectAnyType">JOB1</jsdljcl:member>
         </jsdljcl:byReference>
        </jsdljcl:byRefOrByDef>
       </jsdljcl:jcl>
```

```
      </jsdljcl:JCLParameters>
      <jsdljcl:JOBParameters xsi:type="sdo:EDataObjectAnyType">
        <jsdljcl:jobStreamName xsi:type="sdo:EDataObjectAnyType">JOBS</jsdljcl:jobStreamName>
        <jsdljcl:inputArrival xsi:type="sdo:EDataObjectAnyType">201206131200
        </jsdljcl:inputArrival>
      </jsdljcl:JOBParameters>
     </jsdljcl:jcl>
    </jsdl:application>
    <jsdl:resources/>
   </jsdl:jobDefinition>
  </jmgr:JobDefinitionDocument>
  <jmgr:Alias>ZA86_ZOS1354#JOBS.PROVA.JNUM-622656411</jmgr:Alias>
  <jmgr:JobId>5e2efa42-1dab-31eb-a8f1-6aaa413a4cec</jmgr:JobId>
  <jmgr:ClientNotifyURI>https://ts6087.enervt.com:31116/JobManagerRESTWeb/JobScheduler/job
    </jmgr:ClientNotifyURI>
  <jmgr:ClientNotifyURI>https:/ts6087.enervt.com:31116/JobManagerRESTWeb/JobScheduler/job
    </jmgr:ClientNotifyURI>
</jmgr:submitJobFromJSDL>
```

Note the following keywords:

<jsdl:application name="jcl">
Specifies that the job is an agent for z/OS job.

<jsdljcl:byReference>
Specifies that the JCL that will be run by JES resides in the z/OS system and only the coordinates of its location are specified here.

<jsdljcl:dataset xsi:type="sdo:EDataObjectAnyType">...</jsdljcl:dataset>
Specifies the name of the data set where the JCL is stored. This name can be up to 44 characters long and is optional. If the data set name is not specified, the agent for z/OS will search for the member name in the data set concatenation library declared for the agent at installation time.

Important: Write the keyword also when you do not provide a data set name. The element must be present regardless of whether there is a value or not.

<jsdljcl:member xsi:type="sdo:EDataObjectAnyType">...</jsdljcl:member>
Specifies the name of the JCL member in the data set. This name can be up to 8 characters long and is required.

<jsdljcl:JOBParameters>
In this section you are required to provide the:

- job stream name
- input arrival time

Defining jobs that include the JCL definition

The following example shows the definition of a job named JCLJOB.

The workstation is an agent for z/OS named ZAGE_ZOS1092.

The JCL statement is embedded in the JSDL definition.

```
ZAGE_ZOS1092#JCLJOB
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdljcl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljcl">
<jsdl:application name="jcl">
 <jsdljcl:jcl>
  <jsdljcl:JCLParameters>
 <jsdljcl:jcl>
 <jsdljcl:byRefOrByDef>
  <jsdljcl:byDefinition>
   <jsdljcl:jclDefinition>
      /NORMAL JOB,'TWS JOB',CLASS=A,MSGCLASS=A,
```

```
      //MSGLEVEL=(1,1)
      //*
      //STEP1 EXEC PGM=IEFBR14
    </jsdljcl:jclDefinition>
  </jsdljcl:byDefinition>
 </jsdljcl:byRefOrByDef>
 </jsdljcl:jcl>
 </jsdljcl:JCLParameters>
 <jsdljcl:JOBParameters>
  <jsdljcl:jobStreamName>${tws.jobstream.name}</jsdljcl:jobStreamName>
  <jsdljcl:inputArrival>${tws.job.ia}</jsdljcl:inputArrival>
 </jsdljcl:JOBParameters>
 </jsdljcl:jcl>
</jsdl:application>
</jsdl:jobDefinition>
DESCRIPTION "Sample JCL Job Definition"
```

Note the following keywords:

<jsdl:application name="jcl">
Specifies that the job is an agent for z/OS job.

<jsdljcl:byDefinition>
Specifies that the JCL that will be run by JES is defined within the JSDL.

<jsdljcl:jclDefinition>
Contains the entire JCL definition.

<jsdljcl:JOBParameters>
In this section you are required to provide the:

- job stream name
- input arrival time

In this particular example, the actual values are replaced by IBM Workload Scheduler variables. They will be resolved by dynamic workload broker at submission time.

Specifying that the JCL contains variables that must be resolved at runtime

If the JCL, without regard to whether it is included in the job definition or is referenced by its location in z/OS, includes variables that will be resolved at runtime by the agent, it must be declared in the `jobDefinition` section of the JSDL definition.

The following example is the definition shown in "Defining jobs that include the JCL definition" on page 54 with the addition of the keyword that specifies that there are variables to be resolved at runtime:

```
ZAGE_ZOS1092#JCLJOB
TASK
<?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl"
xmlns:jsdljcl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljcl"
XMLSchema:text="resolveVariableTable" name="jcl">
<jsdl:application name="jcl">
 <jsdljcl:jcl>
  <jsdljcl:JCLParameters>
 <jsdljcl:jcl>
 <jsdljcl:byRefOrByDef>
  <jsdljcl:byDefinition>
   <jsdljcl:jclDefinition>
      /NORMAL JOB,'TWS JOB',CLASS=A,MSGCLASS=A,
      //MSGLEVEL=(1,1)
      //*
      //STEP1 EXEC PGM=&MODULE
    </jsdljcl:jclDefinition>
  </jsdljcl:byDefinition>
 </jsdljcl:byRefOrByDef>
 </jsdljcl:jcl>
```

```
 </jsdljcl:JCLParameters>
 <jsdljcl:JOBParameters>
  <jsdljcl:jobStreamName>${tws.jobstream.name}</jsdljcl:jobStreamName>
  <jsdljcl:inputArrival>${tws.job.ia}</jsdljcl:inputArrival>
 </jsdljcl:JOBParameters>
 </jsdljcl:jcl>
</jsdl:application>
</jsdl:jobDefinition>
DESCRIPTION "Sample JCL Job Definition"
```

where:

XMLSchema:text="resolveVariableTable"
Specifies that there are variables in the JCL that need to be resolved at runtime. When you submit the job to the agent for z/OS, the definition is transmitted to the agent, together with the table that contains the variable and the corresponding value, where it is processed before being passed on to JES.

Attention: Do not misuse this keyword. If there is no correspondence between the keyword and the presence or absence of variables, the job will fail.

Defining the JCL

How to define the JCL.

When you define the JCLs in the Dynamic Workload Console or in the composer command line, it is important to remember that:

- In the JOB card the programmer name can have a maximum length of 19 characters, instead of 20, since the last character (HEX '30x') is reserved for internal use. The job is not processed if the agent detects a longer programmer name and an error message is issued.
- The name of the JCL must conform to the following syntax rules. It must:
  - Start from column 3.
  - Have length from 1 to 8 alphanumeric (capital A to Z, numbers 0 to 9) or national ($, #, @) characters.

    Note: The system recognizes the following hexadecimal representations of the U.S. National characters in EBCDIC format:

    - $ (dollar) as `X'5B'`
    - # (number) as `X'7B'`
    - @ (at) as `X'7C'`

    In countries other than the U.S., the U.S. National characters represented on terminal keyboards might generate a different hexadecimal representation and cause an error.

    For example, in some countries the $ character may generate `X'4A'`. This implies that, depending on the codepage specified with the CODEPAGE parameter of the TWSOPTS initialization statement for the agent, you must use whatever characters correspond to hex `X'7C'`, `X'5B'`, and `X'7B'` in EBCDIC format. For example, if IBM_280 is specified as the system codepage in TWSOPTS, then within the job name you can use the £ (pound) character which is coded as hex `X'7B'`.
  - Start with a letter or national character, but not with a number.
  - Be followed by at least one blank.

HFS or ZFS file triggering

APAR PI09318 provides you with a file watching utility. After installing the APAR, you find this utility as load-module EELFLWAT in the SEELLMD0 library. You can use it to check for file system changes of HFS or ZFS files and directories, for example when you want to make sure that a file exists before running a job that processes that file. By defining a job that runs this utility, you can implement file dependency, that is a relationship between a file and an operation in which specific activity on the file determines the starting of the operation. For example, you can define the job that runs EELFLWAT as predecessor of the operation depending on the file.

The user running EELFLWAT must have execute (x) access to the directory path that contains files to be monitored.

The SEELSAMP library contains EELFLWAS member as sample JCL to call the file watching utility.

Syntax

As shown in the EELFLWAS member of the SEELSAMP library, the following format is supported for the parameter of the EXEC statement that specifies PGM=EELFLWAT:

```
//FLWATCH  EXEC PGM=EQQFLWAT,PARM='ENVAR()/-c condval  -dea deadline
//   -fi file_path -i interval -r rc -t trace_level'
```

Consider that:

- No continuation character is used. To continue the PARM field, interrupt it at column 72 and continue in column 16 of the next card.
- Separate arguments and values by using a blank character.
- Use `ENVAR()` to pass environment variables to EELFLWAT.
- EELFLWAT returns messages and trace information (if required) in the log of the job used to run the utility. If you use the job log retrieval function, set to Y the User Sysout field in the cleanup options at operation level, both in the database and current plan.
- The arguments are not positional.
- You can use an abbreviated format for all the arguments. Generally you can truncate the arguments to any position following the first character, except for `deadline` that requires at least three characters.

Arguments

-condition | -c
The condition to be checked. Valid values are:

wcr | waitCreated
Waits until the file exists. If the file already exists, **filewatch** exits immediately. If the `-filename` argument specifies a directory, the process waits until the directory exists and contains a new file.

wmr | waitModificationRunning
Waits until the file size or modification time changes. If the `-filename` argument specifies a directory, the process waits until the size or earlier file modification time changes, when a file is created, modified, or deleted.

wmc | waitModificationCompleted
Checks that the file size or modification time stopped changing, meaning that **filewatch** waits for three search intervals without any change. If the `-filename` argument specifies a directory, the process checks the size or the earlier file modification time change, for example if the number of directory files and the earlier file modification time does not change within three search intervals.

wmrc | waitModificationRunningCompleted
Waits until the file size or modification time changes and stops changing, meaning that, after the first change, **filewatch** waits for three search intervals without any further change. If the `-filename` argument specifies a directory, the process checks the size or the earlier file modification time change, for example if the number of directory files and the earlier file modification time does not change within three search intervals.

wdl | waitDelete
Stops running when the file is deleted. If the `-filename` argument specifies a directory, the process waits until a file is deleted from the directory.

-deadline | -dea
The deadline period, expressed in seconds. Valid formats are:

- An integer in the range 0 to 31536000 (the upper value corresponds to one year). To have **filewatch** performing an indefinite loop, specify 0.
- *hh:mm:ss*, in the range 00:00:01 to 24:00:00, to select a time within the same day when **filewatch** started.

  It corresponds to the GMT value. To specify a value different from the GMT, use the `ENVAR` parameter as shown in "Example" on page 59. For details about setting the TZ environment variable, see the *UNIX System Services Command Reference*.

-filename | -fi
The file path to be processed. You can embed blank or special characters, by using double quotation marks. Wildcard characters are not supported. To include more than one file in the monitoring process, you can store the files in a directory and use a file path specifying that directory.

-interval | -i
The file search interval, expressed in seconds. Specify an integer in the range:

- 5–3600, when specifying **wcr** or **wdl** as condition value.
- 30–3600, otherwise.

The default is 60.

-returncode | -rc
The exit return code, if the file is not found by the deadline. Specify an integer in the range 0 to 255. The returncode value is ignored if you specify 0 as deadline value. The default is 4.

-trace | -t
Trace level for internal logging and traces. Possible values are:

0 To receive error messages only.

1 Indicates the *fine* level, to receive the most important messages with the lowest volume.

2 Indicates the *finer* level, to activate entry and exit traces.

3 Indicates the *finest* level, to receive the most detailed tracing output.

The default value is 0.

You find the trace output in the log of the job that run **filewatch**.

Example

```
//FLWATCH  EXEC PGM=EELFLWAT,PARM='ENVAR(TZ=GMT-1CET)/-co wcr  -dead 30
//             -fi /u/falsi/prova -int 30 -t 2'
```

In this example, `ENVAR(GMT-1CET)` is used to change the time zone to Central Europe Time. GMT-1 is the time zone value and CET is the selected daylight saving time.

Submitting jobs

All agent for z/OS jobs can be either part of a job stream and be submitted in a plan, or be submitted at any time using the `conman submit` commands or proper Dynamic Workload Console panels.

Submit agent for z/OS jobs just as you submit all other IBM Workload Scheduler jobs.

When you submit a job of type JCL through a production plan, or more specifically from a `conman` command line or the **Submit** windows of the Dynamic Workload Console, the job is processed by dynamic workload broker and dispatched to the agent for z/OS specified in the job definition.

The agent receives the job submission requests. The job submission requests include either the body of the JCL to be passed on to JES*n* for execution, or a reference to a member of a partitioned data set that includes the JCL. If the reference names only the member, but not the data set, the agent will search the member name in the data set concatenation library declared for the agent at installation time (which defaults to EELJBLIB).

If variable substitution is requested in the JCL, the variable tables that include the variables featured in the JCL (and the respective values) are sent with the submission request to the agent.

Upon receiving a job submission request, the HT task briefly stores the job in the HTREF service database, that is part of the agent for z/OS. When the request involves a JCL by reference or variable substitution, the referenced JCL is fetched or the variables are resolved inside of HTREF. A failover mechanism on the agent keeps track of the jobs that are not processed if for some reason the agent should become unlinked or fail, and picks up the submission thread as soon as the communication between IBM Workload Scheduler and the agent has restarted. See "Understanding resynchronization messages" on page 99 for details on this process.

Figure 6 on page 60 tracks the route followed by a job as it arrives to the agent for z/OS and is passed on to JES for processing.

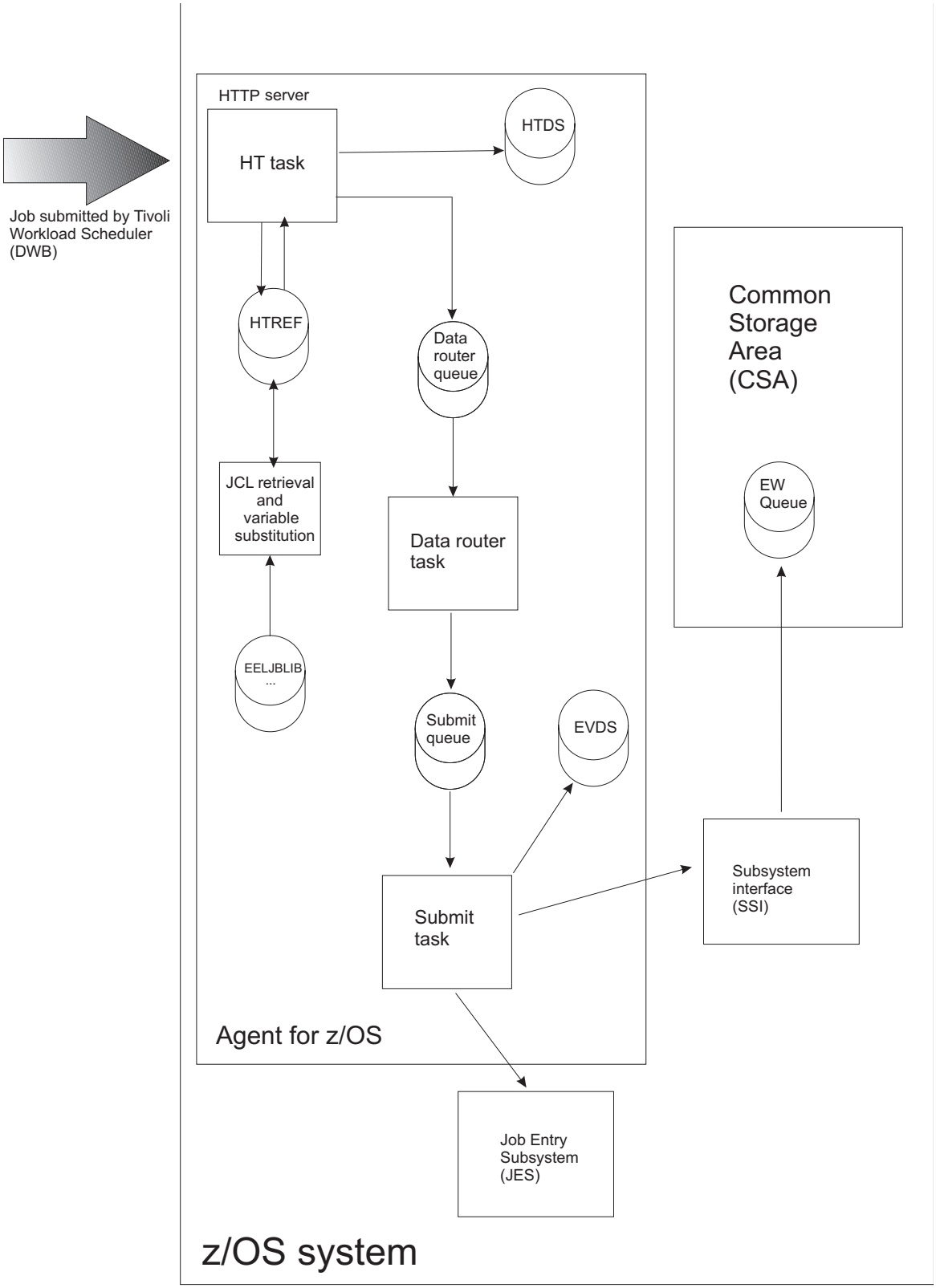


Figure 6. The route followed by a job within the agent for z/OS.

When the JCL arrives at the submit task, the job is submitted to JES via the EELBRDS data set. The EELBRDS data set is used to allocate a JES internal reader.

Note that the JCLs coming from IBM Workload Scheduler are stored on disk (HTDS). This guarantees that, if the agent is interrupted, the jobs are not lost, but sent to JES when the agent resumes.

After the JCL is added to the EELBRDS data set, the event data set (EVDS) is updated with the new job state and an update is also sent to dynamic workload broker through the z/OS subsystem interface (SSI).

After a job is processed, its status is sent back to IBM Workload Scheduler as an event.

Figure 7 on page 62 tracks the route followed by the events related to the statuses of submitted jobs as they are returned by JES, processed by the agent for z/OS, and sent back to IBM Workload Scheduler.

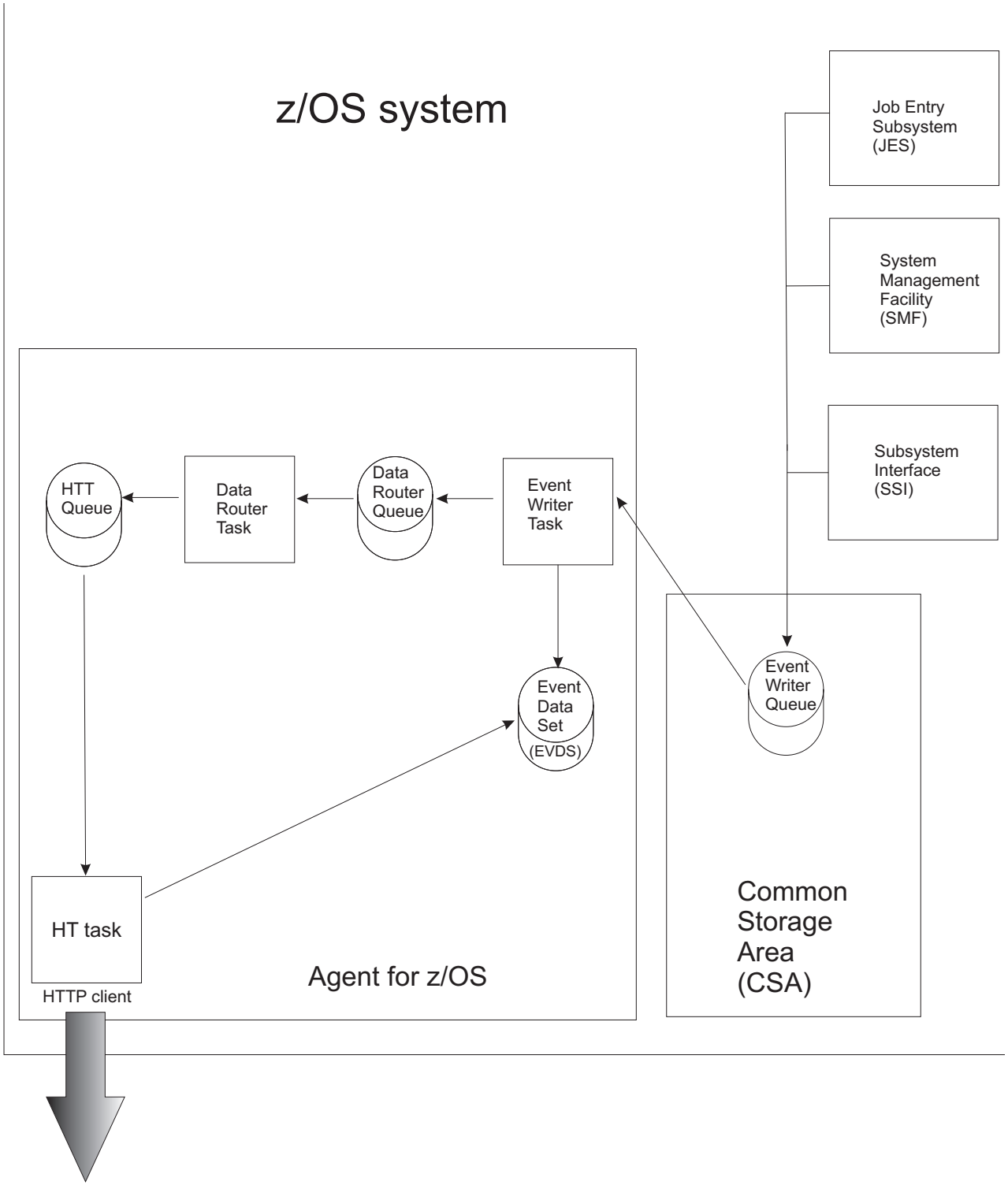


Figure 7. The route followed by a status event within the agent for z/OS as it is returned by JES on its way to IBM Workload Scheduler.

The agent uses the normal tracking mechanism based on SMF and JES exits to track the status of all the jobs in the system and to send them back to the dynamic workload broker.

Note that the events are stored in the event writer queue in CSA and also in the event data set (EVDS) on disk. This guarantees that, if the agent is interrupted, they are not lost, but sent to IBM Workload Scheduler when the agent resumes.

Back in IBM Workload Scheduler, the events filter component of dynamic workload broker receives all the events coming from the agent for z/OS and matches them against a job table. If there is a match, the event is processed, otherwise it is discarded.

The Output manager handles job log requests from IBM Workload Scheduler by getting the job output from the JES spool (if still available).

When a job submission request is received by an agent for z/OS in a sysplex environment, the agent executes the job, but the job can be routed by JES to any other node of the sysplex where another agent tracks it and sends the event back to IBM Workload Scheduler that checks if the event is related to one of the jobs submitted and if this is the case updates its status.

The use of NJE (Network Job Entry) with the agent for z/OS is not supported because it can result in faulty tracking of a job state.

Using variables in your jobs

You can include variables in your job definition.

The variables are resolved at submission time. They can be grouped into two types depending on where they are resolved:

- Variables that are resolved by IBM Workload Scheduler at a dynamic workload broker level before the job is submitted to the agent for z/OS. They must be placed in the JSDL portion of the job definition.

  You can also pass properties variables or job output between 2 jobs in the same job stream instance. For more information about which properties you can use as variables or how you can pass the job output, see "Variables passing between jobs in the same job stream instance" on page 65.
- Variables that are resolved by the agent for z/OS before it submits the JCL to JES. They must be placed in the JCL embedded or referenced in the job definition.

Variables resolved by Tivoli Workload Scheduler

The variables are assigned their values by means of dynamic workload broker at job submission time before the job is passed to the agent for z/OS.

The following variables are supported:

Table 12. Supported variables in JSDL definitions

Variable name	Description
tws.host.workstation	Name of the host workstation
tws.job.date	Date of the submitted job.

Table 12. Supported variables in JSDL definitions (continued)

Variable name	Description
tws.job.fqname	Fully qualified name of the job (UNISON_JOB)
tws.job.ia	Input arrival time of the job
tws.job.interactive	Job is interactive. Values can be `true` or `false`. Applies only to backward-compatible jobs.
tws.job.logon	Credentials of the user who runs the job (LOGIN). Applies only to backward-compatible jobs.
tws.job.name	Name of the submitted job
tws.job.num	Number of the submitted job.
tws.job.priority	Priority of the submitted job
tws.job.promoted	Job is promoted. Values can be `YES` or `No`. For more information about promotion for dynamic jobs, see the section about promoting jobs scheduled on dynamic pools in *Scheduling Workload Dynamically*.
tws.job.recnum	Record number of the job.
tws.job.resourcesForPromoted	Quantity of the required logical resources assigned on a dynamic pool to a promoted job. Values can be `1` if the job is promoted or `10` if the job is not promoted. For more information about promotion for dynamic jobs, see the section about promoting jobs scheduled on dynamic pools in *Scheduling Workload Dynamically*.
tws.job.workstation	Name of the workstation on which the job is defined
tws.jobstream.id	ID of the job stream that includes the job (UNISON_SCHED)
tws.jobstream.name	Name of the job stream that includes the job (UNISON_SCHED)
tws.jobstream.workstation	Name of the workstation on which the job stream that includes the job is defined
tws.master.workstation	Name of the master domain manager (UNISON_MASTER)
tws.plan.date	Start date of the production plan (UNISON_SCHED_DATE)
tws.plan.date.epoch	Start date of the production plan, in epoch format (UNISON_SCHED_EPOCH)
tws.plan.runnumber	Run number of the production plan (UNISON_RUN)

When you include any of these variables in the job definition in composer, put the $ (dollar) sign before the variable and write the variable within braces; for example, ${tws.master.workstation}.

Variables passing between jobs in the same job stream instance

In many scenarios the job output or a job property of the first job in a job stream can be the input for the execution of the successive jobs in the same job stream instance. This is valid also for JOBS job stream.

In the scenario, you have *JobA* and *JobB* in the same job stream instance, and *JobA* passes some variables values to the *JobB* at execution time. You can pass the following variables from *JobA* to *JobB*:

- *JobA* exports some properties and *JobB* references these properties in its definition as variables in a predefined format. At execution time the *JobB* variables are automatically resolved. The job properties that you can export depend on the job type you are defining. See "Passing job properties from one job to another in the same job stream instance."
- *JobA* exports its standard output value and *JobB* references this standard output as property in *JobB* definition as variable. At execution time the *JobB* variable is automatically resolved. See "Passing job standard output from one job to another in the same job stream instance" on page 67.
- *JobA* exports its standard output value and *JobB* references this standard output as its standard input value. This option is valid only for executable jobs. See "Passing job standard output from one job to another as standard input in the same job stream instance" on page 68.

Note: The USERJOBS job stream created by IBM Workload Scheduler processes, does not support the variables passing between jobs that belong to it.

Passing job properties from one job to another in the same job stream instance:

You can export some job properties from one job on IBM Workload Scheduler agent for z/OS to another in the same job stream instance

You can export some job properties from one job on IBM Workload Scheduler agent for z/OS to another in the same job stream instance. To add a job property within another job definition, that it is resolved locally on the agent at run time, use the following syntax:

```
${job:<JOB_NAME>.<property_name>}
```

where <*JOB_NAME*> is the name value or the alias name value of the job from which you are exporting the property values and <*property_name*> is the property that you are referring to. The <*property_name*> value is case insensitive.

Table 13 shows the list of properties that you can pass from dynamic job on IBM Workload Scheduler agent for z/OS to another job and indicate the mapping between the Extra information properties of the job and the properties that you can use.

Table 13. Properties for dynamic jobs on IBM Workload Scheduler agent for z/OS

Dynamic job on IBM Workload Scheduler agent for z/OS properties that can be pass in another job definition	Dynamic job on IBM Workload Scheduler agent for z/OS Extra Information properties
${job:<*JOB_NAME*>.zAgentJESId}	JES Id
${job:<*JOB_NAME*>.zAgentJobName}	Job Name
${job:<*JOB_NAME*>.zAgentStartReaderTime}	Start Reader Time

Example

The following example demonstrates how variables can be passed from job that run on IBM Workload Scheduler agent for z/OS to another executable job that run in the same job stream instance. The WIN92MAS#JS_PROP job stream contains ZSPD_ZOS1274#DDRIVEN_JOB used with alias JOBA and NC112016#JOBB jobs. The NC112016#JOBB executable job references the following properties of the JOBA job that is defined on IBM Workload Scheduler agent for z/OS:

- zAgentJESId
- zAgentJobName
- JOBA.zAgentStartReaderTime

The database definitions:

```
SCHEDULE WIN92MAS#JS_PROP
:
ZSPD_ZOS1274#DDRIVEN_JOB AS JOBA
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdljcl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljcl" name="JCL">
  <jsdl:application name="jcl">
    <jsdljcl:jcl>
   <jsdljcl:JCLParameters>
    <jsdljcl:jcl>
     <jsdljcl:byRefOrByDef>
      <jsdljcl:byDefinition>
       <jsdljcl:jclDefinition>//JOB JOB
//S1 EXEC PGM=IEFBR14</jsdljcl:jclDefinition>
      </jsdljcl:byDefinition>
     </jsdljcl:byRefOrByDef>
    </jsdljcl:jcl>
   </jsdljcl:JCLParameters>
   <jsdljcl:JOBParameters>
    <jsdljcl:jobStreamName>${tws.jobstream.name}</jsdljcl:jobStreamName>
    <jsdljcl:inputArrival>${tws.job.ia}</jsdljcl:inputArrival>
   </jsdljcl:JOBParameters>
  </jsdljcl:jcl>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Added by composer for job stream: WIN92MAS#JS_PROP."
 RECOVERY STOP

NC112016#JOBB
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:XMLSchema="http://www.w3.org/2001/XMLSchema" xmlns:
jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" XMLSchema:
text="resolveVariableTable" name="executable">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false">
            <jsdle:script>
echo JES Id = ${job:JOBA.zAgentJESId}
echo Job Name = ${job:JOBA.zAgentJobName}
echo Start Reader Time = ${job:JOBA.zAgentStartReaderTime}
</jsdle:script>
        </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Added by composer for job stream: WIN92MAS#JS_PROP."
 RECOVERY STOP
 FOLLOWS JOBA
```

Passing job standard output from one job to another in the same job stream instance:
You can export the job standard output from a dynamic job to another in the same job stream instance.

To add a job standard output within another job definition, that it is resolved locally on the agent at runtime, use the following syntax:

```
${job:<JOB_NAME>.stdlist}
```

where <*JOB_NAME*> is the name value or the alias name value of the job from which you are exporting the job standard output.

Example

The following example demonstrates how variables can be passed from job that run on IBM Workload Scheduler agent for z/OS to another executable job that run in the same job stream instance. The WIN92MAS#JS_PROP job stream contains ZSPD_ZOS1274#DDRIVEN_JOB used with alias JOBA and NC112019#JOBD jobs. The NC112019#JOBD executable job references the JOBA standard output. The database definitions:

```
SCHEDULE WIN92MAS#JS_PROP
:
ZSPD_ZOS1274#DDRIVEN_JOB AS JOBA
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdljcl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljcl" name="JCL">
  <jsdl:application name="jcl">
    <jsdljcl:jcl>
   <jsdljcl:JCLParameters>
    <jsdljcl:jcl>
     <jsdljcl:byRefOrByDef>
      <jsdljcl:byDefinition>
       <jsdljcl:jclDefinition>//JOB JOB
//S1 EXEC PGM=IEFBR14</jsdljcl:jclDefinition>
      </jsdljcl:byDefinition>
     </jsdljcl:byRefOrByDef>
    </jsdljcl:jcl>
   </jsdljcl:JCLParameters>
   <jsdljcl:JOBParameters>
    <jsdljcl:jobStreamName>${tws.jobstream.name}</jsdljcl:jobStreamName>
    <jsdljcl:inputArrival>${tws.job.ia}</jsdljcl:inputArrival>
   </jsdljcl:JOBParameters>
  </jsdljcl:jcl>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Added by composer for job stream: WIN92MAS#JS_PROP."
 RECOVERY STOP

NC112019#JOBD
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" name="executable">
  <jsdl:application name="executable">
    <jsdle:executable interactive="false">
            <jsdle:script>echo &quot;stdlist: ${job:JOBA.stdlist}&quot;</jsdle:script>
        </jsdle:executable>
  </jsdl:application>
</jsdl:jobDefinition>
```

```
 DESCRIPTION "Added by composer for job stream: WIN92MAS#JS_PROP."
 RECOVERY STOP
 FOLLOWS JOBA
END
```

Passing job standard output from one job to another as standard input in the same job stream instance:
You can export the job standard output from a dynamic job to another job as standard input in the same job stream instance.

To add a job standard output within another job definition, that it is resolved locally on the agent at runtime, use the following syntax:

```
${job:<JOB_NAME>.stduri}
```

where *<JOB_NAME>* is the name value or alias name value of the job from which you are exporting the job standard output.

Example

The following example demonstrates how variables can be passed from job that run on IBM Workload Scheduler agent for z/OS to another executable job that run in the same job stream instance. The `WIN92MAS#JS_PROP` job stream contains `ZSPD_ZOS1274#DDRIVEN_JOB` used with alias `JOBA` and `NC112019#JOBC` jobs. The `NC112019#JOBC` executable job references the `JOBA` standard output. The database definitions:

```
SCHEDULE WIN92MAS#JS_PROP
:
ZSPD_ZOS1274#DDRIVEN_JOB AS JOBA
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdljcl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdljcl" name="JCL">
  <jsdl:application name="jcl">
    <jsdljcl:jcl>
   <jsdljcl:JCLParameters>
    <jsdljcl:jcl>
     <jsdljcl:byRefOrByDef>
      <jsdljcl:byDefinition>
       <jsdljcl:jclDefinition>//JOB JOB
//S1 EXEC PGM=IEFBR14</jsdljcl:jclDefinition>
      </jsdljcl:byDefinition>
     </jsdljcl:byRefOrByDef>
    </jsdljcl:jcl>
   </jsdljcl:JCLParameters>
   <jsdljcl:JOBParameters>
    <jsdljcl:jobStreamName>${tws.jobstream.name}</jsdljcl:jobStreamName>
    <jsdljcl:inputArrival>${tws.job.ia}</jsdljcl:inputArrival>
   </jsdljcl:JOBParameters>
  </jsdljcl:jcl>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Added by composer for job stream: WIN92MAS#JS_PROP."
 RECOVERY STOP

NC112019#JOBC
 TASK
    <?xml version="1.0" encoding="UTF-8"?>
<jsdl:jobDefinition xmlns:XMLSchema="http://www.w3.org/2001/XMLSchema" xmlns:
jsdl="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdl" xmlns:
jsdle="http://www.ibm.com/xmlns/prod/scheduling/1.0/jsdle" XMLSchema:text="resolveVariableTable"
                                                           name="executable">
  <jsdl:application name="executable">
```

```
    <jsdle:executable input="${job:JOBA.stduri}" interactive="false" path="cat"/>
  </jsdl:application>
</jsdl:jobDefinition>
 DESCRIPTION "Added by composer for job stream: WIN92MAS#JS_PROP."
 RECOVERY STOP
 FOLLOWS JOBA
END
```

Variables resolved by the agent for z/OS

These variables are resolved by the agent for z/OS in the JCL before the JCL is passed on to JES for execution.

The following types of variables are supported:

- User-defined
- Predefined
- JCL tailoring directives

Configuring the agent to run variable substitution

To be able to run this type of variable substitution, you must configure some keywords of the TWSOPTS statement.

To be able to run this type of variable substitution, take one of the following actions:

- Set the VARSUB keyword of the TWSOPTS statement to YES. This means that variable substitution occurs from the beginning of the JCL for all the jobs defined to run on the agent for z/OS.
- Set the VARSUB keyword of the TWSOPTS statement to SCAN and specify the directive `//*%OPC SCAN` in your job. Substitution in the job starts where the SCAN directive is found.

If you want to bypass variable substitution errors, then set the VARFAIL keyword. If you want to apply variable substitution also to inline procedures, then use the VARPROC statement. See the TWSOPTS configuration statement for details.

Coding variables in JCL

Coding variables in JCL follows certain rules.

Variable names, either user-defined or supplied with the product, can be of up to 8 alphanumeric characters, the first of which must be alphabetic. Variable values can be of up to 44 alphanumeric characters.

When using a variable in a job, precede it with an ampersand (&), a percent sign (%), or a question mark (?). The symbol preceding the variable determines how the variable is resolved:

Ampersand (&)

These variables are substituted from left to right within the line. Ampersand variables correspond to the standard variables in z/OS JCL procedures and behave accordingly. Refer to *JCL Reference*.

If an &-variable is immediately followed by a % variable (that is, there is no intervening termination character), a compound variable is formed. See Compound variables. A compound variable is also formed if an &-variable immediately follows a ?-variable.

Any string that begins with && is not substituted. This is because the double ampersand within JCL is usually used to denote a temporary data set. Any such strings are unaffected by the variable substitution.

Percent sign (%)

These variables can be used to form simple variables and compound variables.

Simple variables

If the variable is preceded by a % and ended by a period or any termination character other than %, a value is assigned to the variable, and substitution, for this variable, completes.

Compound variables

Using JCL substitution, you can form *compound variables*. A compound variable is made up of a concatenation of:

- A variable (of any type) followed by a percent variable with no intervening periods or other termination symbols
- A question mark variable followed by an ampersand variable with no intervening periods or other termination symbols

The values of the percent variables making up a compound variable are not substituted directly. Instead, these values are used to form new variables, which have their own values assigned. These variables are resolved in a series of passes. The individual variables making up the compound variable are resolved, moving from right to left.

For example, consider the following line of JCL from a job:

```
//STEPLIB DD DSN=MY.&DATA%SET,DISP=OLD
```

Assume that SET has been given a value of LIB. After the first pass, the variable DATA%SET becomes variable DATALIB because the right-most percent variable is resolved on the first pass. This first pass has now formed a new variable, DATALIB, which the agent will try to resolve on its next pass across this line of JCL.

Compound variables can be made up of a sequence of many %-variables. Consider the following:

```
//DDNAME1 DD DSN=MY.%VAR1%VAR2%VAR3....DATA,DISP=OLD
```

Assume that VAR3 has value SIX and VAR2SIX has value JUNE. On the first pass over this line of JCL, the variable %VAR1%VAR2%VAR3....DATA becomes %VAR1%VAR2SIX...DATA. On the second pass, the variable %VAR1%VAR2SIX. becomes %VAR1JUNE..DATA. The value assigned to %VAR1JUNE. determines the final value that is substituted.

At every substitution, a period was discarded when the variable was substituted. You must specify the correct number of periods to ensure that the substitution is performed correctly. In the preceding example, an extra period was required to denote the beginning of the second-level data set qualifier.

In the next example, you need only one parenthesis to complete the compound variable. This is because the parenthesis is not discarded at substitution.

```
//DDNAME1 DD DSN=MY.%VAR1%VAR2%VAR3(MEMBER),DISP=OLD
```

Question mark (?)

Question mark variables are *positional*; that is, you can specify in which column on the line the variable value should begin when the variable is substituted. The position at which the value is placed is specified in the job where the variable is used. For example:

```
?VAR1.
```

will cause the value of VAR1 to be placed on the line and column that the variable appears on.

```
?nnVAR1.
```

will cause the value of VAR1 to be placed on the line that the variable appears on, starting at the column number specified by *nn*.

More than one ?-variable can appear on a JCL line. The positions of the variables themselves have no influence on the positions of the variable values. These positions are decided by the column number specified for the variable. For example:

```
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
        ?20VAR1.?9VAR2.
```

where VAR1 is APRIL and VAR2 is MAY (the scale line has been included only for example purposes), the result after variable substitution would be:

```
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
       MAY       APRIL
```

The value of ?-variables is evaluated in the same way as for &- and %-variables, and in the same sequence. However, ?-variables are substituted only after all percent and ampersand variables have been substituted. This is because the value of the ?-variable can be placed only in areas of the line that are blank. The agent can only know which areas of a line will be blank after ampersand and percent substitution has occurred.

Tabular variables cannot overlap. That is, the values of two different variables cannot be defined to occupy the same space on a line. The space that the variables themselves originally take up is ignored when substitution occurs. For example:

```
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
        ?20VAR1.?21VAR2.
/*
```

where VAR1 is APRIL and VAR2 is MAY, the substitution would be invalid because the two variables are attempting to use columns 21, 22, and 23.

The agent changes the space occupied by the variable to spaces, if it is not covered by the substituted value. For example:

```
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
THIS IS?40VAR1. THE STANDARD DATA.          IS A WET MONTH.
```

VAR1 is APRIL. After substitution, the line becomes:

```
//SYSIN DD *
....+....1....+....2....+....3....+....4....+....5....+....6....+....7..
THIS IS         THE STANDARD DATA.     APRIL IS A WET MONTH.
```

The agent has changed the space occupied by the variable to spaces. The other data in the line does not move.

Note: Predefined variables do not have an implied position. When these variables are specified as tabular variables, you must include the column number. For example, `?OADID` will not be accepted; however, `?20OADID` is valid: the application ID is substituted at column 20.

You can use a variable repeatedly within the job using different prefix symbols.

An ampersand or percent variable can be assigned a value that is itself a variable.

A period denotes the end of a variable name. To maintain compatibility with variable substitution within z/OS JCL procedures, the agent for z/OS will assume that a variable has ended (even if the completing period is missing) if the variable is followed by one of the symbols listed in the following table.

Table 14. Symbols that mark the end of variables.

Symbol	Description
,	Comma
/	Forward slash
'	Single quote
(	Left parenthesis
)	Right parenthesis
*	Asterisk
+	Plus sign
-	Dash
=	Equals sign
	Blank (b)

For example, if `LIBRARY` is given the value LINKLIB for the following statement:

```
//STEPLIB DD DSN=MY.&LIBRARY.(HDEAQ03),DISP=SHR
```

or the following statement (without the completing period):

```
//STEPLIB DD DSN=MY.&LIBRARY(HDEAQ03),DISP=SHR
```

The JCL line becomes as follows:

```
//STEPLIB DD DSN=MY.LINKLIB(HDEAQ03),DISP=SHR
```

The product assumes that the variable `LIBRARY` ends when it detects the left parenthesis '('.

The completing period is discarded when a variable is substituted. Other termination symbols are left in place.

User-defined variables

You can add your own defined variables in the JCLs.

The names of user-defined variables can be of up to 8 alphanumeric characters, the first of which must be alphabetic. A longer name is taken as not valid and the variable is not processed. An error is recorded in EELMLOG.

Variable values can be of up to 44 alphanumeric characters. Longer values are truncated to 44 characters.

To define these variables, use the variable table definition in the Dynamic Workload Console. To define a variable table and to add variables:

1. From the navigation toolbar, click **Administration>Workload Design>Manage Workload Definitions**.
2. Select the IBM Workload Scheduler engine when prompted, and provide a valid userid and password if required.
3. Select **New** or **Search** and then **Variable table**.

Alternatively, you can use the **composer vartable** command.

Variable tables can be assigned at run cycle, job stream, and workstation level.

The variables (and the values) included in a JCL are forwarded to the agent for z/OS at submission time in a single table. If the scheduler finds that the same variable name is present in more than one of the tables associated with the JCL, it picks the first value according to this sequence:

1. Run cycle
2. Job stream
3. Workstation

Note: The agent does not process variables defined in the global (default) variable table.

Predefined variables

Predefined variables are supplied with IBM Workload Scheduler for use with the agent for z/OS.

The agent never tries to read variable definitions for these variables from a variable table. The variables can be of the following types:

- Job stream-related
- Job-related
- Date-related
- Dynamic-format
- Temporary

Job stream-related variables:

These variables are related to information about the job stream instance.

Predefined job stream-related variables are listed in the following table:

Table 15. Predefined job stream-related variables

Variable name	Length (in bytes)	Description
OADID	16	Job stream name.
ODAY	1	Job stream instance input arrival day of the week (1-7); 1 represents Mon., 7 represents Sun.

Table 15. Predefined job stream-related variables (continued)

Variable name	Length (in bytes)	Description
ODD	2	Job stream instance input arrival day of month, in DD format.
ODDD	3	Job stream instance input arrival day of the year, in DDD format.
ODMY1	6	Job stream instance input arrival date in DDMMYY format.
ODMY2	8	Job stream instance input arrival date in DD/MM/YY format.
OHH	2	Job stream instance input arrival hour in HH format.
OHHMM	4	Job stream instance input arrival hour and minute in HHMM format.
OMM	2	Job stream instance input arrival month in MM format.
OMMYY	4	Job stream instance input arrival month and year in MMYY format.
OWW	2	Job stream instance input arrival week of the year in WW format.
OWWD	3	Job stream instance input arrival week, and day within week, in WWD format, where WW is the week number within the year, and D is the day within the week.
OWWLAST	1	A value, Y (yes) or N (no), that indicates whether the job stream instance input arrival date is in the last week of the month.
OWWMONTH	1	A value between 1 and 6 that indicates the job stream instance input arrival week-in-month, where each new week begins on a Monday. For example, consider these occurrence input arrival dates for the month of March in 1997: **Date OWWMONTH** **Saturday 1st** 1 **Monday 3rd** 2 **Monday 31** 6
OYMD	8	Job stream instance input arrival date in YYYYMMDD format.
OYM	6	Job stream instance input arrival month within year in YYYYMM format.
OYMD1	6	Job stream instance input arrival date in YYMMDD format.
OYMD2	8	Job stream instance input arrival date in YY/MM/DD format.
OYMD3	10	Job stream instance input arrival date in YYYY/MM/DD format.
OYY	2	Job stream instance input arrival year in YY format.
OYYDDD	5	Job stream instance input arrival date as a Julian date in YYDDD format.

Table 15. Predefined job stream-related variables (continued)

Variable name	Length (in bytes)	Description
OYYMM	4	Job stream instance input arrival month within year in YYMM format.
OYYYY	4	Job stream instance input arrival year in YYYY format, for example, 1997.

Job-related variables:

These variables are related to information about the job instance.

Predefined job-related variables are listed in the following table:

Table 16. Predefined job-related variables

Variable name	Length (in bytes)	Description
OSSID	16	Subsystem name of the agent for z/OS.

Date-related variables:

These variables are related to the current date and time; that is, the time and date on which the job was submitted.

Predefined date-related variables are listed in the following table:

Table 17. Predefined date-related variables

Variable name	Length (in bytes)	Description
CDAY	1	Current day of the week; 1 represents Monday, 7 represents Sunday.
CDD	2	Current day of month in DD format.
CDDD	3	Day number in the current year.
CDDMMYY	6	Current date in DDMMYY format.
CHH	2	Current time in HH format.
CHHMM	4	Current hour and minute in HHMM format.
CHHMMSS	6	Current hour, minute, and second in HHMMSS format.
CHHMMSSX	8	Current hour, minute, second, and hundredths of seconds in HHMMSSXX format.
CMM	2	Current month in MM format.
CMMYY	4	Current month within year in MMYY format.
CWW	2	Week number in the current year.
CWWD	3	Current day within week in WWD format, where WW is the week number within the year, and D is the day within the week.
CYMD	8	Current date in YYYYMMDD format.
CYY	2	Current year in YY format.
CYYDDD	5	Current Julian date in YYDDD format.

Table 17. Predefined date-related variables (continued)

Variable name	Length (in bytes)	Description
CYYMM	4	Current month within year in YYMM format.
CYYMMDD	6	Current date in YYMMDD format.
CYYYY	4	Current year in YYYY format, for example, 1997.
CYYYYMM	6	Current month within year in YYYYMM format.

Dynamic-format variables:

Predefined dynamic-format variables are time-and-date-related.

You define the format you require for these variables using the SETFORM directive. For example, if you want to substitute the occurrence input arrival date with the format MM:DD:YY, you define the dynamic variable OCDATE as follows:

```
//*%OPC SETFORM OCDATE=(MM:DD:YY)
```

When you have defined the format of a dynamic-format variable by using the SETFORM directive, you can use a different format later in the job by redefining the same variable with another SETFORM directive.

The following table lists these variables.

Table 18. Predefined dynamic-format variables

Variable name	Description
CDATE	Current date.
CTIME	Current time.
OCDATE	Job stream instance input arrival date.
OCTIME	Job stream instance input arrival time (hours and minutes).
OPIADATE	Job instance input arrival date (if blank, this takes the value of the job stream instance input arrival date).
OPIATIME	Job instance input arrival time (if blank, this takes the value of the job stream instance input arrival time).

Temporary variables:

You can create temporary variables using the SETVAR directive.

You can create temporary variables using one of the following:

- An arithmetic expression on the date-related or time-related variables.
- A substring of another variable.
- The result of an arithmetic addition or subtraction.
- Concatenated strings or variables set to an alphanumeric value.

For example, you might want to refer to the first workday of the next week after the job stream input arrival date in the format YY/MM/DD. You do this by creating a temporary variable from the supplied variable, OYMD2, using the SETVAR directive. The temporary variable is assigned the value (date) of the first workday after the job stream input arrival date like this:

Example

```
//*%OPC SCAN
//*%OPC SETVAR TVAR=(OYMD2+1WK)
```

If the job stream input arrival date is 12/07/20, and the first working day of the following week Monday 12/07/27, TVAR will be assigned the value 12/07/27. You can now refer to TVAR as a normal variable through the rest of the job: you can also give it a new value later in the job.

For details, see "SETVAR directive" on page 80.

JCL tailoring directives

The agent for z/OS uses special comment statements, called *directives*, to manage the inclusion and exclusion of lines and to control aspects of variable substitution.

The directives are:

- SCAN
- SETFORM
- SETVAR
- BEGIN and END
- FETCH

The general syntax of the directives is:

- Each directive must begin on a new 80-byte line.
- All directives begin with `//*%OPC` in columns 1 to 7 followed by at least one space.
- Directive parameters can be coded in any order.
- Directive parameters can occur more than once in the same directive.
- Directive parameters are separated by commas with no embedded blanks between parameters on the same line.
- If more than one parameter value is specified, parentheses are required.
- A directive specification cannot exceed 71 characters. It can be continued on a new line if the directive is split by a comma after a complete or partial parameter.
- Positions 72 to 80 are ignored.
- Each continuation line must begin with `//*%OPC` in columns 1 to 7 followed by a least one space.
- After a directive is successfully executed, the `//*%OPC` string is changed to `//*>OPC`.

If a line begins with `//*%OPC` and none of the known directives is found, the job substitution routines of the agent treat any other directives that it finds as "unknown," and will take no action.

Note: If you set VARSUB(SCAN) in the TWSOPTS statement, the SCAN directive must be present in the JCL in order to process all the subsequent directives.

SCAN directive:
Purpose

If the VARSUB keyword of the TWSOPTS statement is set to SCAN, this directive, when found in the JCL , informs the agent for z/OS that variable substitution

should start from this line. This applies also for processing the directives, meaning that the processing of the directives starts from the line where SCAN is specified.

Syntax

Usage Notes

The SCAN directive is honored only if the VARSUB parameter of the TWSOPTS statement is set to SCAN.

Assuming that VARSUB(SCAN) is specified, in the following example, `MODULE` will not be substituted because it is before the SCAN directive. The variable `LIBRARY`, occurring after the SCAN directive, is substituted.

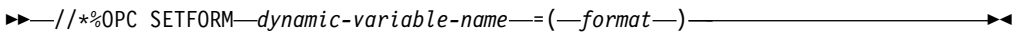
Example

```
//OPSTATUS JOB (ACCOUNT),'Set completed',CLASS=A
//STEP1    EXEC PGM=&MODULE.
//*%OPC SCAN
//STEPLIB  DD DSN=TWS.LOAD.&LIBRARY.,DISP=SHR
//EELMLIB  DD DSN=TWS.MESSAGE.LIBRARY,DISP=SHR
//EELMLOG  DD SYSOUT=A
//SYSIN    DD *
/*
```

SETFORM directive:
Purpose

This directive defines the format of dynamic-format supplied variables. After the agent processes the SETFORM directive, you can refer to the variable and perform arithmetic calculations using the variable. You can redefine the variable many times within the job, if you need to.

Syntax



Parameters

dynamic-variable-name=(*format expression***)**
The dynamic variable uses the format defined in the format expression.

Usage Notes

The dynamic variable name must be one of the predefined dynamic variables (see "Dynamic-format variables" on page 76).

The format expression can contain a combination of time-related keywords, date-related keywords, and delimiters.

The date-related keywords are:

CC Represents the century. This is used in combination with YY to define the format of a full year, such as 2012.

YY Represents the last two figures in the year.

MM Represents the month.

DDD Represents day-in-year. This is substituted before DD: the character string DDDDDD is understood as two DDD keywords, not three DD keywords.

DD Represents the day in the month.

The time-related keywords are:

HH Represents the hour.

MM Represents the minutes.

Any other characters in the format expression are regarded as delimiters. These delimiters can be alphabetic, numeric, or any symbol except the variable substitution characters `&`, `%`, `?`, `=`, and the parentheses ().

For the time-related dynamic variables, OCTIME, OPIATIME, and CTIME, only `HH` and `MM` are recognized. `YY`, for example, is not substituted. `MM` is substituted by the minutes part of the time.

For date-related dynamic variables, only `CCYY`, `YY`, `MM`, `DD`, and `DDD` are recognized. `CC` without `YY` is not recognized. `HH` is not substituted. `MM` is substituted by the month part of the date.

You can use more than one delimiter between keywords.

For example, `MM//DD-- YY` is a valid format expression.

Delimiters are optional; that is, you can define consecutive keywords with no delimiters, such as `DDMMYY`.

In the following examples, assume that the occurrence input arrival time is at 4:10 PM on December 31st 2012.

Example

```
//*%OPC SCAN
//*%OPC SETFORM OCDATE=(YY/MM/DDD)
```

The resulting &OCDATE variable would be: 12/12/365

The examples in Table 19 use the same occurrence input arrival date.

Table 19. Dynamic-format substitution results

Dynamic format variable	Format expression	Result
OCDATE	YY-MM-DDABC	12-12-31ABC
OCTIME	HH MM	16 10 (Note the MM substitutes as minutes for time variables and substitutes as month for date variables.)
OCDATE	DDDDD	36531. DDD is the 365th day of the year, and DD is the day of the month.
OCDATE	DDDD	365D. DDD is the 365th day of the year, but no match was found for the last D.

Table 19. Dynamic-format substitution results (continued)

Dynamic format variable	Format expression	Result
OCDATE	YYMMHHMMSS	1212HH12SS. This is a date variable, so HH is not substituted.

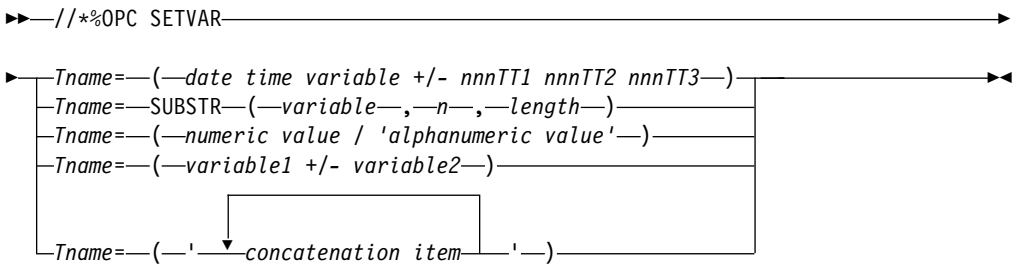
SETVAR directive:
Purpose

This directive creates a temporary variable using one of the following:

- An arithmetic expression together with supplied date or time variables.
- A substring of another variable.
- The result of an arithmetic addition or subtraction.
- Concatenated strings or variables set to an alphanumeric value.

After the agent for z/OS processes the SETVAR directive, you can use the temporary variable in the same way as you use other variables. You can redefine a temporary variable later in the job.

Syntax



Parameters

Tname=(*date time variable +/- nnnTT1 nnnTT2 nnnTT3***)**

Tname
The name of the temporary variable, beginning with the letter *T*.

date time variable
One of the following defined formats:

Date formats:
ODMY1, ODMY2, OYMD, OYMD1, OYMD2, OYMD3, CDDMMYY, CYMD, CYYMMDD, OCDATE, CDATE

Day-in-year formats:
ODDD, OYYDDD, OLYYDDD, CDDD, CYYDDD

Day-in-month formats:
ODD, CDD

Day-in-week formats:
OWWD, CWWD

Day-of-week formats:
ODAY, CDAY

Week formats:
OWW, CWW

Month formats:
OMM, OMMYY, OYM, OYYMM, CMM, CMMYY, CYYMM, CYYYYMM
Year formats:
OYY, OYYYY, CYY, CYYYY
Time formats:
OHHMM, CHHMM, CHHMMSSX, CTIME, OCTIME
Hour formats:
OHH, OLHH, CHH, CYYY

nnn A number in the range 0 to 999.

TT1 The first possible type. You can specify the following values:
For date-related variables
CD Calendar days.
WK Weeks. Weeks are converted to days before the calculation is performed.
MO Months. Performing calculation on the month portion affects only the month, and possibly the year. The calculation always generates valid results, handling actual months durations and leap years. For example, adding one month to 080131 gives 080229 (considering the leap year).
YR Years.
For time-related variables
HH Hours.
MM Minutes.
SS Seconds.

TT2 The second possible type, valid only for time-related variables. You can specify the following values:
MM Minutes.
SS Seconds.

TT3 The third possible type, valid only for time-related variables. You can specify only the value **SS** (seconds).

You can use the format *nnnTT1 nnnTT2 nnnTT3* only for time-related variables, to add or subtract hours, minutes, and seconds to or from a given time. Specify this triple format only if you want to specify hours, minutes, and seconds.

Using duplicated types, as for example in 6HH, 5MM, 7MM, is not allowed.

variable
The name of the source string variable. Use an existing variable name properly defined and accessible. You can specify any supplied or user-defined variable. The length of the variable that is replaced is limited to the remaining JCL row length that is not used by the statement.

n An integer in the range 1 to 60. It defines the starting position, in *variable*, of the substring variable. If it exceeds the length of *variable*, the resulting substring is padded with blank.

length An integer in the range 1 to 60. It defines the length of the substring variable. If it exceeds the length of *variable*, the resulting substring is padded with blank.

numeric value
An integer in the range 0 to 99999, defining the value of the temporary variable.

alphanumeric value
String of alphanumeric characters, defining the value of the temporary variable. Enclose the string in single quotes. It can be up to 48 characters.

If the length of the whole statement in the JCL reaches column 72, an error message is issued.

variable1
An integer in the range 0 to 99999, defining the first operand of an arithmetic addition or subtraction, whose result defines the value of the temporary variable. The arithmetic expression cannot include blank characters.

variable2
An integer in the range 0 to 99999, defining the first operand of an arithmetic addition or subtraction, whose result defines the value of the temporary variable. The arithmetic expression cannot include blank characters.

concatenation item
One of the following:

- A variable previously set to an alphanumeric value.
- A string of alphanumeric characters.

Enclose the item list in single quotes. The result cannot exceed 48 characters.

Usage Notes

These examples show how to use temporary variables created through SETVAR:

Example 1 (with an arithmetic expression)

If the occurrence input arrival date is 12/12/26, the expression is substituted as follows:

```
TVAR=(360+4)
TVAR=364
```

If the occurrence input arrival date is 12/12/30, the expression is substituted as follows:

```
TVAR=(364+4)
TVAR=003
```

Example 2 (with dynamic-format variable)

```
//*%OPC SCAN
//*%OPC SETFORM CDATE=(ACCURATE DATE CCYY MM DD)
//*%OPC SETVAR  TDATE=(CDATE + 1CD)
```

If the occurrence input arrival date is 12/12/26, the expressions are substituted as follows:

```
CDATE = 'ACCURATE DATE 2012 12 26'
TDATE = 'ACCURATE DATE 2012 12 27'
```

If the expression includes dynamic-format supplied variables containing the first or the last day in the month or in the year of the job stream instance IA, the

calculated date must fall within the range of four years earlier and seven years later than the current year. If the current year is 2012, the resulting date of the temporary variable must be later than 07/12/31 and earlier than 20/01/01.

Example 3 (SUBSTR usage)

- Using a variable defined in a JCL variable table:

  ```
  //*%OPC SETVAR  TVAR=SUBSTR(&VAR1,2,4)
  ```

  VAR1 is a variable defined in a JCL variable table.

  TVAR is a substring of VAR1 value, starting from position 2 for a length of 4 characters.

- Using a predefined variable:

  ```
  //*%OPC SETFORM OCDATE=(YYMMDD)
  //*%OPC SETVAR TVAR1=('&OCDATE')
  //*%OPC SETVAR TVAR2=SUBSTR(&TVAR1,3,2)
  //*%OPC SETVAR TVAR3=(OCDATE + 1MO)
  //*%OPC SETVAR TVAR4=SUBSTR(&TVAR3,3,2)
  ```

  If the occurrence input arrival date is 08/06/16, the expressions are substituted as follows:

  ```
  TVAR1 = 080616
  TVAR2 = 06
  TVAR3 = 080716
  TVAR4 = 07
  ```

  In fact the SUBSTR parameter identifies a substring of TVAR1 and TVAR3 values, starting from position 3 for a length of 2 characters. According to the format set by the SETFORM directive, it identifies the MM part of the date value.

Example 4 (arithmetic with temporary variables)

```
//*%OPC SETVAR TX=(1)
//*%OPC SETVAR TY=(2)
//*%OPC SETVAR TZ=(&TX+&TY)
```

TZ is a temporary variable set to the result of the arithmetic addition.

Example 5 (concatenating temporary variables)

```
//*%OPC SETVAR T001=('STRING1')
//*%OPC SETVAR T002=('STRING2')
//*%OPC SETVAR T003=('&T001 &T002 CONCATENATED STRINGS')
```

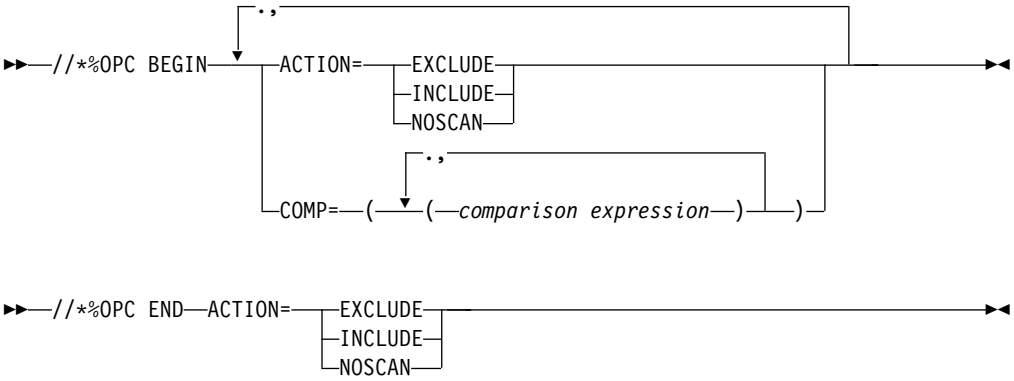
T003 is a temporary variable set to the following value: STRING1 STRING2 CONCATENATED STRINGS

BEGIN and END directives:
Purpose

These directives, used in pairs, denote the following, depending on the value of the ACTION keyword:

- The start and end of the variable substitution action performed by the agent
- The start and end of the lines to be *included in* the tailored job
- The start and end of the lines to be *excluded from* the tailored job

Syntax



Parameters

ACTION=(EXCLUDE|INCLUDE|NOSCAN)
Specifies which BEGIN/END action is required.

EXCLUDE
This specifies that the lines following this BEGIN directive up to the next END ACTION=EXCLUDE directive should be excluded from the job that is submitted for this operation.

INCLUDE
This specifies that the lines following this BEGIN directive up to the next END ACTION=INCLUDE directive should be included as part of the job that is submitted for this operation.

NOSCAN
This specifies that any variables following this BEGIN directive up to the next END ACTION=NOSCAN directive should not be substituted.

COMP=((*comparison expression***), (***comparison expression***),...)**
Specifies comparison expressions that are used to decide whether the BEGIN directive should be acted on. If the comparison expression is true, the BEGIN directive is honored. For details on defining comparison expressions, see "The COMP keyword on BEGIN and FETCH directives" on page 86.

Usage Notes

In a job, every BEGIN directive must have a matching END directive specifying the same ACTION. For example, the directive:

```
//*%OPC BEGIN ACTION=EXCLUDE
```

requires the following matching END directive:

```
//*%OPC END ACTION=EXCLUDE
```

If the agent detects an unpaired BEGIN or END, the processing ends in error. Even a BEGIN statement that is not honored because its comparison expression is not true requires a matching END statement.

Only the following directives can lie within the domain of a BEGIN ACTION=NOSCAN directive and an END ACTION=NOSCAN directive:

- SETFORM
- SETVAR

When these directives are in the range of a NOSCAN directive, they are always acted upon even if there is a comparison condition that is false.

BEGIN and END directives that specify ACTION=INCLUDE or ACTION=EXCLUDE cannot be nested and cannot overlap. They can, however, completely contain a nested NOSCAN domain.

Consider the following examples:

Example 1

```
//*%OPC SCAN
//*%OPC BEGIN ACTION=INCLUDE                    1
//DDNAME1 DD DSN=&HIONE..&DATASET1,DISP=SHR     2
//DDNAME2 DD DSN=&HIONE..&DATASET2,DISP=SHR     3
//*%OPC END ACTION=INCLUDE                      4
```

Example 1 is valid. Lines **2** and **3** will be included in the job for the operation.

Example 2

```
//*%OPC SCAN                                    1
//*%OPC BEGIN ACTION=EXCLUDE                    2
//EXEC PGM=MYPROG                               3
//*%OPC BEGIN ACTION=INCLUDE                    4
//DDNAME1 DD DSN=&HIONE..&DATASET1,DISP=SHR     5
//SYSOUT  DD SYSOUT=A                           6
//*%OPC END ACTION=EXCLUDE                      7
//DDNAME2 DD DSN=&HIONE..&DATASET2,DISP=SHR     8
//*%OPC END ACTION=INCLUDE                      9
```

Example 2 is invalid. An EXCLUDE action (lines **2** and **7**) overlaps an INCLUDE action (lines **4** and **9**). An error message will be issued.

Example 3

```
//*%OPC SCAN                                    1
//*%OPC BEGIN ACTION=INCLUDE                    2
//EXEC PGM=MYPROG                               3
//*%OPC BEGIN ACTION=NOSCAN                     4
//DDNAME1 DD DSN=&HIONE..&DATASET1,DISP=SHR     5
//SYSOUT  DD SYSOUT=A                           6
//*%OPC END ACTION=NOSCAN                       7
//DDNAME2 DD DSN=&HIONE..&DATASET2,DISP=SHR     8
//*%OPC END ACTION=INCLUDE                      9
```

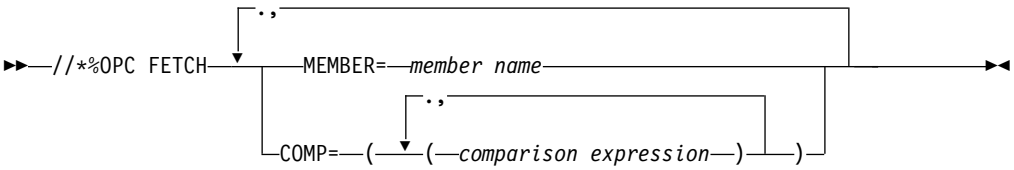
Example 3 is valid. The NOSCAN domain defined by lines **4** and **7** is completely contained within the BEGIN and END ACTION=INCLUDE (lines **2** and **9**).

Note also that the variable `HIONE` on line **8** will be substituted, but the variable `HIONE` on line **5** will not be substituted because it is within a NOSCAN domain.

FETCH directive:
Purpose

This directive lets you include lines, fetched from a partitioned data set member or supplied by an exit, in your job.

Syntax



Parameters

MEMBER=(*member name***)**
Specifies the member name of a partitioned data set allocated to ddname EELJBLIB. The lines in this member are included immediately after the FETCH directive.

COMP=((*comparison expression***), (***comparison expression***),...)**
Specifies comparison expressions used to decide whether the FETCH directive should be acted on. If the comparison expression is true, the FETCH directive is honored. For details on defining comparison expressions, see "The COMP keyword on BEGIN and FETCH directives."

Usage Notes

The FETCH directive is used to include lines from other partitioned data sets or as supplied by an exit. Lines included by a FETCH directive cannot contain another FETCH directive. BEGIN and END directives with action INCLUDE or EXCLUDE cannot be included in lines inserted by a FETCH directive.

Predefined variables can be used to represent the values of any keywords, but not the keywords themselves. A FETCH directive cannot lie between a BEGIN/END directive pair that specifies ACTION=INCLUDE or ACTION=EXCLUDE.

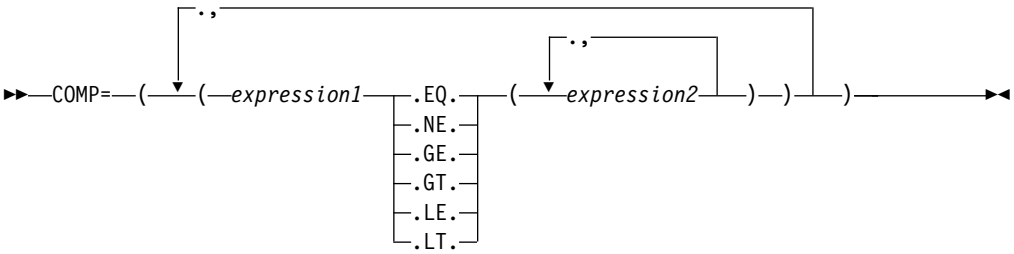
Example

```
//*%OPC SCAN
//*%OPC FETCH,
//*%OPC MEMBER=JCL1,
//*%OPC COMP=(&DAY..EQ.1)
```

The COMP keyword on BEGIN and FETCH directives:
Purpose

A comparison expression lets you specify conditions when BEGIN and FETCH directives will be honored.

Syntax



Parameters

expression1
This specifies a string made up of &-variables and alphanumeric literals. Any global search characters it contains are treated as literals. The value of *expression1*, arrived at by resolving any variables specified, will be tested against the values given by *expression2*.

.Operators.
These values are operators that specify which comparison should be made between *expression1* and any *expression2* values.

.EQ. *Expression1* must equal one of the *expression2* values for the expression to be true.

.NE. All *expression2* values must not equal the *expression1* value for the expression to be true.

.GT. *Expression1* must be greater than the *expression2* value for the expression to be true.

.GE. *Expression1* must be greater than or equal to the *expression2* value for the expression to be true.

.LT. *Expression1* must be less than the *expression2* value for the expression to be true.

.LE. *Expression1* must be less than or equal to the *expression2* value for the expression to be true.

expression2
This parameter can be made up of &-variables, literals, or, if .EQ. or .NE. operators are specified, one of the two global search characters, % and *.

The length of the resolved value cannot exceed 44 characters. The % global search character represents any single alphanumeric character. The * global search character represents any alphanumeric string, including a null string.

If GT, GE, LT, or LE is specified:
- Multiple values of *expression2* are not supported.
- Global search characters are not supported.
- If the strings on both sides of the operators are of different lengths, the comparison is made using the shorter string.

Note: The % symbol does not signify a % predefined variable within a COMP keyword. The %- and ?-variables are not valid within a COMP statement.

Usage Notes

The COMP expression cannot exceed 256 characters unresolved, and cannot be more than 1024 characters after substitution; *expression2* can be any predefined or user-defined &-variable. Neither *expression1* nor *expression2* can have embedded blanks.

Consider the following examples:

Example 1

```
//*%OPC FETCH,
//*%OPC MEMBER=MYJCL,
//*%OPC COMP=(&APPL..EQ.(APPL1,APPL2,APPL3))
```

If &APPL. is equal to APPL1 or APPL2 or APPL3, the expression is true, and the FETCH directive will be honored.

Note the two periods following &APPL.. The first signifies the end of the variable APPL; the second signifies the start of the comparison operator EQ.

Example 2

```
//*%OPC FETCH,
//*%OPC MEMBER=MYJCL,
//*%OPC COMP=(&DAY..NE.(1,3,5))
```

In example 2, if &DAY is not equal to 1 or 3 or 5, the expression is true, and the FETCH directive will be honored. If DAY had been equal to any one of the comparison values, the expression would have been false.

For the COMP keyword to be *true*, all the comparison expressions that it consists of must be *true*. This is shown in the following example:

Example 3

```
//*%OPC BEGIN ACTION=INCLUDE,
//*%OPC       COMP=((&APPL..EQ.(APPL1,APPL2,APPL3)),
//*%OPC             (&DAY..NE.(1,3,5)))
          .
          .
          .
//*%OPC END ACTION=INCLUDE
```

For the COMP statement in example 3 to be true, the expressions (&APPL..EQ.(APPL1,APPL2,APPL3)) and (&DAY..NE.(1,3,5)) must both be true.

The *expression2* values that you specify can be made up of &-variables, alphanumeric literals, and the * and % global search characters. National characters, left and right parentheses; (and), and blanks are not allowed; if they are specified, the results are unpredictable. The * global search character represents a character string of any length; the % global search character represents exactly 1 character. If variables and global search characters are combined, the variables are resolved before any comparisons are made using the global search characters.

Example 4

```
//*%OPC BEGIN ACTION=INCLUDE,
//*%OPC       COMP=(&MYVAR..EQ.(TSO199,TSO2%%.,&VALUE1.*))
          .
          .
          .
//*%OPC END ACTION=INCLUDE
```

In example 4, the variable &MYVAR must have one of the following values for the comparison expression to be true:

- TSO199
- TSO2 followed by any 2 alphanumeric characters except blanks
- The value of variable &VALUE1 followed by an alphanumeric string of any length, including length 0.

Example 5

```
//*%OPC BEGIN ACTION=EXCLUDE,
//*%OPC       COMP=(&CYYMMDD..GE.120101)
          .
          .
          .
//*%OPC END ACTION=EXCLUDE
```

Note that COMP statements can give unexpected results with some of the date formats of the predefined variables. When date variables are substituted, they are compared as numerals, not as dates.

In example 5, `&CYYMMDD` is the current date and `120101` represents 1 January 2012. If the value of *expression1* is greater than 120101, the comparison expression is true.

Managing job instances

You manage jobs submitted to an agent for z/OS with the same commands available for all IBM Workload Scheduler jobs.

Use the conman command line or the Dynamic Workload Console to manage the job instances. Most of the job management commands and actions of IBM Workload Scheduler apply also to agent for z/OS job instances, but not all. The following sections list what you can and cannot do.

Commands and Actions you can run

All except for killing job instances.

Commands and Actions you cannot run

You cannot run the `kill` command on job instances.

This restrains the use of the `maxdur` job stream keyword on agent for z/OS jobs. That is, if in a job stream you specify the `maxdur` keyword with the `onmaxdur kill` argument, the job is not killed if it exceeds the time limit, in spite of the fact that the `MaxDurationExceeded` and `KillSubmitted` flags are reported by the `conman showjobs` command. In the area of event-driven workload automation, the `MaxDurationExceeded` event is generated; it can be used to trigger a number of actions, but not the `KillJob` action.

Tracking jobs

The agent uses the JES/SMF exits to track the status changes for all the jobs submitted through it.

Job states

The status events related to the jobs submitted via the agent for z/OS are generally issued by JES and SMF and intercepted by the agent for z/OS. The agent then forwards them to dynamic workload broker in an XML message. The job states are reported on the Dynamic Workload Console and on the conman command line.

The following table summarizes the states a job undergoes, from submission to completion, and maps the corresponding status names as they are reported by JES/SMF, the agent for z/OS, and on the user interfaces.

Table 20. Job events and statuses as mapped by the involved components

Event description	Agent internal event	Status name as passed by the agent for z/OS to dynamic workload broker in XML	Status name as displayed by the Dynamic Workload Console and conman	Other details
The agent has found an error in the JCL included in the job and stops processing it.	IJ0	-	FAIL	This event is generated by the agent for z/OS when dynamic workload broker submits the job.
Job JCL. A job has been submitted. The job has been submitted by dynamic workload broker and the JCL has been placed in the JES internal reader by the agent for z/OS. The job has not yet been submitted by JES, but the IDs assigned to the job by dynamic workload broker and by JES have been mapped together.	KJ1	INFO	BOUND	This event is generated by the agent for z/OS after copying the JCL in the JES internal reader.
Reader event: a job has entered the JES system.	A1 or B1	SUBMITTED	WAIT	This event is generated by JES on the system hosting the agent for z/OS and sent to dynamic workload broker by the agent for z/OS.
Job-start event: a job has started to execute.	A2 or B2	EXECUTING	EXEC	This event is generated by SMF on the system where the job is submitted and sent to dynamic workload broker by the agent for z/OS.
Step-end event: a job step has finished executing.	A3S or B3S	EXECUTING	EXEC	The event is generated on the system where the job is submitted but is discarded unless there is a step ABEND.
Job-end event: a job has finished executing.	A3J or B3J	SUCCEEDED EXECUTION or FAILED EXECUTION	SUCC or ABEND	This event is generated by SMF on the system where the job is submitted and sent to dynamic workload broker by the agent for z/OS.

Table 20. Job events and statuses as mapped by the involved components (continued)

Event description	Agent internal event	Status name as passed by the agent for z/OS to dynamic workload broker in XML	Status name as displayed by the Dynamic Workload Console and conman	Other details
Job-termination event: a job has been added to the JES output queues.	A3P or B3P	SUCCEEDED EXECUTION or FAILED EXECUTION	SUCC or ABEND	This event is generated by JES on the system where the job is submitted and sent to dynamic workload broker by the agent for z/OS.
Job Print end	A4 or B4	-	-	Optional event that depends on the JCL content. The event is recorded in the event data set (EELEVDS) but is not forwarded to dynamic workload broker.
Job is purged	A5 or B5	-	-	Optional event that depends on the JCL content. The event is recorded in the event data set (EELEVDS) but is not forwarded to dynamic workload broker.

JCL errors

The agent parses the JCL sent by dynamic workload broker before it submits it to JES. If it finds a syntax error in the JCL, it stops the submission process and records the error message in the job log.

Job error codes

Return codes other than zero after a job is submitted are to be considered errors. The return code is sent back to dynamic workload broker by the agent for z/OS via HTTP in an XML POST together with the job ID and name. The z/OS error code is mapped to a numeric code which is displayed in conman or in the Dynamic Workload Console. The following table lists the error codes that can be returned after a job is submitted and their mapping on z/OS and on IBM Workload Scheduler.

Table 21. Error codes returned after a job is submitted.

Return code mapped on IBM Workload Scheduler	Error description	Displayed as job extended property on conman or the Dynamic Workload Console
1*nnnn* (*nnnn* is *xxx* converted to decimal digits)	System abend error codes in hexadecimal notation.	S*xxx*
2*nnnn* (*nnnn* is *xxx* converted to decimal digits)	User abend error codes in hexadecimal notation.	U*xxx*

Table 21. Error codes returned after a job is submitted. (continued)

Return code mapped on IBM Workload Scheduler	Error description	Displayed as job extended property on conman or the Dynamic Workload Console
30007	A JCL error occurred immediately; that is, the error was detected before the job began. This code is also possible when both the job-start event (type A2) and the job-end event (type A3J) are missing.	JCLI
30020	A failure occurred when the agent attempted to submit a job. The operation should be marked as ended-in-error.	OSUB
30021	A failure occurred when the agent attempted to retrieve the JCL for a job.	OSUF
30026	The job was canceled by the operator or by a TSO user before execution. This code is also possible if the job-termination event (type A3P) is missing.	CAN
30027	The completion code is unknown. The job has ended, but no completion code is available. This code is also possible if the job-end event (type A3J) is missing.	CCUN
30029	A JCL error was recognized after the job began to execute, or a JCL error was recognized after syntax checking in the internal reader.	JCL
90000	User-defined error codes in hexadecimal notation.	*xxxx*

The event data sets

The agent uses the EELHTDS and EELEVDS event data sets which contain the records that describe the events created by its job tracking functions. An event-writer task writes and reads records to and from these data sets.

EELHTDS records the events originated by dynamic workload broker and related to the submission of workload.

EELEVDS records the events originated by JES and SMF that are related to job execution and that are to be sent back to dynamic workload broker .

Another data set named EELHTREF is used as a service data set to briefly store the jobs sent within the submission requests coming from IBM Workload

Scheduler. If the jobs call for the retrieval of a JCL stored in the z/OS system, or require variable substitution, these actions are performed here.

Because the event data sets provide a record of each event, events will not be lost if the agent or an event processing component must be restarted. The submit checkpointing process ensures that submit requests are synchronized with dynamic workload broker, thereby preventing lost requests caused by communication failures.

Important: The data sets are formatted the first time they are used. If for some reason you format one of the data sets again, this results in the loss of all the events queued on all data sets.

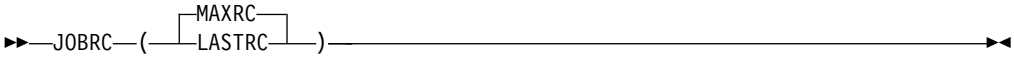
Controlling how the event writer records job completion codes for specific jobs

On z/OS 1.13 and later, you can specify the JOBRC parameter in the JOB card statements of specific jobs to predefine how their completion codes are recorded in the EELEVDS event data set.

For all the jobs submitted to JES by the agent for z/OS, the RETCODE keyword of the EWTROPTS initialization statement defines which completion code the event writer records in the EELEVDS event data set for the job-end (A3J) event record, choosing from the codes returned by the job steps. The default choice for RETCODE is to set the job completion code to the return code of the last step; alternately, you can set it to choose the highest return code of any step.

For particular jobs of your choice, you can override the setting of RETCODE by specifying the JOBRC parameter in the JOB statement of the JCL. Typically, you use JOBRC to reverse for a specific job the setting defined in EWTROPTS RETCODE, which applies generally to all the workload submitted to JES by the agent for z/OS.

JOBRC



Parameters

MAXRC
The job completion code is set to the highest return code of any step in the job, or if the completion of the job fails because of an ABEND, the job completion code is set to the last ABEND code. This is the default parameter.

LASTRC
The job completion code is set to the return code or ABEND code of the last step that is executed in the job.

Example

The EWTROPTS initialization statement of your agent for z/OS is set with `RETCODE=HIGHEST`, whereby for all jobs submitted to JES by the agent, the job completion code returned by the event writer is the highest return code of all the performed steps.

For the ACCT1254JCL multistep job, however, you want the event writer to pick the return code of the last step completed or ABENDED. You therefore write `JOBRC=LASTRC` in the JOB statement of ACCT1254JCL.

Viewing job logs

Viewing the logs of jobs submitted through the agent for z/OS.

You can view the logs of jobs submitted through the agent for z/OS on the Dynamic Workload Console or on the conman command line.

The procedure to do this is standard to all IBM Workload Scheduler jobs:

- On the Dynamic Workload Console click **System Status and Health →Workload Monitoring →Monitor Jobs** and proceed from there to the point of clicking the **Job Log...** button on a selected job instance.
- From the IBM Workload Scheduler command line run `conman showjobs`.

As you follow either of these procedures, the logs are retrieved from the z/OS system and displayed. The logs are available only for jobs that have completed.

The logs are available for your viewing until they are purged by JES. After that time they are no longer available.

Using system commands to control the agent

You can use z/OS system commands to start, stop, cancel, or modify the agent for z/OS.

Use the following operator commands to control the agent:

S START

P STOP

C CANCEL

F MODIFY

You can enter these commands from a Multiple Console Support (MCS) console or from a program such as the Spool Display and Search Facility (SDSF). In both cases, the terminal or console operator must have the required authority to enter operator commands.

Starting the agent

To start the agent for z/OS, enter the following z/OS operator command:

`/S procname`

where *procname* is the agent for z/OS JCL or started task procedure name.

If the agent is to run as a batch job, do not start it with an operator command. Instead, submit a batch job with the same name as the agent for z/OS subsystem. JES starts this job in the same manner as any ordinary job.

Because the agent for z/OS uses JES exits, among other things, to track the progress of z/OS jobs, it does not start before JES is active.

Stopping the agent

To stop the agent for z/OS, enter the following z/OS operator command:

```
/P procname
```

where *procname* is the agent for z/OS JCL procedure name.

Cancelling the agent

If the agent for z/OS is still active 5 minutes after you enter the STOP operator command, you must cancel it.

To cancel the agent for z/OS, enter the following z/OS operator command:

```
/C procname
```

where *procname* is the agent for z/OS JCL procedure name.

If the STOP command is ineffective and you have no earlier documentation of the problem, cancel the agent adding the DUMP option so that the problem can be identified. Enter:

```
/C procname,DUMP
```

This causes the agent for z/OS to end with a dump on the SYSMDUMP file (if the ddname is in the started-task JCL).

Modifying the agent

Use the MODIFY command to start or stop one of the following agent for z/OS subtasks:

- Data router
- Event writer
- Submit

and to list the status of the subtasks.

The syntax of the MODIFY command is:

```
/F procname,modifyoption
```

where:

- *procname* is the agent for z/OS JCL procedure name.
- *modifyoption* can be:

  STATUS, SUBTASK
  Lists all subtasks with their statuses. The status can be ACTIVE or INACTIVE.

  S=*subtask*
  Starts the specified subtask.

  P=*subtask*
  Stops the specified subtask.

  subtask is one of the following agent for z/OS subtasks:

  DRT Data router

  EWTR
  Event writer

SUB Submit

Note that stopping the Event Writer puts the agent for z/OS in OFFLINE status (shown as UNAVAILABLE by dynamic workload broker). Jobs submitted while the agent is OFFLINE are shown in the READY status. Jobs that were submitted but not yet run when the agent went OFFLINE are placed in the WAIT status and moved to FAIL when the timeout configured for dynamic workload broker expires.

You can only start a task that has stopped earlier in the current session. If you attempt to start a started subtask or stop a stopped subtask, error message EELZ049W is issued, and no action is taken.

Switching domain managers

This section describes the mechanism that keeps the agent for z/OS connected to your IBM Workload Scheduler network when you switch to a backup manager.

Understanding how the agent responds to a domain manager switch

Whenever you change the domain manager (or master) associated with the dynamic workload broker instance to which the agent for z/OS is connected, the link between the agent and its dynamic workload broker counterpart is interrupted. The HTTP client of the agent for z/OS is designed to search the next available dynamic workload broker instance (that is, the one running in the newly activated domain manager) and to establish a connection as soon as it acknowledges the interruption.

Following its initialization, the agent for z/OS pings the dynamic workload broker at regular intervals. Each ping is an HTTP post request where the agent sends its identification and other attributes as a dynamic workload broker resource. After the dynamic workload broker accepts and processes the HTTP request, it responds by sending the list of currently defined backup dynamic workload broker instances. This list is based on the list of IBM Workload Scheduler agents defined as backup domain managers.

After the first successful ping, the agent for z/OS has the list of all the backup dynamic workload broker instances available in the IBM Workload Scheduler network. This list is refreshed at each following ping request. If the agent cannot successfully ping the target dynamic workload broker when it is started, then no list of backups is available at all and no switch to a backup dynamic workload broker can occur.

When a network error occurs while the agent for z/OS is issuing a request to the current dynamic workload broker, an offline event is generated that triggers the mechanism by which the HTTP client located in the agent for z/OS pings the next dynamic workload broker present in the list.

- If this dynamic workload broker instance is available, and a connection is established, it becomes the new target dynamic workload broker that the agent for z/OS interacts with. The new dynamic workload broker also provides an updated list of backup dynamic workload broker instances.
- If the instance is unavailable, the HTTP client pings the next instance in the list, and so on. After trying the last instance without success, it starts over from the first. This process goes on until one of the dynamic workload brokers is pinged successfully.

Stopping and restarting the agent after the primary dynamic workload broker has changed

From the first time the connection with the dynamic workload broker is established, the list of backup instances is stored in the agent memory for the duration of the agent runtime. It is lost when you stop the agent. When you stop and restart the agent, the agent pings the original dynamic workload broker instance specified in its configuration parameters. If this instance is unavailable because you have operated a switch or because it has gone down in the meantime, the agent cannot connect with a backup instance since it has no list yet. So, if you do stop and restart the agent after the primary dynamic workload broker has changed, remember to update the agent configuration with the `TDWBHOSTNAME` and `TDWBPORT` values of the new primary dynamic workload broker. After connecting with the new dynamic workload broker, the agent will be sent the list again.

Chapter 4. Troubleshooting and reference

This chapter documents potential problems and reference information.

Understanding resynchronization messages

This section describes the informational messages issued by the agent for z/OS as it synchronizes job submission again with dynamic workload broker following a restart.

Resynchronization overview

The first task the agent for z/OS is called to attempt when it restarts after a planned or involuntary stop, is to synchronize correctly with dynamic workload broker so that there is no loss of information regarding the workload that was being handled by the agent at the time of arrest.

As described in "Tracking jobs" on page 89, the information about the workload assigned to the agent is in term of events that record the work yet to be submitted and the outcome of the already processed work. During normal processing, the agent uses three event data sets to handle these events. The same data sets are also used for recovery purposes after a restart. They are (as known by their DD name within the agent for z/OS started task):

EELEVDS
Stores the events that track the complete life of a submitted job (submission, execution, outcome).

After a stop/restart, the agent searches this data set for the latest emitted events to send them to dynamic workload broker again.

EELHTREF
Stores the job submission requests received from dynamic workload broker. Contains information in terms of type of submission (by reference or by definition), the JCL or its location, the variable table when applicable. As soon as a request stored in EELHTREF has been thoroughly processed (the job has been queued to the `submit` task), it is flagged as completed.

After a stop/restart, the agent searches this data set for incomplete requests and processes them again.

EELHTDS
Stores the JCLs queued to the `submit` task.

After a stop/restart, the agent searches this data set for queued but not yet submitted JCLs in order to queue them again and complete their processing.

The resynchronization messages

After a stop/restart, the agent issues a number of information messages that describe the actions it is taking to resynchronize. While no user response is requested, they are documented here to help you follow the resynchronization process.

© Copyright IBM Corp. 2011, 2016

EELHT28I
EELHTDS REPROCESSING CAUSED RESUBMISSION OF FOLLOWING EVENT:
SSEQ: *SEQ* RECORD: *REC* CYCLE: *CYC* JOBALIAS:
JOBALIA1
JOBALIA2
Explanation: While reprocessing the EELHTDS data set
(recovery data set for pending submissions)after a restart, the agent
found in pending status the job identified by the alias indicated by
JOBALIA1 and *JOBALIA2*.
The EELHTDS record where the job was found is identified by
the *REC* record number and the *CYC* write cycle.
System action: The agent sends the job to the submit task again.

EELHT36I
INFORMATION ABOUT RESTART PROCESSING OF EELHTDS DATA SET FOLLOWS:
HTDS NEXT TO WRITE : REC *INREC* CYC *INCYC*
HTDS RESEND START POSITION: REC *STREC* CYC *STCYC*
HTDS RESEND STOP POSITION : REC *CUREC* CYC *CUCYC*
HTDS LAST SUBMIT DONE : REC *LSREC* CYC *LSCYC* SSEQ: *LSSS*
Explanation: At start up the agent reprocesses the EELHTDS
data set (recovery data set for pending submissions) to find the
jobs that need to be submitted again
because their submission was left pending.
Before the jobs are processed again, the following information is provided:
- The EELHTDS position where the next record will be written, identified by record number *INREC* and write cycle *INCYC*.

- The EELHTDS range that will be analyzed, identified by:
  - start position: record number *STREC*, write cycle *STCYC*
  - end position: record number *CUREC*, write cycle *CUCYC*.

- The last submission made by the agent before the restart, identified by sequence number *LSSS* and the related record in EELHTDS having record number *LSREC* and write cycle *LSCYC*.

System action: Processing continues.

EELHT37I
INFORMATION ABOUT RESTART PROCESSING OF EELHTREF DATA SET FOLLOWS:
HTREF NEXT TO WRITE : REC *INREC* CYC *INCYC*
HTREF START POSITION : REC *STREC* CYC *STCYC*
HTREF LAST POSITION : REC *LAREC* CYC *LACYC*
HTDS LAST SUBMIT STORED: REC *HTREC* CYC *HTCYC* SSEQ: *HTSS*
Explanation: At start up the agent reprocesses the EELHTREF
data set (recovery data set for pending requests) to find the
jobs that need to be processed again as they were left pending.
Before the jobs are processed again, the following information is provided:
- The EELHTREF position where the next record will be written, identified by record number *INREC* and write cycle *INCYC*.

- The EELHTREF range that will be analyzed, identified by:
  - start position: record number *STREC*, write cycle *STCYC*
  - end position: record number *LAREC*, write cycle *LACYC*.

- The last submission recorded in EELHTDS by the agent before the restart, identified by sequence number *HTSS* and the related record in EELHTDS having record number *HTREC* and write cycle *HTCYC*.

System action: Processing continues.

EELHT38I
EELHTREF REPROCESSING RECOVERED FOLLOWING EVENT:
JOBALIA1
JOBALIA2
Explanation: At start up the agent reprocesses the EELHTREF
data set (recovery data set for pending requests) to find the
jobs that need to be submitted again as they were left pending.
This message is issued for each reprocessed job identified by
the alias indicated by *JOBALIA1* and *JOBALIA2*.
System action: Processing continues.

EELHT42I
THE HTTP SERVER FINISHED THE SYNCHRONIZATION WITH THE SUBMIT TASK.
NUM SUBMISSION EVENTS HAVE BEEN REPROCESSED.
Explanation: At start up the agent reprocesses the EELHTDS data set (recovery data set for pending submissions) to find the jobs that need to be submitted again because their submission was left pending. During the process, message EELHT28I is issued for each resubmitted job. At the end of the process, message EELHT42I is issued to communicate the total number of reprocessed events *NUM*.
Note that it can happen that the number of issued EELHT28I messages is lower than *NUM*. This occurs when incomplete events are found (as reported by message EELHT35W) and therefore cannot be submitted.
System action: None.

EELHT44I
THE HTTP SERVER STARTED THE EELHTREF DATA SET PROCESSING.
Explanation: At start up the agent checks the EELHTREF data set (recovery data set for pending requests) to find any pending job submissions that need to be processed.
System action: The agent processes the EELHTREF data set.

EELHT45I
THE HTTP SERVER ENDED THE EELHTREF DATA SET PROCESSING.
NUM SUBMISSION EVENTS HAVE BEEN REPROCESSED.
Explanation: At start up the agent checks the EELHTREF data set (recovery data set for pending requests) to find any pending job submissions that need to be processed.
System action: None.

Component versions must be aligned for the full current functionality

To be able to fully exploit the latest features of the agent for z/OS, make sure that the versions of all components are aligned.

To be able to use the complete functionality currently available in the agent for z/OS, the versions of the agent and of the dynamic workload broker to which it is attached (be it on a dynamic domain manager or a master domain manager) must be concurrent.

Specifically, if you connect the agent for z/OS version 8.6.0.2 with a dynamic domain manager or a master domain manager version 8.6 or 8.6.0.1, you cannot define jobs by `reference` or use variable substitution in the JCLs.

If you do, upon submission the jobs will terminate in error and in some cases will hang indefinitely without issuing an error code.

Saturation of DB2 transaction log halts processing of jobs

Processing of agent for z/OS jobs terminates when the DB2 transaction log fills up.

If the DB2 transaction log becomes full while a job is running, processing of the job is interrupted and the connection between the agent and the dynamic workload broker is stopped. Any agent for z/OS jobs that are yet to run hang idle and, in the particular case that the submitted plan contains only agent for z/OS jobs, it stops altogether.

Look for the following messages to be sure that a saturated DB2 transaction log is the problem:

In the EELMLOG on z/OS:

```
EELHT15E THE HTTP CLIENT FAILED TO PROCESS A REQUEST FOR BROKER
EELHT43I HTTP RESPONSE MESSAGE WITH CODE RDBMS_TRANSPORT_PROBLEM
```

On IBM Workload Scheduler

1. From BATCHMAN (occurs on the running job when the transaction log becomes full):

   ```
   job_name has failed with the error: An error occurred reading the
   job from the job table

   job_name has failed with the error: AWKJDB801E An internal error has
   been found while accessing the database. The internal error message is:
   "Not enough storage is available in the application heap to process"

   AWSBHT032I Workstation broker_workstation_name is now inactive,
   no jobs will be scheduled.
   ```

2. From MAILMAN (reporting that the agent and the dynamic workload broker are unlinking):

   ```
   AWSBCV082I Workstation broker_workstation_name, message: AWSDEB014I
   Connection timed out
   AWSBCV027I Unlinking from broker_workstation_name
   ```

3. From BATCHMAN (after dynamic workload broker has unlinked and job submission to the agent has stopped):

   ```
   AWSBDY103I Received command MY:UNLINK for run number 42 for
   workstation broker_workstation_name from workstation
    workstation_name

   Workstation broker_workstation_name State is being changed: UNSETTING:
   LINKED=TCP AWSBHT032I Workstation broker_workstation_name is now
   inactive, no jobs will be scheduled.
   ```

4. In the SystemOut.log file:

   ```
   AWSJDB801E An internal error has been found while accessing the
   database.The internal error message is: "Not enough storage is
   available in the application heap to process the statement..
   SQLCODE=-954, SQLSTATE=57011, DRIVER=3.61.75".
   ```

In the db2diag.log file of the DB2 server:

```
MESSAGE : ZRC=0x85100009=-2062548983=SQLP_NOSPACE
          "Log File has reached its saturation point"
          DIA8309C Log file was full.

RETCODE : ZRC=0x8B0F0001=-1961951231=SQLO_NOMEM_APPH
          "No memory available in 'Application Heap'"
          DIA8301C No memory available in the application heap.
```

The saturation of the DB2 server transaction log is due to the fact that the log size is insufficient because of its default settings. To prevent it from becoming saturated in the future, change the DB2 configuration settings to at least the following values:

```
Log file size (4KB)               (LOGFILSIZ) = 10000
Number of primary log files       (LOGPRIMARY) = 80
Number of secondary log files     (LOGSECOND) = 40
```

See the DB2 documentation for further information.

Data areas

Provides graphic representation of the data areas used by the agent for z/OS.

The next sections provide graphic representation of the data areas used by the agent for z/OS.

Data areas appear alphabetically, by name of the mapping macro. The data areas in this publication are not intended to be used as a customer user interface, but knowing their layout can be useful for debugging The areas are product sensitive and can be changed at any time during the current release without documentation updates to this publication.

The data area map

Each data area is described field by field. These field descriptions are taken directly from the system code.

For each field in the data area, the table provides the following information:

Offsets
The address of the field, shown in both decimal and hexadecimal (hexadecimal address in parentheses), relative to the beginning of the data area.

Type The kind of program data defined for this field, such as CHARACTER, SIGNED, UNSIGNED.

Len Size of the field in bytes (decimal).

Name The name of the field, bit, or mask.

Bit or mask names are preceded by a description of the bit position and values, as follows:

```
1...  ....  Refers to bit 0.
....  ..11  Refers to bits 6 and 7.
...1  ....  Refers to bit 3.
11..  1111  Refers to bits 0, 1, 4, 5, 6, and 7.
```

Description
A description of the purpose or meaning of the field, bit, or mask.

The cross-reference table

For each data area with more than 40 fields, a cross-reference table shows the following:

- Hex Offset: The hexadecimal offset of the field into the data area (for bits, the hexadecimal offset of the field containing the bit).
- Hex Value: Hex values are shown only for bits. The Hex value shown implies the position of the bit in the field containing the bit.

DQE - Data queue element

Name : DCLDQE
Function:

This segment maps queue elements for several IBM Workload Scheduler queues. The mapping of the DQEDATA field varies depending on the value of DQETYPE. If data buffers are used (indicated by dqebptr ^= 0), they are always allocated in subpool 2 by queue adders, and are freed when no longer needed by queue servers.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	152	dqe	data router q element
0	(0)	CHARACTER	4	dqedesc	block descriptor, DQE
4	(4)	CHARACTER	2	dqever	version number, 01
6	(6)	BITSTRING	2	*	reserved flags
8	(8)	CHARACTER	3	dqetype	data type
11	(B)	BITSTRING	1	dqeflags	flags
		1...		dqeflres	reset seqds
		.1..		*	free
		..1.		dqenckpt	do not chkpt this submit
		...1		*	free
		.... 1...		*	free
		.... .1..		*	free
		.... ..1.		*	free
		1		*	free on=used
12	(C)	ADDRESS	4	*	free
16	(10)	ADDRESS	4	dqebptr	external data buffer ptr
20	(14)	SIGNED	4	dqeblen	size of external buffer
24	(18)	CHARACTER	8	dqedest	destination id
32	(20)	SIGNED	4	dqermax	max # of recs per cyc in eds **Note:** the EELHTDS record number and write cycle are stored in EELEVDS header
36	(24)	SIGNED	4	dqeevtr	used in SUTOP to pass htds rec
40	(28)	SIGNED	4	dqeevtc	used in SUTOP to pass htds cyc
44	(2C)	CHARACTER	100	dqedata	local data buffer
144	(90)	SIGNED	4	dqeadder	Additional recs w buffer
148	(94)	BITSTRING	1	*	free
149	(95)	CHARACTER	1	*	reserved
150	(96)	UNSIGNED	2	dqeevts	used in SUTOP to pass sseq
152	(98)	CHARACTER		dqeend	end of dqe

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	100	dqeevt	event data buffer mapping
0	(0)	SIGNED	2	dqennum	tws node number
2	(2)	SIGNED	2	dqerdrn	event reader number in node
4	(4)	CHARACTER	8	dqepos	event ds position
4	(4)	SIGNED	4	dqecyc	write cycle number
8	(8)	SIGNED	4	dqerec	record number in cycle
12	(C)	CHARACTER	8	*	free
20	(14)	CHARACTER	80	dqeexr	exit record, see dclexr

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	6	dqeclj	a DASD tracker needs
0	(0)	CHARACTER	5	dqecljob	clnjob prefix
5	(5)	CHARACTER	1	dqedsclas	datastore class if JCC

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	20	dqerfw	rfw data
0	(0)	CHARACTER	8	dqerfwdest	destination name
8	(8)	CHARACTER	8	dqeedp	edp information
8	(8)	SIGNED	4	dqeedpwcy	write cycle number
12	(C)	SIGNED	4	dqeedprec	record # of last record
16	(10)	SIGNED	4	dqerfwnnum	node number

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	48	dqerel	release job mapping
0	(0)	CHARACTER	8	dqerjbnm	job name
8	(8)	CHARACTER	8	dqerjbid	job number
16	(10)	CHARACTER	8	dqercnje	current nje node
24	(18)	CHARACTER	8	dqeronje	origin nje node
32	(20)	UNSIGNED	2	dqeaseq	submit sequence #
34	(22)	CHARACTER	4	dqeawsid	work station id

Offsets					
38	(26)	UNSIGNED	2	*	reserved
40	(28)	SIGNED	4	dqerojid	original job number
44	(2C)	CHARACTER	4	dqeassnam	controller subsystem name

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqesub	submit data mapping
0	(0)	CHARACTER	46	*	reserved - do not use
46	(2E)	CHARACTER	50	dqesubesp	DQESUBESP MUST BE MAXIMUM 50 CHARS TO BE FIT IN ESP DATA
46	(2E)	BITSTRING	1	dqesubFLA	DQE submit flags
		1...		dqeERRO	dqe in error
		.1..		dqeUSED	dqe used
		..11 1111		*	free
47	(2F)	CHARACTER	5	*	free
52	(34)	CHARACTER	8	dqetso	Tso user id or blank
60	(3C)	CHARACTER	4	dqewsid	work station id
64	(40)	CHARACTER	4	dqessnam	controller subsystem name
68	(44)	CHARACTER	8	dqejobn	job/STC name
76	(4C)	CHARACTER	8	dqejid	job# of last subbed job
84	(54)	CHARACTER	1	dqeSubType	J, C, D, or O
85	(55)	CHARACTER	1	*	free
86	(56)	UNSIGNED	2	dqesseq	submit sseq (J1)
88	(58)	SIGNED	4	dqerecsub	record in HTDS
92	(5C)	SIGNED	4	dqecycsub	cycle in HTDS
96	(60)	CHARACTER	4	*	reserved - do not use

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	28	dqersseq	sseq# sync request (J0)
0	(0)	CHARACTER	4	dqerwsid	work station id
4	(4)	CHARACTER	8	dqecreat	request evt creation time

Offsets					
4	(4)	SIGNED	4	dqedate	date format (00yydddf)
8	(8)	SIGNED	4	dqetime	time format (secs*100)
12	(C)	CHARACTER	4	dqerssnam	controller subsystem name
16	(10)	UNSIGNED	1	*	free
17	(11)	BITSTRING	1	dqerflg1	flagbyte byte 1
		1...		dqerask	req for curr evds sseq#
		.1..		dqercold	cold start the ws
		..1.		dqerdlte	remove the ws from evds
		...1		dqerset	req to set evds to wseq#
		.... 1...		dqeinit	Initialization event
		.... .111		*	reserved
18	(12)	UNSIGNED	2	dqewseq	Actual WS submit sseq (J0)
20	(14)	CHARACTER	8	dqeJdest	dqedest

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	18	dqecmini	initialize CM for oper
0	(0)	CHARACTER	8	dqecmjob	job/STC name
8	(8)	SIGNED	4	dqecmocc	occurrence number
12	(C)	SIGNED	4	dqecmopr	operation number
16	(10)	CHARACTER	2	dqecmrt	TASK ID requestor : EM = Event Manager ; AR = Automatic Recovery ; GS = General Service

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqerfp	RODM request for parms
0	(0)	CHARACTER	8	dqerfpd	requestor destination
8	(8)	CHARACTER	92	*	reserved

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqersl	RODM subsystem lost
0	(0)	CHARACTER	8	dqersldn	requestor destination
8	(8)	CHARACTER	4	dqerslssn	subsystem name
12	(C)	CHARACTER	88	*	reserved

Dynamic Critical Path:
dqedata containing data sent to critical path handler when an MCP action is performed on a critical predecessor and the dataspace has to be updated consequently

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqecrt	Crit Path recalc
0	(0)	CHARACTER	4	dqews	workstation name
4	(4)	CHARACTER	1	dqestatus	operation status
5	(5)	CHARACTER	1	dqeexstat	oper extended status
6	(6)	BITSTRING	1	dqeflags2	flags
		1...		dqenoped	noped operation
		.1..		dqemanheld	oper manually held
		..1.		dqemheldch	first dqe of a seq
		...1		dqenopch	last dqe of a seq
		.... 1...		dqetmdch	time dep chg
		.... .1..		dqetmdep	time dep
		.... ..1.		dqeopiach	oper ia changed
		1		*	free
7	(7)	BITSTRING	1	dqefla8gs3	free
		1...		dqeisfirst	first dqe of a seq
		.1..		dqeisfirst	add job to dataspace
		..1.		dqeislast	last dqe of a seq
		...1 1111		*	free
8	(8)	CHARACTER	8	dqejobname	job name
16	(10)	SIGNED	4	dqejobenix	job table entry index
20	(14)	CHARACTER	1	dqejobprty	job priority

Offsets					
21	(15)	CHARACTER	1	dqereqtype	MCP request type S - status change A - Add operation/dep D - Delete (oper, dep) M - Modify P - Critical indicat
22	(16)	CHARACTER	2	*	free
24	(18)	SIGNED	4	dqeindpred	pred j table entry ix
28	(1C)	CHARACTER	1	dqeoldstat	operation old status
29	(1D)	CHARACTER	1	dqecritind	critical indicator
30	(1E)	CHARACTER	10	dqeinparr	input arrival time
30	(1E)	CHARACTER	6	dqeinparrd	date
36	(24)	CHARACTER	4	dqeinparrt	time
40	(28)	CHARACTER	2	*	free
42	(2A)	CHARACTER	10	dqedeadl	deadline
42	(2A)	CHARACTER	6	dqedeadld	date
48	(30)	CHARACTER	4	dqedeadlt	time
52	(34)	CHARACTER	2	*	free
54	(36)	CHARACTER	10	dqeastart	actual start time
54	(36)	CHARACTER	6	dqeastartd	date
60	(3C)	SIGNED	4	dqeastartt	time
64	(40)	CHARACTER	2	*	free
66	(42)	CHARACTER	10	dqeaend	actual end time
66	(42)	CHARACTER	6	dqeaendd	date
72	(48)	SIGNED	4	dqeaendt	time
76	(4C)	SIGNED	4	dqeduration	duration
80	(50)	SIGNED	4	dqeactdur	actual duration
84	(54)	CHARACTER	4	dqeoprkey	operation index
84	(54)	UNSIGNED	3	dqeoccidx	occ number
87	(57)	UNSIGNED	1	dqeopridx	oper number
88	(58)	CHARACTER	4	dqeerrcode	job error code
92	(5C)	CHARACTER	4	dqeprekey	pred oper index
92	(5C)	UNSIGNED	3	dqepoccidx	occ number
95	(5F)	UNSIGNED	1	dqepopridx	oper number
96	(60)	CHARACTER	4	*	free

External buffer for dqecrt
It contains WLM data: Policy and Service Class

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	40	dqecrtBUF	DQE Type CRT
0	(0)	CHARACTER	1	dqewlmpol	WLM policy $CRFA
1	(1)	CHARACTER	5	*	free
6	(6)	CHARACTER	8	dqewlmclass	WLM Service Class
14	(E)	CHARACTER	10	dqelstart	latest start
14	(E)	CHARACTER	6	dqelstartd	latest start date
20	(14)	SIGNED	4	dqelstartt	ltst start time
24	(18)	CHARACTER	16	dqeopIA	operation IA
24	(18)	CHARACTER	6	dqeopIAD	date
30	(1E)	CHARACTER	4	dqeopIAT	time
34	(22)	CHARACTER	6	*	

Dynamic Critical Path:
dqedata containing data sent to critical path handler when a status change (EM) occurs or when the job is late or long running (WA)

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqecr1	Crit Path recalc
0	(0)	CHARACTER	1	dqestatu1	operation status
1	(1)	CHARACTER	1	dqeexstat1	oper extended status
2	(2)	CHARACTER	2	*	free
4	(4)	SIGNED	4	dqejobeni1	job table entry index
8	(8)	CHARACTER	1	dqereqtyp1	MCP request type S - status change L - Late R - Long Running
9	(9)	CHARACTER	1	dqeoldsta1	operation old status
10	(A)	BITSTRING	1	dqeflags4	flags
		1...		dqeurgch1	doa urgent queue chng
		.1..		dqedoaur1	doa urgent queue flag
		..1.		dqewlmpro1	WLM promotion flag
		...1 1111		*	free
11	(B)	CHARACTER	3	*	free

Offsets					
14	(E)	CHARACTER	10	dqeastar1	actual start time
14	(E)	CHARACTER	6	dqeastar1d	date
20	(14)	SIGNED	4	dqeastar1t	time
24	(18)	CHARACTER	2	*	free
26	(1A)	CHARACTER	10	dqeaen1	actual end time
26	(1A)	CHARACTER	6	dqeaen1d	date
32	(20)	SIGNED	4	dqeaen1t	time
36	(24)	SIGNED	4	dqeactdu1	actual duration
40	(28)	CHARACTER	4	dqeoprke1	operation index
40	(28)	UNSIGNED	3	dqeoccid1	occ number
43	(2B)	UNSIGNED	1	dqeoprid1	oper number
44	(2C)	CHARACTER	4	dqeerrcod1	job error code
48	(30)	CHARACTER	52	*	free

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqearc	DQE Type TWS Data Sto D90C
0	(0)	CHARACTER	4	dqearctyp	Service Required: D90C LOG = get MVS™ JobLog D90C SLO = get Struct. Log D90A OPI = ask Oper_info D90A SDEL= start delete D90A of old entries D90A
		common part			
4	(4)	CHARACTER	24	dqearcJRkey	Job Log retrieval key D90C
4	(4)	CHARACTER	8	dqearcJobId	Job Log Id D90C
12	(C)	CHARACTER	8	dqearcJobNa	Job Log Name D90C
20	(14)	SIGNED	4	dqearcRdrD	Job Start Rdr Date D90C
24	(18)	SIGNED	4	dqearcRdrT	Job Start Rdr Time D90C
28	(1C)	CHARACTER	8	dqearcdest	output DEST D90C
36	(24)	CHARACTER	16	dqearcADID	Application Name D90C
52	(34)	CHARACTER	10	dqearcIA	IA Date and Time D90C
52	(34)	CHARACTER	6	dqearcIADate	IA Date D90C
58	(3A)	CHARACTER	4	dqearcIATime	IA Time D90C
62	(3E)	CHARACTER	9	dqearcOPkey	Operation VSAM key D90C

Offsets					
62	(3E)	CHARACTER	8	dqearcOCC	OCC token D90C
70	(46)	UNSIGNED	1	dqearcOPR	Opr num D54A
71	(47)	CHARACTER	2	dqearccaller	EM=Event manager D90A WA=WorkStat. Analyzer D90A AR=Automatic Recover D90A GS=General Server D90A
		OPI type only			
73	(49)	CHARACTER	1	dqearcfla1	flags (Ask Oper_Info)
		1...		ARC_PSUpost	1 = post PSU D90A
		.1..		ARC_CleanUp	1 = Stand Alone CLNUP
		..1.		ARC_SL	1 = StepList required
		...1		ARC_DS	1 = DSList required
		.... 1...		ARC_ExpJCL	1 = use expanded JCL
		.... .1..		ARC_BestStep	1 = start from BSTEP
		.... ..1.		ARC_noask	1 = cp14nostr ON
		1		ARC_AskSimGDG	1 = GDG sim required
74	(4A)	CHARACTER	8	*	free
82	(52)	CHARACTER	8	dqearcUSER	original job userid
		Clean Up only			
90	(5A)	CHARACTER	8	dqearcEXdest	execution destination
		AR restart only			
98	(62)	UNSIGNED	1	dqearcopiRet	retry counter
99	(63)	UNSIGNED	1	dqearcARste	AR restart step 01A (currently not used)

AR restart only
structure mapped into DQEARC buffer:

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	16	dqearcBUF	DQE Type TWS Data Sto
0	(0)	CHARACTER	8	dqearcSTEPN	Stepname (AR)
8	(8)	CHARACTER	8	dqearcPSTEPN	Proc Step Name (AR)

Note: For the dqearc SDEL type the information are all contained in the buffer pointed by dqebptr: dqebptr-> CP16 record layout

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqePSU	DQE Type TWS Data Sto
0	(0)	CHARACTER	4	dqePSUtyp	Service Required: APL = Apply logic A CLN = Stand Alone CLN REM = resume suspended DEL = delete request
---------- D90A - common part D90A ---------- ----------D90A					
4	(4)	CHARACTER	45	dqePSUcommon	D90A
4	(4)	CHARACTER	2	dqePSUcaller	FL=Fetch JobLog D90A GS=General Server D90A EM=Event manager D90A
6	(6)	CHARACTER	8	dqePSUJobNa	Job Name D90A
14	(E)	CHARACTER	9	dqePSUOPkey	Operation VSAM key D90A
14	(E)	CHARACTER	8	dqePSUOCC	Occ token D90A
22	(16)	UNSIGNED	1	dqePSUOPR	Opr num D90A
23	(17)	CHARACTER	16	dqePSUADID	Application Name D90A
39	(27)	CHARACTER	10	dqePSUIA	IA Date and Time D90A
39	(27)	CHARACTER	6	dqePSUIADate	IA Date D90A
45	(2D)	CHARACTER	4	dqePSUIATime	IA Time D90A
---------- D90A - GS only D90A ---------- ----------D90A					
49	(31)	CHARACTER	8	dqePSUuser	TSO userid D90A
57	(39)	CHARACTER	4	dqePSUtoken	GS request token D90A
---------- D90A - FL for AR only D90A ---------- --------D90A					
61	(3D)	CHARACTER	8	dqeStepname	AR restart step D90A
69	(45)	CHARACTER	8	dqePStepName	AR restart proc step D90A
---------- D90A - CLN type only -D90A ---------- ----------D90A					
77	(4D)	CHARACTER	8	dqePSUexdest	execution DEST D90A
---------- D90A - APL type only -D90A ---------- ---------D90A					
85	(55)	CHARACTER	1	dqePSUfla1	D90A
		1...		PSU_ExpJCL	1 = use expanded JCL D90A
		.1..		PSU_Suspend	1 = suspend DQE D90A

Offsets					
		..1.		PSU_Operinfo	1 = SL from buffer D90A
		...1		PSU_UserSys	1 = add usersys DD
		.... 1...		PSU_SimGDG	1 = simulate GDG
		.... .1..		PSU_Root	1 = GDG root list
		.... ..1.		PSU_BestStep	1 = start from BSTEP
		1		PSU_AskSimGDG	1 = req simulation
		2 char(2), free 2 dqePSULen,			
86	(56)	UNSIGNED	2	PSU_SLlen	SL length
88	(58)	UNSIGNED	3	PSU_DSlen	DL length
91	(5B)	UNSIGNED	3	PSU_GDGlen	Sim. GDG info
94	(5E)	UNSIGNED	2	PSU_Rootlen	Root len
96	(60)	SIGNED	4	PSU_JCLlen	JCL length D90A

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	dqeWLM	sub WLM reset request D63A
0	(0)	CHARACTER	8	dqetsoW	Tso user id or blank D63A
8	(8)	CHARACTER	8	dqejobnW	job/STC name D63A
16	(10)	CHARACTER	8	dqeSVCnW	WLM Hi perf service class
24	(18)	CHARACTER	4	dqewsidW	work station id D63A
28	(1C)	CHARACTER	16	dqeadidW	application ID D63A
44	(2C)	CHARACTER	10	dqeociaW	occurrence input arrival
44	(2C)	CHARACTER	6	dqeociaD	occurence IA date D63A
50	(32)	CHARACTER	4	dqeociaT	occurence IA time D63A
54	(36)	SIGNED	2	dqeopnumW	operation number D63A
56	(38)	CHARACTER	4	dqessnamW	controller subsystem name
60	(3C)	UNSIGNED	2	*	free
		2 dqejidW char(8) , job of last subbed job 2 char(2) , reserved D63D			
62	(3E)	SIGNED	2	dqeasidW	job asid D63A

2 dqetoken_fullW , operation token D63d 3 dqetoken_preW char(4), operation token prefix 3 dqetokenW unsigned bin(32), operation token 2 dqeaccmpW ptr(31) , acc meth parms Q0A 2 dqenetidW char(8) , APPC netid (final dest) 2 dqenetLUW char(8) ; APPC netLU (final dest)

Offsets					
64	(40)	SIGNED	4	dqeretcW	WLM promot request RC
68	(44)	SIGNED	2	dqersncW	WLM promot request RSN
70	(46)	CHARACTER	10	*	reserved D63A

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	8	dqeTcpIp	tcpip evt mapping
0	(0)	ADDRESS	4	dqeSocketIdPtr	
4	(4)	CHARACTER	4	dqeSocketDomain	

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	18	dqeConfFile	Conf File mapping
0	(0)	CHARACTER	1	dqeRequestType	
1	(1)	CHARACTER	3	*	
4	(4)	CHARACTER	4	dqeRequestCRC	
8	(8)	CHARACTER	8	dqeCRCOwnerDest	
16	(10)	SIGNED	2	dqeDtbDestIndex	

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	44	dqeHTTPaction	HTTP Joblog retrieval Kill Query job status
0	(0)	CHARACTER	8	dqeHTocctoken	HTTP occurence token
8	(8)	CHARACTER	8	dqeHTsubtoken	HTTP submission token
16	(10)	CHARACTER	8	dqeHTjobname	HTTP job name
24	(18)	CHARACTER	4	dqeHTwsname	HTTP ws name
28	(1C)	CHARACTER	4	dqeHTssname	HTTP subsystem name
32	(20)	SIGNED	2	dqeHTopnum	HTTP operation number
34	(22)	SIGNED	2	*	reserved
36	(24)	CHARACTER	8	dqeHTuser	HTTP joblog req user

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqeHTnotify	HTTP notify

Offsets					
0	(0)	CHARACTER	56	dqeHNalias	alias
56	(38)	CHARACTER	8	dqeHNjobid	job number
64	(40)	CHARACTER	4	dqeHNerrc	error code (stat E)
68	(44)	CHARACTER	6	dqeHNstartd	start date
74	(4A)	CHARACTER	6	dqeHNendd	end date
80	(50)	SIGNED	4	dqeHNstartt	start time
84	(54)	SIGNED	4	dqeHNendt	end time
88	(58)	CHARACTER	8	dqeHNxdtoken	XD99 key
96	(60)	CHARACTER	1	dqeHNstatus	status (S/C/E)
97	(61)	CHARACTER	3	*	free

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqeHTinfo	HTTP bind info/failed
0	(0)	CHARACTER	56	dqeHIalias	alias
56	(38)	CHARACTER	8	dqeHIjobname	matched jobname
64	(40)	CHARACTER	10	dqeHIia	matched ia
64	(40)	CHARACTER	6	dqeHIiad	..IA date
70	(46)	CHARACTER	4	dqeHIiat	..IA time
74	(4A)	CHARACTER	1	dqeHItype	I=info, F=failed
75	(4B)	CHARACTER	1	*	free
76	(4C)	CHARACTER	8	dqeHIxdtoken	XD99 key
84	(54)	CHARACTER	4	dqeHIwsname	matched ws name
88	(58)	CHARACTER	12	*	free

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	100	dqeHTsubscr	HTTP subscribe
0	(0)	CHARACTER	35	dqeHSrkeyz	remote job key
0	(0)	CHARACTER	16	dqeHSadid	..remote adid/jsname (D/Z)
16	(10)	CHARACTER	16	dqeHSjsws	..remote jsws (D)
32	(20)	CHARACTER	3	dqeHSopno	..remote opno (Z)
35	(23)	CHARACTER	1	dqeHSflags	flags
		1...		dqeHSzos	..ON: remote eng type is Z
		.111 1111		*	..free
36	(24)	CHARACTER	2	*	free
38	(26)	CHARACTER	10	dqeHSia	IA for match
38	(26)	CHARACTER	6	dqeHSiad	..IA date

Offsets					
44	(2C)	CHARACTER	4	dqeHSiat	..IA time
48	(30)	CHARACTER	52	dqeHSalias	data for alias
48	(30)	CHARACTER	8	dqeHSocctoken	..occurence token
56	(38)	CHARACTER	8	dqeHSsubtoken	..submission token
64	(40)	CHARACTER	8	dqeHSjobname	..job name
72	(48)	CHARACTER	4	dqeHSwsname	..ws name
76	(4C)	CHARACTER	4	dqeHSssname	..subsystem name
80	(50)	SIGNED	2	dqeHSopnum	..operation number
82	(52)	CHARACTER	18	*	free

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	40	dqeHSbuff	DQE Type HTS
0	(0)	CHARACTER	40	dqeHSrjobnm	remote jobname

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	24	dqeJKJes	
0	(0)	CHARACTER	8	dqeJKJJobName	
8	(8)	CHARACTER	8	dqeJKJJobId	
16	(10)	SIGNED	4	dqeJKJRdrDate	
20	(14)	SIGNED	4	dqeJKJRdrTime	

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	56	dqeJKBroker	
0	(0)	CHARACTER	56	dqeJKBJobId	

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	8	dqeJAlias	
0	(0)	SIGNED	4	dqeJALength	
4	(4)	ADDRESS	4	dqeJAptr	

Cross reference

Name	Hex Offset	Hex Value	Level
ARC_AskSimGDG	49	01	3
ARC_BestStep	49	04	3
ARC_CleanUp	49	40	3

Name	Hex Offset	Hex Value	Level
ARC_DS	49	10	3
ARC_ExpJCL	49	08	3
ARC_noask	49	02	3
ARC_PSUpost	49	80	3
ARC_SL	49	20	3
dqe	0		1
dqeactdur	50		2
dqeactdu1	24		2
dqeadder	90		2
dqeaddjob	7	40	3
dqeadidW	1C		2
dqeaend	42		2
dqeaendd	42		3
dqeaendt	48		3
dqeaen1	1A		2
dqeaen1d	1A		3
dqeaen1t	20		3
dqearc	0		1
dqearcADID	24		2
dqearcARste	63		2
dqearcBUF	0		1
dqearccaller	47		2
dqearcdest	1C		2
dqearcEXdest	5A		2
dqearcfla1	49		2
dqearcIA	34		2
dqearcIADate	34		3
dqearcIATime	3A		3
dqearcJobId	4		3
dqearcJobNa	C		3
dqearcJRkey	4		2
dqearcOCC	3E		3
dqearcopiRet	62		2
dqearcOPkey	3E		2
dqearcOPR	46		3
dqearcPSTEPN	8		2
dqearcRdrD	14		3
dqearcRdrT	18		3
dqearcSTEPN	0		2
dqearctyp	0		2
dqearcUSER	52		2

Name	Hex Offset	Hex Value	Level
dqeaseq	20		2
dqeasidW	3E		2
dqeassnam	2C		2
dqeastart	36		2
dqeastartd	36		3
dqeastartt	3C		3
dqeastar1	E		2
dqeastar1d	E		3
dqeastar1t	14		3
dqeawsid	22		2
dqeblen	14		2
dqebptr	10		2
dqeclj	0		1
dqecljob	0		2
dqecmini	0		1
dqecmjob	0		2
dqecmocc	8		2
dqecmopr	C		2
dqecmrt	10		2
dqeConfFile	0		1
dqeCRCOwnerDest	8		2
dqecreat	4		2
dqecritind	1D		2
dqecrt	0		1
dqecrtBUF	0		1
dqecr1	0		1
dqecyc	4		3
dqecycsub	5C		3
dqedata	2C		2
dqedate	4		3
dqedeadl	2A		2
dqedeadld	2A		3
dqedeadlt	30		3
dqedesc	0		2
dqedest	18		2
dqedoaur1	A	40	3
dqedsclas	5		2
dqeDtbDestIndex	10		2
dqeduration	4C		2
dqeedp	8		2
dqeedprec	C		3

Name	Hex Offset	Hex Value	Level
dqeedpwcy	8		3
dqeend	98		2
dqeerrcode	58		2
dqeerrcod1	2C		2
dqeERRO	2E	80	4
dqeevt	0		1
dqeevtc	28		2
dqeevtr	24		2
dqeevts	96		2
dqeexr	14		2
dqeexstat	5		2
dqeexstat1	1		2
dqeflags	B		2
dqeflags2	6		2
dqeflags3	7		2
dqeflags4	A		2
dqeflres	B	80	3
dqeHIalias	0		2
dqeHIia	40		2
dqeHIiad	40		3
dqeHIiat	46		3
dqeHIjobname	38		2
dqeHItype	4A		2
dqeHIwsname	54		2
dqeHIxdtoken	4C		2
dqeHNalias	0		2
dqeHNendd	4A		2
dqeHNendt	54		2
dqeHNerrc	40		2
dqeHNjobid	38		2
dqeHNstartd	44		2
dqeHNstartt	50		2
dqeHNstatus	60		2
dqeHNxdtoken	58		2
dqeHSadid	0		3
dqeHSalias	30		2
dqeHSbuff	0		1
dqeHSflags	23		2
dqeHSia	26		2
dqeHSiad	26		3
dqeHSiat	2C		3

Name	Hex Offset	Hex Value	Level
dqeHSjobname	40		3
dqeHSjsws	10		3
dqeHSocctoken	30		3
dqeHSopno	20		3
dqeHSopnum	50		3
dqeHSrjobnm	0		2
dqeHSrkeyz	0		2
dqeHSssname	4C		3
dqeHSsubtoken	38		3
dqeHSwsname	48		3
dqeHSzos	23	80	3
dqeHTinfo	0		1
dqeHTjobname	10		2
dqeHTnotify	0		1
dqeHTocctoken	0		2
dqeHTopnum	20		2
dqeHTssname	1C		2
dqeHTsubscr	0		1
dqeHTsubtoken	8		2
dqeHTTPaction	0		1
dqeHTuser	24		2
dqeHTwsname	18		2
dqeindpred	18		2
dqeinit	11	08	3
dqeinparr	1E		2
dqeinparrd	1E		3
dqeinparrt	24		3
dqeisfirst	7	80	3
dqeislast	7	20	3
dqeJALength	0		2
dqeJAlias	0		1
dqeJAptr	4		2
dqeJdest	14		2
dqejid	4C		3
dqeJKBJobId	0		2
dqeJKBroker	0		1
dqeJKJes	0		1
dqeJKJJobId	8		2
dqeJKJJobName	0		2
dqeJKJRdrDate	10		2
dqeJKJRdrTime	14		2

Name	Hex Offset	Hex Value	Level
dqejobenix	10		2
dqejobeni1	4		2
dqejobn	44		3
dqejobname	8		2
dqejobnW	8		2
dqejobprty	14		2
dqelstart	E		2
dqelstartd	E		3
dqelstartt	14		3
dqemanheld	6	40	3
dqemheldch	6	20	3
dqenckpt	B	20	3
dqennum	0		2
dqenopch	6	10	3
dqenoped	6	80	3
dqeoccidx	54		3
dqeoccid1	28		3
dqeociaD	2C		3
dqeociaT	32		3
dqeociaW	2C		2
dqeoldstat	1C		2
dqeoldsta1	9		2
dqeopIA	18		2
dqeopiach	6	02	3
dqeopIAD	18		3
dqeopIAT	1E		3
dqeopnumW	36		2
dqeopridx	57		3
dqeoprid1	2B		3
dqeoprkey	54		2
dqeoprke1	28		2
dqepoccidx	5C		3
dqepopridx	5F		3
dqepos	4		2
dqeprekey	5C		2
dqePStepName	45		2
dqePSU	0		1
dqePSUADID	17		3
dqePSUcaller	4		3
dqePSUcommon	4		2
dqePSUexdest	4D		2

Name	Hex Offset	Hex Value	Level
dqePSUfla1	55		2
dqePSUIA	27		3
dqePSUIADate	27		4
dqePSUIATime	2D		4
dqePSUJobNa	6		3
dqePSUOCC	E		4
dqePSUOPkey	E		3
dqePSUOPR	16		4
dqePSUtoken	39		2
dqePSUtyp	0		2
dqePSUuser	31		2
dqerask	11	80	3
dqercnje	10		2
dqercold	11	40	3
dqerdlte	11	20	3
dqerdrn	2		2
dqerec	8		3
dqerecsub	58		3
dqerel	0		1
dqereqtype	15		2
dqereqtyp1	8		2
dqeRequestCRC	4		2
dqeRequestType	0		2
dqeretcW	40		2
dqerflg1	11		2
dqerfp	0		1
dqerfpd	0		2
dqerfw	0		1
dqerfwdest	0		2
dqerfwnnum	10		2
dqerjbid	8		2
dqerjbnm	0		2
dqermax	20		2
dqerojid	28		2
dqeronje	18		2
dqerset	11	10	3
dqersl	0		1
dqersldn	0		2
dqerslssn	8		2
dqersncW	44		2
dqersseq	0		1

Name	Hex Offset	Hex Value	Level
dqerssnam	C		2
dqerwsid	0		2
dqeSocketDomain	4		2
dqeSocketIdPtr	0		2
dqesseq	56		3
dqessnam	40		3
dqessnamW	38		2
dqestatus	4		2
dqestatu1	0		2
dqeStepname	3D		2
dqesub	0		1
dqesubesp	2E		2
dqesubFLA	2E		3
dqeSubType	54		3
dqeSVCnW	10		2
dqeTcpIp	0		1
dqetime	8		3
dqetmdch	6	08	3
dqetmdep	6	04	3
dqetso	34		3
dqetsoW	0		2
dqetype	8		2
dqeurgch1	A	80	3
dqeUSED	2E	40	4
dqever	4		2
dqeWLM	0		1
dqewlmclass	6		2
dqewlmpol	0		2
dqewlmpro1	A	20	3
dqews	0		2
dqewseq	12		2
dqewsid	3C		3
dqewsidW	18		2
PSU_AskSimGDG	55	01	3
PSU_BestStep	55	02	3
PSU_DSlen	58		2
PSU_ExpJCL	55	80	3
PSU_GDGlen	5B		2
PSU_JCLlen	60		2
PSU_Operinfo	55	20	3
PSU_Root	55	04	3

Name	Hex Offset	Hex Value	Level
PSU_Rootlen	5E		2
PSU_SimGDG	55	08	3
PSU_SLlen	56		2
PSU_Suspend	55	40	3
PSU_UserSys	55	10	3

ESP - map of event record in the HTDS

Name : DCLESP

Function:
Allows for continuation of events passed to the Data Router via the Event Writer queue and the event data set. This mapping should be used only when more than 80 bytes should be propagated using the above. The ESP is transformed into a DQE, using the ESP_dqetype as the DQE type, the data in the type F ESP as the DQE internal buffer,and the remainder of ESP data as the DQE external buffer. The subpool of the allocated external buffer should be indicated in the ESP_sp. The EW will freemain this area.
Zeroes yields SP0.
Use only key 0 storage.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	ESP	ESP head or continuation
0	(0)	CHARACTER	1	ESP_exrtyp	event type = N
1	(1)	CHARACTER	1	ESP_type	type of ESP F = first / head ESP N = continued ESP
2	(2)	CHARACTER	1	ESP_syst	contained event type ! blank === above MUST match DCLEXR ==
3	(3)	CHARACTER	3	ESP_dqetype	actual contained event type
6	(6)	SIGNED	2	ESP_seqF	ESP F-type sequence number
8	(8)	SIGNED	4	ESP_seqN	ESP N-type sequence number
12	(C)	SIGNED	4	ESP_seqT	ESP seq totals
16	(10)	ADDRESS	4	ESP_extptr	address of data buffer ! NULL
20	(14)	SIGNED	4	ESP_datasize	Total size in buffer when ESP on WRTQ, and for F-type. Data size for N-types.

Offsets

24	(18)	UNSIGNED	1	ESP_sp	subpool number for ext buffer
25	(19)	CHARACTER	1	ESP_flags	a flag byte
		1...		ESP_app	APP extension buffer
		.111 1111		*	reserved
26	(1A)	CHARACTER	50	ESP_data	actual event data
76	(4C)	CHARACTER	4	ESP_id	event identification (offs 76)
80	(50)	CHARACTER		ESP_xdata0	eXtended data portion plachold

Cross reference

Name	Hex Offset	Hex Value	Level
ESP	0		1
ESP_app	19	80	3
ESP_data	1A		2
ESP_datasize	14		2
ESP_dqetype	3		2
ESP_exrtyp	0		2
ESP_extptr	10		2
ESP_flags	19		2
ESP_id	4C		2
ESP_seqF	6		2
ESP_seqN	8		2
ESP_seqT	C		2
ESP_sp	18		2
ESP_syst	2		2
ESP_type	1		2
ESP_xdata0	50		2

EVT - Map of record layout in event data sets

Name : DCLEVT

Function:
This segment describes the record layout in the event data sets.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	820	evt	event record
0	(0)	CHARACTER	20	evtkey	event record key

Offsets					
0	(0)	SIGNED	4	evtrec	for header = 0 rec number for non-header
4	(4)	SIGNED	4	evtlrrec	latest record number = 0 for non-header
8	(8)	SIGNED	4	evtlrcyc	latest cycle for header
8	(8)	SIGNED	4	evtcyc	write cycle for non-header
12	(C)	SIGNED	4	evtmax	max rcds excl hdr = 0 for non-header
16	(10)	SIGNED	4	evttrcap	track capacity in rcds = 0 for non-header
20	(14)	CHARACTER	80	evtexr (10)	exit record (see dclexr)
20	(14)	CHARACTER	80	evtsur	not used
20	(14)	CHARACTER	80	evtrel	not used

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	*	*	evds header record data
0	(0)	CHARACTER	16	evtckpte (*)	checkpoint entry
0	(0)	BITSTRING	1	*	free
1	(1)	UNSIGNED	1	*	free
2	(2)	CHARACTER	4	evtwsnm	workstation name
6	(6)	UNSIGNED	2	evtsseq	current submit seq
8	(8)	SIGNED	4	evtchkrec	latest submit record num
12	(C)	SIGNED	4	evtchkcyc	latest submit cycle num

Cross reference

Name	Hex Offset	Hex Value	Level
evt	0		1
evtchkcyc	C		3
evtchkrec	8		3
evtckpte	0		2
evtcyc	8		4
evtexr	14		2
evtkey	0		2

Name	Hex Offset	Hex Value	Level
evtlrcyc	8		3
evtlrrec	4		3
evtmax	C		3
evtrec	0		3
evtrel	14		4
evtsseq	6		3
evtsur	14		3
evttrcap	10		3
evtwsnm	2		3

EXI - Ix event definition

Name : DCLEXI

Function:
This segment declares an initialization event. This event contains information about an initiated operation. Initialization events are created by the SUBMIT task and added to the Event Writer queue. Note that the beginning of `exi` must be mapped as `exr`.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	exi	initialization event
0	(0)	CHARACTER	1	exievtyp	event type = I
1	(1)	CHARACTER	3	exitype	type of initiated process J0 = submit seq event J1 = jcl submit J2 = jcl started task J3 = jcl for clean up D90A J4 = jcl submit SE failure $CCUA WTO= WTO message REL= job release OSI= OSI type submit WLM= WLM job promotion
4	(4)	CHARACTER	4	exiwsid	work station id
8	(8)	CHARACTER	16	*	exiadid
8	(8)	CHARACTER	16	exiJ4schenv	Se name (J4)
8	(8)	CHARACTER	8	exircre	Request create time (J0)
8	(8)	SIGNED	4	exircred	date (format: 00YYDDDF)

Offsets					
12	(C)	SIGNED	4	exircret	time (format: secs*100)
16	(10)	CHARACTER	8	exiocctok	
16	(10)	UNSIGNED	2	exiwseq	submit seq from ws (IJ0)
24	(18)	CHARACTER	10	*	
24	(18)	SIGNED	4	exierec	current evds rec (IJ0)
28	(1C)	SIGNED	4	exiecyc	current evds cyc (IJ0)
32	(20)	SIGNED	2	*	
34	(22)	SIGNED	2	exiopnum	operation number
36	(24)	SIGNED	4	exiWLMrc	WLM promotion request RC
36	(24)	BITSTRING	4	exiflags	flag area P10A
36	(24)	BITSTRING	1	exiflag1	flag byte 1 P10C
		1...		exifail	init of process failed P10C
		.1..		exisfail	submit fail
		..1.		exinoedp	no edp updates for this ev31CLVA
		...1		exiWLMah	on= job already Hi perfo D52A
		.... 1...		exiSEund	on=SE not defined
		.... .1..		exiSEnoAva	on=SE not available
		.... ..1.		exiSEnoJPL	on=SE not avail at JPLEX 31Clevel only
		1		exiinit	J0 related to initialization
37	(25)	BITSTRING	1	exiflag2	flag byte 2 P10C
		1...		exi2fail	init of process failed P10C
		.1..		exi2plte	possible late I event P10C
		..1.		exi2bad	the STC funct not active P10C
		...1		exi2njss	not JES managed subsyst P10C
		.... 1...		exi2fjes	STC fail JES input servic31CP10C
		.... .1..		exi2ncf	NCF destination not valid31CP10C
		11		*	reserved P10C
38	(26)	BITSTRING	1	exiflag3	flag byte 3 P10C

Offsets					
		1...		exi3oos	submit out-of-sequence
		.1..		exi3nckp	checkpoint not active
		..1.		exi31st	1st time switch
		...1		exi3misr	seq request mismatch
		.... 1...		exi3allok	submit seq WA vs. SU okay
		.... .1..		*	reserved
		.... ..1.		exi3badj0	bad J0 event (or cp)
		1		exi3rdop	error read cp oper
39	(27)	BITSTRING	1	exiflag4	flag byte 4 P10C
		1...		exi4rdjn	error read cp jnt
		.1..		exi4nsub	sub04 is blank!
		..1.		exi4job	jes04 is blank!
		...1		*	reserved
		.... 1...		exi4nsop	cplsubop is of..!!?
		.... .11.		*	reserved
		1		exi4succ	successful submit
40	(28)	CHARACTER	8	exijobn	job/STC name
40	(28)	UNSIGNED	2	exiwork	current catchup value(J0)
42	(2A)	UNSIGNED	2	*	free
44	(2C)	UNSIGNED	2	exiedoa	current doa value (J0)
46	(2E)	UNSIGNED	2	exijasid	job/STC asid D52C9TA
48	(30)	CHARACTER	8	exijobid	job/STC number (J1,J2,J3)
48	(30)	CHARACTER	8	exiWLMCls	WLM class (WLM only)
48	(30)	CHARACTER	8	exiJ4dest	tracker dest (J4)
48	(30)	CHARACTER	1	exievlog	reserved (WLM) 32C
49	(31)	CHARACTER	7	*	reserved (WLM)
56	(38)	SIGNED	2	exigmtof	gmt offset in minutes
58	(3A)	SIGNED	2	exiWLMrsn	WLM promotion request rsn code31A
58	(3A)	CHARACTER	1	exiVirt	Virtual WS support: Y/N
59	(3B)	CHARACTER	1	*	

Offsets					
60	(3C)	CHARACTER	8	exicreat	event record creation time word bndry
60	(3C)	SIGNED	4	exidate	date format (00yydddf)
64	(40)	SIGNED	4	exitime	time format (secs*100)
68	(44)	CHARACTER	4	exissnm	DQE originating TWS subsys name
72	(48)	UNSIGNED	2	exieseq	current evds seq (IJ0)
74	(4A)	UNSIGNED	2	*	free
76	(4C)	CHARACTER	4	exiid	event id EELx

Cross reference

Name	Hex Offset	Hex Value	Level
exi	0		1
exicreat	3C		2
exidate	3C		3
exiecyc	1C		3
exiedoa	2C		3
exierec	18		3
exieseq	48		2
exievlog	30		5
exievtyp	0		2
exifail	24	80	5
exiflags	24		3
exiflag1	24		4
exiflag2	25		4
exiflag3	26		4
exiflag4	27		4
exigmtof	38		2
exiid	4C		2
exiinit	24	01	5
exijasid	2E		3
exijobid	30		2
exijobn	28		2
exiJ4dest	30		4
exiJ4schenv	8		3
exinoedp	24	20	5
exiocctok	10		4
exiopnum	22		2

Name	Hex Offset	Hex Value	Level
exircre	8		4
exircred	8		5
exircret	C		5
exiSEnoAva	24	04	5
exiSEnoJPL	24	02	5
exiSEund	24	08	5
exisfail	24	40	5
exissnm	44		2
exitime	40		3
exitype	1		2
exiVirt	3A		3
exiWLMah	24	10	5
exiWLMCls	30		3
exiWLMrc	24		2
exiWLMrsn	3A		2
exiwork	28		3
exiwseq	10		5
exiwsid	4		2
exi2bad	25	20	5
exi2fail	25	80	5
exi2fjes	25	08	5
exi2ncf	25	04	5
exi2njss	25	10	5
exi2plte	25	40	5
exi3allok	26	08	5
exi3badj0	26	02	5
exi3misr	26	10	5
exi3nckp	26	40	5
exi3oos	26	80	5
exi3rdop	26	01	5
exi31st	26	20	5
exi4job	27	20	5
exi4nsop	27	08	5
exi4nsub	27	40	5
exi4rdjn	27	80	5
exi4succ	27	01	5

EXK - Kx event definition

Name : DCLEXK

Function:

This segment declares an initialization event. This event contains information about an initiated operation. Initialization events are created by the SUBMIT task and added to the Event Writer queue. Note that the beginning of `exk` must be mapped as `exr`.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	80	exk	initialization event
0	(0)	CHARACTER	1	exkevtyp	event type = K
1	(1)	CHARACTER	2	exktype	type of initiated process J1= jcl submit
3	(3)	CHARACTER	1	*	free
4	(4)	SIGNED	4	exkJKJJobIdN	JES Job Id (only number)
8	(8)	CHARACTER	36	exkJKBJobId	Broker Job Key
44	(2C)	CHARACTER	8	exkJobname	Jobname
52	(34)	SIGNED	4	exkdate	00YYDDDF exkdate
56	(38)	SIGNED	4	exktime	time format (secs*100)exktime
60	(3C)	CHARACTER	4	exkssname	origi subsys
64	(40)	BITSTRING	2	exkflags	
64	(40)	BITSTRING	1	exkflags1	
		1...		exkfail	init of process failed
		.1..		exkSEnoAva	on=SE not available
		..1.		exkSEnoJPL	on=SE not avail at JPLEX
		...1		exkSEund	on=SE not defined
		.... 1...		exk2fail	init of process failed
		.... .1..		exk2plte	possible late I event
		.... ..1.		exk2bad	the STC funct not active
		1		exk2njss	not JES managed subsyst
65	(41)	BITSTRING	1	exkflags2	
		1...		exk2fjes	STC fail JES input servic
		.1..		exk3nckp	checkpoint not active
		..1.		exk3oos	submit out-of-sequence

Offsets					
		...1 1111		*	free
66	(42)	BITSTRING	1	exkreason	free
		1...		exkJCLRopen	open joblib failure
		.1..		exkJCLRdyn	dynalloc failure
		..1.		exkJCLRjnMISS	missing jobname
		...1		exkJCLRjnINVA	invalid jobname
		.... 1...		exkJCLRstor	storage problems
		.... .1..		exkOJCV	variable sub error
		.... ..1.		exknoJCL	JCL not found
		1		*	free
67	(43)	BITSTRING	1	*	free
68	(44)	SIGNED	4	*	free
72	(48)	SIGNED	2	exkgmtof	gmt offset in minutes
74	(4A)	UNSIGNED	2	exksseq	current submit seq
76	(4C)	CHARACTER	4	exkid	event id EELx

Cross reference

Name	Hex Offset	Hex Value	Level
exk	0		1
exkdate	34		2
exkevtyp	0		2
exkfail	40	80	4
exkflags	40		2
exkflags1	40		3
exkflags2	41		3
exkgmtof	48		2
exkid	4C		2
exkJCLRdyn	42	40	3
exkJCLRjnINVA	42	10	3
exkJCLRjnMISS	42	20	3
exkJCLRopen	42	80	3
exkJCLRstor	42	08	3
exkJKBJobId	8		2
exkJKJJobIdN	4		2
exkJobname	2C		2
exknoJCL	42	02	3
exkOJCV	42	04	3
exkreason	42		2
exkSEnoAva	40	40	4

Name	Hex Offset	Hex Value	Level
exkSEnoJPL	40	20	4
exkSEund	40	10	4
exksseq	4A		2
exkssname	3C		2
exktime	38		2
exktype	1		2
exk2bad	40	02	4
exk2fail	40	08	4
exk2fjes	41	80	4
exk2njss	40	01	4
exk2plte	40	04	4
exk3nckp	41	40	4
exk3oos	41	20	4

EXR - Exit record

Name : DCLEXR

Function:
This segment declares an exit record. Exit records are built by SMF and JES exits, passed to the event writer via CSA buffers, and are written to an event data set as part of an event record by the event writer.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	exr	exit record
0	(0)	CHARACTER	2	exrtype	record type
0	(0)	CHARACTER	1	exrsyst	system type A!B
1	(1)	CHARACTER	1	exrevtyp	event type 1!2!3!4!5!6
2	(2)	CHARACTER	1	exrstype	event subtype (only type3)
3	(3)	BITSTRING	1	exrflags	exit flags
		1...		exrretry	retry release command
		.1..		exropcho	this job is in hold
		..1.		exroheld	job was held by tws
		...1		exrjkill	job killed by jes in rdr
		.... 1...		exrjccec	error code from jcc
		.... .1..		exrjccch	checked by jcc
		.... ..1.		exrjccer	error in jcc

Offsets					
		1		exrrlast	retcode(last) active
4	(4)	BITSTRING	1	exrtermf	termination flags
		1...		exrjcset	exrjcout is valid
		.1..		exrjcout	job has jcc sysout
		..1.		*	free D90C
		...1		exrmchld	1= msgclass is held
		.... 1...		exrcmreq	1= requeue msg class
		.... .1..		*	free D90C
		.... ..1.		exrjbtab	0=normal, 1=abend (job)
		.... ..1.		exrabend	0=normal, 1=abend (step)
		1		exrflush	0=normal, 1=flushed step
5	(5)	BITSTRING	1	exrerror	job error switch lcterror
		1...		exrfail	job failed
		.1..		exrJQA	ON= is a JQA
		..1.		exrZ2level	
		...1		exrR4level	
		.... 1...		*	alloc but not unalloc done
		.... .1..		exrcfal	job failed on cond codes
		.... ..1.		exrjcjob	JCJOB processed ok D54A
		1		exrlastfl	last step flushed $BGIA
6	(6)	SIGNED	2	exrgmtof	gmt offset in minutes
8	(8)	CHARACTER	8	exrjobn	job name
16	(10)	CHARACTER	8	exrjobid	job number
24	(18)	CHARACTER	8	exrcreat	event creation time
24	(18)	SIGNED	4	exrdate	date format (00yydddf)
28	(1C)	SIGNED	4	exrtime	time format (secs*100)
32	(20)	CHARACTER	8	exrjsrdr	jes reader date and time
32	(20)	SIGNED	4	exrrdate	date format (00yydddf)

Offsets					
36	(24)	SIGNED	4	exrrtime	time format (secs*100)
40	(28)	SIGNED	4	exrsdate	operation start date
44	(2C)	SIGNED	4	exrstime	operation start time
48	(30)	SIGNED	4	exredate	operation end date
52	(34)	SIGNED	4	exretime	operation end time
52	(34)	SIGNED	4	exrorgid	nje origin job number
56	(38)	CHARACTER	8	exrstepn	job step name
56	(38)	CHARACTER	8	exronje	name of orig nje nod
56	(38)	CHARACTER	1	exrclass	printout class
57	(39)	CHARACTER	1	*	reserved
58	(3A)	SIGNED	2	exrasid	job asid
60	(3C)	SIGNED	4	exrexeid	NJE execution jobn
64	(40)	CHARACTER	8	exrpstep	procedure step name
64	(40)	CHARACTER	8	exrnnje	this/next nje node
64	(40)	CHARACTER	8	exrform	form number
72	(48)	SIGNED	2	exrcode	completion/ condition code
74	(4A)	CHARACTER	1	exrindic	status indicators
		1...		exrjesv4	jes sp4 or above1/3P/
		.1..		exrspun	spun off ds rcd
		..1.		exrterm	oper terminated datagroup
		...1		exrinter	oper interrupted -:-
		.... 1...		exrrstrt	oper restarted -:-
		.... .1..		exrndest	not final f/$sysmsgs 3P
		.... ..1.		exrnods4	no $sysmsgs found 3P
		1		exrsuspd	suspended
75	(4B)	UNSIGNED	1	exrstpnr	step number
75	(4B)	BITSTRING	1	exrpurge	job purge bits
		111.		*	not used
		...1		exrSDEP	SDEP filter used
		.... 1...		exrlastab	last step abended

Offsets					
		.... .1..		exrstall	stepevents(all)
		.... ..1.		exrstnz	stepevents(no) flag
		1		exropcan	cancelled by oper
76	(4C)	CHARACTER	4	exropcid	tws identifier

Cross reference

Name	Hex Offset	Hex Value	Level
exr	0		1
exrabend	4	02	4
exrasid	3A		4
exrcfal	5	04	3
exrclass	38		4
exrcmreq	4	08	3
exrcode	48		2
exrcreat	18		2
exrdate	18		3
exredate	30		2
exrerror	5		2
exretime	34		2
exrevtyp	1		3
exrexeid	3C		4
exrfail	5	80	3
exrflags	3		2
exrflush	4	01	3
exrform	40		4
exrgmtof	6		2
exrindic	4A		2
exrinter	4A	10	3
exrjbtab	4	02	3
exrjccch	3	04	3
exrjccec	3	08	3
exrjccer	3	02	3
exrjcjob	5	02	3
exrjcout	4	40	3
exrjcset	4	80	3
exrjesv4	4A	80	3
exrjkill	3	10	3
exrjobid	10		2
exrjobn	8		2

Name	Hex Offset	Hex Value	Level
exrJQA	5	40	3
exrjsrdr	20		2
exrlastab	4B	08	4
exrlastfl	5	01	3
exrmchld	4	10	3
exrndest	4A	04	3
exrnnje	40		3
exrnods4	4A	02	3
exroheld	3	20	3
exronje	38		3
exropcan	4B	01	4
exropcho	3	40	3
exropcid	4C		2
exrorgid	34		3
exrpstep	40		2
exrpurge	4B		3
exrrdate	20		3
exrretry	3	80	3
exrrlast	3	01	3
exrrstrt	4A	08	3
exrrtime	24		3
exrR4level	5	10	3
exrsdate	28		2
exrSDEP	4B	10	4
exrspun	4A	40	3
exrstall	4B	04	4
exrstepn	38		2
exrstime	2C		2
exrstnz	4B	02	4
exrstpnr	4B		2
exrstype	2		2
exrsuspd	4A	01	3
exrsyst	0		3
exrterm	4A	20	3
exrtermf	4		2
exrtime	1C		3
exrtype	0		2
exrZ2level	5	20	3

HTI - HTTP Interface from C side to PLX side (EELHTCEC module)

Name : DCLHTI

Function:
The array contains information for the PLX inteface implemented in the EELHTCEC module.

Offsets

Dec	Hex	Type	Len	Name (Dim)
0	(0)	STRUCTURE	168	htiSubmitJob
0	(0)	CHARACTER	16	htiOccName
16	(10)	CHARACTER	10	htiOccIA
26	(1A)	CHARACTER	1	htiSubtype
27	(1B)	CHARACTER	1	htiReprocess
28	(1C)	SIGNED	4	htijclNrec
32	(20)	ADDRESS	4	htijclPtr
36	(24)	CHARACTER	36	htibrokerKey
36	(24)	CHARACTER	36	htibrokerKeyId
72	(48)	CHARACTER	8	htialias
72	(48)	SIGNED	4	htialiasLength
76	(4C)	ADDRESS	4	htialiasPtr
80	(50)	CHARACTER	8	htioutput
80	(50)	SIGNED	4	htioutputLength
84	(54)	ADDRESS	4	htioutputPtr
88	(58)	CHARACTER	52	htiREFinfo
88	(58)	CHARACTER	44	htiREFdsname
132	(88)	CHARACTER	8	htiREFmember
140	(8C)	CHARACTER	8	htiPosition
140	(8C)	SIGNED	4	htiwcycle
144	(90)	SIGNED	4	htirecnum
148	(94)	SIGNED	4	htiThreadnum
152	(98)	ADDRESS	4	htiVARTABptr
156	(9C)	CHARACTER	12	htiVARIA

Checkpoint update (UPCP) ==> zHTTPDqe.h eventTypeUpdCP

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	8	htiUpdChkPt	
0	(0)	SIGNED	4	htiUpdcyc	write cycle number

Offsets

4	(4)	SIGNED	4	htiUpdrrec	record number in cycle

Joblog request (JLGT) ==> zHTTPDqe.h eventTypeGetJoblog
Each JL record has 133 chars
htiStageArea is 150 records long

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	32	htiJlRequest	
0	(0)	CHARACTER	8	htiJLJobname	name of the JES job
8	(8)	CHARACTER	8	htiJLJESid	ID of the JES job
16	(10)	SIGNED	4	htiFirstRec	num of the first requested rec (>=1)
20	(14)	SIGNED	4	htiNumOfRecs	num of the requested recs
24	(18)	ADDRESS	4	htiStageArea	stage JL area (150 records long)
28	(1C)	CHARACTER	1	htiIsComplete	'Y': JL retrieved up to the end; 'N': otherwise
29	(1D)	CHARACTER	3	htiJLfiller	free

Cross reference

Name	Hex Offset	Hex Value	Level
htialias	48		2
htialiasLength	48		3
htialiasPtr	4C		3
htibrokerKey	24		2
htibrokerKeyId	24		3
htiFirstRec	10		2
htiIsComplete	1C		2
htijclNrec	1C		2
htijclPtr	20		2
htiJLfiller	1D		2
htiJLJESid	8		2
htiJLJobname	0		2
htiJlRequest	0		1
htiNumOfRecs	14		2
htiOccIA	10		2

Name	Hex Offset	Hex Value	Level
htiOccName	0		2
htioutput	50		2
htioutputLength	50		3
htioutputPtr	54		3
htiPosition	8C		2
htirecnum	90		3
htiREFdsname	58		3
htiREFinfo	58		2
htiREFmember	84		3
htiReprocess	1B		2
htiStageArea	18		2
htiSubmitJob	0		1
htiSubtype	1A		2
htiThreadnum	94		2
htiUpdChkPt	0		1
htiUpdcyc	0		2
htiUpdrrec	4		2
htiVARIA	9C		2
htiVARTABptr	98		2
htiwcycle	8C		3

HTSA - HTTP server task parameter area

Name : DCLHTSA

Function:
This control block is built, initialized, and freed by the HTTP server task PLX mainline module.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	584	htsa	
0	(0)	CHARACTER	456	htsaComm	
0	(0)	CHARACTER	4	htsadesc	block descriptor = 'HTSA'
4	(4)	CHARACTER	2	htsaver	block mapping version
6	(6)	BITSTRING	2	htsaflags	
		1...		htsassl	
8	(8)	ADDRESS	4	htsamcap	mca address
12	(C	ADDRESS	4	htsastopecbptr	pointer to stop ecb
16	(10)	SIGNED	4	htsaPortNum	Local port number of server

Offsets					
20	(14)	ADDRESS	4	htsaSyncecbptr	ptr to Cli to Serv Sync ecb
24	(18)	SIGNED	4	htsaTCPtime	time out for recv
28	(1C)	CHARACTER	8	htsacodep	code page
36	(24)	CHARACTER	9	htsatcpjn	tcpip job name
45	(2D)	CHARACTER	1	*	free
46	(2E)	CHARACTER	1	htsasslamod	ssl auth mode
47	(2F)	CHARACTER	1	htsasslktyp	ssl key ring type
48	(30)	CHARACTER	53	htsahostn	local hostname used
101	(65)	CHARACTER	3	*	free
104	(68)	CHARACTER	65	htsaSSLastr	SSL authorization string
169	(A9)	CHARACTER	3	*	free
172	(AC)	CHARACTER	121	htsaSSLkrnm	SSL key ring name
293	(125)	CHARACTER	3	*	free
296	(128)	CHARACTER	121	htsaSSLkrpw	SSL key password
417	(1A1)	CHARACTER	27	*	free
444	(1BC)	ADDRESS	4	htsaIDMvsPtr	ID pointer
448	(1C0)	ADDRESS	4	htsadiafp	DIAGNOSE flags address
452	(1C4)	ADDRESS	4	htsadiadp	DIAGNOSE data address
456	(1C8)	SIGNED	4	htsaproto	0 HTTP 1 HTTPS
460	(1CC)	ADDRESS	4	htsaioc	pointer to ioc HTDS
464	(1D0)	SIGNED	4	htsannum	entry in EDP table
468	(1D4)	SIGNED	4	htsatnum	num of threads for server
472	(1D8)	SIGNED	4	htsahtsnum	number of hts entries
476	(1DC)	ADDRESS	4	htsahtsptr	hts address
480	(1E0)	CHARACTER	8	htsasyslvl	system level
488	(1E8)	ADDRESS	4	htsaiocHTREF	pointer to ioc HTREF
492	(1EC)	ADDRESS	4	htsaJlibDCB	pointer to JBLIB dcb
496	(1F0)	ADDRESS	4	htsaJBuf	JBLIB buffer
500	(1F4)	CHARACTER	1	htsaVARSUB	Y: varsub needed N: varsub not needed
501	(1F5)	CHARACTER	3	*	free
504	(1F8)	SIGNED	4	htsarecconf	HTREF recovery usage

Offsets					
508	(1FC)	SIGNED	4	htsacycconf	HTREF recovery usage
512	(200)	UNSIGNED	2	htsasseqconf	HTREF recovery usage
514	(202)	CHARACTER	54	*	free
568	(238)	SIGNED	4	*	free
572	(23C)	SIGNED	4	*	free
576	(240)	CHARACTER	8	*	free
584	(248)	CHARACTER		htsaend	end of htsa

Cross reference

Name	Hex Offset	Hex Value	Level
htsa	0		1
htsacodep	1C		3
htsaComm	0		2
htsacycconf	1FC		2
htsadesc	0		3
htsadiadp	1C4		3
htsadiafp	1C0		3
htsaend	248		2
htsaflags	6		3
htsahostn	30		3
htsahtsnum	1D8		2
htsahtsptr	1DC		2
htsaIDMvsPtr	1BC		3
htsaioc	1CC		2
htsaiocHTREF	1E8		2
htsaJBuf	1F0		2
htsaJlibDCB	1EC		2
htsamcap	8		3
htsannum	1D0		2
htsaPortNum	10		3
htsaproto	1C8		2
htsarecconf	1F8		2
htsasseqconf	200		2
htsassl	6	80	4
htsasslamod	2E		3
htsaSSLastr	68		3
htsaSSLkrnm	AC		3
htsaSSLkrpw	128		3
htsasslktyp	2F		3

Name	Hex Offset	Hex Value	Level
htsastopecbptr	C		3
htsaSyncecbptr	14		3
htsasyslvl	1E0		2
htsatcpjn	24		3
htsaTCPtime	18		3
htsatnum	1D4		2
htsaVARSUB	1F4		2
htsaver	4		3

JCFB - JS interface feedback information

Name : DCLJCFB

Function:
This block is always passed back to caller when retrieving data through the JS interface through the GET routine.
Offsets to data are only set when passing data through GS.
NOTE: It is the caller's responsibility to freemain the storage pointed to by jcfdatp.
The amount of storage is in jcfstg, and the subpool number is in jcfsubp.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	88	jcf	feedback area for js interf
0	(0)	CHARACTER	4	jcfdesc	descriptor always 'JCF '
4	(4)	CHARACTER	2	jcfvers	block version
6	(6)	CHARACTER	8	jcfdtype	data type that datp points t see dclcjrq
14	(E)	BITSTRING	1	jcfflags	processing flags
		1...		jcfjcl	jcl indicator
		.111 1111		*	reserved
15	(F)	CHARACTER	1	*	not used
16	(10)	ADDRESS	4	jcfbufp	pointer to jcl record in buf
20	(14)	ADDRESS	4	jcfdatp	pointer to data
24	(18)	SIGNED	4	jcfdato	offset to data (JS record,Variables..)
28	(1C)	SIGNED	4	jcfstg	amount of storage pointed to by jcfdatp
32	(20)	SIGNED	4	jcfsubp	subpool in which stg pointed to by jcfdatp is getmained

Offsets					
36	(24)	SIGNED	4	jcflino	offset to js data lines
36	(24)	SIGNED	4	jcfsvaro	offset to prompt vars
40	(28)	ADDRESS	4	jcflinp	addr to js data lines
40	(28)	ADDRESS	4	jcfsvarp	addr to prompt vars entry
44	(2C)	SIGNED	4	jcflin	number of js data lines
44	(2C)	SIGNED	4	jcfvars	number of prompt vars
48	(30)	CHARACTER	8	jcfjfrom	where jcl retrived ux002name,js, library
56	(38)	CHARACTER	8	jcfjmem	jcl member name if jcfjfrom = library name else blank
64	(40)	CHARACTER	8	*	reserved
72	(48)	CHARACTER	1	jcfvstat	var subst. status V or ' '
73	(49)	CHARACTER	15	*	reserved

Cross reference

Name	Hex Offset	Hex Value	Level
jcf	0		1
jcflin	2C		2
jcfvars	2C		3
jcfbufp	10		2
jcfdato	18		2
jcfdatp	14		2
jcfdesc	0		2
jcfdtype	6		2
jcfflags	E		2
jcfjcl	E	80	3
jcfjfrom	30		2
jcfjmem	38		2
jcflino	24		2
jcflinp	28		2
jcfstg	1C		2
jcfsubp	20		2
jcfsvaro	24		3
jcfsvarp	28		3

Name	Hex Offset	Hex Value	Level
jcfvers	4		2
jcfvstat	48		2

JCL - LAYOUT

Name : DCLJCL

Function:
JCLTWS jcl layout
REFTWS jcl layout

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	JCLTWS	
0	(0)	CHARACTER	136	JCLBDY	FIXED PART OF JCL RECORD
0	(0)	CHARACTER	28	JCLKEY	
0	(0)	CHARACTER	16	JCLADID	APPLICATION NAME
16	(10)	CHARACTER	10	JCLIA	OCC INPUT ARRIVAL DATE+TIME
16	(10)	CHARACTER	6	JCLIAT1	DATE
22	(16)	CHARACTER	4	JCLIAT2	TIME
26	(1A)	SIGNED	2	JCLOPNUM	OPERATION NUMBER
28	(1C)	CHARACTER	4	JCLEYE	EYE CATCHER
32	(20)	UNSIGNED	1	JCLVERS	VERSION NUMBER
33	(21)	CHARACTER	1	*	free
34	(22)	CHARACTER	8	JCLJOBNM	JOBNAME
42	(2A)	CHARACTER	12	JCLVARIA	VAR IA from HTI
54	(36)	CHARACTER	2	*	free

```
============================================================
- 3 JCLWSN CHAR(4), WORK STATION NAME
- 3 JCLUPDAT, LAST UPDATE, DATE+TIME
- 4 JCLUPDT1 CHAR(6), DATE
- 4 JCLUPDT2 CHAR(4), TIME
============================================================
```

56	(38)	CHARACTER	8	JCLUSER	LAST UPDATE, USERID
64	(40)	CHARACTER	1	JCLUPTYP	UPDATE TYPE: not used
65	(41)	CHARACTER	1	JCLSTAT	OP. STATUS: not used
66	(42)	UNSIGNED	2	JCLLINES	NO OF LINES IN RECORD
68	(44)	CHARACTER	1	JCLFLAGS	FLAGBYTE

Offsets					
		1...		JCLJSFND	ON = JCL READ FROM JS
		.1..		JCLEDITD	ONCE BEEN EDITED
		..1.		JCLJDIRNOP	ON = NOP directive set
		...1 1111		*	NOT USED
69	(45)	CHARACTER	1	*	FREE
70	(46)	SIGNED	2	JCLVLINE	NUMBER OF VARIABLES
72	(48)	SIGNED	4	JCLSUBP	SUBPOOL FOR FREEMAIN
76	(4C)	UNSIGNED	2	JCLALIASLEN	Alias name length
78	(4E)	CHARACTER	2	JCLRES4	FREE
80	(50)	CHARACTER	56	JCLBROKERKEYID	Broker key identifier
136	(88)	CHARACTER	*	JCLVARDATA	Variable data

==========
JCL stream records
==========

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	JCLTAB(*)	
0	(0)	CHARACTER	80	JCLREC	
0	(0)	CHARACTER	72	JCLTEXT	COL 1-72 OF THE JCL RECORD
72	(48)	CHARACTER	8	JCLLNNO	COL 73-80 OF THE JCL RECORD

==========
JCL Variables
==========

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	80	JCLVTAB(*)	
0	(0)	CHARACTER	80	JCLV	
0	(0)	CHARACTER	8	CLVNAME	VARIABLE NAME
8	(8)	CHARACTER	16	JCLVTNAM	VARIABLE TABLE NAME
24	(18)	CHARACTER	1	JCLVTYPE	P = PROMPT, Y= SETUP,N=SUB

Offsets					
25	(19)	CHARACTER	1	JCLVSET	E(XIT), D(EFAULT), P(REP), VAR
26	(1A)	CHARACTER	8	JCLVUSER	USER ID
26	(1A)	CHARACTER	8	JCLVEXIT	EXIT NAME
26	(1A)	CHARACTER	8	JCLVSNAM	SETTING VARIABLE NAME
34	(22)	CHARACTER	44	JCLVVAL	VARIABLE VALUE SET
78	(4E)	SIGNED	2	JCLVLGT	LENGTH OF VALUE

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	*	REFTWS	
0	(0)	CHARACTER	188	REFBDY	

COMMON LAYOUT WITH TWSJCL:

0	(0)	CHARACTER	28	REFKEY	
0	(0)	CHARACTER	16	REFADID	ADID
16	(10)	CHARACTER	10	REFIA	IA
16	(10)	CHARACTER	6	REFIAT1	
22	(16)	CHARACTER	4	REFIAT2	
26	(1A)	SIGNED	2	REFOPNUM	ALWAYS 1
28	(1C)	CHARACTER	4	REFEYE	'REF '
32	(20)	UNSIGNED	1	REFVERS	'01'
33	(21)	CHARACTER	1	*	
34	(22)	CHARACTER	8	REFJOBNM	
42	(2A)	CHARACTER	12	REFVARIA	VAR IA from HTI
54	(36)	CHARACTER	2	*	free

- 3 REFWSN CHAR(4), é WORK STATION NAME
- 3 REFUPDAT, é LAST UPDATE, DATE+TIME
- 4 REFUPDT1 CHAR(6), DATE
- 4 REFUPDT2 CHAR(4), TIME

56	(38)	CHARACTER	8	REFUSER	LAST UPDATE, USERID
64	(40)	CHARACTER	1	REFUPTYP	UPDATE TYPE: NOT USED
65	(41)	CHARACTER	1	REFSTAT	OP. STATUS: NOT USED
66	(42)	UNSIGNED	2	REFLINES	NO OF LINES IN RECORD
68	(44)	CHARACTER	1	REFFLAGS	FLAGBYTE

Offsets					
69	(45)	CHARACTER	1	*	FREE
70	(46)	SIGNED	2	REFVLINE	NUMBER OF VARIABLES
72	(48)	SIGNED	4	REFSUBP	SUBPOOL FOR FREEMAIN
76	(4C)	UNSIGNED	2	REFALIASLEN	ALIAS NAME LENGTH
78	(4E)	CHARACTER	2	REFRES4	FREE
80	(50)	CHARACTER	56	REFBROKERKEYID	BROKER KEY IDENTIFIER

ADDITIONAL DATA ONLY FOR BYREF:

136	(88)	CHARACTER	44	REFDSNAME	DATA SET NAME
180	(B4)	CHARACTER	8	REFMEMBER	MEMBER NAME
188	(BC)	CHARACTER	*	REFVARDATA	

JCL VARIABLES

Offsets				
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**
0	(0)	STRUCTURE	80	REFVTAB(*)
0	(0)	CHARACTER	80	REFV
0	(0)	CHARACTER	8	REFVNAME
8	(8)	CHARACTER	16	REFVTNAM
24	(18)	CHARACTER	1	REFVTYPE
25	(19)	CHARACTER	1	REFVSET
26	(1A)	CHARACTER	8	REFVUSER
26	(1A)	CHARACTER	8	REFVEXIT
26	(1A)	CHARACTER	8	REFVSNAM
34	(22)	CHARACTER	44	REFVVAL
78	(4E)	SIGNED	2	REFVLGT

Cross reference

Name	Hex Offset	Hex Value	Level
JCLADID	0		4
JCLALIASLEN	4C		3
JCLBDY	0		2
JCLBROKERKEYID	50		3
JCLEDITD	44	40	4
JCLEYE	1C		3

Name	Hex Offset	Hex Value	Level
JCLFLAGS	44		3
JCLIA	10		4
JCLIAT1	10		5
JCLIAT2	16		5
JCLJDIRNOP	44	20	4
JCLJOBNM	22		3
JCLJSFND	44	80	4
JCLKEY	0		3
JCLLINES	42		3
JCLLNNO	48		3
JCLOPNUM	1A		4
JCLREC	0		2
JCLRES4	4E		3
JCLSTAT	41		3
JCLSUBP	48		3
JCLTAB	0		1
JCLTEXT	0		3
JCLTWS	0		1
JCLUPTYP	40		3
JCLUSER	38		3
JCLV	0		2
JCLVARDATA	88		2
JCLVARIA	2A		3
JCLVERS	20		3
JCLVEXIT	1A		4
JCLVLGT	4E		3
JCLVLINE	46		3
JCLVNAME	0		3
JCLVSET	19		3
JCLVSNAM	1A		5
JCLVTAB	0		1
JCLVTNAM	8		3
JCLVTYPE	18		3
JCLVUSER	1A		3
JCLVVAL	22		3
REFADID	0		4
REFALIASLEN	4C		3
REFBDY	0		2
REFBROKERKEYID	50		3
REFDSNAME	88		3
REFEYE	1C		3

Name	Hex Offset	Hex Value	Level
REFFLAGS	44		3
REFIA	10		4
REFIAT1	10		5
REFIAT2	16		5
REFJOBNM	22		3
REFKEY	0		3
REFLINES	42		3
REFMEMBER	B4		3
REFOPNUM	1A		4
REFRES4	4E		3
REFSTAT	41		3
REFSUBP	48		3
REFTWS	0		1
REFUPTYP	40		3
REFUSER	38		3
REFV	0		2
REFVARDATA	BC		2
REFVARIA	2A		3
REFVERS	20		3
REFVEXIT	1A		4
REFVLGT	4E		3
REFVLINE	46		3
REFVNAME	0		3
REFVSET	19		3
REFVSNAM	1A		5
REFVTAB	0		1
REFVTNAM	8		3
REFVTYPE	18		3
REFVUSER	1A		3
REFVVAL	22		3

JCL1 - JCL used for VARSUB

Name : DCLJCL1

Function:
JCLREC jcl layout

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	JCLREC	
0	(0)	CHARACTER	80	JCLBDY	FIXED PART OF JCL RECORD

Offsets					
0	(0)	CHARACTER	28	JCLKEY	
0	(0)	CHARACTER	16	JCLADID	APPLICATION NAME
16	(10)	CHARACTER	10	JCLIA	OCC INPUT ARRIVAL DATE+TIME
16	(10)	CHARACTER	6	JCLIAT1	DATE
22	(16)	CHARACTER	4	JCLIAT2	TIME
26	(1A)	SIGNED	2	JCLOPNUM	OPERATION NUMBER
28	(1C)	CHARACTER	4	JCLEYE	EYE CATCHER
32	(20)	UNSIGNED	1	JCLVERS	VERSION NUMBER
33	(21)	CHARACTER	1	*	FREE
34	(22)	CHARACTER	8	JCLJOBNM	JOBNAME
42	(2A)	CHARACTER	4	JCLWSN	WORK STATION NAME
46	(2E)	CHARACTER	10	JCLUPDAT	LAST UPDATE, DATE+TIME
46	(2E)	CHARACTER	6	JCLUPDT1	DATE
52	(34)	CHARACTER	4	JCLUPDT2	TIME
56	(38)	CHARACTER	8	JCLUSER	LAST UPDATE, USERID
64	(40)	CHARACTER	1	JCLUPTYP	UPDATING FUNCTION: L: LTP W: WSD R: READY LIST M: MCP
65	(41)	CHARACTER	1	JCLSTAT	OP. STATUS: S:SUB;:ED; V:SAVED; T:TSAVED;
66	(42)	UNSIGNED	2	JCLLINES	NO OF LINES IN RECORD
68	(44)	CHARACTER	1	JCLFLAGS	FLAGBYTE
		1...		JCLJSFND	ON = JCL READ FROM JS
		.1..		JCLEDITD	ONCE BEEN EDITED
		..1.		JCLJDIRNOP	ON = NOP directive set
		...1 1111		*	NOT USED
69	(45)	CHARACTER	1	*	FREE JCLC
70	(46)	SIGNED	2	JCLVLINE	NUMBER OF VARIABLES JCLA
72	(48)	SIGNED	4	JCLSUBP	SUBPOOL FOR FREEMAIN

Offsets					
76	(4C)	CHARACTER	4	JCLRES4	FREE
80	(50)	CHARACTER	80	JCLTAB (*)	JCL RECORDS
80	(50)	CHARACTER	72	JCLTEXT	COL 1-72 OF THE JCL RECORD
152	(98)	CHARACTER	8	JCLLNNO	COL 73-80 OF THE JCL RECORD

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	80	JCLVTAB (*)	
0	(0)	CHARACTER	80	JCLV	
0	(0)	CHARACTER	8	JCLVNAME	VARIABLE NAME
8	(8)	CHARACTER	16	JCLVTNAM	VARIABLE TABLE NAME
24	(18)	CHARACTER	1	JCLVTYPE	P = PROMPT, Y= SETUP,N=SUB
25	(19)	CHARACTER	1	JCLVSET	E(XIT), D(EFAULT), P(REP), VAR
26	(1A)	CHARACTER	8	JCLVUSER	USER ID
26	(1A)	CHARACTER	8	JCLVEXIT	EXIT NAME
26	(1A)	CHARACTER	8	JCLVSNAM	SETTING VARIABLE NAME
34	(22)	CHARACTER	44	JCLVVAL	VARIABLE VALUE SET
78	(4E)	SIGNED	2	JCLVLGT	LENGTH OF VALUE

Cross reference

Name	Hex Offset	Hex Value	Level
JCLADID	0		4
JCLBDY	0		2
JCLEDITD	44	40	4
JCLEYE	1C		3
JCLFLAGS	44		3
JCLIA	10		4
JCLIAT1	10		5
JCLIAT2	16		5
JCLJDIRNOP	44	20	4
JCLJOBNM	22		3
JCLJSFND	44	80	4

Name	Hex Offset	Hex Value	Level
JCLKEY	0		3
JCLLINES	42		3
JCLLNNO	98		3
JCLOPNUM	1A		4
JCLREC	0		1
JCLRES4	4C		3
JCLSTAT	41		3
JCLSUBP	48		3
JCLTAB	50		2
JCLTEXT	50		3
JCLUPDAT	2E		3
JCLUPDT1	2E		4
JCLUPDT2	34		4
JCLUPTYP	40		3
JCLUSER	38		3
JCLV	0		2
JCLVERS	20		3
JCLVEXIT	1A		4
JCLVLGT	4E		3
JCLVLINE	46		3
JCLVNAME	0		3
JCLVSET	19		3
JCLVSNAM	1A		5
JCLVTAB	0		1
JCLVTNAM	8		3
JCLVTYPE	18		3
JCLVUSER	1A		3
JCLVVAL	22		3
JCLWSN	2A		3

JDA - Predefined OPC/ESA variables

Name : DCLJDA

Function:
This block describes the mapping and addressing of variable values defined by the dcljdav block.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	12	jdat	variable data
0	(0)	ADDRESS	4	jdatdavp	pointer to jdav table

Offsets					
4	(4)	SIGNED	4	jdatnum	number of variables
8	(8)	SIGNED	4	jdatsize	total size of varible entrie
12	(C)	CHARACTER		jdattab	end of table header

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	21	jdae	variable table entry
0	(0)	CHARACTER	8	jdaevar	variable name
8	(8)	SIGNED	4	jdaevsz	size of variable name
12	(C)	SIGNED	4	jdaevao	offset to value based jdatda
16	(10)	SIGNED	4	jdaesiz	size of variable value
20	(14)	BITSTRING	1	jdaeflgs	additional flags
		1...		jdaesetp	setup avail on Yes
		.1..		jdaejett	Job using vars is ETT
		..11 1111		*	reserved

This block contains the values of all predefined jcl variables. These variables do not have to be defined in user-defined jcl variable tables.

Note:

If variables are added or removed from the jdav block then the jdavnum value must be updated accordingly.

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	479	jdav	
0	(0)	CHARACTER	4	jdavdesc	descriptor always 'JDAV'
4	(4)	CHARACTER	2	jdavvers	block version
6	(6)	CHARACTER	6	oymd1	occ ia yymmdd
12	(C)	CHARACTER	8	oymd2	occ ia yy/mm/dd
20	(14)	CHARACTER	10	oymd3	occ ia yyyy/mm/dd
30	(1E)	CHARACTER	6	odmy1	occ ia ddmmyy
36	(24)	CHARACTER	8	odmy2	occ ia dd/mm/yy
44	(2C)	CHARACTER	6	oym	occ ia yyyymm
50	(32)	CHARACTER	8	oymd	occ ia yyyymmdd

Offsets					
58	(3A)	CHARACTER	5	oyyddd	occ ia yyddd
63	(3F)	CHARACTER	3	owwd	occ week day within week
66	(42)	CHARACTER	4	oyymm	occ ia date yymm
70	(46)	CHARACTER	4	ommyy	occ ia date mmyy
74	(4A)	CHARACTER	2	oyy	occ ia year
76	(4C)	CHARACTER	4	oyyyy	occ ia year
80	(50)	CHARACTER	2	omm	occ ia month
82	(52)	CHARACTER	2	odd	occ ia day within month
84	(54)	CHARACTER	2	ohh	occ ia hour
86	(56)	CHARACTER	4	ohhmm	occ ia hour minute
90	(5A)	CHARACTER	4	ossid	Subsystem name
94	(5E)	CHARACTER	54	oxjobnam	Extended Job Name
148	(94)	CHARACTER	3	oopno	Oper number
151	(97)	CHARACTER	1	oday	Occ ia day in week 1=monday
152	(98)	CHARACTER	16	oadid	Occ application id
168	(A8)	CHARACTER	4	olhhmm	latest start hour minute
172	(AC)	CHARACTER	2	olhh	latest start hour
174	(AE)	CHARACTER	2	olmm	latest start month
176	(B0)	CHARACTER	6	olymd	latest start yymmdd
182	(B6)	CHARACTER	4	olmd	latest start mmdd
186	(BA)	CHARACTER	2	oldd	latest start dd (day in mon)
188	(BC)	CHARACTER	1	olday	latest start day of wk(1-7)
189	(BD)	CHARACTER	2	olwk	latest start week of year
191	(BF)	CHARACTER	5	olyyddd	latest start yyddd
196	(C4)	CHARACTER	6	cyymmdd	current date yymmdd
202	(CA)	CHARACTER	6	cddmmyy	current date ddmmyy
208	(D0)	CHARACTER	6	cyyyymm	current date year month
214	(D6)	CHARACTER	5	cyyddd	current date year day number
219	(DB)	CHARACTER	4	cyymm	current date year month
223	(DF)	CHARACTER	4	cmmyy	current date month year

Offsets					
227	(E3)	CHARACTER	2	cyy	current date year
229	(E5)	CHARACTER	4	cyyyy	current date year
233	(E9)	CHARACTER	2	cmm	current month
235	(EB)	CHARACTER	2	cdd	current day within month
237	(ED)	CHARACTER	3	cwwd	current week, day in week
240	(F0)	CHARACTER	8	cymd	current date, yyyymmdd
248	(F8)	CHARACTER	1	cday	current day in week,1=monday
249	(F9)	CHARACTER	2	chh	current hour of day
251	(FB)	CHARACTER	4	chhmm	current hour and minute
255	(FF)	CHARACTER	3	cddd	current day within year
258	(102)	CHARACTER	2	cww	current week in year
260	(104)	CHARACTER	8	chhmmssx	current time HHMMSSxx
268	(10C)	CHARACTER	1	cfreeday	cur time freeday Y!N AGLA
269	(10D)	CHARACTER	3	oddd	occ ia day within year
272	(110)	CHARACTER	2	oww	occ ia week in year
274	(112)	CHARACTER	16	oadowner	ad owner
290	(122)	CHARACTER	1	ofreeday	occ ia is freeday Y!N
291	(123)	CHARACTER	8	ojobname	jobname
299	(12B)	CHARACTER	8	oaugroup	authority group
307	(133)	CHARACTER	16	ocalid	calendar name
323	(143)	CHARACTER	1	owwmonth	week number in ia month
324	(144)	CHARACTER	1	owwlast	last week in month = Y else N
325	(145)	CHARACTER	6	chhmmss	current time HHMMSS $CBPA
331	(14B)	CHARACTER	44	oettcrit	ETT criteria
375	(177)	CHARACTER	1	oetttyp	ETT type J or R
376	(178)	CHARACTER	8	oettjob	ETT job name
384	(180)	CHARACTER	8	oettjid	ETT job id
392	(188)	CHARACTER	35	oettgroot	ETT gdg root
427	(1AB)	CHARACTER	44	oettevnam	ETT full event name

Offsets					
471	(1D7)	CHARACTER	8	oettggen	ETT gdg generation
479	(1DF)	CHARACTER		jdavend	end label of block

Cross reference

Name	Hex Offset	Hex Value	Level
cday	F8		2
cdd	EB		2
cddd	FF		2
cddmmyy	CA		2
cfreeday	10C		2
chh	F9		2
chhmm	FB		2
chhmmss	145		2
chhmmssx	104		2
cmm	E9		2
cmmyy	DF		2
cww	102		2
cwwd	ED		2
cymd	F0		2
cyy	E3		2
cyyddd	D6		2
cyymm	DB		2
cyymmdd	C4		2
cyyyy	E5		2
cyyyymm	D0		2
jdae	0		1
jdaeflgs	14		2
jdaejett	14	40	3
jdaesetp	14	80	3
jdaesiz	10		2
jdaevao	C		2
jdaevar	0		2
jdaevsz	8		2
jdat	0		1
jdatdavp	0		2
jdatnum	4		2
jdatsize	8		2
jdattab	C		2
jdav	0		1

Name	Hex Offset	Hex Value	Level
jdavdesc	0		2
jdavend	1DF		2
jdavvers	4		2
oadid	98		2
oadowner	112		2
oaugroup	12B		2
ocalid	133		2
oday	97		2
odd	52		2
oddd	10D		2
odmy1	1E		2
odmy2	24		2
oettcrit	14B		2
oettevnam	1AB		2
oettggen	1D7		2
oettgroot	188		2
oettjid	180		2
oettjob	178		2
oetttyp	177		2
ofreeday	122		2
ohh	54		2
ohhmm	56		2
ojobname	123		2
olday	BC		2
oldd	BA		2
olhh	AC		2
olhhmm	A8		2
olmd	B6		2
olmm	AE		2
olwk	BD		2
olymd	B0		2
olyyddd	BF		2
omm	50		2
ommyy	46		2
oopno	94		2
ossid	5A		2
oww	110		2
owwd	3F		2
owwlast	144		2
owwmonth	143		2
oxjobnam	5E		2

Name	Hex Offset	Hex Value	Level
oym	2C		2
oymd	32		2
oymd1	6		2
oymd2	C		2
oymd3	14		2
oyy	4A		2
oyyddd	3A		2
oyymm	42		2
oyyyy	4C		2

JDT - SETVAR defined temporary variables

Name : DCLJDT

Function:
This block describes the temp variables defined by the SETVAR directive.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	jdt	
0	(0)	CHARACTER	20	JdtHead	fixed part of JDT
0	(0)	CHARACTER	4	Jdteye	descriptor always 'JDT '
4	(4)	CHARACTER	2	Jdtvers	block version
6	(6)	CHARACTER	1	JdtWarnMsg	Y= issue "not ref" messages N= do not issue message
7	(7)	CHARACTER	1	*	not used
8	(8)	SIGNED	4	Jdtvars	of variables in table
12	(C)	SIGNED	4	Jdtmax	max of variables allowed
16	(10)	ADDRESS	4	Jdtnxtp	address of next JDT ! 0
20	(14)	CHARACTER	*	JdtVarTab	Variable part of JDU
20	(14)	CHARACTER	64	JdtVariables (*)	Address of variables
20	(14)	CHARACTER	4	JdtFlags	flags
20	(14)	CHARACTER	1	JdtUsed	Y = variavle was referenced
21	(15)	CHARACTER	1	JdtPhase	U = SETUP, S = SUBMIT
22	(16)	CHARACTER	2	*	Y = variavle was referenced

Offsets					
24	(18)	SIGNED	4	JdtVarL	length of total string that is to be substistuted
28	(1C)	CHARACTER	48	JdtValue	edit value to be used at substitution of this var.
76	(4C)	CHARACTER	8	JdtVname	name of variable

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	64	JdtSvar	mask for JDT substitution
0	(0)	CHARACTER	4	*	NOTE: This mask must much one single entry in JdtVariables
0	(0)	CHARACTER	1	JdtSused	var used in phase
1	(1)	CHARACTER	1	JdtSPhase	current phase
2	(2)	CHARACTER	2	*	
4	(4)	SIGNED	4	JdtSvarL	value length
8	(8)	CHARACTER	48	JdtSValue	substitution value
56	(38)	CHARACTER	8	JdtSVname	name of variable

Cross reference

Name	Hex Offset	Hex Value	Level
jdt	0		1
Jdtmax	C		3
Jdtvars	8		3
Jdteye	0		3
JdtFlags	14		4
JdtHead	0		2
Jdtnxtp	10		3
JdtPhase	15		5
JdtSPhase	1		3
JdtSused	0		3
JdtSValue	8		2
JdtSvar	0		1
JdtSvarL	4		2
JdtSVname	38		2
JdtUsed	14		5
JdtValue	1C		4
JdtVariables	14		3

Name	Hex Offset	Hex Value	Level
JdtVarL	18		4
JdtVarTab	14		2
Jdtvers	4		3
JdtVname	4C		4
JdtWarnMsg	6		3

JDU - Redefined OPC/ESA date and time variables

Name : DCLJDU

Function:
This block describes the contents and the format of OPC predefined variables as redefined by the SETFORM OPC statement in the JCL. NOTE: The JDU and JDUV controlblocks are in contigious storage JudCurr, JduOcc, JduOccl variables are initialized when the Jdu cbs are created. The values are picked up from predefined Occ and Current date /time variables

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	1192	jdu	
0	(0)	CHARACTER	168	JduHead	fixed part of JDU
0	(0)	CHARACTER	4	Jdueye	descriptor always 'JDU '
4	(4)	CHARACTER	2	Jduvers	block version
6	(6)	CHARACTER	2	*	not used
8	(8)	SIGNED	4	JduTotsize	total size for JDU* blocks
12	(C)	CHARACTER	8	JduCurrYmd	current date YYYYMMDD
20	(14)	CHARACTER	5	JduCurrJulian	current julian date YYDDD
25	(19)	CHARACTER	6	JduCurrTime	current time HHMMSS
31	(1F)	CHARACTER	5	JduOlJulian	oper latest start julian
36	(24)	CHARACTER	8	JduOlYmd	oper latest start YYMMDD
44	(2C)	CHARACTER	4	JduOlTime	oper latest start HHMM
48	(30)	CHARACTER	8	JduOiDate	oper ia date YYYYMMDD
56	(38)	CHARACTER	5	JduOiJulian	oper ia julian date YYDDD
61	(3D)	CHARACTER	4	JduOiTime	oper ia time HHMM

Offsets					
65	(41)	CHARACTER	8	JduOccYmd	occ ia date YYYYMMDD
73	(49)	CHARACTER	5	JduOccJulian	occ ia julian date YYDDD
78	(4E)	CHARACTER	4	JduOccTime	occ ia time HHMM
82	(52)	CHARACTER	8	JduOccLw	occ last work day in month
90	(5A)	CHARACTER	5	JduOccLwJulian	occ last work day JULIAN
95	(5F)	CHARACTER	8	JduOccLc	occ last cal day in month
103	(67)	CHARACTER	5	JduOccLcJulian	occ last cal day JULIAN
108	(6C)	CHARACTER	8	JduOccFc	occ first cal day in month
116	(74)	CHARACTER	5	JduOccFcJulian	occ first cal day JULIAN
121	(79)	CHARACTER	8	JduOccFw	occ first work day in month
129	(81)	CHARACTER	5	JduOccFwJulian	occ first work day JULIAN
134	(86)	CHARACTER	8	JduOccFwYear	occ first work day in year
142	(8E)	CHARACTER	5	JduOccFwJulianYr	occ first work day JULIAN
147	(93)	CHARACTER	8	JduOccLwYear	occ last work day in year
155	(9B)	CHARACTER	5	JduOccLwJulianYr	occ last work day JULIAN
160	(A0)	CHARACTER	8	*	spare
168	(A8)	CHARACTER	1024	JduVarTab	Variable part of JDU
168	(A8)	CHARACTER	64	JduVariables (16)	Address of variables
168	(A8)	SIGNED	4	JduOffset	offset to variable from
172	(AC)	SIGNED	4	JduVarL	length of total string that is to be substistuted
176	(B0)	CHARACTER	47	JduValue	edit value to be used at substitution of this var.
223	(DF)	CHARACTER	1	JduVflags	processing flags
		1...		JduvSetup	1= variable aval. at setup
		.111 1111		*	not used

Offsets					
224	(E0)	CHARACTER	8	JduVname	name of variable

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	942	jduv	description of a varible
0	(0)	CHARACTER	46	JduvHead	fixed part of description
0	(0)	SIGNED	2	Jduv	number of values in table
2	(2)	CHARACTER	2	JduvFormat	'DA' = Date related 'TI' = Time format
4	(4)	CHARACTER	40	JduvSdelim	string before date/time
44	(2C)	SIGNED	2	JduvSdelimL	length of delim before strg
46	(2E)	CHARACTER	56	JduvDesc (16)	Description
46	(2E)	CHARACTER	4	JduvType	Keyword type (MM,DD,CC,YY, CCYY,HH,SS)
50	(32)	SIGNED	2	JduvStart	Start position in string
52	(34)	SIGNED	2	JduvEnd	End position in string
54	(36)	CHARACTER	4	JduvVal	Value for this type 1993,10bb and so on
58	(3A)	SIGNED	2	JduvValL	Length of value
60	(3C)	CHARACTER	40	JduvDelim	Delimiter after this one
100	(64)	SIGNED	2	JduvDelimL	length of delimiter

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	64	JduSvariable	Used to conform with other methods for addressing vars to be subst. in JHSLN/JHTRN NOTE: This mapping must map a single entry in JduVartab
0	(0)	SIGNED	4	JduSOffset	offset to variable from

Offsets					
4	(4)	SIGNED	4	JduSVarL	length of total string that is to be substistuted
8	(8)	CHARACTER	47	JduSValue	edit value to be used at substitution of this var.
55	(37)	CHARACTER	1	JduSFlags	processing flags
		1...		JdusSetup	1= variable avail. art setup
		.111 1111		*	not used
56	(38)	CHARACTER	8	JduSVname	name of variable

Cross reference

Name	Hex Offset	Hex Value	Level
jdu	0		1
JduCurrJulian	14		3
JduCurrTime	19		3
JduCurrYmd	C		3
Jdueye	0		3
JduHead	0		2
JduOccFc	6C		3
JduOccFcJulian	74		3
JduOccFw	79		3
JduOccFwJulian	81		3
JduOccFwJulianYr	8E		3
JduOccFwYear	86		3
JduOccJulian	49		3
JduOccLc	5F		3
JduOccLcJulian	67		3
JduOccLw	52		3
JduOccLwJulian	5A		3
JduOccLwJulianYr	9B		3
JduOccLwYear	93		3
JduOccTime	4E		3
JduOccYmd	41		3
JduOffset	A8		4
JduOiDate	30		3
JduOiJulian	38		3
JduOiTime	3D		3
JduOlJulian	1F		3
JduOlTime	2C		3

Name	Hex Offset	Hex Value	Level
JduOlYmd	24		3
JduSFlags	37		2
JduSOffset	0		2
JdusSetup	37	80	3
JduSValue	8		2
JduSvariable	0		1
JduSVarL	4		2
JduSVname	38		2
JduTotsize	8		3
jduv	0		1
Jduv	0		3
JduValue	B0		4
JduVariables	A8		3
JduVarL	AC		4
JduVarTab	A8		2
JduvDelim	3C		3
JduvDelimL	64		3
JduvDesc	2E		2
JduvEnd	34		3
Jduvers	4		3
JduVflags	DF		4
JduvFormat	2		3
JduvHead	0		2
JduVname	E0		4
JduvSdelim	4		3
JduvSdelimL	2C		3
JduvSetup	DF	80	5
JduvStart	32		3
JduvType	2E		3
JduvVal	36		3
JduvValL	3A		3

JHS - Shared parameters for JHSET and JHUTL

Name : DCLJHS

Function:
This block contains parameters used by both JHSET and JHUTL.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	212	JHSETcom	

Offsets					
0	(0)	CHARACTER	60	wformat	format string in local
60	(3C)	CHARACTER	16	Wcalendar	occurrence calendar name
	2 HCPAREA char(length(HCPrm)), parameter area				
76	(4C)	CHARACTER	6	occia_base	ia date (from OYMD1)
82	(52)	CHARACTER	6	olate_base	latest start (from OLYMD)
88	(58)	CHARACTER	6	cdate_base	current d. base (from CYYMMDD)
94	(5E)	CHARACTER	6	occia_baset	ia time (from OHHMM)
100	(64)	CHARACTER	6	olate_baset	latest start time(from OLHHMM)
106	(6A)	CHARACTER	8	ctime_baset	current time (from HHMMSSXX)
114	(72)	CHARACTER	8	varname	variable name
122	(7A)	CHARACTER	3	number	number to add/subtract (date)
125	(7D)	CHARACTER	3	*	free
128	(80)	SIGNED	4	seconds	sec*100 to add/subtract (time)
132	(84)	CHARACTER	2	Ctype	type to add/subtract
134	(86)	CHARACTER	6	fase	= parm phase
140	(8C)	SIGNED	4	signpos	position of sign
144	(90)	SIGNED	4	var	1st pos for category
148	(94)	SIGNED	4	var_end	last pos for category
152	(98)	SIGNED	4	digit	1st pos of digits
156	(9C)	SIGNED	4	digit_end	last pos of digits
160	(A0)	SIGNED	4	type_start	1st pos for type (wd,cd....)
164	(A4)	SIGNED	4	var_cat	variable category
168	(A8)	SIGNED	4	i	
172	(AC)	SIGNED	4	j	
176	(B0)	SIGNED	4	k	
180	(B4)	SIGNED	4	x	
184	(B8)	SIGNED	4	j1	loop indexes
188	(BC)	SIGNED	4	rc	local return code

Offsets

192	(C0)	SIGNED	4	currvar	work index
196	(C4)	SIGNED	4	currJdt	current JDt entry
200	(C8)	ADDRESS	4	wtubptr	= tubptr
204	(CC)	ADDRESS	4	wjduvptr	= jduvptr $CQOC
208	(D0)	ADDRESS	4	wjdtptr	= jdtptr $CQOA

Cross reference

Name	Hex Offset	Hex Value	Level
cdate_base	58		2
ctime_baset	6A		2
Ctype	84		2
currJdt	C4		2
currvar	C0		2
digit	98		2
digit_end	9C		2
fase	86		2
i	A8		2
j	AC		2
JHSETcom	0		1
j1	B8		2
k	B0		2
number	7A		2
occia_base	4C		2
occia_baset	5E		2
olate_base	52		2
olate_baset	64		2
rc	BC		2
seconds	80		2
signpos	8C		2
type_start	A0		2
var	90		2
var_cat	A4		2
var_end	94		2
varname	72		2
Wcalendar	3C		2
wformat	0		2
wjdtptr	D0		2
wjduvptr	CC		2
wtubptr	C8		2
x	B4		2

JV - JCL Variable table layout

Name : DCLJV

Function:
JV layout.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	*	jv	jcl variable table
0	(0)	CHARACTER	98	jvcommon	identifier
0	(0)	CHARACTER	2	*	reserved for vsam mods/02
2	(2)	CHARACTER	23	jvkey	key of record table
2	(2)	CHARACTER	16	jvtable	jcl variable table id
18	(12)	CHARACTER	7	*	always blank
25	(19)	CHARACTER	1	*	not used
26	(1A)	CHARACTER	8	jvlu	last updating user
34	(22)	CHARACTER	4	jvlt	last update time hhmm
38	(26)	CHARACTER	6	jvld	last update date yymmdd
44	(2C)	CHARACTER	24	jvdes	table description
68	(44)	SIGNED	2	jvvar	number of vars in table
70	(46)	CHARACTER	16	jvown	owner id
86	(56)	CHARACTER	2	*	not used
88	(58)	CHARACTER	8	jvluts	last update timestamp
96	(60)	CHARACTER	2	*	not used
98	(62)	CHARACTER	*	jvarsect	variable part of table

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	476	jvtab	jcl var table, var part
0	(0)	CHARACTER	8	jvvar	jcl variable name
8	(8)	CHARACTER	44	jvdfl	jcl variable def value
52	(34)	CHARACTER	1	jvuc	Y = uppercase, N=mixed
53	(35)	CHARACTER	1	jvstp	prompt / setup / submit
54	(36)	SIGNED	2	jvlg	value length
56	(38)	CHARACTER	7	jvtyp	verification type

Offsets					
63	(3F)	CHARACTER	8	jvex	substitution exit name
71	(47)	CHARACTER	1	jvinp	input required
72	(48)	SIGNED	2	jvpos	replace position jcl data
74	(4A)	CHARACTER	1	jvnum	numeric
75	(4B)	CHARACTER	2	jvcmp	comparison operator
77	(4D)	CHARACTER	44	jvpat	validation pattern
121	(79)	CHARACTER	102	jvvld	valid values
121	(79)	CHARACTER	51	jvvld1	first line
172	(AC)	CHARACTER	51	jvvld2	second line
223	(DF)	CHARACTER	204	jvtxt	dialog text
427	(1AB)	CHARACTER	20	jvtdes	description
447	(1BF)	CHARACTER	1	*	reserved
448	(1C0)	SIGNED	2	jvnrp	number of dep values
450	(1C2)	CHARACTER	8	jvind	independent variable name
458	(1CA)	CHARACTER	2	jvvers	version number
460	(1CC)	CHARACTER	2	jvsubs	substring start posVJA
462	(1CE)	CHARACTER	2	jvsubl	substring length
464	(1D0)	CHARACTER	12	*	reserved

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	88	jvd	dependencies
0	(0)	CHARACTER	44	jvdiv	value of setting variable
44	(2C)	CHARACTER	44	jvddv	dependent variable value

Cross reference

Name	Hex Offset	Hex Value	Level
jv	0		1
jvvar	44		3
jvarsect	62		2
jvcmp	4B		2
jvcommon	0		2
jvd	0		1
jvddv	2C		2
jvdes	2C		3

Name	Hex Offset	Hex Value	Level
jvdfl	8		2
jvdiv	0		2
jvex	3F		2
jvind	1C2		2
jvinp	47		2
jvkey	2		3
jvld	26		3
jvlg	36		2
jvlt	22		3
jvlu	1A		3
jvluts	58		3
jvnrp	1C0		2
jvnum	4A		2
jvown	46		3
jvpat	4D		2
jvpos	48		2
jvstp	35		2
jvsubl	1CE		2
jvsubs	1CC		2
jvtab	0		1
jvtable	2		4
jvtdes	1AB		2
jvtxt	DF		2
jvtyp	38		2
jvuc	34		2
jvvar	0		2
jvvers	1CA		2
jvvld	79		2
jvvld1	79		3
jvvld2	AC		3

MCA - TWS/ESA common area

Name : DCLMCA

Function:
This segment declares the TWS/ESA common area.
Most TWS/ESA control blocks can be reached via the MCA.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	392	mca	TWS/ESA communication area

Offsets					
0	(0)	CHARACTER	4	mcadesc	block descriptor = 'MCA '
4	(4)	CHARACTER	10	mcalevel	MCA block code level
4	(4)	CHARACTER	2	mcaver	block mapping version
6	(6)	CHARACTER	8	mcafmid	TWS/ESA tracker fmid
14	(E)	BITSTRING	2	mcaflags	mca options
		1...		mcatso	1: tso user built mca
		.1..		mcaasxb	1: asxb was modified
		..1.		mcaacee	1: acee built
		...1		mcaracf	1: subres profiles
		.... 1...		mcaesa	: active on mvs/esa
		.... .1..		mcasp4	1: mvs/sp 4.1 or later
		.... ..1.		mcamlog	1: msg log is reserved
		1		mca313	1: mvs/sp 3.1.3 or later
15	(F)	1...		mcaqfcan	1: ss cancld due to qfull
		.1..		mcanoprm	1: do not open parmlib
		..1.		mcaedpw	1: waiting for edp
		...1		mcasp422	1: mvs/sp 4.2.2 or later
		.... 1...		mcapace	1: event inflow paced
		.... .1..		mcaupace	1: emgr ask for resume
		.... ..1.		mcartrq	1: queue to DRTQ
		1		mcatcpr	1: ta task restarting
TWS/ESA control block addresses follow					
16	(10)	ADDRESS	4	mcaascb	eqqmajor ascb address
20	(14)	ADDRESS	4	mcamtcb	eqqmajor tcb address
24	(18)	ADDRESS	4	mcaopts	addr of options block
28	(1C)	ADDRESS	4	mcaftab	file table address
32	(20)	ADDRESS	4	mcastemj	addr 1st major subtask

Offsets					
36	(24)	ADDRESS	4	mcastenm	addr 1st nmm subtask
40	(28)	ADDRESS	4	mcastegs	addr 1st genserv subtask
44	(2C)	ADDRESS	4	mcaopecb	addr of oper cmd ecb
48	(30)	ADDRESS	4	mcassvt	addr of tws ssvt
52	(34)	ADDRESS	4	mcassct	addr of tws ssct
56	(38)	ADDRESS	4	mcatsob	addr of tso table block
60	(3C)	ADDRESS	4	mcanmmb	addr of nmm parameters
64	(40)	ADDRESS	4	mcaedpb	addr of edp block
68	(44)	ADDRESS	4	mcaprip	addr of pri block
72	(48)	ADDRESS	4	mcasrap	addr of serv routn array
76	(4C)	ADDRESS	4	mcasure	addr of su/re table
80	(50)	ADDRESS	4	mcanabb	addr of vtam i/o params
84	(54)	ADDRESS	4	mcadiap	addr of diagnose options
88	(58)	ADDRESS	4	mcacpnq	addr of cp enq statistics
92	(5C)	ADDRESS	4	mcatmlog	tcb addr of mlog reserver
96	(60)	ADDRESS	4	mcajancp	js interface anchor block
100	(64)	ADDRESS	4	mcaxsip	XCF system info cb
104	(68)	ADDRESS	4	mcaqfecb	addr to q full ecb
108	(6C)	ADDRESS	4	mcaasip	APPC system info cb
112	(70)	ADDRESS	4	mcaSyncEcbPtr	HT Client to Server Sync
116	(74)	ADDRESS	4	mcaTSRAp	addr of Topology parsing
120	(78)	ADDRESS	4	mcaFSRAp	addr of Data Store service routines
124	(7C)	ADDRESS	4	mcasub	addr of sub parm area
128	(80)	ADDRESS	4	mcassxp	SSX block address
132	(84)	ADDRESS	4	mcarsip	RODM system info cb
136	(88)	ADDRESS	4	mcajltbl	mca job log table ptr
140	(8C)	ADDRESS	4	mcarodmopt	RODM options table
144	(90)	ADDRESS	4	mcahcm	HCMain address

Offsets					
148	(94)	ADDRESS	4	mcaID	Local Id event area
152	(98)	ADDRESS	4	mcaJopts	address of JOBOPTS parms
156	(9C)	ADDRESS	4	mcaanmmp	always nmmpointer
160	(A0)	ADDRESS	4	mcaqueptr	current queue elem ptr
TWS/ESA service routine addresses follow					
164	(A4)	ADDRESS	4	mcamsgx	message routine address
168	(A8)	ADDRESS	4	mcaseqx	seq i/o service routine
172	(AC)	ADDRESS	4	mcapdsx	pds i/o service routine
176	(B0)	ADDRESS	4	mcaprmx	param member parse rtn
180	(B4)	ADDRESS	4	mcaquex	queue server routine
184	(B8)	ADDRESS	4	mcanowx	current time routine
188	(BC)	ADDRESS	4	mcavsam	addr vsam file handler
192	(C0)	ADDRESS	4	mcavsamb	addr bex vsam file handler
196	(C4)	ADDRESS	4	mcasubx	addr job submit routine
200	(C8)	ADDRESS	4	mcarelx	addr job release routine
204	(CC)	ADDRESS	4	mcaevhx	addr of event handler
208	(D0)	ADDRESS	4	mcamcpx	addr of modify curr plan
212	(D4)	ADDRESS	4	mcabexp	bex services address
216	(D8)	ADDRESS	4	mcaaidx	ZNOWX format clone address
220	(DC)	ADDRESS	4	mcalvck	level check routine adress
224	(E0)	ADDRESS	4	mcaznqd	ZNQDX lock dsname
228	(E4)	ADDRESS	4	mcaettp	ETT info for CP04
232	(E8)	ADDRESS	4	mcaIDecb	EW ID event ECB
miscellaneous Tivoli Workload Sscheduler constants					
236	(EC)	SIGNED	2	mcansubs	number of subsys subtasks
238	(EE)	SIGNED	2	mcagmtof	gmt offset, minutes
240	(F0)	SIGNED	2	mcaracrtrc	racroute trace level

Offsets					
242	(F2)	BITSTRING	2	mcaDSTORE	Data Store task status
		1...		DBAReady	Data base init OK
		.1..		DBAFail	Data base ended
		..1.		JQUReady	Jes queue init OK
		...1		JQUFail	Jes queue ended
		.... 1...		mcaRefrCP	
242	(F2)	BITSTRING	1	*	
243	(F3)	.... .1..		Fprocin	
		.... ..1.		Fmethod	Data Store method trace
		1		FParser	Data Store parser trace
244	(F4)	SIGNED	4	mcagmtSEC	gmt offset, seconds
248	(F8)	CHARACTER	1	mcajes	primary jes, A=js2, B=js3
249	(F9)	CHARACTER	1	mcacjes	jes command 1st character
250	(FA)	CHARACTER	4	mcassnm	TWS/ESA subsystem name
254	(FE)	CHARACTER	8	mcamajnm	TWS/ESA major enq name
262	(106)	CHARACTER	8	mcaclass	racf resource class name
270	(10E)	CHARACTER	8	mcanjenm	nje node name
278	(116)	CHARACTER	4	mcaqfqn	name of full queue
282	(11A)	CHARACTER	8	mcanvid	Netview Receiver ID
290	(122)	CHARACTER	1	mcadsclas	JES class for Datastore
291	(123)	CHARACTER	1	mcaSPIN	Y = SPIN available; N = SPIN not available to server block
292	(124)	ADDRESS	4	mcaphbp	
296	(128)	CHARACTER	5	mcaclnjob	clean up job name
301	(12D)	CHARACTER	1	mcaddrspc	address space type: O = controller/tracker; S = server; D = data store; B = batch; T = Trial EQQDTTOP; L = Batch Loader
302	(12E)	CHARACTER	8	MCAJesfmid	JES fmid
310	(136)	CHARACTER	1	MCAtraces	
		1...		MCAzzSPIN	SPIN traces
		.111 1111		*	

Offsets					
311	(137)	CHARACTER	1	mcallopt	LISTLOG option A!F!N
312	(138)	BITSTRING	6	mcasubrs	protected subresources
		1...		mcaadnm	ada.adname is a resource
		.1..		mcaadow	ado.owner is a resource
		..1.		mcaadgr	adg.group is a resource
		...1		mcaadjb	adj.jobname is a resource
		.... 1...		mcacpad	cpa.adname is a resource
		.... .1..		mcacpow	cpo.owner is a resource
		.... ..1.		mcacpgr	cpg.group is a resource
		1		mcacpjb	cpj.jobname is a resource
313	(139)	1...		mcacpws	cpw.wsname is a resource
		.1..		mcajcad	jsa.adname is a resource
		..1.		mcajcjb	jsj.jobname is a resource
		...1		mcajcws	jsw.wsname is a resource
		.... 1...		mcajcow	jso.owner is a resource
		.... .1..		mcajcgr	jsg.group is a resource
		.... ..1.		mcaltad	lta.adname is a resource
		1		mcaltow	lto.owner is a resource
314	(13A)	1...		mcaoiad	oia.adname is a resource
		.1..		mcawsws	wsw.wsname is a resource
		..1.		mcarlad	rla.adname is a resource
		...1		mcarlow	rlo.owner is a resource
		.... 1...		mcarlgr	rlg.group is a resource
		.... .1..		mcarlws	rlw.wsname is a resource

Offsets					
		.... ..1.		mcaclcn	clc.calname is a resource
		1		mcaprpn	prp.pername is a resource
315	(13B)	1...		mcaetnm	ete.name is a resource
		.1..		mcaetad	eta.name is a resource
		..1.		mcasrnm	srs.name is a resource
		...1		mcavjvo	jv.owner is a resource
		.... 1...		mcavjvn	jv.tabname is a resource
		.... .1..		mcacpwo	cpz.wsname is a res
		.... ..1.		mcacpgd	cpd.OCCgrp is a res
		1		mcaltgd	ltd.OCCgrp is a res
316	(13C)	1...		mcaadgd	add.adgrp is a res
		.1..		mcarlwst	rl.wsstat is a res
		..1.		mcardrn	rdr.name is a res
		...1		mcaadex	ade.extname is a res
		.... 1...		mcacpex	cpe.extname is a res
		.... .1..		mcaadse	ad.secelem is a res
		.... ..1.		mcacpse	cp.secelem is a res
		1		mcadbrp	rp.reptype is a res
317	(13D)	1...		mcaADinuse	AD used by batch
		.1..		mcaADVERrun	AD VER done
		..1.		mcapif	
		...1 1111		*	
318	(13E)	BITSTRING	2	mcaflags2	flags
		1...		mcasp52	1: mvs/sp 5.2 or later
		.1..		mcasymb	1: perform symbol subst
		..1.		mcaux002	exit2 invoked
		...1		mcawaenq	deq after abend?
		.... 1...		mcadbg	for debug purpose
		.... .1..		mcajtblock	ON: JTB is locked
		.... ..1.		mcaTWSCntlStart	On at controller StartUp
		1		mcaBulkDiscovery	bulc disc is already running
319	(13F)	1...		mcalock2b	

Offsets					
		.1..		mcastopc	On = stop command issued
		..11 1111		*	free
320	(140)	ADDRESS	4	mcatplgyp	Topology CB address
324	(144)	ADDRESS	4	mcaSCLIBdcb	sclib dcb ptr
324	(144)	SIGNED	4	mcaHTDSlastrec	
328	(148)	ADDRESS	4	mcaCtoken	token for C environment
328	(148)	SIGNED	4	mcaHTDSlastcyc	
332	(14C)	SIGNED	4	mcauserf	reserved for tws exits
336	(150)	SIGNED	2	mcaquelen	QUEUELEN changed value
338	(152)	BITSTRING	2	mcaperf	Performance flags
		1...		mcaexiDB	EXIT debug
		.1..		mcajclDB	JCL debug
		..1.		mcaE105	E105 msg flag
		...1		mcaZ308	Z308 msg flag
		.... 1...		mcan069	N069 msg flag
338	(152)	BITSTRING	1	*	free
340	(154)	SIGNED	2	mcaTimeSta	stats msg interval time
342	(156)	SIGNED	2	mcablrc	BL rc with EQQY221E
344	(158)	ADDRESS	4	MCADBGp	
348	(15C)	ADDRESS	4	mcamlogd	mlog dsname address

---

2 mcafarb ptr(31) , farb ptr
2 lockrc bin(31) ,

352	(160)	UNSIGNED	2	mcaHT_evtsseq	
354	(162)	UNSIGNED	2	*	
356	(164)	SIGNED	4	mcaHT_evtchkcyc	
360	(168)	CHARACTER	4	mcaFINDmem	
360	(168)	BITSTRING	3	mcaTTR	
363	(16B)	BITSTRING	1	mcaconc	
364	(16C)	SIGNED	4	mcaENFTOK57	ENFREQ 57 dtoken
364	(16C)	SIGNED	4	mcaHTDSespN	
368	(170)	SIGNED	4	mcaWLMQsz	WLM query size
368	(170)	SIGNED	4	mcaHTDSespT	
372	(174)	SIGNED	4	mcaENFTOK41	ENFREQ 41 dtoken

Offsets					
376	(178)	ADDRESS	4	mcamsgh	bufmsg routine address
380	(17C)	ADDRESS	4	mcaEXTp	address of extended MCA
384	(180)	SIGNED	4	mcaENFTOK53	ENFREQ 53 dtoken
388	(184)	ADDRESS	4	mcaux014	address of eqqux014
392	(188)	CHARACTER		mcaend	end of mca

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	324	mcaEXT	extended MCA
0	(0)	ADDRESS	4	mcaANCQp	adress of mcaFLque
4	(4)	CHARACTER	8	mcaSYSNAME	&SYSNAME
12	(C)	ADDRESS	4	mcajtab	address of EQQZJTAB
16	(10)	ADDRESS	4	mcadsiox	address of EQQDSIOX
20	(14)	ADDRESS	4	mcadsini	address of EQQDSINI
24	(18)	ADDRESS	4	mcajtbp	ptr to JTB
28	(1C)	ADDRESS	4	mcaX14tabp	exit14 tabptr
32	(20)	SIGNED	4	mcaX14numr	exit14 numrow
36	(24)	SIGNED	4	mcaX14rsiz	exit14 recsize
40	(28)	ADDRESS	4	mcaDSViewp	ptr to DSV command area
44	(2C)	CHARACTER	8	mcaoptmem	options member
52	(34)	ADDRESS	4	mcadtbaux	pointer to refresh dest
56	(38)	SIGNED	4	mcaavildst	destination slots available for refresh
60	(3C)	ADDRESS	4	mcahtca	pointer to htca
64	(40)	SIGNED	4	mcahtcauxn	len of aux htc
68	(44)	ADDRESS	4	mcahtcaux	pointer to auxiliary htc
72	(48)	SIGNED	4	mcahtsauxn	len of aux hts
76	(4C)	ADDRESS	4	mcahtsaux	pointer to auxiliary hts
80	(50)	ADDRESS	4	mcahtsa	pointer to htsa
84	(54)	ADDRESS	4	mcahtcp	ptr to HTC block
88	(58)	SIGNED	4	mcahtcl	length of HTC block
92	(5C)	ADDRESS	4	mcahtsp	ptr to HTS block
96	(60)	SIGNED	4	mcahtsl	length of HTS block
100	(64)	CHARACTER	2	*	free
102	(66)	UNSIGNED	4	mcasseqconf	last sseq confirmed

Offsets					
104	(68)	SIGNED	4	mcarecconf	last HTDS record confirmed
108	(6C)	SIGNED	4	mcacycconf	last HTDS cycle confirmed
112	(70)	ADDRESS	4	*(52)	free
320	(140)	ADDRESS	4	mcaHTdbfP	free

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	12	mcaFLque	
0	(0)	ADDRESS	4	mcaANCQp1	floptmsgqu1
4	(4)	ADDRESS	4	mcaANCQp2	floptmsgqu2
8	(8)	ADDRESS	4	mcaANCQpT	floptmsgqut

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	28	mcaDSView	
0	(0)	CHARACTER	16	mcaDSV_AD	adid
16	(10)	CHARACTER	10	mcaDSV_IA	ia
26	(1A)	SIGNED	2	mcaDSV_OP	opnum

Offsets				
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**
0	(0)	STRUCTURE	*	mcaHTpage
0	(0)	CHARACTER	12	mcaHTfix
0	(0)	ADDRESS	4	mcaHTnext
4	(4)	ADDRESS	4	mcaHTtot
8	(8)	SIGNED	4	mcaHToff
12	(C)	CHARACTER	*	mcaHTend

Cross reference

Name	Hex Offset	Hex Value	Level
DBAFail	F2	40	3
DBAReady	F2	80	3
Fmethod	F3	02	3
FParser	F3	01	3
Fprocin	F3	04	3
JQUFail	F2	10	3
JQUReady	F2	20	3
mca	0		1
mcaacee	E	20	3

Name	Hex Offset	Hex Value	Level
mcaadex	13C	10	3
mcaadgd	13C	80	3
mcaadgr	138	20	3
mcaADinuse	13D	80	3
mcaadjb	138	10	3
mcaadnm	138	80	3
mcaadow	138	40	3
mcaadse	13C	04	3
mcaADVERrun	13D	40	3
mcaaidx	D8		2
mcaANCQp	0		2
mcaANCQpT	8		2
mcaANCQp1	0		2
mcaANCQp2	4		2
mcaanmmp	9C		2
mcaascb	10		2
mcaasip	6C		2
mcaasxb	E	40	3
mcaavildst	38		2
mcabexp	D4		2
mcablrc	156		2
mcaBulkDiscoveryRunning	13E	01	3
mcacjes	F9		2
mcaclass	106		2
mcaclcn	13A	02	3
mcaclnjob	128		2
mcaconc	16B		3
mcacpad	138	08	3
mcacpex	13C	08	3
mcacpgd	13B	02	3
mcacpgr	138	02	3
mcacpjb	138	01	3
mcacpnq	58		2
mcacpow	138	04	3
mcacpse	13C	02	3
mcacpwo	13B	04	3
mcacpws	139	80	3
mcaCtoken	148		2
mcacycconf	6C		2
mcadbg	13E	08	3
MCADBGp	158		2

Name	Hex Offset	Hex Value	Level
mcadbrp	13C	01	3
mcaddrspc	12D		2
mcadesc	0		2
mcadiap	54		2
mcadsclas	122		2
mcadsini	14		2
mcadsiox	10		2
mcaDSTORE	F2		2
mcaDSV_AD	0		2
mcaDSV_IA	10		2
mcaDSV_OP	1A		2
mcaDSView	0		1
mcaDSViewp	28		2
mcadtbaux	34		2
mcaedpb	40		2
mcaedpw	F	20	3
mcaend	188		2
mcaENFTOK41	174		2
mcaENFTOK53	180		2
mcaENFTOK57	16C		2
mcaesa	E	08	3
mcaetad	13B	40	3
mcaetnm	13B	80	3
mcaettp	E4		2
mcaevhx	CC		2
mcaexiDB	152	80	3
mcaEXT	0		1
mcaEXTp	17C		2
mcaE105	152	20	3
mcaFINDmem	168		2
mcaflags	E		2
mcaflags2	13E		2
mcaFLque	0		1
mcafmid	6		3
mcaFSRAp	78		2
mcaftab	1C		2
mcagmtof	EE		2
mcagmtSEC	F4		2
mcahcm	90		2
mcaHT_evtchkcyc	164		2
mcaHT_evtsseq	160		2

Name	Hex Offset	Hex Value	Level
mcahtca	3C		2
mcahtcaux	44		2
mcahtcauxn	40		2
mcahtcl	58		2
mcahtcp	54		2
mcaHTdbfP	140		2
mcaHTDSespN	16C		3
mcaHTDSespT	170		3
mcaHTDSlastcyc	148		3
mcaHTDSlastrec	144		3
mcaHTend	C		2
mcaHTfix	0		2
mcaHTnext	0		3
mcaHToff	8		3
mcaHTpage	0		1
mcahtsa	50		2
mcahtsaux	4C		2
mcahtsauxn	48		2
mcahtsl	60		2
mcahtsp	5C		2
mcaHTtot	4		3
mcaID	94		2
mcaIDecb	E8		2
mcajancp	60		2
mcajcad	139	40	3
mcajcgr	139	04	3
mcajcjb	139	20	3
mcajclDB	152	40	3
mcajcow	139	08	3
mcajcws	139	10	3
mcajes	F8		2
MCAJesfmid	12E		2
mcajltbl	88		2
mcaJopts	98		2
mcajtab	C		2
mcajtblock	13E	04	3
mcajtbp	18		2
mcalevel	4		2
mcallopt	137		2
mcalock2b	13F	80	3
mcaltad	139	02	3

Name	Hex Offset	Hex Value	Level
mcaltgd	13B	01	3
mcaltow	139	01	3
mcalvck	DC		2
mcamajnm	FE		2
mcamcpx	D0		2
mcamlog	E	02	3
mcamlogd	15C		2
mcamsgh	178		2
mcamsgx	A4		2
mcamtcb	14		2
mcanabb	50		2
mcanjenm	10E		2
mcanmmb	3C		2
mcanoprm	F	40	3
mcanowx	B8		2
mcansubs	EC		2
mcanvid	11A		2
mcan069	152	08	3
mcaoiad	13A	80	3
mcaopecb	2C		2
mcaoptmem	2C		2
mcaopts	18		2
mcapace	F	08	3
mcapdsx	AC		2
mcaperf	152		2
mcaphbp	124		2
mcapif	13D	20	3
mcaprip	44		2
mcaprmx	B0		2
mcaprpn	13A	01	3
mcaqfcan	F	80	3
mcaqfecb	68		2
mcaqfqn	116		2
mcaquelen	150		2
mcaqueptr	A0		2
mcaquex	B4		2
mcaracf	E	10	3
mcaracrtrc	F0		2
mcardrn	13C	20	3
mcarecconf	68		2
mcaRefrCP	F2	08	3

Name	Hex Offset	Hex Value	Level
mcarelx	C8		2
mcarlad	13A	20	3
mcarlgr	13A	08	3
mcarlow	13A	10	3
mcarlws	13A	04	3
mcarlwst	13C	40	3
mcarodmopt	8C		2
mcarsip	84		2
mcartrq	F	02	3
mcaSCLIBdcb	144		2
mcaseqx	A8		2
mcaSPIN	123		2
mcasp4	E	04	3
mcasp422	F	10	3
mcasp52	13E	80	3
mcasrap	48		2
mcasrnm	13B	20	3
mcassct	34		2
mcasseqconf	66		2
mcassnm	FA		2
mcassvt	30		2
mcassxp	80		2
mcastegs	28		2
mcastemj	20		2
mcastenm	24		2
mcastopc	13F	40	3
mcasub	7C		2
mcasubrs	138		2
mcasubx	C4		2
mcasure	4C		2
mcasymb	13E	40	3
mcaSyncEcbPtr	70		2
mcaSYSNAME	4		2
mcatcpr	F	01	3
mcaTimeSta	154		2
mcatmlog	5C		2
mcatplgyp	140		2
MCAtraces	136		2
mcatso	E	80	3
mcatsob	38		2
mcaTSRAp	74		2

Name	Hex Offset	Hex Value	Level
mcaTTR	168		3
mcaTWSCntlStart	13E	02	3
mcaupace	F	04	3
mcauserf	14C		2
mcaux002	13E	20	3
mcaux014	184		2
mcaver	4		3
mcavjvn	13B	08	3
mcavjvo	13B	10	3
mcavsam	BC		2
mcavsamb	C0		2
mcawaenq	13E	10	3
mcaWLMQsz	170		2
mcawsws	13A	40	3
mcaxsip	64		2
mcaX14numr	20		2
mcaX14rsiz	24		2
mcaX14tabp	1C		2
mcaznqd	E0		2
MCAzzSPIN	136	80	3
mcaZ308	152	10	3
mca313	E	01	3

TJCB - Tailoring JCL control block

Name : DCLTJCB

Function:
This is the js handler tailoring jcl control block. It holds information about all imbedded jcl.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	92	tjcb	tailoring jcl control block
0	(0)	CHARACTER	4	tjcbdesc	descriptor always 'tjcb'
4	(4)	CHARACTER	2	tjcbvers	block version
6	(6)	CHARACTER	2	*	not used
8	(8)	SIGNED	4	tjcblin	number of lines in this jcl block
12	(C)	SIGNED	4	tjcbclin	current jcl line being proc

Offsets					
16	(10)	ADDRESS	4	tjcbtubp	pointer to tub block
20	(14)	ADDRESS	4	tjcbstgp	pointer to storage allocated for jcl , or null if it is the first tjcb in chain, it also points to the common part of jcl record
24	(18)	SIGNED	4	tjcbofst	from start of orig/fetched JCL (the value is number of lines)
28	(1C)	ADDRESS	4	tjcbexpp	pointer to expansion work area
32	(20)	SIGNED	4	tjcbexps	size of exp. work area
36	(24)	SIGNED	4	tjcbamnt	amount of storage getmained for jcl pointed to by this tjcb
40	(28)	ADDRESS	4	tjcbjclp	pointer to first jcl line
44	(2C)	ADDRESS	4	tjcbnxtp	pointer to next tjcb block
48	(30)	ADDRESS	4	tjcbprep	pointer to previous tjcb block
52	(34)	SIGNED	4	tjcbnslv	nesting level of current tjcb
56	(38)	CHARACTER	1	tjcbtype	type of data pointed to block J= original JCL M= fetched nominated member X= fetched by user exit
57	(39)	CHARACTER	1	tjcbactn	action Y= include JCL N= not included / exclude
58	(3A)	CHARACTER	8	tjcblib	ddname of jcl library
66	(42)	CHARACTER	8	tjcbmem	member of jcl library
74	(4A)	CHARACTER	8	tjcbusrx	user exit name
82	(52)	CHARACTER	1	tjcbfin	data for this block processed

Offsets					
83	(53)	CHARACTER	1	tjcbstrm	JCL stream switches at start of JCL governed by this tjcb values as in prsbsw1
84	(54)	CHARACTER	4	tjcbint	directive introducing block command,main keyword,domain
88	(58)	CHARACTER	1	tjcbstsa	saved values for stream sw
89	(59)	CHARACTER	3	*	reserved values as in prsk $LBC
92	(5C)	CHARACTER		tjcbend	end label of block

Cross reference

Name	Hex Offset	Hex Value	Level
tjcb	0		1
tjcbclin	C		2
tjcblin	8		2
tjcbactn	39		2
tjcbamnt	24		2
tjcbdesc	0		2
tjcbend	5C		2
tjcbexpp	1C		2
tjcbexps	20		2
tjcbfin	52		2
tjcbint	54		2
tjcbjclp	28		2
tjcblib	3A		2
tjcbmem	42		2
tjcbnslv	34		2
tjcbnxtp	2C		2
tjcbofst	18		2
tjcbprep	30		2
tjcbstgp	14		2
tjcbstrm	53		2
tjcbstsa	58		2
tjcbtubp	10		2
tjcbtype	38		2
tjcbusrx	4A		2
tjcbvers	4		2

TUB - Tailoring JCL control block

Name : DCLTUB

Function:
This is the js handler tailoring jcl control block.
It holds information about all imbedded jcl.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	800	tub	tailoring user block
0	(0)	CHARACTER	4	tubdesc	descriptor always 'TUB '
4	(4)	CHARACTER	2	tubvers	block version
6	(6)	CHARACTER	8	tuboreqd	original data type
14	(E)	CHARACTER	8	tubident	tailoring identifier
14	(E)	CHARACTER	8	tubuser	..tso; userid
22	(16)	CHARACTER	28	tubopid	operation identifier
22	(16)	CHARACTER	16	tubadid	..application; id
38	(26)	CHARACTER	6	tubadiad	..input; arrival day
44	(2C)	CHARACTER	4	tubadiat	..input; arrival time
48	(30)	SIGNED	2	tubopno	..operation; number
50	(32)	CHARACTER	1	tubSimulate	time simulation Y or N
51	(33)	CHARACTER	1	*	not used
52	(34)	ADDRESS	4	tubocp	addr of PIF format CP occ
56	(38)	ADDRESS	4	tubopp	addr of PIF format CP opr
60	(3C)	ADDRESS	4	tubwsp	address of PIF format CP WS
64	(40)	SIGNED	4	tubocl	length of PIF format CP occ
68	(44)	SIGNED	4	tubopl	length of PIF format CP opr
72	(48)	SIGNED	4	tubwsl	length of PIF format CP WS
76	(4C)	ADDRESS	4	tubdcbp	current jcl library dcb
80	(50)	ADDRESS	4	tubbufp	address of jblib buffer
84	(54)	SIGNED	4	tubasubp	subpool for subseq allocs

Offsets					
88	(58)	ADDRESS	4	tubworkp	ptr to tailoring work area
92	(5C)	ADDRESS	4	tubmcap	pointer to mca
96	(60)	SIGNED	4	tubworkl	length of tail work area
100	(64)	ADDRESS	4	tubjcbcu	current tjcb
104	(68)	ADDRESS	4	tubjcbfp	pointer to first tjcb
108	(6C)	ADDRESS	4	tubjcblp	pointer to last tjcb
112	(70)	ADDRESS	4	tubjvptr	pointer to jv record buffer
116	(74)	ADDRESS	4	tubdatp	pointer predef variables and their values
120	(78)	ADDRESS	4	tubtvp	pointer to jcl vars found when searching jcl
124	(7C)	SIGNED	4	tubtvarl	length of allocated var stg
128	(80)	CHARACTER	2	tubtask	current task (GS,WA)
130	(82)	CHARACTER	8	tubjclib	current jcl library
138	(8A)	CHARACTER	1	tuballv	Y= variable proc. complete
139	(8B)	CHARACTER	1	*	not used
140	(8C)	CHARACTER	16	tubjvtab	occurrence variable table
156	(9C)	CHARACTER	16	tubsearch (16)	current table search order
412	(19C)	CHARACTER	16	tubTabName (16)	table names array $CXWC
668	(29C)	ADDRESS	4	tubTabNameP (16)	table ptrs array $CXWA
732	(2DC)	CHARACTER	8	tubfoot	current footprint
740	(2E4)	CHARACTER	2	tubdlm	current dlm
742	(2E6)	CHARACTER	1	*	reserved
743	(2E7)	BITSTRING	1	tubflags	flag byte
		1...		tubosi	osi operation JCL
		.1..		tubboj	osi operation JCL
		..11 1111		*	reserved
744	(2E8)	SIGNED	4	tubosi	number of lines inserted
748	(2EC)	ADDRESS	4	tubjcfp	pointer to feedback area
752	(2F0)	ADDRESS	4	tubjdup	addr of user SETFORM date dates

Offsets					
756	(2F4)	ADDRESS	4	tubtvsp	addr of SAVEVAR variable in stg
760	(2F8)	ADDRESS	4	tubjdtp	addr of SETVAR variables
764	(2FC)	ADDRESS	4	tubysimp	addr of simulation parms
768	(300)	ADDRESS	4	tuboca	addr of PIF format occ alwaysOEA
772	(304)	ADDRESS	4	tubopa	addr of PIF format opr alwaysOEA
776	(308)	ADDRESS	4	tubwsa	address of PIF format ws alw.OEA
780	(30C)	ADDRESS	4	tubxinfp	Extended Job Info addr.
784	(310)	ADDRESS	4	tubDOA	DOA address needed to $CAEC check DOAPSUJCL $CAEA
788	(314)	ADDRESS	4	tubcp3c	CP occurrence
792	(318)	ADDRESS	4	tubcp3p	CP operation record
796	(31C)	ADDRESS	4	tubcp3r	CP op record user fields
800	(320)	CHARACTER		tubend	end of tub block

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	200	tubxinf	
0	(0)	CHARACTER	54	tubxjnm	
54	(36)	CHARACTER	146	*	

Cross reference

Name	Hex Offset	Hex Value	Level
tub	0		1
tubosi	2E8		2
tubadiad	26		3
tubadiat	2C		3
tubadid	16		3
tuballv	8A		2
tubasubp	54		2
tubboj	2E7	40	3

Name	Hex Offset	Hex Value	Level
tubbufp	50		2
tubcp3c	314		2
tubcp3p	318		2
tubcp3r	31C		2
tubdatp	74		2
tubdcbp	4C		2
tubdesc	0		2
tubdlm	2E4		2
tubDOA	310		2
tubend	320		2
tubflags	2E7		2
tubfoot	2DC		2
tubident	E		2
tubjcbcu	64		2
tubjcbfp	68		2
tubjcblp	6C		2
tubjcfp	2EC		2
tubjclib	82		2
tubjdtp	2F8		2
tubjdup	2F0		2
tubjvptr	70		2
tubjvtab	8C		2
tubmcap	5C		2
tuboca	300		2
tubocl	40		2
tubocp	34		2
tubopa	304		2
tubopid	16		2
tubopl	44		2
tubopno	30		3
tubopp	38		2
tuboreqd	6		2
tubosi	2E7	80	3
tubsearch	9C		2
tubSimulate	32		2
tubTabName	19C		2
tubTabNameP	29C		2
tubtask	80		2
tubtvarl	7C		2
tubtvp	78		2
tubtvsp	2F4		2

Name	Hex Offset	Hex Value	Level
tubuser	E		3
tubvers	4		2
tubworkl	60		2
tubworkp	58		2
tubwsa	308		2
tubwsl	48		2
tubwsp	3C		2
tubxinf	0		1
tubxinfp	30C		2
tubxjnm	0		2
tubysimp	2FC		2

TV - JCL variable table record description

Name : DCLTV

Function:
Defines the layout of each non-preset variable encountered in the jobstream. The information in the main section of the record and the dependency information are taken unaltered from the corresponding variable description in the JCL variable type 3 table record.
NOTE: This block is expanded by getmains to the next multiple of 32K whenever there is insufficient space for the next entry. Current address and getmained length are held in tub.

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	16	tv	JCL variable table
0	(0)	CHARACTER	4	tveye	descriptor always 'tvar'
4	(4)	CHARACTER	2	tvvers	block version
6	(6)	CHARACTER	2	*	not used
8	(8)	SIGNED	4	tvvars	number of variables in storage
12	(C)	SIGNED	4	tvnoff	offset to next free byte
16	(10)	CHARACTER		tvdata	variable part of table

Offsets

Dec	Hex	Type	Len	Name (Dim)	Description
0	(0)	STRUCTURE	552	tvtab	
0	(0)	CHARACTER	88	tvrun	tailoring run-time info

Offsets					
0	(0)	CHARACTER	16	tvtable	source table name
16	(10)	CHARACTER	8	tvasusr	assigned by USER/EXIT/VAR
24	(18)	CHARACTER	44	tvasg	value assigned
68	(44)	CHARACTER	16	tvfrst	first occurrence
68	(44)	SIGNED	4	tvotjcb	address of source tjcb
72	(48)	SIGNED	4	tvoline	JCL line within tjcb scope
76	(4C)	SIGNED	2	tvovnr	seq of identification in line
78	(4E)	CHARACTER	6	*	reserved
84	(54)	CHARACTER	1	tvasgtyp	type of assignment P/E/V/D
85	(55)	CHARACTER	1	tvset	Y(es) ! N(o) ! D(elayed)
86	(56)	CHARACTER	1	tvusg	type of usage % / & /
87	(57)	CHARACTER	1	*	reserved
88	(58)	CHARACTER	464	tvfxd	unchanged from VSAM
88	(58)	CHARACTER	8	tvvar	JCL variable name
96	(60)	CHARACTER	44	tvdfl	JCL variable default value
140	(8C)	CHARACTER	1	tvuc	uppercae Y/N
141	(8D)	CHARACTER	1	tvstp	prompt / setup / submit
142	(8E)	SIGNED	2	tvlg	value length
144	(90)	CHARACTER	7	tvtyp	verification type
151	(97)	CHARACTER	8	tvex	substitution exit name
159	(9F)	CHARACTER	1	tvinp	input required
160	(A0)	SIGNED	2	tvpos	replace position JCL data
162	(A2)	CHARACTER	1	tvnum	numeric
163	(A3)	CHARACTER	2	tvcmp	comparison operator
165	(A5)	CHARACTER	44	tvpat	validation pattern
209	(D1)	CHARACTER	102	tvvld	valid values
311	(137)	CHARACTER	204	tvtxt	dialog text
515	(203)	CHARACTER	20	tvdes	description
535	(217)	CHARACTER	1	*	reserved
536	(218)	SIGNED	2	tvnrp	number of dependent values

Offsets					
538	(21A)	CHARACTER	8	tvind	independent variable name
546	(222)	CHARACTER	2	*	reserved
548	(224)	CHARACTER	2	tvsubs	substring start pos
550	(226)	CHARACTER	2	tvsubl	substring length

Offsets					
Dec	**Hex**	**Type**	**Len**	**Name (Dim)**	**Description**
0	(0)	STRUCTURE	88	tvd(*)	
0	(0)	CHARACTER	44	tvdiv	value of independent
44	(2C)	CHARACTER	44	tvddv	value of dependent

Cross reference

Name	Hex Offset	Hex Value	Level
tv	0		1
tvvars	8		2
tvasg	18		3
tvasgtyp	54		3
tvasusr	10		3
tvcmp	A3		3
tvd	0		1
tvdata	10		2
tvddv	2C		2
tvdes	203		3
tvdfl	60		3
tvdiv	0		2
tvex	97		3
tveye	0		2
tvfrst	44		3
tvfxd	58		2
tvind	21A		3
tvinp	9F		3
tvlg	8E		3
tvnoff	C		2
tvnrp	218		3
tvnum	A2		3
tvoline	48		4
tvotjcb	44		4

Name	Hex Offset	Hex Value	Level
tvovnr	4C		4
tvpat	A5		3
tvpos	A0		3
tvrun	0		2
tvset	55		3
tvstp	8D		3
tvsubl	226		3
tvsubs	224		3
tvtab	0		1
tvtable	0		3
tvtxt	137		3
tvtyp	90		3
tvuc	8C		3
tvusg	56		3
tvvar	58		3
tvvers	4		2
tvvld	D1		3

Messages

Messages issued by the agent for z/OS.

All messages issued by the agent for z/OS are described in *Tivoli Workload Automation: Messages and Codes*.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

© Copyright IBM Corp. 2011, 2016

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

© (your company name) (year).
Portions of this code are derived from IBM Corp. Sample Programs.
© Copyright IBM Corp. _enter the year or years_.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a Registered Trade Mark of AXELOS Limited.

Linear Tape-Open, LTO, the LTO Logo, Ultrium, and the Ultrium logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

ITIL is a Registered Trade Mark of AXELOS Limited.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

Index

A

accessibility ix
agent for z/OS
  address space
    batch job definition 23
    JCL procedure definition 23
  data sets
    diagnostic 8
    event 8
    message library 8
    message log 8
    parameter library 8
  default initialization statements 9
  download 4
  hostname and port specification 32
  libraries 5
  runtime options 35
  STC name 17
  subsystem name 17
  workstation name 25
agent for z/OS exits 39
agent for z/OS job 48
agent for z/OS optional datasets 24
agent for z/OS required datasets 23
agent for z/OS workstation 48, 49
agent restart 99
APARs
  PI09318 57
  PI35949 33
application keyword 48

B

backup domain managers 96
BEGIN JCL directive 83

C

cancel 2
CANCEL command 95
cancelling the agent 95
CDATE, JCL variable 76
CDAY, JCL variable 75
CDD, JCL variable 75
CDDD, JCL variable 75
CDDMMYY, JCL variable 75
CHH, JCL variable 75
CHHMM, JCL variable 75
CHHMMSS, JCL variable 75
CHHMMSSX, JCL variable 75
Cloud & Smarter Infrastructure technical training ix
CMM, JCL variable 75
CMMYY, JCL variable 75
codepage definition 35
CODEPAGE parameter 56
coding JCL variables 69
commands
  unavailable 89
COMP keyword on JCL directives 86
compound variables 70
computer name 47
conman commands 89
connection options 30
considerations when allocating job library (EELJBLIB) 29
CTIME, JCL variable 76
CWW, JCL variable 75
CWWD, JCL variable 75
CYMD, JCL variable 75
CYY, JCL variable 75
CYYDDD, JCL variable 75
CYYMM, JCL variable 76
CYYMMDD, JCL variable 76
CYYYY, JCL variable 76
CYYYYMM, JCL variable 76

D

data areas
  DCLDQE 103
  DCLESP 125
  DCLEVT 126
  DCLEXI 128
  DCLEXK 132
  DCLEXR 135
  DCLHTI 140
  DCLHTSA 142
  DCLJCFB 145
  DCLJCL 147
  DCLJCL1 152
  DCLJDA 155
  DCLJDT 161
  DCLJDU 163
  DCLJHS 167
  DCLJV 170
  DCLMCA 172
  DCLTJCB 187
  DCLTUB 190
  DCLTV 194
data router subtask 95
DB2 transaction log full 101
default variable table 73
directives 77
dump options 20
dynamic workload broker 59
  changing 96
  hostname and port specification 32
  synchronizing with 99
  updating 97
dynamic workload brokerbackup instances 96
Dynamic Workload Console
  accessibility ix

E

EBCDIC format 56
ECSA storage 17
education ix
EELBRDS data set 60
EELEVDS 92, 93, 99
EELEXIT macro 11
EELHTDS 92, 99
EELHTREF 59, 92, 99
EELINST
  data set creation 8
  libraries 6
  sample JCL 6
  sample job creation 7
  starting 6
EELJBLIB 59
EELJBLIB (job library data set)
  considerations when allocating 29
EELUX002 (job-library-read exit) 40
END JCL directive 83
event data set 26
event data sets 92, 99
event filtering exit 43
event writer 26
  runtime options 30
event writer subtask 95
event writer task 63
event-writer 93
events
  troubleshooting 28
events generated 26
EWTROPTS 30
exit policy 30
exit51 13
exits
  event filtering 43
  job library read (EELUX002) 40
  SMF
    job initiation 19
    job-end and step-end 19
    record write 19
  start/stop 39

F

Failover mechanism 59
FETCH JCL directive 85
file watching utility 57

G

global variable table 73

H

HTTP
  connection options 31
HTTP client 96
HTTPOPTS 22, 31
HTTPOPTS initialization statement 25
  SSLKEYRING 10

© Copyright IBM Corp. 2011, 2016

I

IBM Workload Scheduler for z/OS 2
  SSL 23
  SUBSYS STC 19
IBM Workload Scheduler variables 63
IEFACTRT 19
IEFU83 19
IEFUJI 19
in conman 90
initialization statements 30
  event writer 30
  exit policy 31
  HTTP options 31
  runtime options 35

J

JCL
  data set 53
  definition 2
  definition rules 56
  location in z/OS 53
  member 53
  name rules 56
  national characters 56
  programmer name 56
  statement 54
  tailoring 2
  variable substitution
    syntax 69
JCL error 92
JCL errors 91
JCL tailoring directives 77
JES
  internal reader 90
JES2 exits
  entry points 12
  installation commands 13
  load modules 12
jesexit7 13
job
  definition interfaces 2
  definition type 48
  log 2
  resiliency 61
  status event 61
  submission 59
job canceled 92
job completion codes 93
job definition
  by JCL definition 54
  by JCL reference 53
  composer 53
    example 53, 54, 55
  Dynamic Workload Console 49
  example
    Dynamic Workload Console 51
  JSDL 48
  variable resolution 55
job error codes 91
job hangs 101
job instance management 89
job library data set (EELJBLIB)
  considerations when allocating 29
job log requests 63
job logs
  viewing
    in conman 94
    in Dynamic Workload Console 94
job print event 91
job purged event 91
job return codes 91
job state
  in Dynamic Workload Console 90
job state description 89
JOB statement 93
job submission events 26
job submission flow 59
job variables
  definition
    in JCL 63
    in JSDL 63
job-end event 90
job-end events 26
job-library-read exit (EELUX002) 40
job-start event 90
job-start events 26
job-step end events 26
job-termination event 91
job-termination events 26
JOBRC 30, 93
JSDL 2

K

kill 2, 89
kill job 89

L

link library 20
listing agent for z/OS workstations 48

M

managing job instances 89
maxdur, job stream keyword 89
maxecsa 17
MODIFY command 95
modifying the agent 95

N

national characters 56
network job entry 46
NJE 46, 63

O

OADID, JCL variable 73
OCDATE, JCL variable 76
OCTIME, JCL variable 76
ODAY, JCL variable 73
ODD, JCL variable 74
ODDD, JCL variable 74
ODMY1, JCL variable 74
ODMY2, JCL variable 74
OHH, JCL variable 74
OHHMM, JCL variable 74
OMM, JCL variable 74
OMMYY, JCL variable 74
OPIADATE, JCL variable 76
OPIATIME, JCL variable 76
OS type 48
OSSID, JCL variable 75
output manager 63
OWW, JCL variable 74
OWWD, JCL variable 74
OWWLAST, JCL variable 74
OWWMONTH, JCL variable 74
OYM, JCL variable 74
OYMD, JCL variable 74
OYMD1, JCL variable 74
OYMD2, JCL variable 74
OYMD3, JCL variable 74
OYY, JCL variable 74
OYYDDD, JCL variable 74
OYYMM, JCL variable 75
OYYYY, JCL variable 75

P

plan interruption 101
print events 26
product version 101
programmer name length 56
purge events 26

R

RACF 21
  data set protection 22
reader event 90
reader events 26
rerun 2
resynchronization messages 99
runtime options 30

S

sample exits
  JCT I/O exit for JES2, purge 12
  JES2 QMOD phase change 12
  job and step completion 12
  job print end 12
  job start 12
sample started task
  parameters 23
  procedure 23
SCAN JCL directive 77
SETFORM JCL directive 78
SETVAR JCL directive 80
showjobs 94
simple variables 70
SMF exits 19
SMP/E 4
special characters 56
SSL 22
  importing default certificates 10
  security certificates 22
SSL security options 32
START command 94
START subtask 95
start/stop exit 39
started tasks 2
starting the agent 94

step-end event 90
STOP command 95
STOP subtask 95
stopping and restarting the agent 97
stopping the agent 95
submission error 92
submit subtask 95
submit task 59
subsystem definition
  example 17
  record 17
subsystem name 17, 47
subtasks
  data router 95
  event writer 95
  submit 95
switching domain managers 96
sysplex 45
SYSPLEX 63
system abend 91
system commands 94
system name 47

T

TDWBHOSTNAME 97
TDWBPORT 97
technical training ix
tracker
  exits 2
training
  technical ix
TWSOPTS 35
  codepage 56

U

updating agent configuration 97
user abend 91
utilitiesfile watching 57

V

variable resolution 63
variable table
  default 73
  global 73
variables
  in JCL 69
  predefined in JCL 73
    date-related 75
    dynamic-format 76
    job stream-related 73
    job-related 75
    temporary 76
  user-defined in JCL 73
VARSUB keyword of TWSOPTS 69

W

workstation name 47
  changing 47
workstation type 48

Z

z/OS 48
z/OS commands 94
z/OS internal event 90

Product Number: 5698-WSH

Printed in USA