ESP Workload Manager

Version 5.5

Installation and Configuration Guide

ESP-5.5-IG-01

First Edition (June 2007)

Contents

About this Guide

This guide helps you

- Install ESP Workload Manager
- Install ESP Workload Manager High Availability Option
- Install ESP Workload Manager Service Governor
- Install ESP Encore
- Connect ESP Workstation to ESP Workload Manager
- Upgrade from an earlier version of ESP Workload Manager to the current release
- Add another ESP Workload Manager to your installation
- Configure ESP Workload Manager and ESP Encore to suit your requirements

Using this Guide

When to use this guide

- As you install ESP Workload Manager to step you through the necessary tasks
- After the installation to reconfigure the appropriate ESP Workload Manager initialization parameters
- When installing a new version of ESP Workload Manager over an existing version
- When adding another ESP Workload Manager system into an existing network.

Entering commands

ESP Workload Manager commands can be entered in line mode, page mode or batch, loaded from a data set (using the LOAD command) or entered via ESP Workstation. You can also enter many commands from a system console.

Entering statements

You must enter ESP Workload Manager statements in a data set specific to the type of statement used.

Note: The content of this section also applies to commands and initialization parameters when they are entered in a data set.

Continuation

To continue a line of input, type either a hyphen (-) or a plus sign (+) as the last non-blank character on a line. The hyphen attaches the next line, including any leading blank position. The plus sign strips leading blanks from a continuation line. ESP Workload Manager retains blanks that precede the hyphen or the plus sign. Statements cannot extend beyond column 72. You can place the continuation character in or before column 72.

Note: You can also use a hyphen as a wildcard character. If the wildcard hyphen is the last character on the line, ESP Workload Manager will interpret the hyphen as a continuation character and not as a wildcard. For ESP Workload Manager to interpret the hyphen as a wildcard, follow the hyphen with a semicolon or something else on the line, such as a comment: (/* */).

Wildcards and masking

Many statements, commands or initialization parameters permit the use of the following wildcard characters (also called masking):

- An asterisk matches a single character.
- A hyphen matches zero or more characters. It must be used as the last character of the operand. If the wildcard hyphen is the last character on the line, ESP Workload Manager interprets the hyphen as a continuation character and not as a

wildcard. For ESP Workload Manager to interpret the hyphen as a wildcard, follow the hyphen with a semicolon or something else on the line, such as a comment: (/* */).

Comments

Enclose comments between /* and */. You can write comments anywhere in an ESP Workload Manager Procedure.

Data sets

Enclose all data set names in single quotation marks (‘’), otherwise ESP Workload Manager adds your TSO data set prefix to the name. Use LIB- or PAN- prefixes to identify Librarian and Panvalet data sets.

Delimiters

Use single quotation marks when you want to denote character strings and literal data in expressions, assignment statements, and built-in functions.

Note: You must include single quotation marks around a string that contains blanks.

Indentation

Use indentation to improve readability.

Examples

Issuing commands from the console

To issue a command from the system console, use the MODIFY command, as in F ESP, where ESP is the started task name, for example

```
F ESP,STATUS
```

Issuing OPER commands

To issue a command that requires OPER authority, precede it with OPER, for example

```
OPER STATUS
```

Issuing commands in batch

To issue a command in batch, use the following JCL after your job card.

If the subsystem name for ESP is not ESP, you need to type the subsystem name when entering a command in batch. In the following example, the subsystem name is ESPA.

```
//EXEC PGM=ESP,PARM='SUBSYS(ESPA)',REGION=4000K
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
```

```
DEFSPEC MYSPECIALDAY REPEAT('2 TIMES FIRST WORKDAY OF YEAR')+
CAL(MYCAL)
```

Steplib

If the TSO command processor is not in a LINKLIST library, you need a STEPLIB or JOBLIB statement in your JCL. Your statement might look like the following:

```
//STEPLIB DD DSN=CYB.ESP.SSCPLINK,DISP=SHR
```

Wildcards and masking

To display all calendars, type

```
LISTCAL –
```

To display all calendars with names containing XY in character positions three and four, type

```
LISTCAL **XY-
```

To display all calendars with two-character names, type

```
LISTCAL **
```

Conventions Used in this Guide

Input

You can type commands in uppercase or lowercase.

Syntax conventions

The syntax diagrams in this guide use the following conventions:

Notation	Meaning
Quotation marks " or '	Must be entered as shown
Comma ,	Must be entered as shown
Ellipsis …	The operand can be repeated. Do not enter ellipsis.
Lowercase Italics operand	User-supplied variable or character string. A value must be substituted.
Uppercase operand	The operand must be spelled as shown. You can enter the command and the operand in either uppercase or lowercase.
OR bar (|)	Indicates an exclusive value on left or right of the bar. You must enter one of the items. You cannot enter more than one.
Underline ________	If you do not enter one of the operands, the system supplies the underlined operand. The underline indicates the default.

Notation	Meaning
Parentheses ( ) and special characters	An operand enclosed in parentheses is mandatory and must be entered as shown.
Single operand in square brackets [ ]	Optional operand. Do not type the brackets.
Stacked operands in braces { } { }	Mandatory operand. You must enter one of the operands. You cannot enter more than one.
Stacked operands in square brackets [ ] [ ]	Optional operand. You can enter one value, or none.
Operands with OR-bars (|) and square brackets [ ]|[ ]	Optional, mutually exclusive operands. Enter one or none.
Stacked operands in square brackets within braces {[ ] [ ]}	Mandatory operand. You must enter one of these operands. You can enter more than one.

If an operand has complex syntax, “See expanded syntax” will follow the operand. Expanded syntax is documented in a separate table after the main syntax table.

Summary of Changes

- PTF SU01811, defect 16499: In ENCPARM PRINT initialization parameter, added section name “RETRIEVAL”.
- PTF SU02077: Added GENERATION operand to SADLINK initialization parameter
- PTF SU02363: Added FROMADDRESS operand to SMTPPARM initialization parameter.
- PTF SU02536: In the XCF SET SERVICE initialization parameter, added operand TIMER and changed the default for the SERVICE operand from 120 seconds to 60 seconds.
- PTF SU02767: You can now control the status of a user exit with the EXIT initialization parameter. For example, you can disable a user exit. See “Activating and Controlling a Supplied User Exit” on page 224 and the EXIT initialization parameter.

- PTF SU02815: Added initialization parameters LOGTRACE, XDAB, and XFRB.
- RFEs 635 and 2310: New CUSTCTBL initialization parameter allows you to load custom ASCII-EBCDIC and EBCDIC-ASCII character-conversion tables. See “CUSTCTBL: Load custom ASCII-EBCDIC conversion tables” on page 279.
- RFE 636: New Application Monitor
  - Added Application Monitor information to “Application Data Set (APPLFILE)” on page 16 and “Jobindex Data Set (JOBINDEX)” on page 27
  - Added checklist steps for installing CSF and Application Monitor REXX extensions. See “Started Task Checklist” on page 56.
  - Added utility install in “Installing utilities accessed by the ESP Utilities panel” on page 60
  - Added Application Monitor scoreboard to “XCF scoreboard service” on page 133
  - Added “Implementing Application Monitor Extensions” on page 203
- RFE 1155: New JOBSTATS data set contains job statistics and job-profiling information. This information enhances ESP Workload Manager’s anticipated end-time and critical-path calculations. See “Allocating Data Sets” on page 13. Related initialization parameters are: “JOBSTATS: Jobstats data set” on page 382, “JOBPROF: Defining a Job Profile Name” on page 379, “BKUPJSTS: Back up the jobstats data set” on page 259.
- RFE 1935, defect 12162
  - In “Setting Up Tracking on Your System” on page 64, removed reference to SMF record type 132 subtype 1. ESPDST (alias CYBESDT1) no longer produces those records.
  - Added EXPDSTRG initialization parameter
- RFE 2254: Added operand WOBPRIORITY to the EXPEDITE initialization parameter to change the process priority of a UNIX or Windows job that is executing.
- RFE 2319: Added a section on APPLFILE requirements for ESP Procedure caching. See “ESP Procedure caching” on page 17. Added information to the PCACHE initialization parameter regarding the new Procedure Caching Record (PCR).
- RFE 2753: Added three new usermods, numbers 58, 59, and 187 in “ESP Workload Manager USERMODs” on page 213.
- RFE 2931, defect 13313, defect 22634, defect 22644: Added variable FULLNAME to the JOBONCSF function and added function CSFJOB. See “Using the CSFJOB Function” on page 184 and “Using the JOBONCSF Function” on page 188.

- RFE 3300
  - Added LOG initialization parameter.
  - Added MAILLOG initialization parameter.
  - Added TSOSEND initialization parameter.
  - In “MAILLIST Data Set” on page 49, added information about LOG initialization parameter.
  - Added ORIGUSER operand to the SMTPPARM initialization parameter.
- RFE 3409: Added the BIND operand to the MANAGER initialization parameter and added the HOME operand to the AGENTRCV initialization parameter. These operands enable ESP Workload Manager TCP/IP connections with ESP Agents to use the same local TCP/IP address. See “MANAGER: Specify the Subsystem Controlling the ESP Agents” on page 400 and “AGENTRCV: Define Agent receiver” on page 246.
- RFE 21353: Added USERMOD 79 for step ESPCCCHK generated by the CCCHK initialization parameter or statement.
- DEF 11074: Added information on configuring ESP Workload Manager to use Agents. See
  - “Adding Agents” on page 8
  - “Sample Library Members Containing Initialization Parameters” on page 42
  - “Required Initialization Parameters” on page 44
  - “Configuring ESP Workload Manager to Use Agents” on page 103
  - “Workload object modules” on page 540

  Reviewed and updated the workload object type table to ensure it is complete and accurate.
- Bug 15426: Added the DFLTDSN initialization parameter. See “DFLTDSN: Specify Parameters for Data Sets Used by ESP Workload Manager” on page 285.
- Bug 15489 (documentation corrections)
  - In “ESP Workload Manager DASD requirements” on page 16 and “Resource Data Set (RESFILE)” on page 30, increased the recommended size of the RESFILE from 2 M to 4 M.
  - In “Creating and Customizing the Initialization Parameter Data Set” on page 44, documented a recommended primary track size, secondary track size, and record length for the initialization parameter data set.
  - In “Setting Up ESP Workload Manager Resources on Your System” on page 78, added a section on defining resources within an IF RESFORM THEN loop.
- Defect 15506: Added “NETWORK START SERVICE ROUTING” to table under “Configuring ESP Workload Manager for NDS” on page 148.

- Bug 15658: In “Agent Communications Data Set (COMMQ)” on page 19, removed the reference to abbreviation CQ because anything from CQ to CQFORMAT is recognized as CQFORMAT.
- Bug 15739: In “TAPETRAK: Control tape usage count” on page 503, added note about valid use under “Usage notes”.
- Bug 16619: Corrected “USERMOD 25” on page 217. The criteria listed govern job execution, not job submission.
- Bug 16731: In “WSSCTL: Set parameters and control message tracing” on page 545, in the operand table, changed default for MAXCON_TOTAL from 2508 to 250.
- Bug 16818: Added new parameter “SUBSYS: Set subsystem name (ESP Workstation Server)” on page 493.
- Bug 16826: In “EVENTOPT: Set Event options” on page 342, underlined the default operands.
- Bug 16905: In the Usage Notes for the MCS initialization parameter, added information on what happens when the MCS command is issued without operands.
- Bug 17013: In “APPLFILE: Define Application data set” on page 252 and “TRAKFILE: Define tracking data set” on page 529, noted that the CACHE operand is ignored on proxy systems.
- Bug 17073: Revised the space requirements note under “Resource Data Set (RESFILE)” on page 30. Added info on specifying secondary extent size and increasing the value of the SIZE parameter to the Usage Notes in “RESFILE: Specify the resource data set” on page 468.
- Bug 17247: Added Usermod 60. See “ESP Workload Manager USERMODs” on page 213.
- Bug 17580: In “Network Job Entry (NJE)” on page 154, corrected the NJE examples.
- Bug 18390: Revised the description of the MAXENTRIES operand of “SCHDFILE: Identify schedule and work data set names” on page 480. MAXENTRIES is now obsolete, so you no longer have to set this operand.
- Bug 18724: In “TIMECHK: Check time” on page 511, documented a new feature that warns the operator if the system clock changes by more than a specified time.
- Bug 19086: In “USERDEF: Identify user definition data set” on page 533, added a usage note stating that you do not need to specify USERDEF if your installation uses SAF.

- Bug 19316
  - Revised the description and space requirements of “Schedule Data Set (SCHDFILE)” on page 36. This data set is now optional.
  - In “LOADSCHF: Load schedule data set into CSF” on page 389, updated the example to show how to populate the work data set, load the schedule data set, and view the schedule data in CSF.
  - In “SCHDFILE: Identify schedule and work data set names” on page 480, added the operand NONE, updated the description and default of the MAXENTRIES operand, and changed the default to SECURE from NOSECURE.
- Bug 19477: In “CCCHK: Specify Job Processing Based on Condition Codes” on page 261, you can now use wildcards to specify the JOB, STEP, PROC, and PROGRAM operands.
- Bug 19631: When setting up the ESP Workload Manager started task during installation, you no longer have to run the CYBESINI CLIST. Removed section “Defining Required Entities to ESP Workload Manager”.
- Bug 20273: In “Updating CLIST” on page 59, added a sample CLIST to allocate ESP Workload Manager libraries and to display the ESP Workload Manager Primary menu.
- Bug 20321: In “Sample Library Members Containing Initialization Parameters” on page 42, added name WSSPARM for sample member CYBESS32 (Workstation Server initialization parameters) and WSSREMOT for CYBESS33 (Remote Workstation Server initialization parameters).
- Bug 20769: Under “Masters and Proxies” on page 122, added INDEX as a data set that must be shared, added descriptions for data sets that can be shared, and re-worded Performance considerations for clarity.
- Bug 21695: Changed the description of the MAXLRECL initialization parameter to indicate it is obsolete.
- Bug 21866: Reorganization of HAO and Service Governor (formerly HPO) information
  - Added information on XCF services in “XCF services” on page 132.
  - Moved information on using XCF in a sysplex to “Configuring ESP Workload Manager subsystems for a sysplex environment” on page 134.
  - Added information on XCF status monitoring in “XCF status monitoring” on page 134.
  - Added chapter “Using Shadow Manager and Automatic Restart Management” on page 137.
- Bug 21993: In “USERMOD 58” on page 220, added more conditions that result in message 2069W.

- Bug 22010: Added note to RESDEF WLM operand stating that if you code WLM, you must code the SYSPLEX operand in the NODE command or initialization parameter when you define your system nodes.
- Bug 22206: In the RESDEF initialization parameter, corrected the paragraph on workload balancing in “Resources with MONITOR(CPU) operand (ESP Workload Manager Service Governor only)” on page 459.

Part 1

Installing, Upgrading, and Configuring ESP Workload Manager

This part takes you through the process of installing ESP Workload Manager on your system.

This part contains the following chapters:

1

Installing and Upgrading ESP Workload Manager

This chapter contains the following topics:

- Installing ESP Workload Manager
- Upgrading to ESP Workload Manager Version 5.5
- Adding ESP Workload Manager High Availability Option (HAO)
- Adding ESP Workload Manager Service Governor
- Adding ESP Encore
- Adding Agents
- Setting up TP Server
- Setting up Workstation Server
- Installing Application Programming Interface (API)
- Installing the Software From the Tape
- Interfacing with AllFusion CA-Librarian

Installing ESP Workload Manager

Follow the steps outlined below to install ESP Workload Manager Version 5.5. When you have completed the basic installation, you can customize ESP Workload Manager for your site.

What you need

To install ESP Workload Manager, you need the following documents:

- Program Directory for Cybermation Programming Environment (CPE)—FMID SCP5100
- Program Directory for Enterprise Systems Platform Workload Manager (ESP Workload Manager)—FMID SEP5500
- This manual

Sample library

The tape you received containing the ESP Workload Manager software also contains a sample library, SSCPSAME, with several sample members for your use. You can edit these to fit your requirements.

To install ESP Workload Manager for the first time, follow these steps:

Step	Action	Page No.	√
1	Install the software from the tape.	11	
2	Allocate the ESP Workload Manager data sets.	13	
3	Set up the ESP Workload Manager initialization parameters.	41	
4	Define basic security.	51	
5	Set up the ESP Workload Manager started task.	55	
6	Define what ESP Workload Manager will track.	63	

Upgrading to ESP Workload Manager Version 5.5

Follow the steps outlined below to upgrade ESP Workload Manager to Version 5.5. When you have completed the upgrade, you can make any changes to customize ESP Workload Manager for your site.

What you need

To upgrade ESP Workload Manager, you need the following documents:

- Program Directory for CPE—FMID SCP5100
- Program Directory for Enterprise Systems Platform Workload Manager—FMID SEP5500
- This manual

Sample library

The tape you received containing the ESP Workload Manager software also contains a sample library, SSCPSAME, with several sample members for your use. You can edit these to fit your requirements.

To upgrade ESP Workload Manager from an earlier release, follow these steps:

Step	Action	Page No.	√
1	Install the software from the tape.	11	
2	Review the allocation of the ESP Workload Manager data sets.	13	
3	Modify any ESP Workload Manager initialization parameters.	41	
4	Update the LOGON proc or CLIST with any new library names.	58	
5	Follow the instruction under “Starting ESP Workload Manager after upgrades are installed”	61	

Adding ESP Workload Manager High Availability Option (HAO)

Follow the steps outlined below to install the HAO. When you have completed the installation, you can customize ESP Workload Manager for your site.

Note: You must install ESP Workload Manager before or with HAO.

What you need

To install HAO you need the following documents:

- Program Directory for Enterprise Systems Platform Workload Manager High Availability Option—FMID SEP5501
- This manual

Sample library

The tape you received containing the HAO software also contains a sample library, SSCPSAME, with several sample members for your use. You can edit these members to fit your requirements.

To add HAO to your ESP Workload Manager installation, follow these steps:

Step	Action	Page No.	√
1	Install the software from the tape.	11	
2	Modify any ESP Workload Manager initialization parameters.	41	
3	Update the LOGON proc or CLIST with any new library names.	58	
4	Follow the instruction under “Starting ESP Workload Manager after upgrades are installed”.	61	
5	Update the started task for shadow manager.	138	

Adding ESP Workload Manager Service Governor

Follow the steps outlined below to install Service Governor. When you have completed the installation, you can customize ESP Workload Manager for your site.

Note: You must install ESP Workload Manager before or with Service Governor.

What you need

To install Service Governor, you need the following documents:

- Program Directory for Enterprise Systems Platform Workload Manager Service Governor—FMID SEP5502
- This manual

Sample library

The tape you received containing the Service Governor software also contains a sample library, SSCPSAME, with several sample members for your use. You can edit these members to fit your requirements.

To add Service Governor to your ESP Workload Manager installation, follow these steps:

Step	Action	Page No.	√
1	Install the software from the tape.	11	
2	Modify any ESP Workload Manager initialization parameters.	41	
3	Update the LOGON proc or CLIST with any new library names.	58	
4	Follow the instruction under "Starting ESP Workload Manager after upgrades are installed".	61	

Adding ESP Encore

Follow the steps outlined below to install ESP Encore and then configure it for your site.

Note: You must install ESP Workload Manager before or at the same time as you install ESP Encore.

Note: To use remote ESP Encore, you must have NDS (Network Delivery Services); TP Server does not work with remote ESP Encore.

What you need

To install ESP Encore, you need the following documents:

- Program Directory for ESP Encore—FMID SEN3100
- This manual

Sample library

The tape you received containing the ESP Encore software includes sample members that are added to sample library SSCPSAME. You can edit these members to fit your requirements.

To add ESP Encore to your ESP Workload Manager installation, follow these steps:

Step	Action	Page No.	√
1	Install the software from the tape.	11	
2	Allocate the EXH data set.	21	

Step	Action	Page No.	√
3	Modify any ESP Workload Manager initialization parameters.	41	
4	Update the LOGON proc or CLIST with any new library names.	58	
5	Follow the instruction under “Starting ESP Workload Manager after upgrades are installed”.	61	

Adding Agents

What you need

To add Agents, you need the following document:

- This manual

To add Agents to your ESP Workload Manager installation, follow these steps:

Step	Action	Page No.	√
1	Code Agent-related initialization parameters in the ESPPARM data set.	106	
2	Code initialization parameters in the AGENTDEF data set.	107	
3	Set up encryption for communication with Agents.	110	
4	Set up security access for using Agents	113	
5	Enable or update the Agent configuration.	116	
6	Verify that the Agents work with ESP Workload Manager.	117	

Setting up TP Server

TP Server is shipped and installed as part of CPE Version 5.1.

Note: To use remote ESP Encore, you must have NDS (Network Delivery Services); TP Server does not work with remote ESP Encore.

Follow the steps outlined below to set up TP Server.

What you need

To set up TP Server, you need the following document:

- This manual

Sample library

The tape you received containing the TP Server software also contains a sample library, SSCPSAME, with several sample members for your use. You can edit these members to fit your requirements.

To set up TP Server in your ESP Workload Manager environment, follow these steps:

Step	Action	Page No.	√
1	Allocate the TP Server checkpoint data set.	39	
2	Modify the sample ESP Workload Manager initialization parameters for VTAM LU 6.2.	149	
3	Modify the sample TP Server Initialization Parameters.	149	
4	If you are using INFOSERV, tailor TP Server parameters for INFOSERV. For more information, see the *ESP Infoserv Installation and User's Guide.*	NA	
5	Start ESP Workload Manager. The messages ESP1413 and ESP1419 should appear, indicating that TP Server has initialized its checkpoint.	61	

Setting up Workstation Server

Follow the steps outlined below to set up Workstation Server.

What you need

To set up Workstation Server, you need the following document:

- This manual

Sample library

The tape you received containing the ESP Workload Manager software also contains a sample library, SSCPSAME, with several sample members for your use. You can edit these members to fit your requirements.

Workstation Server

You must set up Workstation Server on every ESP Workload Manager master and proxy group to which you want to connect ESP Workstation.

To set up Workstation Server in your ESP Workload Manager environment, follow these steps:

Step	Action	Page No.	√
1	Define the initialization parameters for Workstation Server.	96	
2	Define the Workstation Server started task.	97	

Consolidated Workstation Server

You can set up one Workstation Server as a Consolidated Workstation Server to allow one ESP Workstation client to access multiple ESP Workload Manager master and proxy groups. The Consolidated Workstation Server will allow an ESP Workstation client to have one consolidated view of the workload running on all the ESP Workload Manager groups that it is connected to.

To set up a Consolidated Workstation Server in your ESP Workload Manager environment, follow these steps:

Step	Action	Page No.	√
1	Define the initialization parameters for Consolidated Workstation Server.	99	
2	Update the Workstation Server started task.	99	
3	Set the Workstation Server authority.	100	

Installing Application Programming Interface (API)

You can install the API that is included on the ESP Workload Manager product tapes while you install ESP Workload Manager or at a later time.

A user-written application program can use the API to communicate directly with the ESP Workload Manager subsystem.

What you need

To install the API, you need the following documents:

- Program Directory for CPE—FMID SCP5100
- Program Directory for the API—FMID SCP3200
- This manual

Sample library

The tape you received containing the API software also contains a sample library, SSCPSAME, with several sample members for your use. You can edit these members to fit your requirements.

To add the API to your ESP Workload Manager installation, follow this step:

Step	Action	Page No.	√
1	Install the software from the tape.	11	

Installing the Software From the Tape

The ESP products must be installed (received, applied, and accepted) using SMP/E. Follow the instructions in the program directories to install the required software.

Note: You must install the CPE first or at the same time as the ESP products because

- ESP Workload Manager requires CPE.
- ESP Encore, HAO, and Service Governor require ESP Workload Manager.

Interfacing with AllFusion CA-Librarian

To form a CA-Librarian interface

1. Link-edit module CYBXIl00 into an appropriate library (the ESP Workload Manager load library or a linklist library).

   The CPE samples library includes Assembly Language source statements for CYBXIL00. When the SMP/E ACCEPT process for the CPE product is complete, the DD name for the PDS library is SSCPSAME. Index member names end with $$.

2. Find member CYBXXS$$, which lists the following members used:

   - CYBXIL00 (mentioned above)
   - CYBXXS01 (a sample USERMOD to SMP/E install the interface)
   - CYBXXS10 (macros and DSECTs needed to assemble the interface modules)
   - CYBXXS11 (a sample IBM High Level Assembler job stream for assembling interface modules)

   Note: The Librarian FAIRDS DSECT is used during the assembly, so the data set with this DSECT must be concatenated to the SYSLIB DD statement.

3. Link-edit the newly assembled CYBXIL00.
4. Use member CYBXXS01 from the SSCPSAME library to install the interface.

5. Edit the sample job stream to meet your installation standards and then run it to install the interface as a USERMOD.

   The sample job stream executes SMP/E to RECEIVE and then APPLY CHECK the USERMOD.

6. After successfully completing step 5, run the sample job stream to APPLY the USERMOD.

   To install a load module, CYBXILP00, in your load library, the sample job stream uses your assembled object module, CYBXIL00, and the FAIR routines from the Librarian load library specified on the LIBRLIB DD statement.

   Note: CYBXIL00 is link-edited with AMODE(24) RMODE(24). Should you install a new release of CA-Librarian, CYBXIL00 must be re-assembled and then re-linked. An SMP/E APPLY REDO of the USERMOD will suffice if you use the same data set name for the Librarian load module library. Otherwise, an SMP/E RESTORE and REJECT of the USERMOD will remove the load module CYBXIL00. Then you can modify the USERMOD as required (remember to keep a copy around to edit) and do the SMP/E RECEIVE and APPLY again to create the new version.

2

Allocating Data Sets

You need to allocate several data sets that ESP Workload Manager uses in its day-to-day operation. This chapter contains the following topics:

- Sample Batch Job to Allocate Data Sets
- Data Set Summary
- Sizing Data Sets
- Application Data Set (APPLFILE)
- Checkpoint Data Set (CKPT)
- Agent Communications Data Set (COMMQ)
- Event Data Set (EVENTSET)
- Execution History (EXH) Data Set
- History Data Set (HISTFILE)
- Index Data Set (INDEX)
- Jobindex Data Set (JOBINDEX)
- Jobstats Data Set (JOBSTATS)
- Netqueue Data Set(NETQUEUE)
- Queue Data Set (QUEUE)
- Resource Data Set (RESFILE)
- Schedule Data Set (SCHDFILE)
- Tracking Data Set (TRAKFILE)
- User Definition Data Set (USERDEF)
- TP Checkpoint Data Set
- VSAM Data Set Considerations

Sample Batch Job to Allocate Data Sets

SSCPSAME

In the sample library *prefix*.SSCPSAME (*prefix* represents the high-level data set qualifiers you assign) that was shipped with ESP Workload Manager there is a sample batch job you can edit and run to allocate the ESP Workload Manager data sets. The batch job is in a member called CYBESS01.

To allocate the data sets

1. To allocate the ESP Workload Manager data sets to meet your site requirements, edit CYBESS01 as noted in the JCL comments.
   - See "Data Set Summary" on page 14 for descriptions of the ESP Workload Manager data sets.
   - See "ESP Workload Manager DASD requirements" on page 16 to estimate your space requirements.
   - If you are using MIM or GRS, the ESP Workload Manager data set placement does not matter. In fact, it might be advantageous to distribute the data sets across DASD packs. However, if you are using RESERVE/RELEASE serialization, ensure all ESP Workload Manager data sets are on the same pack.
2. To allocate all data sets except for the EXH data set, run CYBESS01.
3. To allocate the EXH data set, run CYBRMALC.

Data Set Summary

Data Set	Contains
APPLFILE	Data about each Application generation
CKPT	Information relating to, for example, schedule times, Event and history data set information, CLASS activities, automatic data set backup schedule, and tracking information
COMMQ	A VSAM linear data set used to store information about Agent communications
EVENTSET	One or more VSAM KSDS containing Event definitions. The data set suffix includes a unique number, *n*. The first Event data set has suffix EVENT1, the second, EVENT2, and so on.
EXH	ESP Encore job execution history **Note:** The CYBRMALC utility allocates this data set.

Data Set	Contains
HISTFILE	One or more VSAM KSDS used to store historical data
INDEX	Information relating to data set triggering, calendars, global variable tables, passwords, and encryption keys
JOBINDEX	An index to the most recent jobs and Applications ESP Workload Manager tracked
JOBSTATS	A VSAM KSDS that contains job statistics, job-profiling information, and the number of instances of each record type to keep
NETQUEUE	A VSAM linear data set used to support Network Delivery Service (NDS) tracking services
QUEUE	Job-tracking data. Also a vehicle for communication between multiple ESP Workload Manager subsystems. Up to 32 ESP Workload Manager subsystems can share one QUEUE data set.
RESFILE	A VSAM linear data set used to store information about ESP resources, implicit and explicit
SCHDFILE	A VSAM data set that provides information to the Consolidated Status Facility (CSF) about future, scheduled workload **Note:** To allocate SCHDFILE, uncomment it in CYBESS01.
TRAKFILE	Short-term job-tracking data about jobs ESP Workload Manager tracks
TP.CKPT	TP Server tracking information
USERDEF	Information about each user allowed to access the system

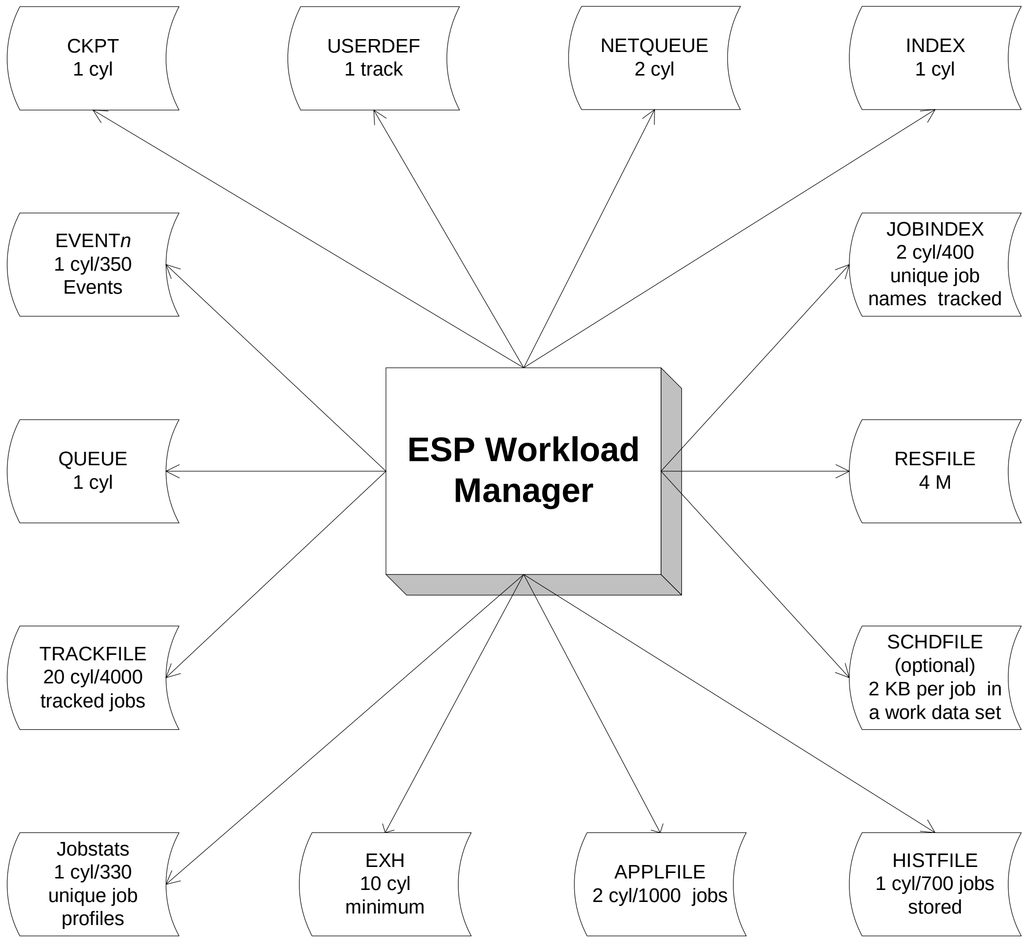
Sizing Data Sets

When allocating the data sets for ESP Workload Manager to use, consider the size of the data set required and those criteria that might require you to allocate more space to a data set.

The following sections of this chapter help you determine the size of each data set.

ESP Workload Manager DASD requirements

The following diagram outlines the DASD requirements for ESP Workload Manager. The basic recommended size is under each data set in the diagram. You must adapt each recommendation to the particular circumstances of your site using the information given in the following sections.



Application Data Set (APPLFILE)

The APPLFILE is needed when ESP Workload Manager Applications are used. This data set might be shared among multiple copies of ESP Workload Manager executing on different members of a MAS complex. Determine the name that best suits your installation standards.

CSF and Application Monitor use information from APPLFILE to help you monitor and control workload.

Space requirements

The APPLFILE is a non-VSAM data set, internally formatted into 4K (default) or larger slots that are managed on a wrap-around basis. The data set must be allocated contiguous space. The more space allocated to the APPLFILE, the longer individual Application information is retained before a slot needs to be reused. The size of the APPLFILE varies depending on

- The number of Applications you run.
- The number of jobs in each Application.
- The number of job relationships specified.
- The number of resources specified.
- The use of ESP Procedure caching.

Distributed workload specifications require more space than z/OS job specifications. We recommend that you allocate at least 40 cylinders. If you use very large Applications, you probably require more space. Use the following information to estimate your needs.

Each logical record (an Application Tracking Record [ATR]) consists of one or more complete slots. The slots are not necessarily contiguous. The space required for a single record can be estimated as follows:

- Each ATR records the information for a single generation of a specific Application. Use this formula to calculate its size:

  352 bytes + 400 bytes per workload object + 24 bytes per related job pair + 24 bytes per subApplication.

- Additional information is stored in an ATR, but much of it is difficult to estimate because its length varies. Additional information includes

  - Data set names for procedure libraries.
  - JCL libraries.
  - Documentation libraries.
  - Additional data required for distributed workload objects, such as an Agent name.

- Add 10% to 25% to the calculated size to get a final estimate that is reasonable in most cases. Use the larger figure for Applications with significant distributed content.

- Add an additional amount to allow for wasted space in each record's final slot, which averages to half a slot. The default slot size for the APPLFILE is 4096 bytes, but you can increase it to 16384 bytes maximum in 4096 byte increments.

ESP Procedure caching

If you are planning to use ESP Procedure caching or the TEMPPROC Event definition command, you should allocate additional space to the APPLFILE. A cached copy of an ESP Procedure is stored in the Application data set in the Procedure

Caching Record (PCR). One PCR in a 4K slot APPLFILE can keep approximately 50 lines of a cached ESP Procedure.

The PCR allows you to keep an original cached copy of an ESP Procedure after restarting ESP Workload Manager. The Application Tracking Record (ATR) refers to the PCR.

Rerunning jobs in completed Applications

To rerun jobs in a completed Application, the Application Tracking Record (ATR) in the APPLFILE data set must be intact. If any of the slots of the Application's ATR have been reused by a new Application, the rerun request will fail with an error message.

If you are planning to rerun jobs in completed Applications, consider allocating additional space to the APPLFILE. However, keep in mind that the following considerations also affect whether jobs in a completed Application can be rerun:

- Pattern of Application completions
- Number and size of the Applications created after an Application completed
- How long users wait before rerunning a completed Application

  The older the ATR, the likelier that its slots will get reused. When an Application is complete, the ATR's slots are released, making them available for new Applications.
- Fragmentation of the APPLFILE

  The more fragmented the APPLFILE, the likelier that released slots of a recently completed Application will get reassigned to a new Application. Fragmentation occurs when new Applications are allocated non-contiguous slots because of slot reuse over time.

Note: Jobs in a completed Application cannot be rerun if the Application-descriptor-index entry (ADXE) for the specified generation of the Application is no longer available in the JOBINDEX data set.

Initialization Parameter

Code the APPLFILE initialization parameter to specify your Application data set. For information about the initialization parameter, see "APPLFILE: Define Application data set" on page 252.

Backup facilities

There are no specific backup facilities for this data set.

Checkpoint Data Set (CKPT)

A unique checkpoint data set is required for each ESP Workload Manager subsystem. It stores, for example, scheduled times, Event and history data set information, CLASS activities, automatic data set backup schedule, and tracking information.

The checkpoint data set is reformatted on a cold start.

You must specify checkpointed information in the cold start initialization parameters or enter the commands after each cold start.

Space requirements

The data set must reside on a device whose control unit supports the Define Extent and Locate Record channel commands. All 3990 and later class-control units satisfy this requirement, but only some models of 3880 class controllers do (3880 controllers cannot attach 3390 drives).

You should initially define the checkpoint data set as a single cylinder. If one cylinder is insufficient (for example, if you experience insufficient space errors), define a bigger checkpoint data set. There is no limit to the checkpoint data set's size, except that two buffers will exist in virtual storage; therefore, virtual memory must be available for the buffers.

The checkpoint data set must be allocated as one contiguous extent. ESP Workload Manager cannot process multiple extents.

Initialization parameter

Code the CKPT initialization parameter to specify your checkpoint data set. For information about the initialization parameter, see "CKPT: Specify checkpoint data set" on page 267.

Agent Communications Data Set (COMMQ)

The COMMQ data set is a VSAM linear data set used to store information about ESP Agent communications. If your installation does not use ESP Agents, then you do not need to define this data set.

Description

ESPCOM stores its status information in the COMMQ data set. The COMMQ data set also contains ESP Agent command information.

Space requirements

We recommend a primary allocation of 1 cylinder.

Initialization parameter

Code the COMMQ initialization parameter to specify the name of the checkpoint data set that ESPCOM uses. For information about the initialization parameter, see "COMMQ: Define ESPCOM checkpoint data set" on page 270.

Backup facilities

The ESPCOM checkpoint data set is too volatile for a backup to be of use. If the data set is lost or destroyed, define a new one, if required, and then restart ESP Workload Manager with PARM=CQFORMAT. Unprocessed and unsent messages will be lost.

Event Data Set (EVENTSET)

ESP Workload Manager Events are maintained in an Event data set. One Event data set is required, and is usually sufficient, but you might need additional ones. After reviewing the *ESP Workload Manager System Programmer's Guide*, determine if multiple Event data sets are required to support the users at your site.

The Event data set is a VSAM data set (KSDS).

Space requirements

Events can vary in size depending on their complexity. An allocation of one cylinder accommodates roughly 350 Events.

This data set contains one record for each Event defined. The records consist of a fixed portion and a variable portion, depending on the actual Event definition. The main items in the variable portion are

- Schedule definitions (including NOSCHED definitions)
- Action items (for example, procedure invocations, ESP or z/OS commands)
- Comments

Estimate the size as follows:

> 252 bytes + 120 bytes per schedule definition + (8 + text length) bytes per action + (6 + text length) bytes per comment.

Initialization parameter

Code the EVENTSET initialization parameter to specify your Event data set. For information about the initialization parameter, see “EVENTSET: Identify Event data set” on page 343.

Backup facilities

ESP Workload Manager can internally back up and recover the Event data set. The backup data set is a cataloged, non-VSAM data set given a primary space allocation that is roughly 50% of the size of the corresponding Event definition data set and a secondary allocation of 10%.

You do not need to specify DCB attributes for this data set. Backups are automatically taken at the scheduled Event data set scan time (SCANTIME), as coded in your initialization parameter cold member.

Execution History (EXH) Data Set

You require the EXH data set only if you use ESP Encore. This slotted data set is the repository for information about jobs ESP Encore tracked for restart and rerun purposes.

To allocate and format the ESP Encore EXH data set, run CYBRMALC from the CPE Sample Library (*prefix.*SSCPSAME).

- Specify the KEEP count, which is the maximum number of occurrences of each job group to be kept on the EXH data set.
- Specify the size of the index or use the default (40). The index points to the job data records.
- Change @EXHFILE to the name of your EXH data set.
- Replace the @NUMCYLS by the number of cylinders to be allocated.
- Change @LOADLIB to the name of your SSCPLINK library.

Space requirements

There are two approaches to determining the size of the EXH data set: testing and calculation.

Testing to determine the EXH data set’s size

The testing approach involves creating an EXH data set with an arbitrary amount of space. After using the data set during a test period, you create a new EXH data set based on how much space your jobs require.

To create a new EXH data set

1. Allocate an EXH data set with 200 cylinders of space and run utility CYBRMALC to initialize the EXH data set with 200 index slots.
2. During a test period, run utility CYBRMANA to track EXH data set usage.
3. When you have determined the size you need for your EXH data set, allocate a new EXH data set with the correct size and run utility CYBRMALC to initialize the EXH data set with the correct number of index slots.
4. Run utility CYBRMCPY to copy the old EXH data set allocated in step 1 to the new EXH data set allocated in step 3.

Important: Keep the old EXH data set until the new EXH data set is successfully activated. Cybermation support may need the old and the new EXH data set to help you solve any problems.

Calculating the EXH data set's size

Estimating the size of your EXH data set is not an exact science because the size of jobs (number of steps, DD statements, data set names, and others) differs. Here are some guidelines:

- The EXH data set is a slotted file, formatted into 4K slots.
- The file contains three to five slots for system information, followed by a certain number of index slots as specified by the user at allocation time.
- The index is followed by the job data records. Every job data record uses at least one slot and for very large jobs it will use more than one slot.

Sample EXH data set calculation

Given 1000 unique job names run a day, of which 800 are average (one slot) jobs and 200 are very large jobs (10 slots), the KEEP count is 4.

Sample size of index calculation:

The first instance of a unique job name takes 48 bytes in the index. Subsequent instances take 32 bytes.

There are 3960 usable bytes per index slot.

Based on a scenario of 1000 unique job names active within your organization:

48 bytes x 1000 jobs = 48,000

48,000 / 3960 = 13 slots

Add a safety factor of approximately 50% giving an index count of 20.

Calculation of the EXH data set size

Index record = (4K x 20) +

Job data records for storing average jobs = 4(4K x 1 x 800) +

Job data records for storing very large jobs on the
EXH data set = 4(4k x 10 x 200)

= 44,880k

To convert to number of cylinders

48k per track (44,880/48 = 935 tracks)

1 cylinder = 15trks (935/15) = 62 cylinders

Note: The space per track and the number of tracks per cylinder can be different depending on the type of DASD you use.

Initialization parameter

Code the EXHFILE initialization parameter to specify your EXH data set. For information about the initialization parameter, see "EXHFILE: Identify Execution History data set" on page 345.

History Data Set (HISTFILE)

ESP Workload Manager job history data sets retain job statistics on a long-term basis, as opposed to the tracking data set that stores a more detailed record of job processing for shorter periods. Usually one history data set is sufficient. Job tracking definitions specify where data set job history data should be written.

HISTFILE is a VSAM data set (KSDS).

Space requirements

Space allocation depends on the number of job executions the history data set is expected to maintain. The job history data set contains one record for each individual job instance an ESP subsystem tracks. These records are 400 bytes, fixed length. One cylinder accommodates roughly 700 jobs. For example, if you run 1000 jobs per day and want to keep the history data for 30 days, you require at least 45 cylinders. Alter this to fit your requirements. Historical data might also be archived to tape or deleted at regular intervals to free up space.

Initialization Parameter

Code the HISTFILE initialization parameter to specify your history data set. For information about the initialization parameter, see "HISTFILE: Define or alter a job history data set" on page 359.

Backup facilities

ESP Workload Manager can internally back up and recover history data sets. The backup data set is a cataloged, non-VSAM data set given a primary space allocation that is roughly 50% of the size of the corresponding history data set and a secondary allocation of 10%. You do not need to specify DCB attributes for this data set.

Backups are automatically taken at a scheduled time specified in the BACKUPTIME keyword on the HISTFILE initialization parameter. Enter an ESP Workload Manager schedule statement (like 6:00 am DAILY) if you want automatic history data set backup.

Index Data Set (INDEX)

The index data set contains information about

- Calendars
- Global variable tables
- Password entries
- Cryptkey entries
- Symbolic variables in SYMLIB
- Data set triggers

INDEX is a VSAM data set (KSDS).

Space requirements

Use the following table to calculate the space required for the index data set. A one-cylinder allocation should be sufficient for most sites using a small number of global variable tables and symbolic variables in SYMLIB.

Data element	Size for each data element
Calendar	112 bytes + 32 bytes per holiday or special day
Global-variable table	Consists of a 160-byte header that describes the table and individual items Each item is composed of • Variable, composed of • A 20-byte name element (NEL) • A one-byte name length followed by the item name • An attribute element (AEL). If the length of the variable current value is four bytes or less, the AEL is 24 bytes long; if the current value length is greater than four bytes, the AEL size is 24 bytes plus the length of the current value, rounded up to next multiple of eight. • Trigger target, which has a trigger element (TEL). A TEL refers to either an Event trigger or a WOB trigger. • The TEL for an Event trigger is 60 bytes • The TEL for an WOB trigger is 44 bytes • Variable and trigger combination, which has a 16-byte trigger link element (TLL)
Password	Approximately 200 bytes per password

Data element	Size for each data element
Cryptkey	Approximately 100 bytes per cryptkey
Symbolic variable in SYMLIB	• Zero or more 120-byte symbol table records (STR) per Event (one per SYMLIB definition) • One or more 60-byte symbol library records (SLIB) per STR
Data set trigger	For data set trigger records, the key consists of the first eight characters of the data set name. A single index record indexes all data sets having the same first eight characters. The record consists of a fixed-header portion followed by a variable portion for each of the data sets with the same first eight characters. If there are more data sets than will fit into a single VSAM record, continuation index records are created, up to 255 index records for an eight-byte name prefix. *To estimate data set trigger space requirements* 1. Assume one index record for each unique eight-byte prefix. 2. Add the requisite number of index record headers for any prefixes including enough data set names to require extension records. You need to know local data set naming conventions and volume to produce a reasonably accurate estimate. The fixed portion of each index data set trigger record requires 84 bytes. You must add a variable portion for each additional data set represented by that eight-byte name prefix. The length of this variable portion can vary significantly, depending on the type of data set trigger (Event or Application level, FTP, explicit or other) and the options specified (for example, SAF and FTP user ID requested, FTP host name requested, count specified). The maximum possible size for the variable portion is 217 bytes, which is needed when the host TCP/IP name is requested for an FTP trigger; it will occupy up to 101 bytes. The best approach is to estimate sizes for FTP and non-FTP triggers separately, taking an average size for each kind.

Initialization Parameter

Code the INDEX initialization parameter to specify your index data set. For information about the initialization parameter, see “INDEX: Identify index data set” on page 362.

Backup facilities

ESP Workload Manager backs up and recovers the index data set by using the BKUPINDX command or initialization parameter. The backup data set is a cataloged, non-VSAM data set given a primary space allocation that is roughly 50% of the size of the index data set and a secondary allocation of 10%. DCB attributes do not need to be specified for this data set.

Jobindex Data Set (JOBINDEX)

The jobindex is a VSAM data set (KSDS) that contains

- Tracking definitions and an index to the most recent jobs and Application generations being ESP Workload Manager is tracking
- Pointers to the TRAFKILE and APPLFILE

CSF and Application Monitor uses information from JOBINDEX to help you monitor and control workload.

Space requirements

Space allocation depends on the number of unique job names and associated ancestor jobs that ESP Workload Manager tracks, and the number of pnodes and tracking model definitions.

You should allocate one cylinder plus an additional cylinder for every 400 unique job names that ESP Workload Manager tracks. On average, an installation that runs 10,000 unique jobs daily requires 26 cylinders.

The important records in this data set are the index records for the jobs and the Applications being tracked. There are a few other items in the data set, their size being negligible compared to the job and Application data. Both job and Application index records consist of a fixed portion, plus an additional portion for each job execution instance or Application generation (as defined by the INDEX option in the DEFTM command and the APPL statements).

Job index (J) record

136 bytes + 136 bytes per job execution instance

Application index (B) record

132 bytes + 124 bytes per Application generation

Initialization parameter

Code the JOBINDEX initialization parameter to specify your jobindex data set. For information about the initialization parameter, see “JOBINDEX: Jobindex data set” on page 377.

Backup facilities

ESP Workload Manager backs up and recovers the jobindex data set by using the BKUPJNDX command or initialization parameter. The backup data set is a cataloged, non-VSAM data set given a primary space allocation that is roughly 50% of the size of the jobindex data set and a secondary allocation of 10%. You do not need to specify DCB attributes for this data set.

Jobstats Data Set (JOBSTATS)

The jobstats is a VSAM data set (KSDS) that contains

- Job statistics
- Job-profiling information
- Number of instances of each record type to keep

Space requirements

Space allocation depends on the number of unique job names that start from ESP Workload Manager Applications, and the number of job profiles that are active.

You should allocate one cylinder plus an additional cylinder for every 330 unique job names or 330 unique job profile names. On average, an installation that runs 10,000 unique ESP Workload Manager jobs daily requires 30 cylinders.

Both job and profiling records consist of a fixed portion, plus an additional portion for each job execution instance.

Job Statistics (J) record

216 bytes + 160 bytes per job execution instance

Job Profile (P) record

216 bytes + 160 bytes per job execution instance

Initialization parameter

Code the JOBSTATS initialization parameter to specify your jobstats data set. For information, see “JOBSTATS: Jobstats data set” on page 382.

Backup facilities

ESP Workload Manager backs up and recovers the jobstats data set by using the BKUPJSTS command or initialization parameter. The back-up data set is a cataloged, non-VSAM data set given a primary space allocation that is roughly 50% of the size of the jobstats data set, and has a secondary allocation of 10%. You do not need to specify DCB attributes for this data set.

Netqueue Data Set(NETQUEUE)

The NETQUEUE data set is a VSAM LINEAR data set used to store job-tracking records when a Network Delivery Service (NDS) tracking service internodal transmit or receive request is made. ESP Workload Manager stores job-tracking records regardless of whether a connection between the local and remote ESP Workload Manager subsystems currently exists.

When the ESP Workload Manager subsystem on the remote node confirms receipt of an internodal job-tracking record, the transmit record is purged from the NETQUEUE data set. A received internodal job-tracking record is purged from the NETQUEUE data set after the local ESP Workload Manager subsystem has written the record to its checkpoint data set.

Space requirements

The amount of space specified in the example is two cylinders, providing approximately one Megabyte of 3380-E storage. The secondary allocation of one cylinder allows for expansion if required.

Queue Data Set (QUEUE)

The ESP Workload Manager QUEUE data set contains job-tracking data to preserve the current state of an ESP Workload Manager subsystem and allow data to pass between a master subsystem and its associated proxy subsystems. DCB attributes are not required for this data set because ESP Workload Manager sets them internally. QUEUE is a non-VSAM data set.

The QUEUE data set has a record of Abended Job Information (available with the DAB command) and information on jobs in the Input, Execute, and Output pnodes.

Note: The QUEUE data set is always required, even if you are using XCF tracking and data set triggering services.

Space requirements

Initially, you should define the QUEUE data set as a single cylinder. If one cylinder is insufficient (for example, if you experience insufficient space errors), define a bigger QUEUE data set. If you require a larger data set, the following rules apply:

- The data set must reside on a device whose control unit supports the Define Extent and Locate Record channel commands. All 3390 and later class-control units satisfy this requirement, but only some models of 3880 class controllers do (3880 controllers cannot attach 3390 drives).
- If the data set is shared, the maximum usable size is limited to approximately 2.4 M. The exact limit is calculated during formatting, and the lesser of the limit and the allocated space is formatted and used.

Initialization Parameter

Code the QUEUE initialization parameter to specify your QUEUE data set. For information about the initialization parameter, see “QUEUE: Specify QUEUE data set” on page 451.

Resource Data Set (RESFILE)

The resource data set is a VSAM linear data set used to store information about resources. The resource data set contains the following classes of information:

- System topology
- Resource definition, both explicit and implicit
- Jobs

Space requirements

We recommend that you specify a minimum of 4 M.

To calculate the size for a resource data set, you need to know

- The numbers and kinds of resources to be managed.
- The maximum number of jobs that the resource manager is expected to manage at any point in time.
- The job’s resource requirement.s

Note: When the resource data set runs out of space, ESP Workload Manager automatically tries to expand the file up to 4 MB per session. The expansion will succeed only if the VSAM data set was allocated with sufficient amount of space; therefore, we recommend that you specify secondary extent size during the file allocation.

Control blocks

The resource data set contains a number of control blocks. The sizes, in bytes, of each of these control blocks are as follows:

Control Block	Size (Bytes)	
Data set header block (ResFHB) 1 in total	312	+ 4 × maxRES + 4 × maxCPU
Node descriptor (ResFND) 1 per node	152	
CPU descriptor (ResFCD) 1 per CPU	168	
Resource descriptor (ResFRD) 1 per resource	120	+ 4 × maxCPU
Resource counter (ResFRC) • 1 for each ENTERPRISE resource • 1 per node for each GLOBAL resource • 1 per CPU for each LOCAL resource	72	
Job descriptor (ResFJD) 1 per job being managed *	160	+ TDR + parameters
Requested resource (ResFRR) 1 per resource per job *	48	
Device list (ResFDL) 1 per resource counter for real resources	48	+ 8 × number of specified devices
Resource UCB list (ResMUE) 1 per UCB per FDL	8	
Enqueue descriptor (ResFED)	66	+ average name length × number of unique defined names
Enqueue request (ResFER)	64	× number of active jobs with enqueue requests × average number of requests per job

* The ResFJD and ResFRR vary dynamically as the system is executing.

There is one ResFED for every unique name for which at least one ENQUEUE request exists—including both those explicitly defined by an ENQUEUE statement and those implicitly defined by NOTWITH statements and the ENQSELF command or initialization parameter. The length of the control blocks varies, consisting of a fixed 66-byte header plus the name. The maximum length of the name is 132 bytes. Names generated by the NOTWITH/ENQSELF feature have a defined format discussed in the "Specifying mutually exclusive run requirements" section of the *ESP Workload Manager User's Guide*. The names resulting from explicit ENQUEUE statements can be any length with a maximum of 132 bytes.

There is one ResFER for each ENQUEUE request (explicit or implicit) in any active job. The ResFER control blocks are a fixed length of 64 bytes.

In addition to these directly specified names, there are control blocks for phantom enqueue names. They are specific names that are implied by a directly specified name including wildcards. For example, a NOTWITH statement referring to a job in a given Application with no generation specified generates

- An entry for the specified generic name
- Phantom entries for currently active generations of that Application

Note: All these control blocks are dynamically created and deleted, according to the actual workload on the system. When attempting to estimate the data set size, you must realistically estimate the peak workload.

In addition to the main control blocks described above, the resource data set provides a home for more transient data, including request blocks used to pass requests to the Resource Manager and return the results. These can be safely ignored for overall data set sizing purposes.

Job descriptors

Of particular concern are the job descriptors, which, in addition to the basic job data, contain

- The TDR used for scheduling the job
- The Event parameters
- Other optional data items

The TDR itself is 104 bytes long; the other items vary in length. A regularly scheduled Event has 24 bytes of basic Event parameters plus

- 10 to 18 bytes of triggering information if the Event was manually triggered
- From 18 bytes up to over 100 bytes if the Event was activated by a data set trigger
- Space for the USER1 to USER4 options if they are specified
- Optionally, space for
  - An inforec number
  - Application tag and resubmission data
- Any symbolic variables associated with the job

Initialization Parameter

Code the RESFILE initialization parameter to specify your resource data set. For information about the initialization parameter, see “RESFILE: Specify the resource data set” on page 468.

Examples of RESFILE size calculation

Example 1

Assumptions

- 200 resources are to be managed:
  - 3 have ENTERPRISE scope
  - 47 have GLOBAL scope
  - 150 have LOCAL scope with three of these local resources being real resources
- The maximum number of jobs being managed concurrently by the resource manager is 1500.
- Each job being managed requests, on average, four resources.
- All the jobs are of the regularly scheduled variety, with
  - No USERn parameters
  - No inforec data
  - No tag or symbolic variables
- There are 10 CPUs, split among four nodes, in the installation.
- The values of maxCPU and maxRES are set to 15 and 250 to allow for expansion.
- The average number of devices per real resource is two.
- The average number of UCBs per device list is 16.

Calculation

This setup would require an estimated resource data set size of

$312 + (4 \times 250) + (4 \times 15) + (152 \times 4) + (168 \times 10)$

$+ ((120 + (4 \times 15)) \times 200) + (72 \times \text{ctrs})$

$+ ((160 + 104 + 24) \times 1500) + (48 \times 4 \times 1500) + ((48 + (8 \times 2)) \times \text{rctr})$

$+ (8 \times 16 \times \text{rctr})$

$= (3660 + 157752 + 721920 + 3840)$

$= 887172$ bytes

where ctrs = $(3 + (4 \times 47) + (10 \times 150)) = 1691$ and is the number of resource counters, and rctr = $(10 \times 3) = 30$ and is the number of real resource counters.

Calculation explained

Control Block	Assumption	Value	Calculation	Total
Data set header	maxRes	**250**	312 + (4 × **250**)	1312
	maxCPU	**15**	(4 × **15**)	60
Node descriptor	nodes	4	(152 × 4)	608
CPU descriptor	CPU	**10**	(168 × **10**)	1680
Resource descriptor	maxCPU	**15**	((120 + (4 × **15**)) × **200**)	36000
	Resources	**200**		
Resource counter	ENTERPRISE	**3**	(3 +	121752
	GLOBAL	47	(4 × 47) +	
	nodes	4		
	LOCAL	**150**	(**10** × **150**))	
	CPU	**10**		
			× 72)	
Job descriptor	maximum number of jobs	**1500**	((160 + 104 + 24) × **1500**)	432000
Requested resource	maximum number of jobs	**1500**	(48 × **4** × **1500**)	288000
	average resource requested	4		
Device list	average device per real resource	**2**	((48 + (8 × **2**) ×	1920
	real resource	3	(3 × 10))	
Resource UCB	average number of UCB per device	**16**	(8 × **16** ×	3840
	real resource	3	(3 × 10))	
Total				887172

This estimate should be increased by a factor of at least 50%, to allow for

- Transient data
- Unexpected peaks in activity
- Fragmentation
- Natural expansion

A size of 1.3 MB would be a reasonable starting point for this example.

Increase the resource data set if a significant number of jobs are expected

- To be triggered manually
- To be triggered via DSTRIG logic
- To have associated symbolic variables

This example demonstrates that a reasonably accurate size estimation for a resource data set depends on a good understanding of the environment to be managed. In the absence of detailed information, you must err on the safe side. It is better to have some wasted space in the data set than to have ESP Workload Manager terminate because the resource manager cannot obtain space to store a new entity.

Example 2

Assumptions

- 9000 resources are to be managed, all of them global, with no real resources.
- The maximum number of jobs being managed concurrently by the resource manager is 2000.
- Each job being managed requests, on average, 20 resources.
- All the jobs are of the regularly scheduled variety, with
  - No USERn parameters
  - No inforec data
  - No tag or symbolic variables
- A single node with two CPUs.
- The values of maxCPU and maxRES are set to 4 and 10000 to allow for expansion.

Calculations

This setup would require an estimated resource data set size of

$312 + (4 \times 10000) + (4 \times 4) + 152 + (168 \times 2)$

$+ ((120 + (4 \times 4)) \times 9000) + (72 \times 9000)$

$+ ((160 + 104 + 24) \times 2000) + (48 \times 20 \times 2000)$

$= (40816 + 1872000 + 2496000)$

$= 4408816$ bytes

Calculation explained

Control Block	Assumption	Value	Calculation	Total
Data set header	maxRes	**10000**	312 + (4 × **10000**)	40312
	maxCPU	4	(4 × 4)	16
Node descriptor	nodes	**1**	(152 × **1**)	152
CPU descriptor	CPU	**2**	(168 × **2**)	336
Resource descriptor	maxCPU	4	((120 + (4 × 4)) × **9000**)	1224000
	Resources	**9000**		

Control Block	Assumption	Value	Calculation	Total
Resource counter	ENTERPRISE	**0**	(0 +	
	GLOBAL	**9000**		
	nodes	1	(**1** × **9000**) +	
	LOCAL	**0**		
	CPU	**2**	(**2** × **0**))	
			× 72)	648000
Job descriptor	maximum number of jobs	**2000**	((160 + 104 + 24) × **2000**)	576000
Requested resource	maximum number of jobs	**2000**	(48 × **20** × **2000**)	1920000
	average resource requested	**20**		
Device list	average device per real resource	**0**	((48 + (8 × **0**) ×	0
	real resource	**0**	(0 × 10))	
Resource UCB	average number of UCB per device	**0**	(8 × **0** ×	0
	real resource	**0**	(0 × 10))	
Total				4408816

In this case, applying an adjustment of approximately 50% gives a reasonable initial estimate of 7 MB for this data set.

Comments on the examples

The most significant factors affecting the overall size of the data set are

- The total number of defined resources
- The peak number of jobs being managed concurrently by the resource manager
- The number of resource requests made by each of these jobs

The first of these factors is the major influence on the base (static) data set size. This is estimated in the first two lines of the calculations, especially the second line. The other two factors are the main determining components of the dynamically varying portion of the data set size. This is estimated mainly in line 3 (Requested resource) of the calculations.

Schedule Data Set (SCHDFILE)

The schedule data set is a VSAM data set (KSDS) that is populated by a work data set generated by a SADGEN job, providing information to the Consolidated Status Facility (CSF) about future, scheduled workload. The schedule data set is not required unless you need to use the LOADSCHF command to view future, scheduled workload from CSF.

For an example of a SADGEN job that populates the work data set, see “LOADSCHF: Load schedule data set into CSF” on page 389 or the LOADSCHF command in the *ESP Workload Manager Reference Guide*.

If the schedule data set contains a record for a complete job, the job will persist in a CSF display across restarts of ESP Workload Manager. Issue the PURGSCHF command to purge the records for completed workload objects.

Space requirements

The LOADSCHF command loads a work data set generated by a SADGEN job. The number of records in the schedule data set depends on the number of jobs generated by the SADGEN job according to its specified scheduling period.

In general, assuming an average record size per job of 2KB gives a reasonable quick estimate, unless the set of Applications generated is unusually complex.

For example, a schedule data set containing 4000 average jobs occupies approximately

4000 records x 2048 bytes = 8,192,000 bytes

which is approximately 7.8MB.

Initialization parameter

If you want to see future, scheduled workload in CSF, code the SCHDFILE initialization parameter to specify your schedule data set. For information about the initialization parameter, see “SCHDFILE: Identify schedule and work data set names” on page 480.

Tracking Data Set (TRAKFILE)

The tracking data set retains job-tracking information and might be shared among multiple instances of ESP Workload Manager executing on different members of an XCF group. TRAKFILE is a non-VSAM data set.

Space requirements

The tracking data set is internally formatted into 1 K slots that are managed on a wrap-around basis. The more space allocated to the tracking data set, the longer job tracking information is retained before a slot needs to be reused. The tracking data set must be big enough to accommodate all the active jobs at any one time. User experience has shown that 20 cylinders holds the tracking data for 4000 jobs. We recommend that you allocate at least 20 cylinders.

Each logical record (a job-tracking record (JTR)) consists of one or more complete slots. The slots are not necessarily contiguous. You can estimate the space required for a single record in each of these data sets as follows:

Each JTR records the information for a single execution of a specific job. Use this formula to calculate its size:

560 bytes + 112 bytes per step + 48 bytes per pnode.

A job will usually pass through two predefined pnodes (INPUT and EXEC). If tracking through purge is in effect, there will be another predefined pnode (OUTPUT). You must also include any user-defined post-processing pnodes in the calculation.

Initialization parameter

Code the TRAKFILE initialization parameter to specify your tracking data set. For information about the initialization parameter, see “TRAKFILE: Define tracking data set” on page 529.

Backup facilities

There are no specific backup facilities for this data set.

Upgrading the TRAKFILE for six-digit job numbers

Pre-ESP Workload Manager Version 5.4 TRAKFILEs must be upgraded to support six-digit job numbers. To upgrade your TRAKFILE with the CYBESUT4 utility, you can use one of the two sample JCL library members SSCPSAME(CYBESS36) or SSCPSAME(CYBESS37).

User Definition Data Set (USERDEF)

ESP Workload Manager user data is maintained in the user definition data set. It is a VSAM (KSDS) data set.

Space requirements

Allocate one track.

Initialization parameter

Code the USERDEF initialization parameter to specify your user definition data set. For information about the initialization parameter, see “USERDEF: Identify user definition data set” on page 533.

Backup facilities

ESP Workload Manager can internally back up and recover the user definition data set. The backup data set is a cataloged, non-VSAM data set given a primary space allocation that is roughly 50% of the size of the user definition data set and a secondary allocation of 10%. You do not need to specify DCB attributes for this data set.

TP Checkpoint Data Set

The TP checkpoint data set is a VSAM linear data set that queues transactions until they successfully pass to the target transaction handler or a partner server. Use the TP checkpoint data set to ensure transmission integrity.

The CYBESS01 member of the sample library creates the ESP data sets, including the TP checkpoint data set. The amount of space specified in the example is two cylinders. For ESP intersystem job tracking, the amount used for a long duration queue of tracking data is approximately 250 bytes per job plus 150 bytes per step for each system to which tracking information will be routed.

Note: You require a TP checkpoint data set only if you use TP Server to communicate between remote ESP Workload Manager master subsystems or between an ESP Workload Manager master subsystem and ESP Infoserv.

Initialization Parameter

Code the TPCKPT initialization parameter to specify your TP checkpoint data set. For information about the initialization parameter, see “TPCKPT: Allocate the TP Server checkpoint data set” on page 516.

VSAM Data Set Considerations

You can code the AMS (Access Method Services) control statements as a //SYSIN data set within an IDCAMS batch job, or as a TSO input stream if you are authorized.

Note: ESP Workload Manager internal processing depends on how the VSAM record keys are defined; altering the key size or offset prevents ESP Workload Manager from functioning.

REUSE

We recommend that you use of the REUSE keyword to permit reloading of the data set without having to delete and redefine it. When allocated in a non-ICF catalog, a reusable data set must be non-unique. If the reusable data set is defined in an ICF catalog, it cannot be defined as SUBALLOC.

RECSZ

The second number of the RECSZ parameter, for example, the 16300 in RECSZ(200 16300), defines the largest record size allowed in the VSAM data set. ESP Workload Manager's MAXLRECL initialization parameter must indicate the largest record in all VSAM data sets.

VSAM control interval

If the VSAM control interval size is left as the default, AMS calculates the optimum based on the record size. If you want a control interval size (CISZ) smaller than the largest record size, code the SPANNED option. This allows large records to span two or more control intervals.

VSAM share options

We recommend that all VSAM SHAREOPTIONS be set to (3,3), with the exception of four data sets, as indicated below. This takes advantage of ESP Workload Manager's data integrity functions and reduces possible interference with VSAM utility functions.

Exception

There are four linear data sets ESP Workload Manager uses that do not fall under the share options (3,3) rule. The following data sets need share options set to (2,3):

- Resource data set
- TP Server checkpoint data set
- Commq
- Netqueue

3

Setting the Initialization Parameters

ESP Workload Manager reads initialization parameters when it starts. These parameters define the ESP Workload Manager environment In this chapter, you create an initialization parameter data set, copy members to it from a sample library, and edit some of the members you copied.

This chapter contains the following topics:

- Sample Library Members Containing Initialization Parameters
- Creating and Customizing the Initialization Parameter Data Set
- Required Initialization Parameters
- Recommended Initialization Parameters
- Performance Considerations
- Specifying Default Access for Users
- MAILLIST Data Set

Sample Library Members Containing Initialization Parameters

Some members of sample library *prefix*.SSCPSAME (where *prefix* represents the data set qualifiers specific to your system) contain ESP Workload Manager initialization parameters. These parameters define your ESP Workload Manager environment. For example, they define data set names and job-tracking information.

This chapter helps you edit sample library members CYBESS03, CYBESS04, CYBESS05 and CYBESS71. Later steps of the installation procedure, under "Installing ESP Workload Manager" on page 4, refer to other chapters that help you edit the other members in the following list.

The sample library contains the following members related to ESP Workload Manager initialization parameters:

Note: Each member description is followed by the name used in the documentation. For example, CYBESS03 is called ESPPARM in the documentation.

- CYBESS03 — Startup initialization parameters (ESPPARM)

  Every time you start ESP Workload Manager, it reads the startup initialization parameters. DD statement ESPPARM in the ESP Workload Manager started task JCL references the ESP Workload Manager start initialization parameters member (see "Creating the Started Task Procedure" on page 56).

- CYBESS04 — Cold-start initialization parameters (ESPCOLD)

  Every time you cold start ESP Workload Manager, it reads the cold-start initialization parameters. DD statement ESPCOLD in the ESP Workload Manager started task JCL refers to the ESP Workload Manager cold-start initialization parameters member (see "Creating the Started Task Procedure" on page 56).

- CYBESS05 — Warm start initialization parameters (ESPWARM)

  Every time you warm start ESP Workload Manager, it reads the warm-start initialization parameters. DD statement ESPWARM in the ESP Workload Manager started task JCL refers to the ESP Workload Manager warm-start initialization parameters member (see “Creating the Started Task Procedure” on page 56).

- CYBESS06 — TP Server initialization parameters (TPPARM)

  You edit your copy of this member in “TP Server” on page 149.

- CYBESS08 — Job-Tracking Definition Table (JTDT)

  The JTDT specifies the characteristics of the jobs you want tracked, for example, job name, execution class, and so on. You edit your copy of this member in “Defining What ESP Workload Manager Tracks” on page 63.

- CYBESS09 — Initialization parameters for VTAM LU 6.2 used with TP Server

  You edit your copy of this member in “TP Server” on page 149.

  See also member CYBESS06 in this list

- CYBESS17 — User Profile Definition Table (UPDT)

  The UPDT enables a user’s access to

  - The Event data set
  - Specified history data sets
  - Specified calendars

  You edit your copy of this member in “Specifying Default Access for Users” on page 48.

- CYBESS32 — Workstation Server initialization parameters (WSSPARM)

  You edit your copy of this member in “Configuring ESP Workload Manager for ESP Workstation” on page 96.

- CYBESS33 — Remote Workstation Server initialization parameters (WSSREMOT)

  You edit your copy of this member in “Configuring ESP Workload Manager for ESP Workstation” on page 96.

- CYBESS35 — ESP Agent definition data set (AGENTDEF)

  In the AGENTDEF data set, you code initialization parameters to allow ESP Workload Manager to work with Agents. You edit your copy of this member in “Configuring ESP Workload Manager to Use Agents” on page 103.

- CYBESS38 — ESP Workload Manager LOADNET initialization data set

  You edit your copy of this member in “Network Delivery Services (NDS)” on page 147.

- CYBESS71 — ESP Workload Manager MAILLIST data set

Creating and Customizing the Initialization Parameter Data Set

1. Allocate your initialization parameter data set.
   - You can create a PDS or PDSE.
   - We recommend you specify a primary track size of 20, a secondary track size of 10, and a record length of 80.
   - Specify any BLKSIZE provided that it is a multiple of 80.
2. Copy the required members from sample library *prefix*.SSCPSAME into your initialization parameter data set (you can rename the copied members if you want).

   For a description of the sample library members, see "Sample Library Members Containing Initialization Parameters" on page 42.
3. Edit the initialization parameters in your copy of sample members CYBESS03, CYBESS04, CYBESS05. CYBESS17, and CYBESS71

   For details, see
   - "Required Initialization Parameters" on page 44.
   - "Recommended Initialization Parameters" on page 46.
   - "Performance Considerations" on page 46.
   - "MAILLIST Data Set" on page 49.
   - The initialization parameters listed in the reference section of this guide.
   - Comments in the library member you are editing.
4. Continue with the next step of the checklist in "Installing and Upgrading ESP Workload Manager" on page 3.

Required Initialization Parameters

For details on all initialization parameters, see "Installation Reference" on page 227.

Initialization Parameter	Use	Additional Information
AGENTRCV	If you use Agents	"Configuring ESP Workload Manager to Use Agents" on page 103
APPLFILE	Always	"Data Set Summary" on page 14
AUDITLOG	Always	
CKPT	Always	"Data Set Summary" on page 14
COMMQ	Always	"Data Set Summary" on page 14

Initialization Parameter	Use	Additional Information
CPU NODE RESFILE	If you use resources, enqueues or notwith	“Setting Up ESP Workload Manager Resources on Your System” on page 78
ERMSTEP	If you use ESP Encore	
ESPGROUP	If you use ESP Encore or CCCHK in a master and proxy environment	
EVENTSET	Always	“Data Set Summary” on page 14
EXHFILE	If you use ESP Encore	“Customizing ESP Encore” on page 85
INDEX	Always	“Data Set Summary” on page 14
JOBINDEX	Always	“Data Set Summary” on page 14
JOBSTATS	Always	“Data Set Summary” on page 14
LOADAGDF	If you use Agents	“Configuring ESP Workload Manager to Use Agents” on page 103
LOADJTDT SMFINT TCELL TRACKOPT	Always	“Defining What ESP Workload Manager Tracks” on page 63
LOADUPDT	Always	
QUEUE	Always	“Data Set Summary” on page 14
SAFCLASS	Always	“Defining Security” on page 51
SHADGOAL	If you use shadow manager (requires HAO)	“Shadow manager” on page 138
SUBSYS	Always	
SVC	Always	
SYSID	Always	
SYSPLEX	If you have a master and proxies	
TIMEZONE	Always	“Setting Up Time Zones” on page 73
TRAKFILE	Always	“Data Set Summary” on page 14
WOBDEF	Always	
XCF START SERVICE DSTRIG XCF START SERVICE TRACKING	• If you have, at least, a monoplex • If you have a master and proxies	“Configuring Master-Proxy Subsystems in a Sysplex” on page 131
XCF START SERVICE ROUTING XCF START SERVICE SCOREBD **Note:** Both of the above require either HAO or Service Governor	• If you need to issue ESP Workload Manager commands from a proxy • If you need to view the CSF and Application Monitor scoreboards from a proxy	“Configuring Master-Proxy Subsystems in a Sysplex” on page 131

Recommended Initialization Parameters

For details on all initialization parameters, see “Installation Reference” on page 227.

Initialization Parameter	Use	Additional Information
MINHOLD MINDORM MAXDORM	If you have master and proxies	“Performance Considerations” on page 127

Performance Considerations

Using the PREALLOC initialization parameter

If a data set is pre-allocated, ESP Workload Manager uses that data set rather than dynamically allocating one. This increases I/O efficiency when ESP Workload Manager frequently accesses that data set. For example, pre-allocating JCL libraries and COPYJCL libraries increases I/O efficiency for those libraries.

Pre-allocating data sets

- Create a PREALLOC ALLOC initialization parameter for each data set you want to pre-allocate.

You can also issue PREALLOC as an operator command.

De-allocating data sets that were pre-allocated

- Use the PREALLOC UNALLOC initialization parameter or command for each data set you want to de-allocate.

After you issue PREALLOC UNALLOC for a data set, ESP Workload Manager dynamically allocates and de-allocates the data set again. For a pre-allocated data set that is currently in use, PREALLOC UNALLOC causes the data set to be dynamically de-allocated when it is no longer needed.

Generation Data Groups (GDGs)

You cannot pre-allocate a GDG data set using the PREALLOC initialization parameter since the current generation number can change. However, you can issue the PREALLOC command when you know the full data set name. You can also use ESP Workload Manager scheduled Events to allocate or de-allocate data sets at various times instead of, or in addition to, using an initialization parameter.

You can increase the efficiency of I/O operations on ESP Workload Manager data sets by pre-allocating the appropriate data sets.

Using Event Streaming

Event initiators

ESP Workload Manager uses an Event initiator to process an ESP Workload Manager Event. ESP Workload Manager may use multiple Event initiators to allow concurrent processing of Events. The number of Event initiators allotted determines the number of Events that can be processed concurrently.

During default ESP Workload Manager operation, ESP Workload Manager uses one Event initiator class and one Event initiator. Each Event waits in a queue until its turn to process.

Specifying multiple Event initiators

You can specify multiple Event initiators, with a single class (class 0) for processing. In this case, several Event initiators might be executing Events in parallel, but only one queue of Events is waiting for execution. All Events share the same pool of Event initiators.

Event streaming allows you to create multiple streams or queues of Events by specifying multiple Event initiator classes. Each class has its own dedicated set of Event initiators.

Specifying too many Event initiators might result in increased data set contention as more Events execute simultaneously. We recommend that you specify at least three initiators. You can adjust the number of initiators as ESP Workload Manager workload increases.

Setting priorities using Event streaming

You can use Event streaming to give certain Events priority over other Events. For example, if you have an Event that must run every 10 minutes, you can't afford to have the Event wait in a queue for a free initiator. You want to assign this Event a higher priority—a separate Event initiator class.

Using the EICLASS initialization parameter

You use the EICLASS initialization parameter to create multiple Event initiator classes. The EICLASS initialization parameter specifies the class and the number of initiators assigned to that class. If you do not specify any values for EICLASS, ESP Workload Manager uses one class 0 initiator.

The following example defines 5 initiators to class 0, and defines 2 additional classes, each with 2 initiators:

```
EICLASS SET CLASS(0) MPL(5)
EICLASS SET CLASS(1) MPL(2)
EICLASS SET CLASS(2) MPL(2)
```

To assign an initiator class to an Event

Edit the Event to add an EICLASS statement to the Event.

To add an EICLASS statement to an Event

Specify the following in the Event definition:

```
EICLASS(nnn)
```

where *nnn* is a number from 0 to 255, matching one of the EICLASS setup.

Event initiator security

Host security authorization controls the use of Event initiator classes. For any non-zero class specified on an Event, a host security check occurs. The requester must have read access to the resource EVENTINITCLASS.*nnn* to use the Event initiator class. *nnn* is the three-digit requested class with leading zeros.

Event initiator processing

All Application jobs are routed through the specified Event initiator class. If an initiator is not available, and one cannot be created, the Event waits until an initiator of the required class becomes available. If an Event is triggered for which no class is defined, it is routed through class 0.

The Application Manager uses Event initiator classes for workload submission. Optionally, JOBEND and STEPEND monitors can use Event initiator classes if USERMOD 36 is active. (USERMOD 36 allows JOBEND and STEPEND monitor Events to honor non-zero Event initiator class requests.)

Specifying Default Access for Users

Setting up the User Profile Definition Table

Add one or more PROFILE initialization parameters to the User Profile Definition Table (UPDT) to enable a user's access to the following:

- A specified Event data set
- Specified history data sets
- Specified calendars

The UPDT is in your copy of sample library member CYBESS17 (see "Sample Library Members Containing Initialization Parameters" on page 42). The UPDT must contain at least one PROFILE initialization parameter that sets the default Event data set for all new Events, for example

```
PROFILE USER(-) EVENTSET(EVENT1)
```

ESP Workload Manager scans the UPDT from top to bottom until it finds a match.

Note: The LOADUPDT initialization parameter (see “Required Initialization Parameters” on page 44) loads the UPDT when ESP Workload Manager starts.

Loading the User Profile Definition Table

After defining the UPDT, load it into ESP Workload Manager.

To load the table

1. Use the LOADUPDT initialization parameter to load the table from the data set.
2. Issue the initialization parameter in the ESP Workload Manager initialization parameter data set or, when it is active, from your console.

When you have loaded the table, ESP Workload Manager refers to the table, not the data set. If any changes are to be made to the table while ESP Workload Manager is active, the table needs to be reloaded, using this command.

The following example loads member PROFILE1 of data set CYB1.ESP.INSTALL as the user profile definition table:

```
LOADUPDT CYB1.ESP.INSTALL(PROFILE1)
```

MAILLIST Data Set

Use the MAILLIST data set to define a notification list. MAILLIST includes the following initialization parameters:

- A single SMTPPARM initialization parameter that defines a Simple Mail Transfer Protocol (SMTP) server to ESP Workload Manager
- Any number of MAILBOX initialization parameters to specify the name of each mailbox
- A single, optional LOG initialization parameter for each mailbox to specify whether mailbox messages are written to the mail log
- Any number of EMAIL initialization parameters to specify an email address in mailboxes. You can specify the same email address in different mailboxes
- Any number of TSOUSER initialization parameters to include TSO user IDs in mailboxes. You can specify the same TSO user ID in different mailboxes

Note: You can use the member CYBESS71from the SSCPSAME data set as a sample to define your own MAILLIST.

If you have a complex distribution list, we recommend that you use email group addresses for each distribution point. Each group can include a number of different individual email addresses and each individual recipient can belong to any number of groups.

You must use the LOADNL initialization parameter to load the MAILLIST data set.

You can use the MAILBOX command to list how the mailboxes are defined.

You can use the mailboxes in Events and in NOTIFY statements. When you specify a mailbox in an Event or a NOTIFY statement, all the messages generated by the Event or the NOTIFY statement are sent to all email addresses and TSO users defined in that mailbox.

You can specify a mailbox in the PROFILE initialization parameter. When you specify a mailbox in a PROFILE initialization parameter, any Event using the prefix defined in this PROFILE initialization parameter uses that mailbox as a default mailbox. The default mailbox is used when the Event does not specify a mailbox.

Example of MAILLIST data set

```
SMTPPARM CLASS(A) JOBNAME(SMTP)
MAILBOX  PAYROLL MAXLINES(0)
   LOG ON
   TSOUSER CYBPAY1 SYSID(SYSC)
   EMAIL paymaster@company.com
   EMAIL payservice@payservice.com
MAILBOX CYBACCOUNTING MAXLINES(300)
   TSOUSER (CYBACC1 CYBACC2) SYSID(SYSA)
   LOG OFF
```

4

Defining Security

You use a host security product to determine the access users and groups have to ESP Workload Manager.

For detailed information on ESP Workload Manager security, see the *ESP Workload Manager Security Guide*.

This chapter contains the following topics:

ESP Workload Manager Users and Security

ESP Workload Manager security

ESP Workload Manager security deals with two areas:

- External objects — Defined outside ESP Workload Manager, such as a JCL library ESP Workload Manager uses to submit jobs
- Internal objects — Defined in ESP Workload Manager and stored in one of its data sets (for example, Events, Applications, calendars, and job-tracking models)

ESP Workload Manager supplies security to its objects through the SAF interface. To update an ESP Workload Manager data set, the user issues a request to the ESP Workload Manager command processor. The command processor passes the request to your security system, which verifies the user's authority to update the data set. For more information about ESP Workload Manager security, see the *ESP Workload Manager Security Guide*.

ESP security process for external objects

ESP Workload Manager submits jobs or accesses data sets when an ESP Event is triggered (with the exception of ESP Workload Manager own data sets). ESP Workload Manager executes the workload specified by the Event under the Event owner's user ID rather than under its own user ID. To determine the Event owner, see the *ESP Workload Manager Security Guide* for the process ESP Workload Manager to determine user ID.

Setting up Host Security

You need to enable a host security product to control access to ESP Workload Manager. ESP Workload Manager needs to know the class of the resource profiles defined for ESP Workload Manager resources. You can also set some global parameters.

When you have enabled the host security product, you then need to set some ESP Workload Manager defaults for the users.

To activate host security processing

1. Identify a host security resource class (SAFCLASS parameter).
2. Specify host security processing options (SAFOPTS parameter).
3. Define and load a user profile definition table (LOADUPDT parameter).
4. Code the RACROUTE initialization parameter with the ON operand to enable ESP Workload Manager calls to the system security product.

Identifying the Host Security Resource Class

When ESP Workload Manager initializes, it checks for the SAFCLASS initialization parameter. This parameter indicates that your security product is used for ESP Workload Manager objects.

Note: If you installed ESP Workload Manager prior to 1995 and are still using the original internal security method, do not specify SAFCLASS.

To identify the host security resource class to which you define ESP Workload Manager objects

- Code the SAFCLASS initialization parameter in your ESPPARM data set.

In SAFCLASS, you can also specify a prefix, such as ESP, for the resource profiles.

Note: We recommend that you use the class FACILITY or a class with the same characteristics as FACILITY.

Example

The following example identifies the resource class as FACILITY with a prefix of ESP:

```
SAFCLASS FACILITY PREFIX(ESP)
```

Defining RACROUTE

Use the RACROUTE initialization parameter to specify how an Event is processed for security purposes. You specify whether ESP Workload Manager should issue a RACINIT prior to executing Events, and how the user ID associated with the Event is determined.

Example

The following example specifies that ESP Workload Manager should issue a RACINIT before executing Events:

```
RACROUTE ON
```

Specifying Host Security Processing Options

Use the SAFOPTS initialization parameter to control processing options for the host security interface, for example, issuing z/OS commands with ESP Workload Manager and issuing OPER commands.

Note: If you are installing ESP Workload Manager for the first time, we recommend that you use the SAFOPTS defaults to get started. You can revise it as needed later.

To display the current host security options, class name, and resource prefix

Use the SAFOPTS command without any operands.

For example

```
oper safopts
Saf Options:MsgSupp DsAlloc MvsCmd Utoken NoSim3rdParty
SAF Class:FACILITY, Prefix ESP
```

Defining a Security Profile for ESP Workload Manager Setup

Contact your security administrator to set up the following profile that gives you unlimited access to set up ESP Workload Manager:

1. Define security profile *xxx*.**

   For *xxx*, substitute the value of the PREFIX operand for the SAFCLASS initialization parameter (see “Identifying the Host Security Resource Class” on page 53), for example

   ESP.**

2. Grant a universal access (UACC) of NONE and ALTER access to the TSO ID you are using to set up ESP Workload Manager.

Note: This generic profile is not intended for running the system in production. For production, you will need to define specific profiles for specific users.

5

Setting Up the Started Task

ESP Workload Manager executes as a z/OS subsystem and is initiated as a started task. The started task requires a procedure started from a system procedure library.

This chapter contains the following topics:

- Started Task Checklist
- Authorizing the Started Task and Link Library
- Creating the Started Task Procedure
- Setting Up TSO and ISPF for ESP Workload Manager
- Setting up the Procedure for ESP Encore
- Starting ESP Workload Manager

Started Task Checklist

You can use the following checklist to help you set up the ESP Workload Manager started task:

Step	Action	Page No.	√
1	Authorize the ESP Workload Manager started task and link library.	56	
2	Create the ESP Workload Manager started task procedure.	56	
3	Set up TSO and ISPF for ESP Workload Manager.	58	
4	Set up the ESP Encore Procedure (if applicable).	60	
5	Start ESP Workload Manager for the first time.	61	
6	Update the started task for Shadow Manager (if applicable).	138	
7	Install any CSF extensions that you require.	167	
8	Install any Application Monitor extensions that you require.	203	

Authorizing the Started Task and Link Library

1. Consult your security administrator to determine what is required to allow the ESP Workload Manager started task procedure to execute in your environment.
2. Define the ID for the ESP Workload Manager started task to the host security product.
3. APF-authorize the *prefix*.SSCPLINK library or put the members in a LINKLIST library.

Creating the Started Task Procedure

ESP Workload Manager requires a procedure that can be started from a system PROCLIB, such as SYS1.PROCLIB.

Note: Create one started task for each instance of ESP Workload Manager.

1. Customize a copy of sample library procedure CYBESS02 or create your own procedure.

   Ensure the JCL points to the correct data set names for the initialization parameter data sets. For details, see “Data sets allocated by the started task procedure for ESP Workload Manager” on page 57.

   Note: If you use shadow manager, you must specify the relevant parameters in CYBESS02 or in the start command.

2. Move the procedure to a system proclib.

Data sets allocated by the started task procedure for ESP Workload Manager

The following DD statements in the ESP Workload Manager started task procedure allocate the data sets indicated (see also “Sample Library Members Containing Initialization Parameters” on page 42.

ESPPARM

The ESPPARM DD statement points to the data set containing most of the ESP Workload Manager initialization parameters. ESPPARM is a mandatory data set; read every time you cold-start or warm-start ESP Workload Manager. A sample ESPPARM is available in sample library SSCPSAME, member CYBESS03.

ESPCOLD

The ESPCOLD DD statement points to the data set containing the ESP Workload Manager cold start initialization parameters. ESPCOLD is an optional data set, read every time you cold start ESP Workload Manager. A sample ESPCOLD is available in sample library SSCPSAME, member CYBESS04.

Note: You must do a cold-start the first time you start ESP Workload Manager.

ESPWARM

This DD statement points to the data set containing the warm start initialization parameters for ESP Workload Manager. ESPWARM is read only during a warm start. A warm start is the default startup option.

JOBDOC

This optional DD statement specifies the default job documentation library to be used for the JOBINFO command.

STEPLIB

The STEPLIB DD statement must be included to allocate the load library that contains the ESP Workload Manager modules if they do not reside in a LINKLIST library.

Setting Up TSO and ISPF for ESP Workload Manager

Making the ESP Workload Manager subsystem available from ISPF

1. Open for edit your ISPF primary option or user option menu.
2. Add the ESP Workload Manager option description to the body section.
3. In the processing section for each subsystem you need to access, insert the following statement into the TRANS function list of the ZSEL assignment statement:

   ```
   ESP,'PGM(ESP) PARM(MENU(CYBESMSM))'
   ```

   Note:

   - If the subsystem name for ESP Workload Manager is not ESP, you must also include the subsystem name as a parameter. In this example, the subsystem name for ESP Workload Manager is ESPX:

     ```
     ESP,'PGM(ESP) PARM(SUB(ESPX) MENU(CYBESMSM))'
     ```

   - The ROUTING of subsystem requests defaults to ROUTING LOCAL. The following example shows how you can set routing to MASTER as the default:

     ```
     ESPS,'PGM(ESP) PARM(SUB(ESPS) MENU(CYBESMSM) MASTER)'
     ```

   - If you want to invoke ESP Encore from the ESP Workload Manager primary options menu, change the menu name to CYBERMSM as in the following example:

     ```
     ESP,'PGM(ESP) PARM(MENU(CYBERMSM))'
     ```

Making ESP Workload Manager libraries available to TSO and ISPF

You need to set up the ISPF interface to identify target data sets for the panel, CLIST, skeleton, and message libraries. Target data sets can be existing ISPF libraries or newly allocated data sets that will be concatenated with existing data sets within your ISPF environment.

Either update the LOGON procedure or update the CLIST as described below.

To update the logon procedure

1. Change the ISPPLIB DD statement to include the *prefix*.SSCPPENU library.
2. Change the ISPMLIB DD statement to include the *prefix*.SSCPMENU library.
3. Change the ISPTLIB DD statement to include the *prefix*.SSCPTENU library.
4. Change the SYSPROC DD statement to include the *prefix*.SSCPCLST library.
5. Add or change the STEPLIB DD statement to include the *prefix*.SSCPLINK library if the ESP Workload Manager modules are not stored in a LINKLIST library.

Updating CLIST

As an alternative, you can use a CLIST to allocate these libraries using a combination of LIBDEF and ALTLIB.

Here is a sample CLIST:

```
PROC 0 DEBUG
CONTROL NOMSG NOFLUSH
FREE FI(ESPLLIB)
FREE FI(ESPPROC)
FREE FI(SYSEXEC)
/*********************************************************************/
/*   THIS CLIST ALLOCATES ESP LIBRARIES AND DISPLAYS THE ESP         */
/*   PRIMARY MENU.                                                   */
/*   MODIFICATIONS                                                   */
/*********************************************************************/
IF &DEBUG = DEBUG  THEN +
         CONTROL PROMPT SYMLIST CONLIST LIST MSG
         ISPEXEC CONTROL ERRORS RETURN
         ISPEXEC VGET ZTRAIL
/*********************************************************************/
/*    ALLOCATE THE DATASETS REQUIRED TO RUN ESP AND CALL THE         */
/*    MAIN PROGRAM. NOTE THAT THE LIBDEF FACILITY IS USED TO FREE    */
/*    THE USER FROM CODING THE ESP DATASETS IN THEIR LOGON PROCS.    */
/*********************************************************************/
ALLOC FI(ESPLLIB) DA('ESP.PRDLIB.LOAD') SHR REUSE
ALLOC FI(ESPPROC) DA('ESP.PRDLIB.CLIST') SHR REUSE
ALLOC FI(SYSEXEC) DA('ESP.PRDLIB.SYSEXEC') SHR REUSE
EXECUTIL SEARCHDD(YES)
ALTLIB ACTIVATE APPLICATION(CLIST) LIBRARY(ESPPROC)
ISPEXEC LIBDEF ISPPLIB DATASET ID('ESP.PRDLIB.PLIB')
ISPEXEC LIBDEF ISPMLIB DATASET ID('ESP.PRDLIB.MLIB')
ISPEXEC LIBDEF ISPSLIB DATASET ID('ESP.PRDLIB.SLIB')
ISPEXEC LIBDEF ISPTLIB DATASET ID('ESP.PRDLIB.TLIB')
ISPEXEC LIBDEF ISPLLIB LIBRARY ID(ESPLLIB)
ISPEXEC SELECT PGM(ESP) PARM(MENU(CYBERMSM)) NEWAPPL(ESP) PASSLIB
```

```
/*******************************************************************/
/*    FREE THE ESP DATASETS. THEY ARE NO LONGER REQUIRED BY          */
/*    THE USER.                                                      */
/*******************************************************************/
ISPEXEC LIBDEF ISPSLIB
ISPEXEC LIBDEF ISPMLIB
ISPEXEC LIBDEF ISPPLIB
ISPEXEC LIBDEF ISPTLIB
ISPEXEC LIBDEF ISPLLIB
ALTLIB DEACTIVATE APPLICATION(CLIST)
FREE  FI(ESPLLIB)
FREE  FI(ESPPROC)
FREE  FI(SYSEXEC)
EXIT
```

Installing utilities accessed by the ESP Utilities panel

Copy the following REXX execs from PDS *prefix*.SSCPSAME to a SYSEXEC PDS defined within the TSO logon procedures used to access ESP Workload Manager:

- CYBRES — Calls CYBRESNA to list ESP subsystem resources
- CYBRESNA — Lists ESP subsystem resources (called by CYBRES)
- CYBVAR — List Global Variable Tables

Setting up the Procedure for ESP Encore

CYBRMENC is the cataloged procedure that ESP Workload Manager inserts into any job you wish to be tracked by ESP Encore. It must be copied to a system PROCLIB. Do the following:

1. Copy member CYBRMENC from the CPE Sample Library (*prefix*.SSCPSAME) to a system cataloged procedure library, for example, SYS1.PROCLIB or equivalent.
2. Change @LOADLIB to the name of your SSCPLINK library.
3. Specify CYBRMENC as the cataloged procedure in the ERMSTEP initialization parameter.

Starting ESP Workload Manager

Starting ESP Workload Manager for the first time

The first time you start ESP Workload Manager, you must do a cold start.

- Issue one of the following z/OS operator commands and reply to all messages. If you use resources, ENQUEUEs, or NOTWITHs, format the resource data set by including the RESFORM parameter in the start command.

```
S ESP_started_task_name,PARM=(COLD)
S ESP_started_task_name,PARM=(COLD,RESFORM)
S ESP_started_task_name,,,COLD
S ESP_started_task_name,,,(COLD,RESFORM)
```

The last two commands are used if you are merging parameters in the started task JCL and parameters in the ESP start command.

Once ESP Workload Manager completes its initialization tasks, it issues message 499I and is ready for communication.

Note: ESP Workload Manager begins tracking immediately upon start up.

Starting ESP Workload Manager after upgrades are installed

- Follow the instructions in the release notes.

Once ESP Workload Manager completes its initialization tasks, it issues a message with ID ESP499, and is ready for communication.

The release notes instructions may require that you use the RELOAD option. The RELOAD option of the START command simulates an ESP Workload Manager start following an IPL. This causes the following modules to be loaded into CSA:

- CYBSS010—CPE SVC routine, used for inter-address space communication and data transmission
- CYBSS011—End of Memory (EOM) and End of Task (EOT) cleanup routines
- CYBSS023—RACF administration functions
- CYBXPI04—Freeform filter compiler/interpreter
- CYBES079—Scoreboard text API processor
- CYBES086—SMF record write intercept routine
- CYBJS031—WTO intercept and command substitution routines

6

Defining What ESP Workload Manager Tracks

ESP Workload Manager can track the progress of all jobs or selected jobs. It can also track the progress of jobs it did not schedule. You get real-time information for each job step, including step name, completion code, CPU time, elapsed time, and number of tape mounts. After you install ESP Workload Manager, a tracking model called MODEL1 tracks all jobs by default. A history data set called HIST1 stores the history data.

This chapter contains the following topics:

- Setting Up Tracking on Your System
- Specifying Tracking Options
- Using Job-Tracking Definition Tables
- Defining a Tracking Model

Setting Up Tracking on Your System

ESP Workload Manager can track the progress of all jobs or selected jobs, including those it did not submit. ESP Workload Manager provides real-time tracking information for each job step, including stepname, completion code, CPU time, elapsed time, and number of tape mounts.

You can have ESP Workload Manager track jobs, started tasks (STCs), TSO users (TSUs), and system messages. The tracking environment must be set up to track jobs in an ESP Workload Manager Application. The process of defining what ESP Workload Manager tracks involves:

- Specifying tracking options
- Defining and loading a job tracking definition table
- Defining one or more tracking models

SMF record collection

ESP Workload Manager needs the following SMF record types:

- Type 30
- Types 14, 15, and 64

  These record types are required for implicit data set triggers.
- Type 90, subtype 30 (collected only if Service Governor is on)

  This record is required for job expediting.
- Type 118 or 119

  These record types represent the reception or transmission of a data set via FTP. They are required for FTP data set triggers.

You can disable the SMF intercept using the SMFINT initialization parameter. Disabling the SMF intercept is useful when you are running multiple instances of ESP Workload Manager in a master and proxy configuration because it prevents duplicate SMF data collection.

Specifying Tracking Options

The tracking options specify what you want ESP Workload Manager to track. You can turn an option on or off using the TRACKOPT initialization parameter or command. The TRACKOPT initialization parameter is in the member you created from sample library (SSCPSAME) member CYBESS08 (see “Required Initialization Parameters” on page 24).

If you enter the TRACKOPT command, the options you specify override those in the initialization parameter. However, when you restart ESP Workload Manager, ESP Workload Manager uses the options in the initialization parameter.

Considerations

Before setting up the tracking environment, consider

- Where the job-tracking definitions will be stored
- Which jobs will need to be tracked
- How these jobs will be identified

What ESP Workload Manager can track

ESP Workload Manager can track jobs, started tasks, TSO users, and system messages.

Tracking options

You can track jobs on a master or proxy subsystem by specifying the TRACKOPT initialization parameter on all subsystems.

To turn a tracking option on

- Specify the TRACKOPT initialization parameter followed by the option. You can specify multiple options at the same time.

Using Job-Tracking Definition Tables

A job-tracking definition table identifies the characteristics of the jobs you want tracked. ESP Workload Manager can track jobs based on job name, execution class, programmer name, account number, job type or the user ID associated with the job.

Defining a job-tracking definition table

A job tracking definition table consists of two parts:

1. WILDCARD initialization parameters (optional) — A description of wildcard characters

   Characters described with WILDCARD are in addition to the asterisk (*) and hyphen (-) wildcard characters. You can use wildcard characters to simplify the job-tracking definition entries in the table.

2. TRACKDEF initialization parameters — Job-tracking definitions

   Each TRACKDEF entry identifies a job or a group of jobs and specifies the tracking parameters for them. When ESP Workload Manager sees a job, it finds the first TRACKDEF entry that matches the job. If ESP Workload Manager tracks the job, it uses the tracking model specified for the entry. If it does not find a matching entry in the table, it does not track the job.

Use the TRACKDEF initialization parameter or command to create job-tracking definition entries in the table.

You can edit a job-tracking definition table at any time and reload it using the LOADJTDT command.

Example

Note: You can also see examples in the TRACKDEF and WILDCARD initialization parameter reference pages in this guide.

Here is a sample job-tracking definition table:

```
Line
      /******************************************************/
      /*JOB TRACKING DEFINITION TABLE*/
      /******************************************************/
1     WILDCARD # 0-9            /*NUMERICS*/
2     WILDCARD $ A-Z            /*ALPHABETICS*/
3     WILDCARD + 0-9A-Z         /*ALPHANUMERIC*/

4     TRACKDEF JOB NAME(DUMMYJOB) NOTRACK
5     TRACKDEF JOB NAME(CYBJOB-)  MODEL(CYBMODEL)
6     TRACKDEF JOB NAME($$$$####) MODEL(PRODJOBS)
7     TRACKDEF JOB NAME($$#####-) MODEL(NAME(1:2),MODEL)
8     TRACKDEF ACCOUNT1(CYB3000)  MODEL(TESTJOBS)TRACKDEF
9     RACID(CYBFM-)               MODEL(DEMOMDL)
10    TRACKDEF JOB CLASS(G)       MODEL(MODELG)
11    TRACKDEF JOB NAME(-)        MODEL(MODEL1)
12    TRACKDEF STC NAME(-)        MODEL(MODEL1)
```

ESP Workload Manager processes the table entries in the order in which they appear. When a condition is satisfied, processing stops.

If ESP Workload Manager searches the preceding table for job CYBJOBAB and first account CYB3000, the condition on line 5 is satisfied first. Therefore, ESP Workload Manager tracks job CYBJOBAB with model CYBMODEL.

The following table explains each numbered statement in the preceding table:

Line	Explanation
1	The number sign (#) wildcard character matches all numeric characters.
2	The dollar sign ($) wildcard character matches alphabetic characters A through Z inclusive.
3	The wildcard character plus sign (+) matches alphanumeric characters A through Z inclusive and digits 0 through 9 inclusive.

Line	Explanation
4	NOTRACK disables tracking for jobs called DUMMYJOB.
5	Jobs whose name starts with CYBJOB are tracked using model CYBMODEL.
6	Jobs whose name starts with four alphabetic characters followed by four numeric characters are tracked using model PRODJOBS.
7	Jobs whose name starts with two alphabetic characters followed by five numeric characters are tracked using a model whose name is the first two characters of the job name followed by "MODEL". For example, tracking model DXMODEL tracks job DX123245.
8	Jobs whose first account number is CYB3000 are tracked using model TESTJOBS.
9	Jobs owned by user IDs beginning with CYBFM are tracked using model DEMOMDL.
10	Class G jobs are tracked using model MODELG.
11 and 12	All jobs and started tasks not covered by preceding job tracking definition table entries are tracked using model MODEL1.

Loading a job-tracking definition table

The person installing ESP Workload Manager at your site should, using the LOADJTDT initialization parameter, load the job-tracking definition table into the system each time ESP Workload Manager initializes. ESP Workload Manager uses this table to determine what to track. This table overrides any other tracking definition in the system.

To replace a job-tracking definition table

Issue the LOADJTDT command with the data set name that contains the new table, for example

```
LOADJTDT 'CYB.ESP.PARMLIB(JOBDEF1)'
```

Note: If more than one ESP Workload Manager subsystem uses the same table, you need to issue the LOADJTDT command on each ESP Workload Manager subsystem.

Changing the tracking model a job uses

To change the tracking model a job or group of jobs uses

1. Update the job-tracking definition table by specifying the new model you want to use.
2. Issue the LOADJTDT command to load the new table.
3. Issue the ALTTJ command to alter the index entry for the job. Specify the new model you want. You require this step for jobs that ESP Workload Manager might have already tracked using a different model.

The following example associates all jobs that begin with the letter J with the model called NEWMOD:

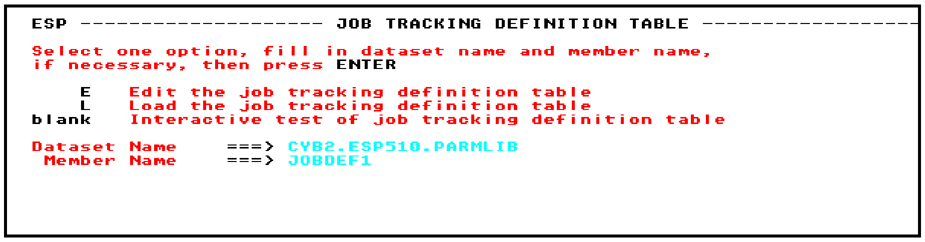
```
ALTTJ J- MODEL(NEWMOD)
```

Testing a job-tracking definition table

After defining a job tracking definition table, you may need to see what tracking model, if any, ESP Workload Manager uses for a job. This review may be necessary for troubleshooting.

To display the job tracking definition table

1. Select option **M** from the Main menu.
2. Select option 4 from the next panel.
3. Enter the name of the data set that contains the job tracking definition table on the panel. Specify the member name also if the job tracking definition table is a member of a PDS. Use the LDSN command in Page mode to display the name if you do not know it.



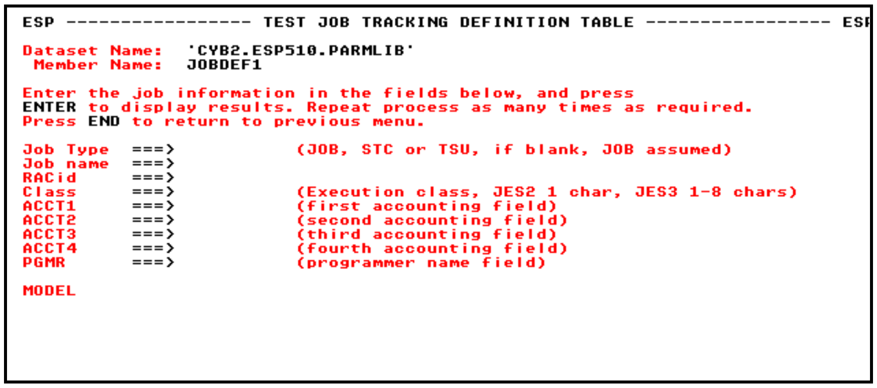
```
ESP -------------------- JOB TRACKING DEFINITION TABLE ------------------

Select one option, fill in dataset name and member name,
if necessary, then press ENTER

    E    Edit the job tracking definition table
    L    Load the job tracking definition table
blank    Interactive test of job tracking definition table

Dataset Name     ===> CYB2.ESP510.PARMLIB
 Member Name     ===> JOBDEF1
```

4. Enter any combination of job information to inquire about.



```
ESP ---------------- TEST JOB TRACKING DEFINITION TABLE ---------------- ESP

Dataset Name:   'CYB2.ESP510.PARMLIB'
 Member Name:   JOBDEF1

Enter the job information in the fields below, and press
ENTER to display results. Repeat process as many times as required.
Press END to return to previous menu.

Job Type   ===>            (JOB, STC or TSU, if blank, JOB assumed)
Job name   ===>
RACid      ===>
Class      ===>            (Execution class, JES2 1 char, JES3 1-8 chars)
ACCT1      ===>            (first accounting field)
ACCT2      ===>            (second accounting field)
ACCT3      ===>            (third accounting field)
ACCT4      ===>            (fourth accounting field)
PGMR       ===>            (programmer name field)

MODEL
```

Job tracking definition table results

The following table shows you the result corresponding to job tracking definition table entries:

If your input...	ESP Workload Manager...
Matches a tracking entry in the job tracking definition table...	...indicates the tracking model it uses.
Does not match a tracking entry in the job tracking definition table...	...informs you that it is not tracking the specified object.

Examples

Below are two examples of testing a job tracking definition table:

Example - 1	Example - 2
`Job Type===>JOB`	`Job Type===>Job`
`Job Name===>ABCD1234`	`Job Name===>DUMMYJOB`
`RACid===>`	`RACid===>`
`Class===>`	`Class===>`
`ACCT1===>`	`ACCT1===>`
`ACCT2===>`	`ACCT2===>`
`ACCT3===>`	`ACCT3===>`
`ACCT4===>`	`ACCT4===>`
`PGMR===>`	`PGMR===>`
`MODELPRODJOBS`	`MODEL-No Tracking-`

Defining a Tracking Model

Each job you want ESP Workload Manager to track must have an associated tracking model.

A tracking model specifies

- A name for the model
- The number of instances of a job ESP Workload Manager should keep on the jobindex data set
- The logical name of a history-recording data set
- Job-monitor Events or the prefix of job-monitor Events
- The manual processing nodes (pnodes) through which a job passes

To define a tracking model

- Use the DEFTM command.

Examples 1, 2, and 3 show how to define different types of tracking models.

Note: You can use the LTM command to list tracking models and the DELTM command to delete a tracking model. For more information, see the *ESP Workload Manager Reference Guide*.

Example 1

```
DEFTM PRODJOBS INDEX(10) HISTFILE(HIST1)
```

When you define the PRODJOBS model, you tell ESP Workload Manager to keep the 10 most recent executions of each job on the jobindex data set, and store history data for jobs using this model in the history data set whose logical name is HIST1.

Example 2

```
DEFTM TESTJOBS INDEX(5)
```

When you define the TESTJOBS model, you tell ESP Workload Manager to keep the five most recent executions of each job on the jobindex data set.

Example 3

```
DEFTM STEPMON INDEX(10) HISTFILE(HIST1)-
   STEPEND EVENT(CYBER.CHECK_STEPEND)
```

When you define the STEPMON model, you tell ESP Workload Manager to invoke a monitor Event called CYBER.CHECK_STEPEND at the end of each step for jobs using this model. Keep the 10 most recent executions of each job on the jobindex data set, and store history data for those jobs in the history data set whose logical name is HIST1.

Models specifying due-out time

In a tracking-model definition, you can specify a due-out time for each of the job's pnodes. The due-out time is the time when you expect a particular phase to be complete. If you have a lot of jobs with the same due-out requirements, using a tracking model can reduce the need for DUEOUT statements when you define a job. For more information on due-out times for jobs in an Application, refer to the "Using Applications" chapter in the *ESP Workload Manager User's Guide*.

Relative and absolute time

Due-out time can be either absolute or relative. Use the format *hh.mm* (*hh* represents the hour and *mm* represents the minutes).

- Absolute time specifies the actual time using the 24-hour clock.
- Relative time specifies the time relative to the submission time of the job.

Example — Defining a model with actual due-out time

The following statement defines a tracking model named DISTJOBS:

```
DEFTM DISTJOBS PNODE(DISTRIB(13:45))
```

Any job using this model must finish the distribution phase by 1:45 pm. This translates to 13:45 in the 24-hour clock. A job submitted at 2 pm (14:00) today has a due-out time of 1:45 pm (13:45) tomorrow.

If ESP Workload Manager has submitted a job and any phase of processing is incomplete by its due-out time, the job is considered overdue. ESP Workload Manager sends a message to the operator console stating the job name and the incomplete processing phase. If you want to take additional actions, see the "Job Monitoring and Alert Processing" chapter of the *ESP Workload Manager Advanced User's Guide*.

7 Setting Up Time Zones

ESP Workload Manager processes time-related information from a number of different sources. Some of this information comes from Agents or other ESP Workload Manager subsystems running on machines with different time zones set up. To interpret the received time values correctly, ESP Workload Manager must adjust them according to the difference between its own time zone, called LOCAL, and the zone from which the data came.

Your installation might consist of ESP Workload Manager subsystems located in different time zones. This chapter shows you how to specify the time zone for your subsystems.

This chapter contains the following topic:

- The TIMEZONE Initialization Parameter

The TIMEZONE Initialization Parameter

The TIMEZONE initialization parameter for time zone LOCAL must specify the offset between ESP Workload Manager local time and a reference time zone that is common to all communicating systems. UTC (Coordinated Universal Time) or GMT (Greenwich Mean Time) is a typical common reference for a time zone.

ESP Workload Manager accepts time expressions that include time zone names, for example

```
SCHEDULE 14:00 EST
```

You must define a TIMEZONE initialization parameter for any time zone used in a time expression. For each time zone you define, you must include an offset of that time zone from the common reference zone.

For details on the TIMEZONE initialization parameter, see “TIMEZONE: Define time zone” on page 512.

The Offset and SYSZONE operands

The shift in time from the reference zone is a sum of two parts: the offset coded on the TIMEZONE initialization parameter and, optionally, the z/OS time zone offset. Code SYSZONE(YES) to include the z/OS time zone offset.

When a time zone is defined with SYSZONE(YES), it immediately reflects any change in the z/OS time zone, allowing the time zone definition to remain current when seasonal time changes occur, such as the adjustment for Daylight Saving Time. Use SYSZONE(YES) only if the operating system's time zone offset is correctly maintained.

The SYSZONE(YES) operand should not be used

- If your installation resets the hardware clock, rather than local time, for seasonal time changes.
- If a TIMEZONE definition for a zone in a different country requires a seasonal change on a date that is different from the date of that change in your country.
- If the TIMEZONE definition is for a time zone name that does not undergo seasonal changes, such as GMT or EST.

The following example illustrates the use of the SYSZONE operand:

Note: Offsets are specified as West (W) or East (E) of the reference zone.

```
TIMEZONE 0  LOCAL    0     SYSZONE(YES)
TIMEZONE 21 NFNDLAND 1.30E SYSZONE(YES)
TIMEZONE 20 ATLANTIC 1E    SYSZONE(YES)
TIMEZONE 16 EASTERN  0     SYSZONE(YES)
TIMEZONE 17 CENTRAL  1W    SYSZONE(YES)
TIMEZONE 18 MOUNTAIN 2W    SYSZONE(YES)
TIMEZONE 19 PACIFIC  3W    SYSZONE(YES)
TIMEZONE  1 UTC 0
TIMEZONE  2 GMT 0
TIMEZONE  3 Z   0
TIMEZONE  5 EST 5.0W
```

The following is an example of a time zone table that does not use z/OS time zone:

```
TIMEZONE  1 UTC 0
TIMEZONE  2 GMT 0
TIMEZONE  3 Z   0
TIMEZONE  5 EST 5.0W
IF DAYS_FROM('1ST SUNDAY OF APRIL') GE 0 +
   AND DAYS_FROM('LAST SUNDAY OF OCTOBER') < 0 +
   THEN DO
     TIMEZONE 0  LOCAL    4
     TIMEZONE 21 NFNDLAND 2.30E
     TIMEZONE 20 ATLANTIC 3E
     TIMEZONE 16 EASTERN  4
     TIMEZONE 17 CENTRAL  5W
     TIMEZONE 18 MOUNTAIN 6W
     TIMEZONE 19 PACIFIC  7W
   ELSE DO
     TIMEZONE 0  LOCAL    5
     TIMEZONE 21 NFNDLAND 3.30E
     TIMEZONE 20 ATLANTIC 4E
     TIMEZONE 16 EASTERN  5
     TIMEZONE 17 CENTRAL  6W
     TIMEZONE 18 MOUNTAIN 7W
     TIMEZONE 19 PACIFIC  8W
  ENDDO
```

Important: If you use logic like the preceding example, you must restart ESP Workload Manager after the appropriate date that the condition is in effect (in this case, after midnight on the 1st Sunday of April), otherwise the offsets will not change.

The TIMEREF operand

Usually, ESP Workload Manager relies on the operating system to provide the local time, but, in some cases, it needs to operate at a different time. The optional TIMEREF operand of the TIMEZONE initialization parameter defines the way ESP Workload Manager obtains local time. You code this operand for the time zone LOCAL definition only.

Time reference can be defined as either z/OS-reported local time (SYSTIME) or z/OS-reported GMT adjusted by a fixed offset.

Important: The TIMEZONE statement with TIMEREF operand should be one of the first ESP initialization statements, so that subsequent statements could be processed after the time logic is configured.

The following example sets up ESP Workload Manager to work on Pacific time on a machine that is configured for the Eastern time zone (the Pacific time zone is three hours ahead of the Eastern time zone):

```
TIMEZONE 0  LOCAL    3W    SYSZONE(YES) TIMEREF(SYSTIME-3)
TIMEZONE 16 EASTERN  0     SYSZONE(YES)
TIMEZONE 17 CENTRAL  1W    SYSZONE(YES)
TIMEZONE 18 MOUNTAIN 2W    SYSZONE(YES)
TIMEZONE 19 PACIFIC  3W    SYSZONE(YES)
TIMEZONE  1 UTC 0
TIMEZONE  2 GMT 0
TIMEZONE  3 Z   0
TIMEZONE  5 EST 5.0W
```

8

Setting up Resources

This chapter contains the following topics:

- Setting Up ESP Workload Manager Resources on Your System
- Resource Data Set (RESFILE) Considerations
- System Topology
- Default Resource Assignments

Setting Up ESP Workload Manager Resources on Your System

ESP Workload Manager maintains counters for resource availability where the Application Manager resides so that resource management has one central point of control. As resources are used and replenished, the counters are adjusted.

ESP Workload Manager resources can be classified into the following four groups:

- Local resources — Resources that are maintained for each z/OS system image
- Global resources — Resources that are available to all processors in a node. Processors share a resource-availability counter.
- Enterprise resources — Resources that are available to all components of the enterprise. One availability counter exists for these resources.
- Implicit resources — ENQUEUE and NOTWITH

To set up resources

1. Specify the resource file using the RESFILE initialization parameter. For more information, see “RESFILE: Specify the resource data set” on page 468.
2. Define the topology with the NODE and CPU initialization parameters. For more information, see “NODE: Define JES node” on page 436 and “CPU: Central Processing Unit” on page 273.
3. Code the RESDFLT initialization parameter if you want to use default resources. For more information, see “RESDFLT: Identify default resources” on page 462. If you are unsure, you might want to omit this step until you are more familiar with default resources. For more information on default resources, see “Default Resource Assignments” on page 81.
4. Use the RESDEF command to define individual resources. For more information, see the *ESP Workload Manager Reference Guide*.

Defining resources within an IF RESFORM THEN loop

Resource definitions are stored in the RESFILE. If ESP Workload Manager starts with the RESFORM option, information stored in the RESFILE is lost. To ensure that system topology and resource definitions are not lost when the RESFILE is reformatted (RESFORM), code the following in your ESP Workload Manager initialization parameters:

```
IF RESFORM THEN DO
  NODE TORONTO ADD ROUTEJCL('/*XEQ TORONTO')
  CPU T1 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T1')
ORDER(1) CURRENT
  CPU T2 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T2')
ORDER(2)
  CPU T3 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T3')
ORDER(3)
  RESDEF T3480 ADD LOCAL RENEWABLE MAX(5) CPU(T1)
  RESDEF CICSUP ADD NODAL THRESHOLD AVAIL(0)
  RESDEF DB2TAB1 ADD NODAL RENEWABLE AVAIL(3)
ENDDO
```

If you use the resource feature extensively and have numerous resources defined, you might prefer not to code all your resource definitions in the ESP Workload Manager initialization parameters. Alternatively, store your resource definitions in a data set and issue the LOAD command to process the RESDEF commands in the event the RESFILE is reformatted.

Note: The LOAD command cannot be used in the ESP Workload Manager initialization parameters.

Resource Data Set (RESFILE) Considerations

The Resource data set contains the following information classes:

- System Topology
- Resource definitions
- Jobs

Note: You do not need a resource data set if you do not use resources. However, the statements described in the “Using Implicit Resources” section of *ESP Workload Manager User’s Guide* use resources implicitly. Therefore, a resource data set (RESFILE) and the network topology must be defined before you can use these statements.

System topology

The system topology is a definition of all the processors in the enterprise. One or more processors in a shared-spool environment are collectively known as a node. The enterprise can consist of one or more nodes, which are connected by NJE (Network Job Entry).

Resource definitions

A descriptor exists for each resource defined to the enterprise. The resource characteristics and availability counters are also maintained in this file.

Jobs

Jobs that have already allocated or are awaiting resources have descriptors within the resource file.

When ESP Workload Manager starts with a RESFILE, commands can be entered to define the topology and resources. The CPU/NODE topology can be changed at any time. The definitions need to be reloaded only when the RESFILE is re-created or reformatted.

System Topology

The way your processors are arranged into one or more multi-access-spool (MAS) configurations is known as the topology of the network. Specifying the topology provides vital information to ESP Workload Manager about how you want to schedule resources and route jobs.

When specifying the topology, you can

- Name individual processors
- Group processors into one or more nodes
- Arrange for jobs to be routed automatically to a processor
- Define a preference order for CPUs
- Make CPUs temporarily unavailable

NODE and CPU

To specify your typology, you name and group the individual processors in the enterprise into one or more nodes by using two ESP Workload Manager statements: NODE and CPU. For more information on NODE and CPU, see “NODE: Define JES node” on page 436 and “CPU: Central Processing Unit” on page 273.

Routing jobs

Sometimes in a multiple node environment a job may need a mix of resources that are not available on the ESP Workload Manager subsystem submitting the job. However, these resources may be available on another processor—either on the current node or on a remote node.

ESP Workload Manager can automatically insert JCL statements at submission time to ensure the job is automatically routed to the required node and processor. However, ESP Workload Manager only ships a job to a remote node if one of the resources the job requires has the GRAVITY attribute in its definition.

CPU preference order

When defining CPUs, you can also define the order in which ESP Workload Manager scans the CPUs for available resources. If sufficient resources to execute a job reside on two or more CPUs, the order in which ESP Workload Manager scans the CPUs determines the CPU the job is sent to. For more information, see "CPU: Central Processing Unit" on page 273.

Default Resource Assignments

ESP Workload Manager can determine some of the significant resource requirements a job might have by looking at the job's run history. If required, ESP Workload Manager can automatically make default resource assignments. The use of default resource assignments is optional.

Resources for default assignments

The resources for which ESP Workload Manager can make default assignments are

- Number of tape drives needed
- Number of scratch tapes needed
- CPU Absorption (percentage used)
- Total CPU time used
- Total elapsed time used
- Total print lines generated
- Number of unused CPU service units over a specified period (IBM WLM resources)

How ESP Workload Manager determines default resource assignments

ESP Workload Manager uses information from the jobstats data set to calculate the default resources it assigns to jobs. ESP Workload Manager bases the default resource levels it assigns on an average of the records from the jobstats data set. If job profiles are used, the number of records included in the average depends on the number of entries for the job profile. If job profiles are not used, the number of records included in the average depends on the number of entries for the job, based on its full name, in the jobstats data set. The average includes only completed jobs; abended jobs are ignored. ESP Workload Manager updates the information each time it creates a new Application generation.

Setting up default resources

To set up default resources, code the RESDFLT initialization parameter and the optional default resources. For more information, see “RESDFLT: Identify default resources” on page 462.

Using ELAPSED resource in a master and proxy environment

In a master and proxy environment, you can use a default resource representing elapsed time to prevent jobs from being submitted if there is not enough elapsed time available. If the ELAPSED resource is defined as a local resource for each CPU, each CPU can have its own counter. ESP Workload Manager submits a job to JES only if enough ELAPSED resource is available on any CPU. Even though JES controls where the job executes, ESP Workload Manager can automatically insert a system affinity card to control routing.

When a job’s predecessor and time requirements are met, ESP Workload Manager checks if the required resources are available. For example, ESP Workload Manager checks if enough of the ELAPSED resource is available for job ABC on each of the CPUs. If no ORDER operand is specified for the CPU definitions in the ESP Workload Manager master’s initialization parameters, ESP Workload Manager checks each CPU in the order in which they are defined.

Using the ROUTEJCL parameter on the CPU definition controls routing. The parameter contains the JCL representing a system affinity card, causing ESP Workload Manager to route a job to the system where there is a sufficient amount of the ELAPSED resource. For example, if enough of the ELAPSED resource exists on CPU B, ESP Workload Manager submits the job and adds a system affinity card to route the job to CPU B. Similarly, if enough of the ELAPSED resource exists on CPU A, ESP Workload Manager submits the job and adds a system affinity card to route the job to CPU A. In other words, when a local resource becomes available, ESP Workload Manager submits the job and routes it to that system.

Part 2

Extending ESP Workload Manager with Other ESP Products

This part discusses how to extend ESP Workload Manager with other ESP products:.

This part contains the following chapters:

- Customizing ESP Encore
- Configuring Workstation Server
- Configuring ESP Workload Manager to Use Agents

9

Customizing ESP Encore

This chapter describes the options for customizing ESP Encore and contains the following topics:

- Initialization Parameters
- Prerequisites to Customizing ESP Encore
- Establishing Installation Default Values
- ENCPARM
- Where ENCPARMs Are Used
- Listing, Adding or Deleting ENCPARMs While ESP Workload Manager is Running
- Examples of Default Initialization Parameters
- Compressing EXH Records
- Interfacing With Your Tape-Management System

Initialization Parameters

Required initialization parameters

If you use ESP Encore, you must add the following initialization parameters:

- ERMSTEP

  ERMSTEP specifies the ESP Encore step to be inserted into each job ESP Encore will track.

  ESP Workload Manager inserts this JCL automatically.

  ```
  ERMSTEP ('//ENCORE EXEC CYBRMENC')
  ```

  Note: Specify the name of the cataloged procedure, for example, CYBRMENC.

- EXHFILE

  EXHFILE specifies the name of the ESP Encore Execution History (EXH) data set.

  ```
  EXHFILE data_set_name
  ```

- ESPGROUP (only for a master and proxy environment)

  An ESP Workload Manager master subsystem and its corresponding proxy ESP Workload Manager subsystems require the same ESPGROUP identifier. Having the same identifier ensures the correct ESP Encore tracking data is sent to the correct ESP Workload Manager master subsystem. The name has a limit of eight characters.

  ```
  ESPGROUP name
  ```

ENCPARM initialization parameters

You can customize ESP Encore's features and functions to your environment by adding ENCPARM initialization parameters to your ESP Workload Manager initialization parameters. ENCPARM initialization parameters affect all jobs on a global basis. For recommended default ENCPARM initialization parameters, see "Examples of Default Initialization Parameters" on page 89.

```
ENCPARM CLEANUP INITIAL(YES)
```

Prerequisites to Customizing ESP Encore

To customize ESP Encore, you must have complied with the instructions listed on “Adding ESP Encore” on page 7:

- You must have allocated the EXH data set that stores information stored about jobs ESP Encore tracks for restart and rerun purposes. Refer to “Execution History (EXH) Data Set” on page 21.
- You must have added the mandatory initialization parameters for ESP Encore. Refer to “Initialization Parameters” on page 86 and to
  - “ERMSTEP: Add restart step” on page 339
  - “ESPGROUP: Specify ESP complex ID” on page 341
  - “EXHFILE: Identify Execution History data set” on page 345

Establishing Installation Default Values

You specify the default ESP Encore values that apply to every job ESP Encore tracks in the ESP Workload Manager initialization parameters. To distinguish ESP Encore initialization parameters from ESP Workload Manager initialization parameters, you must include the keyword ENCPARM before each ESP Encore initialization parameter. When ESP Workload Manager submits a job with the ESP Encore step, these default values pass to the job unless they are overridden in the Application or the job details.

ENCPARM

There are two categories of ESP Encore ENCPARMs:

- User ENCPARMs — These initialization parameters describe options and features users might need to control. These are the ENCPARMs the user enters in ESP Workload Manager Procedures.
- System ENCPARMs — ESP Agent uses these initialization parameters to communicate with ESP Encore by inserting the system ENCPARMs into the job stream.

Note: For more information on what ENCPARMS are initialization parameters required for installation, see “Initialization Parameter Summary” on page 230 and

Where ENCPARMs Are Used

You can use ESP Encore ENCPARMs in the following places:

- ESP Workload Manager initialization parameters

  ```
  ENCPARM SECURITY TAPE(NO)
  ENCPARM CLEANUP INITIAL(YES)
  ```

  Note: You must code the ENCPARM initialization parameters in the ESPPARM member for every ESP Workload Manager subsystem (master or proxy) that executes or tracks jobs.

- ESP Workload Manager Procedure

  ```
  APPL CYBER
  JCLLIB .....
  OPTIONS RESTARTSTEP
  ENCPARM HONORCC NONE
  JOB JOB1
    RUN DAILY
  ENDJOB
  ```

- ESP Encore Resubmission Panel

  ```
  Restart  ===> Y
  Jobid    ===> J1234
  Fromstep ===> STEP003
  Tostep   ===>
  Do you wish to enter any ESP Encore statements? ===> Y
  ```

- ENCPARM Panel

  ```
  Backout  ===> Y(Yes/No)
  Cleanup  ===>(Yes/No)
  Mode     ===> (Normal/Scan)
  Force  ===>(Yes/No)
  Auto-restore===>(Yes/No/Always)
  Honor cond codes===>(All/None)
  Enter one or more ESP Encore statements below. Do not prefix the
  statements with 'ENCPARM'.

  ===>
  Press ENTER to resubmit the object or END to cancel the request.
  ```

- Page Mode

  ```
  OPER ENCPARM WARNING MISSING(YES)
  ```

- Console

  ```
  F ESP,ENCPARM CLEANUP INITIAL(YES)
  ```

Listing, Adding or Deleting ENCPARMs While ESP Workload Manager is Running

You can list, add or delete ENCPARMs while ESP Workload Manager is running by issuing an ENCPARM command from the following table.

ENCPARM Command	Description
ENCPARM	List all ENCPARMs, their values, and a number representing the order in which each ENCPARM was issued.
ENCPARM *xxx yyy zzz*	Add an ENCPARM with the values *xxx, yyy, zzz*
ENCPARM DELETE(*n*)	Delete the *n*th ENCPARM, where *n* is the order in which the ENCPARM was issued. To list *n* for the ENCPARM you want to delete, issue the ENCPARM command.

Note: The effect of the preceding ENCPARM addition and deletion commands is temporary. The next time ESP Workload Manager restarts, additions and deletions are undone.

Examples of Default Initialization Parameters

The following are examples of ESP Encore ENCPARM default initialization parameter examples you may use in your ESP Encore environment. To use any of these default ENCPARM examples, specify them in the ESP Workload Manager initialization parameters for all subsystems. For more information on ENCPARM initialization parameters, see “Installation Reference” on page 227.

Preventing NOT CATLGD 2 errors

- Indicate whether you want ESP Encore to automatically clean up data sets on the job’s initial run to prevent NOT CATLGD 2 errors:
  ```
  ENCPARM  CLEANUP  INITIAL(YES)
  ```

Volumes requiring special processing

- If you have a data set migration or archival product, such as DFHSM, indicate the special volume name that the product uses to indicate that a data set was migrated/archived:
  ```
  ENCPARM  VOLUME  MIGRATE(migvol)
  ```
- If you have DFHSM, add the following initialization parameter:
  ```
  ENCPARM  VOLUME  MIGRATE(MIGRAT)
  ```

Data sets requiring special processing

- If there are some data sets that ESP Encore should never clean up or auto-restore, then specify them with the following initialization parameter:

```
ENCPARM  IGNOREDS  DSNAME(dsname-mask)
```

- If you have complex data set masking standards, see “Write a Data Set Processing Exit for ESP Encore” on page 225.

Interfacing with your tape-management system

- If you have a tape-management system, you should specify the name of your TAPESCR exit in the following initialization parameter:

```
ENCPARM  TAPESCR  PROG(pgmname)
```

- For more information on TAPESCR, see “Interfacing With Your Tape-Management System” on page 92.

Describing special abending programs

- If you have programs that abend on behalf of other programs, name the abending programs with this initialization parameter:

```
ENCPARM  ABENDER  program
```

Modifying internal processing functions

- If you want to save space on the EXH data set, use the COMPRESS(YES) option. See “Compressing EXH Records” on page 91 for details on how compression works.
- If you want to change the number of EXH requests in the ESP Workload Manager address space, use the PACING(nn) option.
- If you want to invoke a user-defined exit routine to support a complex naming convention for data sets you want ESP Encore to ignore, use the DSEXIT(module) option. See “Write a Data Set Processing Exit for ESP Encore” on page 225 for details.

```
ENCPARM  MODIFY COMPRESS(YES) PACING(20) DSEXIT(module)
```

Issuing warning messages from the Restart Analysis panel

- If you want a warning message issued when a generation from a GDG concatenation is missing upon a job restart, use GDGALL(YES).
- If you want a warning message issued when a data set is missing during a job restart that ESP Encore wanted to clean up, use MISSING(YES).
- If you want a warning message issued when a data set has DISP=MOD specified during a job restart and ESP Encore cannot automatically restore the data set, use DISPMOD(YES).

```
ENCPARM  WARNING GDGALL(YES) MISSING(YES) DISPMOD(YES)
```

Automatically purging jobs from the EXH data set

- If you want to automatically purge jobs from the EXH data set that have different criteria than the defaults you set up, use this initialization parameter:

```
ENCPARM  PURGE JOB(job-mask) AGE(days) KEEP(number)
```

For more information on ESP Encore ENCPARM initialization parameters, see “Installation Reference” on page 227.

Compressing EXH Records

The EXH data set describes each job ESP Encore tracks. The EXH data set can grow very large depending on

- The installation’s workload
- The timeframe you require that a job remain restartable

Each job is described by a single EXH record. Each record in turn is saved into as many fixed-length slots as required. Shorter records use up fewer slots in the data set. Therefore, if many records can be significantly shortened, a great deal of EXH data set space can be saved.

EXH data set records

The life of an EXH data set record is as follows:

a. The ESP Encore step creates a record that describes the job as it looks before it executes.
b. ESP Encore task CYBRM312 in the ESP Workload Manager address space writes the record to the EXH data set and saves it in memory until the job finishes execution.
c. As the job executes and SMF data is created, the copy of the record is updated and the updated portions are rewritten to the EXH data set.

d. When the job ends, the final update is made to the EXH data set and the memory copy of the record is deleted.
e. The record is later read from the EXH data set to display its contents or to help restart the job.

Compression

Compression takes place at stage d above. The record is originally written to the data set in an uncompressed state. When the job ends, the record is compressed and written to the data set a final time. The slot access method automatically frees the slots that are no longer needed to hold the uncompressed record.

Turning on compression

The default setting for EXH data set compression is NO. You can have ESP Encore compress EXH records by coding the following initialization parameter:

```
ENCPARM MODIFY COMPRESS(YES)
```

Note: The amount of compression varies. Only records for new jobs are compressed; existing records are not.

Decompression

Decompression takes place at stage e above. When the record is read from the EXH data set, a special marker in the record indicates that it is compressed.

The record is compressed by using run-length encoding of repeated blanks and zeroes. That is, each string of 3 to 255 blanks (or zeroes) is replaced by a three-byte sequence that describes the string. This sequence consists of an escape character, the byte repeated, and the length of the repeating string.

For more information on the ESP Encore ENCPARM initialization parameters, see the *ESP Encore User's Guide.*

Interfacing With Your Tape-Management System

If no tape-management system is installed, ESP Encore does not act when scratching a tape data set, except to uncatalog it. However, if a tape-management system is present, that system must be notified whenever ESP Encore cleans up or backs out a tape data set.

There are two ways for ESP Encore to interact with a tape-management system. One is by issuing an operator command, and the other is by executing an interface program. The method you choose depends on your tape-management system and how it scratches tapes.

Issuing an operator command

```
ENCPARM  TAPESCR  COMMAND('cmd text') -
                  CONSOLE(consname)  CART(token)
```

The console name and token are optional. The command text might contain any of the following variables. ESP Encore substitutes each before the command is issued.

Variables	Description
&DSN	The data set name
&VOL	The tape volume
&EXPDT	The expiry date, as six hexadecimal digits. This is in binary YYDDD format, with the year and Julian day, for example X'63016D' is EXPDT=99365
&RETPD	The retention date, also in binary YYDDD format

&TYPE	I, R, B, OR C depending on the TYPE:
	`TYPE=INSTALL`
	`TYPE=RESTART`
	`TYPE=BACKOUT`
	`TYPE=CLEANUP`

Executing an interface program

```
ENCPARM  TAPESCR  PROG(pgmname)
```

The interface program must be a separate load module in the ESP Encore load library (or load library in the systems LINKLIST). It is attached in the AMODE it was link-edited with. A standard z/OS parameter list is used, containing these parameters, all of them in below-the-line storage:

Parameter	Description
Data set Name	44-byte name of the tape data set
Number of Volumes	Full-word integer. Number of volumes in the following list
Volume List	List of six-byte volume serials
Expiry Date	three-byte expiry date in Julian format: X'yydddd'
Retention Period	three-byte retention date in Julian format: X'yydddd'
Flags	Two bytes of flags: X'8000': `TYPE=INITIAL` X'4000': `TYPE=RESTART` X'2000': `TYPE=BACKOUT` X'1000': `TYPE=CLEANUP`

The tape scratch program should return a code in register 15. A return code of zero usually indicates success.

10

Configuring Workstation Server

This chapter contains the following topics:

- Configuring ESP Workload Manager for ESP Workstation
- Testing the Installation Connections to the Network

Configuring ESP Workload Manager for ESP Workstation

Initialization parameters for Workstation Server

Workstation Server provides communication between ESP Workload Manager and ESP Workstation and requires initialization parameters.

CA provides sample initialization parameter data sets that you can use to code your initialization parameters. The sample initialization parameter data sets are in the SSCPSAME data set.

To use your initialization parameter data set

1. Rename the members (optional).
2. Copy the member CYBESS32 from *prefix*.SSCPSAME into your own initialization parameter data set.
3. Edit the copy as required.

Initialization parameters

The initialization parameter data set for Workstation Server might contain the following initialization parameters:

- ENCRYPT
- INET
- MODE
- PORT
- ROUTING
- SUBSYS
- TCPIP
- WSSCTL

  Note: If you coded a WSSCTL initialization parameter and have OPER authority, you can issue WSSCTL from an ESP Workstation client. You can use WSSCTL to dynamically turn message tracing on or off without requiring a TRACELOG DD statement.

- WSSSET (can also be coded as SET)

  Note: If you coded a WSSSET initialization parameter, you can use the WSSSET or SET commands only to dynamically change a Workstation Server initialization parameter through a MODIFY command. You cannot issue the WSSSET or SET commands from an ESP Workstation client.

For information on the initialization parameters, see “Installation Reference” on page 227.

Scoreboard-Scan Interval

One of the Workstation Server's functions is to inform ESP Workstation of the status of the selected workload objects. The Workstation Server also notifies ESP Workstation when the status of existing workload objects changes and when a new workload object satisfies the filtering criteria specified. If no filtering criteria are specified, the Workstation Server reports the status of the workload objects a user is authorized to view among all the workload objects ESP Workload Manager monitors.

The Workstation Server obtains the status of workload objects by scanning the ESP Workload Manager scoreboard, executing a single scoreboard scan on behalf of all users as specified in the WSSPARM file. A scan might occur sooner, for example, when a user begins a new ESP Workstation session. When a scoreboard scan is in progress, all sessions receive the status updates that apply to them.

If you want a value different than the five-second default, you can set the INTERVAL operand in the WSSPARM data set. For details about the WSSCTL initialization parameter, see "WSSCTL: Set parameters and control message tracing" on page 545. For details about the WSSSET initialization parameter, see "WSSSET: Set Workstation Server connections and timing" on page 548.

Example of Workstation Server initialization parameter data set

The following is an example of a valid initialization parameter data set for Workstation Server:

```
MODE     NORMAL
PORT     5300
SUBSYS   Q530
TCPIP    TCPHPNS(TCPIP)
SET
ENCRYPT key(X'0307195303071953')
WSSCTL SET PARMS INTERVAL()
WSSCTL SET PARMS RPOLLINT()
WSSCTL SET PARMS RCONNINT()
WSSCTL SET PARMS RRSPWAIT()
WSSCTL SET PARMS PROXYEXP()
WSSCTL SET PARMS TCPWAIT(120)
WSSCTL SET PARMS MAXCON_LOIP(10)
WSSCTL SET PARMS MAXCON_PERIP(100)
WSSCTL SET PARMS MAXCON_TOTAL()
WSSCTL SET TRACE SYSOUT(A)
WSSCTL START TRACE
```

Workstation Server started task

ESP Workstation connects to ESP Workload Manager via an z/OS started task called Workstation Server. Commands and requests issued against objects from within ESP Workstation are routed through the Workstation Server to ESP Workload Manager.

To set up a Workstation Server, you must create a JCL procedure that executes as a started task. The procedure executes program CYBES064. Follow the instructions provided.

Files allocated by the Workstation Server started task procedure

The following DD statement in the ESP Workload Manager started task procedure allocates the following required and optional files:

- WSSPARM
- STEPLIB

To define the started task

1. Edit the sample member CYBESS34 in library *prefix*.SSCPSAME and customize it as noted in the comments.
2. Ensure the JCL points to the correct data sets for the initialization parameter data set WSSPARM and STEPLIB. You coded WSSPARM according to the instructions in "Initialization parameters for Workstation Server" on page 96.

WSSPARM

This DD statement points to the data set containing the Workstation Server initialization parameters.

STEPLIB

If the ESP Workload Manager modules do not reside in a LINKLIST library, you must include this DD statement to allocate the load library that contains the ESP Workload Manager modules.

Usage Notes

The Workstation Server does not allow the use of concatenated data sets on the WSSPARM DD statement.

If a Workstation Server encounters a WSSPARM DD statement in its JCL, it processes the WSSPARM file before parsing the parameters listed on the EXEC statement. The EXEC parameters override those specified in the WSSPARM file. The Workstation Server comes to a halt if it detects an error in a WSSPARM parameter or an EXEC parameter. Diagnostic messages in the Workstation Server's JES message log identify the problems that arise. A system programmer can test the Workstation Server parameters and terminate by including the SCAN EXEC parameter or the MODE SCAN WSSPARM parameter. The NORMAL EXEC parameter cancels the effect of a MODE SCAN parameter.

EXEC parameters

You can use parameters equivalent to those listed in “Initialization parameters” on page 96. The format of those equivalent parameters is described in “Installation Reference” on page 227.

Initialization parameters for Consolidated Workstation Server

A Consolidated Workstation Server is a Workstation Server with the additional ability to connect to other Workstation Servers. It requires an additional initialization parameter data set.

CA provides sample initialization parameter data sets that you can use to code your initialization parameters. The sample initialization parameter data sets are in the SSCPSAME data set.

To use your initialization parameter data set

1. Rename the members (optional).
2. Copy the member CYBESS33 from *prefix*.SSCPSAME into your own initialization parameter data set.
3. Edit the copy as required.

Initialization parameters

The initialization parameter data set for Consolidated Workstation Server contains one or more REMOTE initialization parameter, one for each Workstation Server that Consolidated Workstation Server will connect to. For information about the REMOTE initialization parameter, see “REMOTE: Specify connected Workstation Server” on page 455.

Consolidated Workstation Server started task

To set up a Consolidated Workstation Server, you must edit the Workstation Server JCL procedure according to the instructions provided below.

Files allocated by the Workstation Server started task procedure

The following DD statement in the ESP Workload Manager started task procedure allocates the following required and optional files:

- WSSPARM
- WSSREMOT
- STEPLIB

To define the started task

1. Uncomment the WSSREMOT DD statement in the sample member CYBESS34, which are in library *prefix*.SSCPSAME, and customize it as noted in the comments.
2. Ensure the JCL points to the correct data set for the initialization parameter data WSSREMOT. You coded this data set according to the instructions in "Initialization parameters for Consolidated Workstation Server" on page 99.

The following example shows you how to allocate a WSSREMOT file:

```
//WSSREMOT DD DISP=SHR,DSN=name of PDS member or sequential data
set
```

WSSREMOT

This DD statement points to the data set containing the additional initialization parameters for Consolidated Workstation Server.

Usage Notes

The Workstation Server does not allow the use of concatenated data sets on the WSSREMOT DD statement. If you update the WSSREMOT file while the Consolidated Workstation Server is running, you must stop and restart the Consolidated Workstation Server to implement the changes.

Workstation Server authority

When you sign on to ESP Workstation, you log on with your own RACF mainframe user ID and ESP Workload Manager uses the authority associated with that user ID. Ensure the user ID assigned to the Consolidated Workstation Server has maximum access to all the resources any ESP Workstation user may require.

For more information on ESP Workload Manager security, see the *ESP Workload Manager Security Guide*.

Testing the Installation Connections to the Network

ESP Workstation is a client-server application that requires an active network connection to one or more Workstation Servers. This network connection uses TCP/IP.

To successfully install and use ESP Workstation, you must

- Verify the TCP/IP connection to the Workstation Server
- Verify the Workstation Server's operation
- Connect ESP Workstation to ESP Workload Manager

Verifying the TCP/IP connection to the Workstation Server

You need to verify the TCP/IP connection between your ESP Workstation and the Workstation Server.

To verify the connection, use one of two methods:

- Ping
- Telnet

Verify the TCP/IP connection between your ESP Workstation and the Workstation Server by pinging or telneting the host from your workstation, using the appropriate TCP/IP addresses.

See your Network Administrator if you need assistance with this process.

Verifying the Workstation Server's operation

To verify the Workstation Server's operation

1. Start the Workstation Server.
2. Issue the status command.

Starting the Workstation Server

To start the Workstation Server, type

```
/S ESPWSS
```

In this example, ESPWSS is the started task name.

Issuing the STATUS command

You can use the Workstation Server STATUS command to display the Workstation Server's status. The number of connections attempted and the number of current active sessions appear.

To issue the STATUS command, type

```
/F ESPWSS, STATUS
```

When you issue the Workstation Server STATUS command prior to connecting Workstation, you see messages like the following:

```
F ESPWSS,STATUS
ESPWSS468I ESP Workstation Server 5.4.0 STATUS
ESPWSS472I 0 Client connections have been made, currently 0 are
active
```

Issuing the STOP command

To stop the Workstation Server, issue the following command:

```
/P ESPWSS
```

Connecting ESP Workstation to ESP Workload Manager

To verify the Workstation Server is connected and test the installation

1. Start ESP Workload Manager.
2. Start the Workstation Server using the S *wssname* command.
3. Start ESP Workstation by double-clicking on the ESP Workstation icon.
4. Connect using ESP Workstation. For information on connecting, see the *ESP Workstation User's Guide*.

11

Configuring ESP Workload Manager to Use Agents

This chapter contains the following topics:

About Agents

ESP Agents enable you to automate and manage workload on distributed systems.

How ESP Workload Manager works with ESP Agents

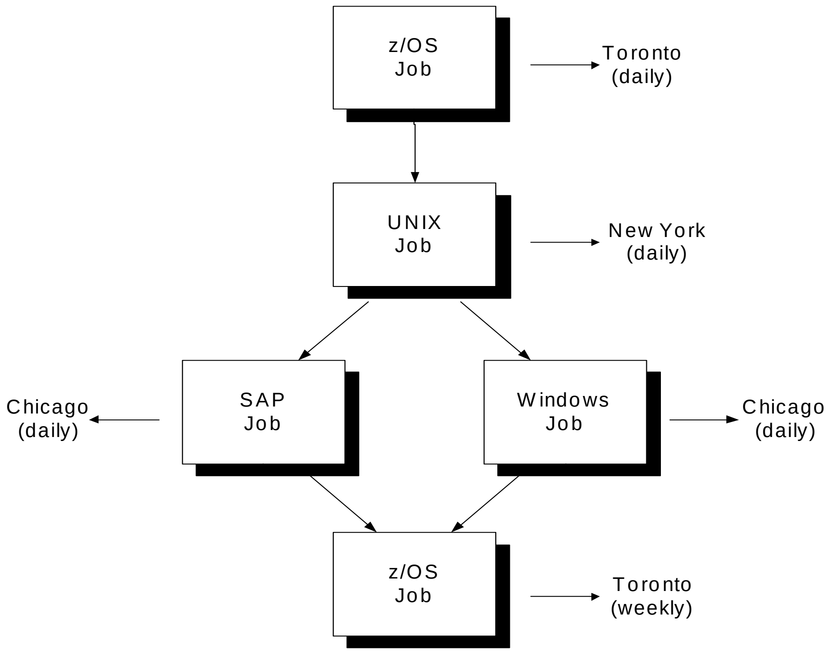
You schedule a distributed job on ESP Workload Manager. When the job is selected to run, ESP Workload Manager passes a request to the appropriate Agent, which submits the job on the distributed system.

Agents receive commands from ESP Workload Manager, carry out those commands, and transmit data and messages back to ESP Workload Manager. The following table summarizes the relationship between the ESP host and the Agents it works with:

ESP Workload Manager	The Agent
Is aware of the entire network	Is aware of the local environment
Issues commands and sends messages to Agents	Responds to commands and messages sent by ESP Workload Manager
Receives data from Agents	Transmits data to ESP Workload Manager
Makes decisions	Is directed by ESP Workload Manager

Example—Running distributed workload

The following job flow shows z/OS jobs, a UNIX job, an SAP job, and a Windows job running on different machines, in different locations, and at different times. To run workload on these distributed systems, you need to install ESP Agents on them.



Communication between ESP Workload Manager and ESP Agents

ESP Workload Manager and ESP Agent communicate by sending Automated Framework Messages (AFMs) to each other. Communication is asynchronous using message queues via TCP/IP ports. Some Agents before version R5 and all R5 Agents and above require encrypted communications.

Receiver ports

ESP Workload Manager and ESP Agents each have TCP/IP ports to receive messages. The receiver listens on its designated port for a message from one or more senders. When the sender has messages to transmit, it connects to the receiver's port, sends the messages, and closes the connection.

Receiver port configuration is restricted as follows:

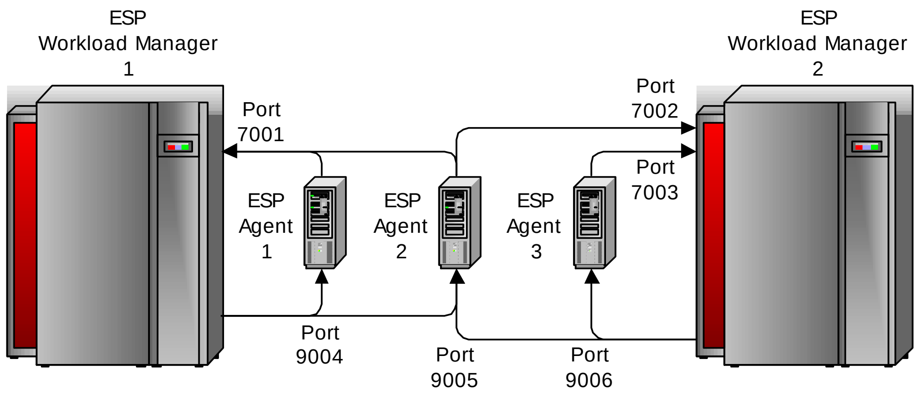
- ESP Workload Manager can have multiple receiver ports. Each of these ports can receive messages from multiple Agents.
- An ESP Agent has only one receiver port. This port can receive messages from multiple ESP Workload Managers.

Note: When there is a need to limit ESP Agent connections to a single TCP/IP address, you can specify the BIND operand in the MANAGER initialization parameter, and the HOME operand in the AGENTRCV initialization parameter. Using these operands guarantee that all ESP Workload Manager TCP/IP connections with ESP Agents will have the same TCP/IP address.

Example—Communication configuration

The following diagram shows some possible communication configurations between ESP Workload Manager and ESP Agents. Note that

- ESP Workload Manager 1 communicates with ESP Agent 1 and ESP Agent 2. It receives messages from both Agents via port 7001. It sends messages to port 9004 on Agent 1 and port 9005 on Agent 2.
- ESP Workload Manager 2 communicates with ESP Agent 2 and ESP Agent 3. It receives messages from Agent 2 via port 7002 and from Agent 3 via port 7003. It sends messages to port 9005 on Agent 2 and port 9006 on Agent 3.



Note: All ports shown are receiver ports

Configuring the ESPPARM Data Set

In the ESPPARM data set, code the Agent-related initialization parameters listed in this section. The ESPPARM data set is created from the sample library (see “Sample Library Members Containing Initialization Parameters” on page 42).

- AGENTRCV

  The AGENTRCV initialization parameter defines the ESP Workload Manager receiver for communication with Agents.

- COMMQ

  The COMMQ initialization parameter specifies the name of the VSAM LINEAR data set used to store information about communication between Agents and ESP Workload Manager.

- LOADAGDF

  The LOADAGDF initialization parameter loads the AGENTDEF data set when ESP Workload Manager starts. For details on the AGENTDEF data set, see “Configuring the AGENTDEF Data Set” on page 107.

- MGRADDR (optional)

  The MGRADDR initialization parameter notifies one or more Agents of the ESP Workload Manager IP address and port number, so the Agent knows the Agent receiver to connect to.

- WOBDEF

  The WOBDEF initialization parameter specifies the workload object modules that are loaded when ESP Workload Manager starts. You specify a workload object corresponding to each Agent type you want to use with ESP Workload Manager.

Configuring the AGENTDEF Data Set

In the AGENTDEF data set, code the initialization parameters listed in this section. The AGENTDEF data set is created from the sample library (see “Sample Library Members Containing Initialization Parameters” on page 42). For examples, see “Examples of an AGENTDEF Data Set” on page 115.

Certain settings in the initialization parameters must match settings in the Agent parameter file agentparm.txt. For a list of these settings, see “Relationship of ESP Workload Manager initialization parameters to Agent parameters” on page 109.

Code the following initialization parameters:

- MANAGER

  The MANAGER initialization parameter defines the ESP Workload Manager subsystem that controls the Agents.

  Code the MANAGER initialization parameter first.

- ENCRYPT (optional)

  The ENCRYPT initialization parameter specifies encryption for communication between an ESP Workload Manager subsystem and one or more Agents.

  If needed, code one or more ENCRYPT initialization parameters

  - After the MANAGER initialization parameter.
  - Before any AGENT initialization parameters that ENCRYPT applies to.

  For details, see “Encrypting Communications” on page 110.

- AGENT

  The AGENT initialization parameter defines an Agent that is controlled by ESP Workload Manager.

  Code each AGENT initialization parameter

  - After the MANAGER initialization parameter
  - After an ENCRYPT initialization parameter that applies to the AGENT initialization parameter

  For details about encryption, see “Encrypting Communications” on page 110.

- MAPUSER (optional)

  The MAPUSER initialization parameter maps distributed system user IDs to authorized mainframe user IDs.

  If needed, code the MAPUSER initialization parameter anywhere in the AGENTDEF data set.

Relationship of ESP Workload Manager initialization parameters to Agent parameters

Some ESP Workload Manager initialization parameter operands must match Agent parameters specified in the agentparm.txt file on the system where each Agent runs. The relationship is given in the following table. Some table entries below deal with encrypting communication; for details, see "Encrypting Communications" on page 110.

ESP Workload Manager Initialization Parameters	Agent Parameters
AGENTDEF Data Set	**agentparm.txt file**
`MANAGER`	
`[NAME(`*mgrname*`)]`	`communication.managerid_`*n* `=` *mgrname*
`[TCPIP[(`*ipaddress*`)]]`	`communication.manageraddress_`*n* `=` *ipaddress*
`ENCRYPT`	
`KEY(`*key*`)|`	`security.cryptkey =` *key*
`KEYNAME(`*keyname*`)` *	`security.cryptkey =` *path*`\cryptkey.txt` *
`[ALL|NOALL]`	
`AGENT`	
name	`agentname`
`...`	
`{PORT(`*port*`)}`	`communication.inputport`
`...`	
`[ENCRYPT|`	
`ENCRYPT(`*key*`)|`	`security.cryptkey =` *key*
`KEYNAME(`*keyname*`)` *	`security.cryptkey =` *path*`\cryptkey.txt` *
`NOENCRYPT]`	
ESPPARM Data Set	
`AGENTRCV`	
`[PORT(`*port*`)]`	`communication.managerport_`*n*

* The key referenced by ESP Workload Manager operand *keyname* must match the key stored in Agent file *path*\cryptkey.txt, where *path* is the directory path. For details, see "Securing the encryption key on ESP Workload Manager" on page 112.

Encrypting Communications

This section shows you how to encrypt communication between ESP Workload Manager and Agents.

Types of encryption

Some Agents before version R5 and all R5 Agents and above require encrypted communication with ESP Workload Manager using one of the following methods:

- DES (Data Encryption Standard) encryption using a 56-bit key

  DES encryption is included in the Agents.

- Strong encryption using the 256-bit key Blowfish algorithm

  Strong encryption is an option you can add to the Agents and ESP Workload Manager.

Setting up strong encryption

To set up strong encryption

- Install Cybermation Programming Environment FMID SCP5101 on ESP Workload Manager.
- Set up strong encryption on the Agent (for details, see the administration guide for the Agent).

The encryption key

ESP Workload Manager and Agents cannot communicate unless they use the same key to encrypt and decrypt communication.

You specify the key in the AGENTDEF data set for ESP Workload Manager and in the Agent parameter file (agentparm.txt) on the Agent system. For details, see “Configuring encryption in the AGENTDEF data set” on page 111 and the administration guide for the Agent.

You can secure the key specified on ESP Workload Manager and Agents by encrypting them. For details, see “Securing the encryption key on ESP Workload Manager” on page 112 and the administration guide for the Agent.

Configuring encryption in the AGENTDEF data set

Encrypting individual Agents

To encrypt communications for an individual Agent, specify the key in the ENCRYPT operand of the AGENT initialization parameter, for example

```
AGENT CYBUNIX ADDRESS(113.11.11.11) Port(3990)+
UNIX ASCII TCPIP ENCRYPT(X'1111222233334444')
```

Encrypting multiple Agents using the same key

To encrypt communications for multiple Agents using the same key, first code the ENCRYPT initialization parameter followed by the AGENT initialization parameters. The ENCRYPT initialization parameter applies to all subsequent AGENT initialization parameters until the next ENCRYPT initialization parameter.

Example — Encrypting multiple Agents

In the following example, the first ENCRYPT initialization parameter encrypts Agents A and B using key X’1111111111111111’) and the second ENCRYPT initialization parameter encrypts Agents C and D using key X’2222222222222222’).

```
ENCRYPT KEY(X'1111111111111111') ALL
AGENT A...
AGENT B...
ENCRYPT KEY(X'2222222222222222') ALL
AGENT C...
AGENT D...
```

Controlling encryption for each Agent

The following table shows you how to control encryption for each Agent following an ENCRYPT initialization parameter.

ENCRYPT Initialization Parameter Operand	Result	Key
ALL	Encrypt communications for all Agents except those defined with AGENT...NOENCRYPT.	Use the key in the ENCRYPT initialization parameter except for Agents defined with AGENT...ENCRYPT(*key*).
NOALL	Encrypt communications only for Agents defined with AGENT...ENCRYPT or AGENT...ENCRYPT(*key*).	Use the key in the ENCRYPT initialization parameter except for Agents defined with AGENT...ENCRYPT(*key*).

Example — Controlling encryption for each Agent

In the following example, Agent A and B are encrypted using key X'1111111111111111', Agent C is encrypted using its own key, X'2222222222222222', and Agent D is not encrypted.

```
ENCRYPT KEY(X'1111111111111111') ALL
AGENT A...ENCRYPT
AGENT B...
AGENT C...ENCRYPT(X'2222222222222222')
AGENT D...NOENCRYPT
```

In the following example, Agent A is encrypted using key X'1111111111111111', Agent B and D are not encrypted, and Agent C is encrypted using its own key, X'2222222222222222'.

```
ENCRYPT KEY(X'1111111111111111') NOALL
AGENT A...ENCRYPT
AGENT B...
AGENT C...ENCRYPT(X'2222222222222222')
AGENT D...NOENCRYPT
```

Securing the encryption key on ESP Workload Manager

You can secure a communications encryption key by creating a key name for it. You then specify the key name instead of the actual key in the AGENTDEF data set. This prevents anyone with access to the AGENTDEF data set from seeing the encryption key.

Note: To use a strong encryption key, you must create and specify a key name of type BLOWFISH for the key.

To create a key name for an encryption key

- Issue the CRYPTKEY command, specifying the key you want to secure and the type of encryption required.

ESP Workload Manager encrypts the key and stores it with a key name you specify.

For example, you could issue CRYPTKEY to create key name AGTKEY1 for key X'010203030501ADDD'. Then, instead of coding

```
ENCRYPT KEY(X'010203030501ADDD')
```

you can code

```
ENCRYPT KEYNAME(AGTKEY1)
```

Note: When you encrypt the key on ESP Workload Manager, also encrypt the key on the Agent. For details, see the administration guide for the Agent.

Making encrypted communication mandatory

Some Agents before version R5 don't require encrypted communication. However, you can make encrypted communication mandatory by ensuring an Agent receiver in ESP Workload Manager only accepts encrypted messages.

- Code the ENCRYPTEDONLY operand in the AGENTRCV initialization parameter.

You code the AGENTRCV initialization parameter when you get to "Configuring the ESPPARM Data Set" on page 106.

Controlling Access

Note: Contact your mainframe security administrator to grant access to the security profiles discussed in this section. For details about security profiles, see the *ESP Workload Manager Security Guide*.

Allowing users to submit work to an Agent

- Add the required user IDs to the AGENT security profile.

Allowing users to send commands and messages to an Agent

You can authorize users to issue commands and send messages to an Agent, which they do using the AGENTMSG command.

- Add the required user IDs to the AGENTMSG security profile.

Allowing users to specify a user ID under which jobs are submitted to an Agent

You can authorize specified users to submit jobs to an Agent under a user ID specified in the USER statement. The USER statement is coded in the ESP Workload Manager Procedure that submits the jobs.

- Add the required user IDs to the AGENTUSR security profile.

For example, you can authorize users JSMITH and VLEE to submit jobs that specify USER(PROD) to Agent HP51 by adding JSMITH and VLEE to an AGENTUSR security profile.

Specifying a password to enable an Agent to run workload

Before it can run workload, an Agent might require a password to log on to the distributed system it runs on. The Agent logs on under the user ID specified in the Procedure's USER statement.

To specify a password for the Agent's user ID

- Issue the PASSWORD command.

  The PASSWORD command securely stores a user ID and encrypted password pair. You can also issue the PASSWORD command to update and delete passwords, and list user IDs that have passwords defined.

Note: Ensure the password and user ID match the ones set up on the distributed system running the Agent.

Example

```
PASSWORD DEFINE USER(jdoe) PASSWORD(A2B3C) TYPE(NT_JOB)
```

When a USER statement is coded in an NT_JOB definition, a search occurs for an entry that matches the specified user ID with an NT_JOB and a qualifier that is the same as the Agent name specified in the ESP Workload Manager Procedure. This search returns the password that most closely matches these parameters. If there are no matching entries, an entry with no type or qualifier is returned.

Allowing a user to update or delete a password without specifying the old password

- Create a PASSWORD security profile for the user ID and give that user ID update access to the profile.

Allowing users to update or delete an encrypted key without specifying the old encrypted key

You can allow users to issue the CRYPTKEY command to update or delete an encrypted key without having to specify the old encrypted key.

- Add the required user IDs to the PASSWORD security profile.

Allowing users to specify distributed system user IDs corresponding to mainframe user IDs

To authorize messages sent to ESP Workload Manager using the ESPmgr command, distributed system user IDs must be mapped to mainframe user IDs.

To map distributed system user ID to mainframe user IDs, do one of the following:

- Specify the user IDs to be mapped in the MAPUSER initialization parameter in the AGENTDEF file.

  This method requires access to the AGENTDEF data set. For details, see "Configuring the AGENTDEF Data Set" on page 107.

- Issue the OPER command MAPUSER.

Controlling Agent access to ESP Workload Manager

When ESP Workload Manager receives an Agent message, it checks to see if the distributed system user ID has the required access. This access is defined in the ESP Workload Manager security profiles. For details on security profiles, see the *ESP Workload Manager Security Guide*.

The access defined by a specific security profile is granted if the distributed system user ID in the Agent message is

- The same as an authorized mainframe user ID defined in the security profile.
- Mapped to an authorized mainframe user ID defined in the security profile.

  You map user IDs with the MAPUSER initialization parameter or command. For details, see "Configuring the AGENTDEF Data Set" on page 107.

Examples of an AGENTDEF Data Set

In the following example, the ENCRYPT initialization parameter on the second line specifies the encryption key for communication between ESP Workload Manager and all the Agents except TORAS4001. For communication between ESP Workload Manager and Agent TORAS4001, the encryption key is specified in the ENCRYPT operand of the last AGENT initialization parameter.

```
MANAGER NAME(CM_CENTRAL) TCPIP
ENCRYPT KEY(X'010203030501ADDD')
MAPUSER JDOE TO(ABCJD01) AGENT(TORSUN1)
AGENT TORSUN1 ADDRESS(123.45.67.89) PORT(9999) UNIX ASCII +
TCPIP PREFIXING
AGENT TORNT1 ADDRESS(123.67.89.01) PORT(9998) NT ASCII +
TCPIP PREFIXING
AGENT TORAS4001 ADDRESS(123.89.01.23) PORT(9997) AS400 EBCDIC +
TCPIP PREFIXING ENCRYPT(X'010203040506AFFC')
```

The following example is the same as the preceding example, except key names AGTKEY1 and AGTKEY2 are used in place of the actual keys. For details on key names, see "Controlling Access" on page 113.

```
MANAGER NAME(CM_CENTRAL) TCPIP
ENCRYPT KEYNAME(AGTKEY1)
MAPUSER JDOE TO(ABCJD01) AGENT(TORSUN1)
AGENT TORSUN1 ADDRESS(123.45.67.89) PORT(9999) UNIX ASCII +
TCPIP PREFIXING
AGENT TORNT1 ADDRESS(123.67.89.01) PORT(9998) NT ASCII +
TCPIP PREFIXING
AGENT TORAS4001 ADDRESS(123.89.01.23) PORT(9997) AS400 EBCDIC +
TCPIP PREFIXING ENCRYPT KEYNAME(AGTKEY2)
```

Displaying Agent Definitions

- Issue the AGENT command (Display Agent Definition).

Displaying Agent communications activity

You can display communications activity between the ESP Agents and ESP Workload Manager version 5.3 and higher.

- Issue the ESPCOM LIST command.

Enabling Agent Configurations

When you configure ESP Workload Manager for new Agents or update the configuration for existing Agents, you must load the configuration into ESP Workload Manager. You can do this while ESP Workload Manager is active or inactive.

Enabling configurations while ESP Workload Manager is inactive

- Restart ESP Workload Manager so that it reads the initialization parameters.

Enabling configurations while ESP Workload Manager is active

1. If you want the new or updated configuration to load the next time ESP Workload Manager restarts, code the configuration in the Agent-related initialization parameters.

   See

   - “Configuring the ESPPARM Data Set” on page 106
   - “Configuring the AGENTDEF Data Set” on page 107
2. Issue the Agent-related commands for the Agent configuration you want.

   If you edited the Agent-related initialization parameters, ensure the commands you issue reflect the settings in those initialization parameters.

   You can issue the following commands:

   - AGENTRCV
   - LOADAGDF
   - MGRADDR
   - WOBDEF

Note: WOBDEF LOAD loads new workload object modules only. To update existing modules, restart ESP Workload Manager.

Testing Agents

Testing communication

For communication to occur between ESP Workload Manager and Agents, each must have a valid IP address and a unique port.

Use the PING and NETSTAT commands to test network communication between ESP Workload Manager and ESP Agents.

Testing workload processing

In this section, you perform an end-to-end functional test of your ESP Workload Manager configuration for using Agents by initiating and tracking workload on an Agent. When you submit the workload

1. ESP Workload Manager looks in the AGENTDEF data set for the Agent's address and port.
2. ESP Workload Manager sends a message to the defined address and port.
3. If the Agent returns an acknowledgment, the status of the job changes from

   ```
   QUEUED FOR SUBMISSION
   ```

   to

   ```
   AGENT NOTIFIED
   ```

   If the address and port defined in the AGENTDEF file are incorrect, the status remains

   ```
   QUEUED FOR SUBMISSION
   ```

To set up and run the workload test

1. Ensure that ESP Workload Manager and the Agent are active.
2. Create a simple UNIX script or Windows batch file on the system where the Agent runs.
3. Create a test procedure that runs the UNIX script or Windows batch file you created in the preceding step.

   See the examples following this procedure.
4. Create an Event to invoke the procedure you created in the preceding step.
5. Simulate the Event and correct errors.

6. Trigger the Event and use CSF or ESP Workstation to monitor the Application's progress.

   You should see a series of status changes on the CSF lines for your job as the Application progresses. Also, you should be able to confirm that the Agent is running the script or batch file.

Example — a UNIX Agent test procedure

```
APPL TSTUNXAG
UNIX_JOB TSTJOBU
  RUN DAILY
  RELEASE B
  AGENT MY_UNIX_AGENT
  SCRIPTNAME /mfg/test/sort
ENDJOB
```

Example — a Windows Agent test procedure

```
APPL TSTWINAG
NT_JOB TSTJOBW
  RUN DAILY
  AGENT MY_WINDOWS_AGENT
  CMDNAME C:\mfg\test\sort.bat
ENDJOB
```

Part 3

Configuring Multiple ESP Workload Manager Subsystems

When your installation requires multiple systems running ESP Workload Manager, or multiple copies of ESP Workload Manager running on a single z/OS image, you need to perform some additional tasks to configure the ESP Workload Manager subsystems. This part describes the steps you need to perform to configure multiple ESP Workload Manager subsystems.

This part contains the following chapters:

- Scheduling and Monitoring Workload on Multiple z/OS Images
- Configuring Master-Proxy Subsystems in a Sysplex
- Using Shadow Manager and Automatic Restart Management
- Setting Up Communications Between Two or More Masters

MSGPRFX

You can alter the message prefix from ESP Workload Manager to another string of up to seven characters in length (the default value is ESP). The message prefix helps to identify the source of a message when more than one ESP Workload Manager subsystem is active. Code the prefix on the MSGPRFX initialization parameter.

To Initialize MSGPRFX:

Code this initialization parameter first in the initialization parameter data set because it only applies to features specified after it, not to what is specified ahead of it.

12

Scheduling and Monitoring Workload on Multiple z/OS Images

This chapter describes how to create an ESP Workload Manager master and proxy configuration to track workload on more than one z/OS image.

Note: Before using this chapter, you must have ESP Workload Manager installed with a single master subsystem.

This chapter contains the following topics:

- Masters and Proxies
- Adding and Configuring Proxies
- Ensuring the Integrity of Shared Data Sets
- Performance Considerations

Masters and Proxies

The ESP Workload Manager concept of a master and proxies allows you to process daily workload on multiple z/OS images with a central point of control: the ESP Workload Manager master subsystem.

Master

The master controls

- The ESP Workload Manager data sets
- Job release

The master communicates with the proxies through the queue data set or XCF.

Proxy

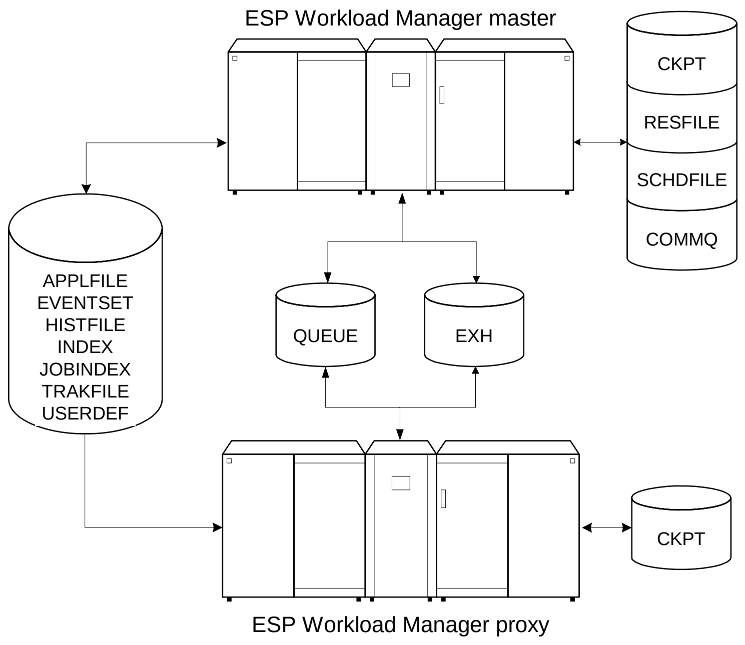
Each proxy

- Runs any or all workload as required
- Tracks the execution of the workload through SMF
- Provides a limited user interface via a subset of Line mode commands and Application Status panels

The proxies communicate with the master through the queue data set or XCF.

Data set sharing

The following diagram illustrates the way a master and proxy share and access data sets:



Data sets specific to the master subsystem

The following data sets can only be allocated on the master subsystem:

- AGENTDEF
- RESFILE
- SCHDFILE

Data set unique to each ESP Workload Manager subsystem

The following data set must exist on every ESP Workload Manager subsystem:

- CKPT

Data set that must be shared

Master and proxy subsystems must share the following data set:

- INDEX
- QUEUE

Data sets that can be shared

Master and proxy subsystems can share the following data sets, depending on the how you configure your systems:

- APPLFILE — should be shared if the proxy subsystem generates applications
- INDEX
- JOBINDEX — should be shared if the proxy subsystem executes commands
- TRAKFILE — does not need to be shared

The proxy subsystems can have UPDATE access to these data sets.

The proxy subsystems must have at least READ access to the APPLFILE.

If either APPLFILE or JOBINDEX is shared, all must be.

Performance considerations

If performance is a problem when the master and proxy share data sets, with the exception of the queue data set, mark the data sets in the initialization parameters on the master NOSHR and mark the data sets in the initialization parameters on the proxy NOSHR READONLY.

Adding and Configuring Proxies

Configuring the master for a proxy

You need to specify options on the master to allow access to data sets and enable communication between the master and proxies.

On the ESP Workload Manager master subsystem

1. Ensure the TRACKOPT initialization parameter specifies MASTER.
2. Ensure the following initialization parameters specify SHR:
    - APPLFILE
    - EVENTSET
    - HISTFILE
    - INDEX
    - JOBINDEX
    - QUEUE
    - TRAKFILE
    - USERDEF

Important: The queue data set must be shared for the master and proxies to communicate.

Do not share the following data sets:

- CKPT
- RESFILE
- SCHDFILE

JOBINDEX data set

Specify SHAREOPTIONS(3,3) when defining the jobindex data set.

USERDEF data set and initialization parameter

Will the same collection of users, with the same authorities and functions, access ESP Workload Manager across all systems at your site? If so, a common user definition data set should be shared across all systems. Specify SHR on the USERDEF initialization parameter and SHAREOPTIONS(3,3) on the user definition data set.

EVENTSET data set and initialization parameter

Event databases are associated with ESP Workload Manager users or groups of users. For those Event definition data sets that are shared across all systems, specify SHR on the appropriate EVENTSET initialization parameters, and SHAREOPTIONS(3,3) on the Event data set IDCAMS statements.

INDEX data set and initialization parameter

Unless data set triggering, calendars, global variable tables, passwords, and cryptkeys are limited to specific subsystems, you should share the index data set. Specify SHR on the INDEX initialization parameter, and SHAREOPTIONS(3,3) on the index data set IDCAMS statements.

Adding a proxy

You need to specify that the ESP Workload Manager subsystem is a proxy.

- On each proxy subsystem, ensure the TRACKOPT initialization parameter specifies PROXY.

Example

The following example shows how to specify that an ESP Workload Manager subsystem is a proxy:

```
TRACKOPT PROXY
```

For details about the rest of the data in the TRACKOPT initialization parameter, see “TRACKOPT: Set tracking options” on page 527.

Configuring additional initialization parameters for both the master and proxies

1. Specify the ESPGROUP initialization parameter to access ESP Encore and to use CCCHK. All subsystems must have the same ESPGROUP. For more information, see “ESPGROUP: Specify ESP complex ID” on page 341.
2. In the RSVLOGIC initialization parameter, specify how subsystems should access shared data sets (the serialization option).

   For example, if you want a hardware reserve (locking the DASD while one ESP Workload Manager is accessing the data set) to occur, specify RESERVE. For more information, see “Ensuring the Integrity of Shared Data Sets” on page 127.

   If you are using MIM or GRS or you would like more information about the RSVLOGIC initialization parameter, see “RSVLOGIC: Reserve logic” on page 471.
3. Specify which subsystems capture SMF records using the SMFINT initialization parameter. Each system must capture its records, unless you are running a master and proxy on the same z/OS image. In that case, capture SMF on the proxy.

   If you have a master and a proxy on the same JES image, and the proxy captures SMF records, the master is portable to another subsystem when necessary.

   To specify SMFINT on a proxy

   - Specify the following on the master:
     `SMFINT OFF`
   - Specify the following on the proxy:
     `SMFINT ON`

   For more information on SMFINT, see “SMFINT: Specify SMF interface” on page 484.
4. If you have not specified a SYSID for the master, do so now.

   The SYSID must be unique in the z/OS image. For more information, see “SYSID: Specify system identifier” on page 496.
5. Specify access criteria to the QUEUE data set using the following initialization parameters:

   - MINDORM
   - MAXDORM
   - MINHOLD

   These initialization parameters define the amount of time each ESP Workload Manager has access to the queue data set, and how long it must wait before it has access again. The settings you choose can dramatically affect your ESP Workload Manager subsystem’s performance.

Ensuring the Integrity of Shared Data Sets

ESP Workload Manager uses serialized access to ensure the integrity of shared data sets when multiple access is requested in a multiple-CPU environment. In this case, ESP Workload Manager uses one of the following serialization techniques:

- RESERVE/RELEASE
- ENQ/DEQ

Serialization techniques ESP Workload Manager uses

RESERVE/RELEASE

RESERVE/RELEASE uses a hardware lock to force serialization. However, under adverse conditions, RESERVE/RELEASE might lock out other applications that use the same pack. Also, this technique requires that data sets in the ESP Workload Manager database reside on the same volume to avoid a z/OS RESERVE lockout and ensure the data sets are synchronized in the event of a full-volume restore.

ENQ/DEQ

ENQ/DEQ is available if you have GRS or a similar product active for cross-system serialization. A cross-system ENQUEUE is used, which locks a single data set only and not the entire volume on which the data set resides. You can use ENQ/DEQ when you spread data sets out over several online volumes. No deadly embrace occurs because a single ENQ locks the entire logical database.

Specifying the serialization technique

1. Code the RSVLOGIC initialization parameter to specify the appropriate serialization technique.
2. If you are using Multi-Image Manager (MIM), add the QNAME from RSVLOGIC to your MIM inclusion table.

   For more information, see “RSVLOGIC: Reserve logic” on page 471.

Performance Considerations

Because the master and each proxy must update the queue data set periodically, and have exclusive access to it, you need to specify criteria that defines how the ESP Workload Manager subsystems share the data set. For example, how long a system must wait for access, a minimum (MINDORM) and maximum (MAXDORM) amount of time, and for how long the system can hold the data set before it must relinquish it to another system (MINHOLD).

MINDORM

MINDORM specifies the minimum amount of time a system waits for exclusive access to the queue data set.

MAXDORM

MAXDORM specifies the maximum amount of time a system waits for exclusive access to the queue data set. When this time elapses, ESP Workload Manager forces a read of the queue data set. This value is critical to the performance of proxy systems.

MINHOLD

MINHOLD specifies the amount of time a system has exclusive access to the queue data set.

Relationship

MINHOLD specifies exactly how long a system holds the queue data set each time it acquires it. When it is released, it waits at least the value specified in MINDORM before acquiring it again. After MINDORM time elapses, the system seeks control of the queue data set when it has a need to add to it, or when the time specified in MAXDORM has elapsed.

Because most of the flow is from proxy to master, the master seldom needs to write data to the queue data set; therefore, its MAXDORM should be small, not much larger than its MINDORM value. Proxies can have a fairly large MAXDORM, but not more than several seconds.

To determine the required values for these initialization parameters

1. Determine the required MINHOLD values.
2. Determine the MINDORM values.
3. Determine the MAXDORM values.

To determine the MINHOLD values, you need to consider the activity expected on the master and proxies. MINHOLD values should realistically reflect the volume of activity each system has relative to the others. For example, if two proxies forward data to the master, and proxy A is twice as busy as proxy B, you might assign MINHOLD values (in centiseconds) as follows:

Proxy A 100, proxy B 50, and the master 200.

Now, add the values up to see how much time is required for each system to have a turn:

100+50+200=350

Each system can have a turn in 3.5 seconds.

Determining MINDORM values

Each sharing system needs a MINDORM value equal to that part of the 3.5 seconds it does not own. Therefore,

The sum of all MINHOLDs - the MINHOLD of this system = MINDORM

From the previous example, the recommended MINDORM values are

Proxy A 350-100=250

Proxy B 350-50=300

Master 350-200=150.

When all systems are working, a subsystem productivity owns the queue data set at all times, the master with the biggest share.

Determining MAXDORM values

When a proxy is idle, it does not need to access the queue data set as often, but MAXDORM should not be too large a multiple of the cycle time. If it is acceptable to have an idle proxy wait 10 cycles before accessing the queue data set to look for work, then you can specify 3500 for each proxy. On the master, however, the MAXDORM time needs to be kept small—probably not more than double the MINDORM. In cases where the master has the SMF interface turned off, MAXDORM should be equal to, or very slightly greater than MINDORM:

```
Proxy A 3500, Proxy B 3500, Master 300.
```

Changing number of systems

If the number of systems changes, use the new sum of MINHOLD values to calculate dormancy values. If the workload balance changes, the relative size of the MINHOLD values must be adjusted and the new dormancy values calculated.

Scenario 1 — master, 1 proxy

In the following scenario, the ESP Workload Manager master is busier than the proxy:

ESP Workload Manager	MINHOLD	MINDORM	MAXDORM
Master	200	100	200
Proxy	100	200	3000

Scenario 2 — master, 1 proxy

In the following scenario, the master and proxy run equal workload:

ESP Workload Manager	MINHOLD	MINDORM	MAXDORM
Master	100	100	200
Proxy	100	100	2000

Scenario 3 — 1 busy proxy, 1 idle

In the following scenario, one proxy is busy, the other idle:

ESP Workload Manager	MINHOLD	MINDORM	MAXDORM
Master	200	150	300
Proxy—busy	100	250	3500
Proxy—idle	50	300	3500

Scenario 4 — master with SMFINT OFF

In the following scenario, the master does not intercept SMF records—the proxy does:

ESP Workload Manager	MINHOLD	MINDORM	MAXDORM
Master	200	100	100
Proxy	100	200	3000

Changing these values

If you are unclear what values to set for MINHOLD, MINDORM, and MAXDORM, or are making environmental changes that affect these values, contact CA Technical Support.

13

Configuring Master-Proxy Subsystems in a Sysplex

This chapter gives you information for configuring ESP Workload Manager to work with the Cross-System Coupling Facility (XCF) in a sysplex environment.

This chapter contains the following topics:

- XCF services
- Configuring ESP Workload Manager subsystems for a sysplex environment
- XCF status monitoring

XCF services

XCF services are functions of ESP Workload Manager that use sysplex architecture. ESP Workload Manager currently registers these four XCF services:

- Data set triggering (DSTRIG)
- Job tracking (TRACKING)
- Routing (ROUTING)
- Scoreboard (SCOREBD)

Note: The ROUTING and SCOREBD services require ESP Workload Manager High Availability Option (HAO) or ESP Workload Manager Service Governor.

Job tracking and data set triggering

ESP Workload Manager uses XCF for job-tracking and data set triggering in a sysplex environment. A proxy transmits job-tracking and data set triggering records to the master using XCF.

However, if XCF services are not enabled, a proxy writes job-tracking and data set triggering records to the QUEUE data set and the master reads those records from the QUEUE data set.

If you use XCF instead of the QUEUE data set for job-tracking and data set triggering information, contention is reduced on the QUEUE data set.

Universal login

Note: Universal login requires ESP Workload Manager High Availability Option (HAO) or ESP Workload Manager Service Governor.

Universal login allows ESP Workload Manager clients connected to a proxy to

- Route subsystem requests to the master (XCF routing service).
- View the CSF and Application Monitor scoreboards (XCF scoreboard service).

With universal login, ESP Workload Manager clients and the ESP Workload Manager master can run on any system in the sysplex.

ESP Workload Manager clients can be TSO/ISPF users or Workstation Servers.

XCF routing service

The ROUTING service is an XCF connection between a proxy and the master that enables ESP Workload Manager clients connected to the proxy to route subsystem requests, usually ESP Workload Manager commands, to the master. The master processes the subsystem requests and routes the command responses back to the ESP Workload Manager client through the local proxy.

Note:

- To route subsystem requests to the master from a proxy, the current routing status must be set to MASTER. Issue the ROUTING STATUS command to display your current routing status. To set your current routing status to MASTER, issue the ROUTING MASTER command.
- We recommend that the default routing be changed from LOCAL to MASTER during installation. For more information, see “ROUTING: Route ESP Workstation requests” on page 470. You can also set the default routing in the ISPF client. For more information, see “Making the ESP Workload Manager subsystem available from ISPF” on page 58.

XCF scoreboard service

The SCOREBD service is an XCF connection between a proxy and the master that enables ESP Workload Manager clients connected to the proxy to view the CSF and Application Monitor scoreboards.

When a proxy’s XCF SCOREBD service connects to the master, the master transmits the contents of the scoreboards to the proxy. Thereafter, the master transmits every scoreboard update to all proxies connected to it through the XCF SCOREBD service.

Enabling XCF services

XCF services are registered and started with the XCF START SERVICE initialization parameters. These initialization parameters must be coded in the initialization parameter data set. XCF services must be coded in all the subsystems in the XCF group.

When an XCF service is coded and started on the master, a listener is activated. When an XCF service is started on a proxy, the XCF service connects to the master’s listener for that service.

To start XCF services

The following is the syntax for the XCF START SERVICE initialization parameters in the initialization parameter data set:

```
XCF START SERVICE DSTRIG
XCF START SERVICE TRACKING
XCF START SERVICE ROUTING
XCF START SERVICE SCOREBD
```

Note: The ROUTING and SCOREBD services require ESP Workload Manager High Availability Option (HAO) or ESP Workload Manager Service Governor.

Configuring ESP Workload Manager subsystems for a sysplex environment

ESP Workload Manager uses the Cross-System Coupling Facility (XCF) component of z/OS to provide service to the entire sysplex.

To use ESP Workload Manager in a sysplex environment

1. Determine the name you want to call your XCF group.

   The XCF group name is specified in the SYSPLEX initialization parameter.

2. Code the SYSPLEX initialization parameter in the initialization parameter data set.

   The SYSPLEX initialization parameter must be coded in all subsystems (master and proxies) in the XCF group. For more information, see "SYSPLEX: Define an XCF Group and Its Members" on page 501.

3. Code the following XCF START SERVICE initialization parameters in the initialization parameter data set.

   ```
   XCF START SERVICE DSTRIG
   XCF START SERVICE TRACKING
   XCF START SERVICE ROUTING
   XCF START SERVICE SCOREBD
   ```

   Note: The ROUTING and SCOREBD services require ESP Workload Manager High Availability Option (HAO) or ESP Workload Manager Service Governor.

   The XCF START SERVICE initialization parameters must be coded in all subsystems (master and proxies) in the XCF group. For more information, see "XCF START SERVICE: Enable XCF connection" on page 553.

XCF status monitoring

Note: XCF status monitoring requires ESP Workload Manager High Availability Option (HAO) or ESP Workload Manager Service Governor.

XCF status monitoring monitors XCF members' activity for any possible problems. An XCF member requests XCF status monitoring when it joins the XCF group.

When XCF status monitoring is enabled, ESP Workload Manager notifies XCF of an interval period. The interval period is specified in seconds. ESP Workload Manager then updates its status with XCF according to the specified interval.

XCF status monitoring is enabled by coding the INTERVAL operand on the SYSPLEX initialization parameter.

How XCF status monitoring works

If a subsystem has not updated its XCF status monitoring field for at least two intervals, XCF will warn the other members in the XCF group of a possible problem with that subsystem. There can be either a loop or hang in the ESP Workload Manager main task. Message ESP4316I is issued.

```
ESP4316I XCF status update missing, possible main task loop or hang
```

Note: This message might not necessarily indicate a problem with the subsystem. For example, the system that ESP Workload Manager is running on can be heavily loaded, and z/OS might not be dispatching ESP Workload Manager.

Second warning message

When XCF does not receive a positive response from the failing subsystem, XCF will broadcast to all the other XCF members in the XCF group that a status update missing condition exists for that member.

Message ESP4331W is issued:

```
ESP4331W XCF status update missing
```

XCF member failure

If the failing subsystem terminates, message ESP4317I is issued:

```
ESP4317I XCF member state change
```

Normal operating status resumed

If the failing subsystem resumes its normal operating status, message ESP4318I is issued:

```
ESP4318I XCF status update resumed
```

14

Using Shadow Manager and Automatic Restart Management

This chapter describes how shadow manager and support for the IBM Automatic Restart Management (ARM) function of z/OS can increase enterprise IT performance by improving the availability of system resources throughout the organization.

Note: Shadow Manager and Automatic Restart Management require ESP Workload Manager High Availability Option (HAO).

This chapter contains the following topics:

- Shadow manager
- ARM registration

Shadow manager

Note: Shadow manager requires ESP Workload Manager High Availability Option (HAO).

Shadow manager allows your environment to shift the workload to another master, so processing can continue should a system fail.

Shadow manager is an ESP Workload Manager subsystem that reads the same initialization parameters as the master subsystem, joins an XCF group, and monitors the master subsystem in that XCF group for the master's termination. When the master subsystem terminates, the shadow manager automatically takes action based on its shadow goal for that type of termination. A shadow manager can also take action from a direct command to take over as the master subsystem.

A master subsystem can be shadow-enabled or shadow-disabled.

Shadow-enabled master subsystem

A shadow-enabled master subsystem is a subsystem that will become a shadow manager if, when it is started, it detects that an active master subsystem already exists in its XCF group.

Shadow-disabled master subsystem

A shadow-disabled master subsystem is a subsystem that will terminate if, when it is started, it detects that an active master subsystem already exists in its XCF group.

Data set usage

Shadow managers must

- Read the same initialization parameter data set as the master subsystem.
- Have the same checkpoint data set as the master subsystem.
- Have the same subsystem ID as the master subsystem.

The master subsystem and shadow manager subsystems must have different XCF member names.

Shadow manager START parameters

A master subsystem is defined as shadow-enabled through either the PRIMARY or SHADOW parameters. These parameters can be specified in the START command or in the started task JCL.

For information on using the START command, see the *ESP Workload Manager Operator's Guide*.

PRIMARY parameter

The PRIMARY parameter specifies that the subsystem is a shadow-enabled master if no other master subsystem is active. If a master is already active in the XCF group, the subsystem is a shadow manager. If PRIMARY is specified and SHADOW is omitted, the shadow number defaults to zero.

SECONDARY parameter

The SECONDARY parameter specifies that the subsystem is a shadow manager. If SECONDARY is specified and SHADOW is omitted, the shadow number defaults to 1, if it is the first shadow manager or to a subsequent number otherwise.

SHADOW parameter

The SHADOW parameter specifies a shadow number between 0 and 31.

The symbolic variable %SHADOW is set to this value. If SHADOW is specified without a number, zero is the default. If SHADOW is specified and both PRIMARY and SECONDARY are omitted, then the subsystem becomes

- A shadow-enabled master if its shadow number is zero.
- A shadow manager if its shadow number is between 1 and 31.

Usage notes

To avoid confusion between a shadow-enabled master subsystem and shadow managers, and to ensure the master subsystems and shadow managers are all assigned unique shadow numbers, we recommend that

- Both the PRIMARY and SHADOW parameters be explicitly specified when defining a shadow-enabled master subsystem.
- Both the SECONDARY and SHADOW parameters be explicitly specified when defining a shadow manager.

We also recommend that you use the SHADGOAL initialization parameter to ensure that, if a shadow manager takes over as the master subsystem, another shadow manager is started, so that you always have at least one shadow manager active.

Symbolic variable: %SHADOW

%SHADOW is a symbolic variable that can be used anywhere in the initialization parameters. The %SHADOW symbolic variable is created only if the SHADOW, PRIMARY or SECONDARY parameters are used in the START parameters or in the started task JCL for the subsystem. This includes the use of default values.

Using the %SHADOW symbolic variable is one way to ensure the master and shadow managers have different XCF member names when they read the same initialization parameter data set.

For example:

```
SYSPLEX GROUP(ESP) MEMBER(ESPM%SHADOW)
```

Shadow goals

A shadow goal is a set of actions that a shadow manager is instructed to perform when the master terminates.

How a shadow goal works

A shadow manager can have one shadow goal for each way the master can terminate. A shadow manager can have a QUIESCE shadow goal, a LEAVE shadow goal, and a FAIL shadow goal.

A shadow manager takes action automatically only if it has a corresponding shadow goal for the way in which the master terminates. For example, if the master terminates and enters an XCF quiesced state, the shadow manager must have a QUIESCE shadow goal to take any specified action.

In addition to specifying actions that a shadow manager must perform, a shadow goal also specifies how long the shadow manager must wait before performing the action.

What actions a shadow goal can take

A shadow goal can specify one or more of the following actions to take when the master terminates:

- Issue warning message #4397
- Issue a system command
- Take over as the master
- Take over as the master and trigger an Event

XCF member termination

The master (or any XCF group member) can terminate normally or abnormally. A normal termination can be either QUIESCE or LEAVE, as defined by the SYSPLEX initialization parameter. The shadow goal for a normal termination must match what is specified in the SYSPLEX initialization parameter. The shadow goal for an abnormal termination is FAIL.

QUIESCE

The master terminates normally but leaves a trace of its membership in the group behind. The command XCF DISPLAY GROUP will show the XCF member in a quiesced state.

LEAVE

The master terminates normally but leaves no trace of its membership in the group behind. The XCF member goes into an undefined state. The command XCF DISPLAY GROUP will not show the XCF member because it no longer exists.

FAIL

The master terminates abnormally (ABENDs). The command XCF DISPLAY GROUP will show the XCF member in a failed state.

SHADGOAL

You can code SHADGOAL in the initialization parameter data set of a shadow manager or a shadow-enabled master. You can also issue the SHADGOAL command dynamically in a shadow manager with the z/OS MODIFY command. You can also use the SHADGOAL command to delete a shadow goal or to list the shadow goals.

Examples

The following example specifies that 180 seconds (3 minutes) after the master terminates and enters an XCF quiesced state, the shadow manager issues warning message #4397 and issues z/OS command S ESPM:

```
SHADGOAL MASTER(QUIESCE) AFTER(180) WARN +
COMMAND('S ESPM')
```

The following example specifies that 300 seconds (5 minutes) after the master abnormally terminates, the shadow manager issues warning message #4397 and takes over as the master:

```
SHADGOAL MASTER(FAIL) AFTER(300) WARN TAKEOVER
```

The following example specifies the shadow goal for the LEAVE option is deleted immediately:

```
SHADGOAL DELETE MASTER(LEAVE)
```

Note: Once a shadow-enabled master becomes the active master in the XCF group, all shadow goals for that shadow-enabled master are deleted and the SHADGOAL command ceases to be available for that master.

Manually moving control

You can switch control of the workload to shadow manager without shadow goals defined in your initialization parameters. You can issue an XCF command or an z/OS MODIFY command manually. XCF commands can be issued from the master or from any proxy that belongs to the same XCF group as the target shadow manager.

Important: When you are connected to a proxy, you must have the XCF ROUTING service enabled, and the ROUTING LOCAL command must be issued to instruct the proxy to process subsystem requests itself. Then you can issue the XCF commands from the proxy you are logged on to. For more information on XCF ROUTING, see "XCF routing service" on page 132.

XCF command examples

The following is an example of shutting down the master (ESPM), displaying the group to ensure ESPM is in a QUIESCE state, and then moving control of the workload to shadow manager (ESPS):

```
XCF STOP MEMBER ESPM
XCF DISPLAY GROUP
XCF SET MEMBER ESPS MASTER
```

The following is an example of the master (ESPM) being disabled and moving control of the workload to shadow manager (ESPS):

```
XCF FORCE MEMBER ESPM
XCF DISPLAY GROUP
XCF SET MEMBER ESPS MASTER
```

z/OS MODIFY command

Shadow manager will take control of the workload if the following z/OS MODIFY command is issued to the shadow manager when the master is not active:

```
F jobname,SHADOW TAKEOVER
```

In the preceding command, *jobname* is the name of the shadow manager started task.

ARM registration

Note: Automatic Restart Management requires ESP Workload Manager High Availability Option (HAO).

ESP Workload Manager provides support for the IBM Automatic Restart Management (ARM) function of z/OS.

The ARMELEM initialization parameter enables ESP Workload Manager to register with ARM. If an ESP Workload Manager subsystem fails, it can be automatically restarted with ARM.

How it works

ARM will restart a registered ESP Workload Manager subsystem that fails if an ARM policy (either the IBM default or a user-defined policy) is active on the system where the failure occurs and on the system where the restart is to occur.

If the master fails or the system the master runs on fails, the registered master can be restarted on the same system or restarted on another system in the sysplex.

If a proxy is registered with ARM and the proxy fails, the proxy will be restarted on the same system by default. Restarts should be limited to the system the proxy runs on.

ARM couple data set

The ARMELEM initialization parameter requires that the system ESP Workload Manager runs on be connected to an ARM couple data set.

ARM and shadow manager

If an installation uses shadow manager, it can consider ARM unnecessary for the master and shadow managers. However, ARMELEM co-exists well with shadow manager. The default RESTART operand value for shadow-enabled masters and shadow managers has deliberately been set to ELEMENT_FAILURE, so that restarts on other systems in the sysplex do not occur when the system the ARM-registered ESP Workload Manager is running on fails.

An installation can use ARM instead of shadow manager for master restarts.

Note: Shadow manager does not require an ARM couple data set and ARM policy.

15

Setting Up Communications Between Two or More Masters

This chapter contains the following topics:

- Internodal Processing
- Selecting your Communication Method
- Network Delivery Services (NDS)
- TP Server
- Network Job Entry (NJE)
- Examples

Internodal Processing

For multi-site processing, build all the Applications at the same site (that is, the one you will be monitoring from) and route jobs to the appropriate site. Put the route cards directly into the JCL or request that ESP Workload Manager do so using the ROUTE statement in the ESP Procedure. Alternatively, use ESP Workload Manager resources to carry out this process.

JES numbers

With JES numbers, partition job numbers so different nodes use different ranges of job numbers. For example, use job numbers 1 to 4999 for one node and 5000 to 9999 for another node. With JES for OS/390 Version 2 Release 4, use job numbers from 1 to 65534 so each site (in a two-node environment) can use 32762 job numbers. You benefit by being able to identify a job's source very easily. ESP Workload Manager can handle changed job numbers when one node sends job information to another node.

Example

To submit jobs between both copies of the ESP Workload Manager subsystem, define the other jobs as `MANUAL`. ESP Workload Manager will not submit manual jobs, but will wait for it before releasing a successor. This ensures ESP Procedures that all subsystem uses are identical.

ESP Procedure for internodal tracking

```
JCLLIB ...
APPL XSYS
N1=''
N2='MANUAL'
JOB A %N1
   RELEASE B
   RUN DAILY
JOB B %N2
   ROUTE XEQ(NODE2)
   RELEASE C
   RUN DAILY
JOB C %N1
   RUN DAILY
ENDJOB
```

Setting a variable using SMFID

For NODE2, you can use the same ESP Procedure. However, set the N1 variable to `MANUAL` and the N2 variable to " ". Automate this by testing the SMFID and setting the appropriate variable.

```
JCLLIB ...
APPL XSYS
IF SMFID='A081' THEN -
   DO
     N1=''
     N2='MANUAL'
   ENDDO
   ELSE DO
     N1='MANUAL'
     N2=''
   ENDDO
```

Selecting your Communication Method

You can use the following communication methods to exchange tracking information between multiple ESP Workload Manager master subsystems that exist on separate JES nodes:

- Network Delivery Services (NDS). NDS uses the standard TCP/IP protocol.
- TP Server. TP Server uses the LU6.2 protocol.
- Network Job Entry (NJE). NJE uses JES communication protocol.

Note: If you use ESP Infoserv, you must code the ISCOPT initialization parameter with the LU62 operand even if you want to use NDS for exchanging tracking information between multiple ESP Workload Manager master subsystems.

Network Delivery Services (NDS)

NDS transfers ESP Workload Manager job-tracking information between ESP Workload Manager master subsystems that exist on separate JES nodes, using standard TCP/IP protocol.

Data sets

NETQUEUE data set

NDS requires the NETQUEUE data set to store job-tracking records when an NDS tracking service internodal transmit request is made. This occurs regardless of whether a connection between the local and remote ESP Workload Manager subsystems currently exists. When the ESP Workload Manager on the remote node confirms receipt of an internodal job-tracking record, the record is purged from the NETQUEUE data set.

Also, internodal job-tracking records received from a remote ESP Workload Manager subsystem via the NDS tracking service are stored in the NETQUEUE data set. A received internodal job-tracking record is purged from the NETQUEUE data set after the local ESP Workload Manager subsystem has written the record to its checkpoint data set.

Create your NETQUEUE data set. You can use SSCPSAME(CYBESS01) for creating the NETQUEUE data set.

LOADNET data set

The LOADNET data set includes the initialization parameters related to NDS.

Create your LOADNET data set in the ESP.PARMLIB library. You can use SSCPSAME(CYBESS38) as a sample LOADNET data set.

Configuring ESP Workload Manager for NDS

Code all the initialization parameters in the following table in your ESPPARM data set:

Initialization Parameter	Description
LOADNET	Include a LOADNET initialization parameter in your ESPPARM data set to load your LOADNET data set.
ISCOPT	Include an ISCOPT initialization parameter in your ESPPARM data set to specify the communication method (NJE, LU 6.2 or NETWORK) used to transmit internodal tracking information. **Note:** If you use ESP Infoserv, you must specify ISCOPT LU62.
XMITMDL	Include an XMITMDL ADD MODEL initialization parameter specifying each remote node that requires internodal tracking with the local network node. All nodes defined with a NETDEST initialization parameter in the LOADNET data set must be referenced with a NETNODE operand in the XMITMDL initialization parameter.

Code the initialization parameters listed in the following table in your LOADNET data set.

Initialization Parameter	Description
NETNODE	Defines your local node. Required.
NETDEST	Defines each remote node. Required.
NETQUEUE	Points to your NETQUEUE data set. Required.
NETWORK SET AUTHENTICATE	Authenticates remote node host by name or IP address at the time you connect.
NETAUTH	Defines an authorization profile for each remote host allowed to connect to the local NDS network. Required if you code NETWORK SET AUTHENTICATE.
NETWORK START LISTENER	The TCP port that you want remote nodes to connect to. Required.
NETWORK START CONNECTION	Connects to each defined remote node. Required.
NETWORK START SERVICE ROUTING	For remote ESP Encore requests (ESP Encore 3.1 and higher). Enables routing commands in internodal communications.
NETWORK START SERVICE TRACKING	Enables internodal job tracking. Required.

TP Server

TP Server transfers ESP Workload Manager job-tracking information between ESP Workload Manager subsystems that exist on separate JES nodes, using the LU 6.2 protocol.

Internodal tracking and TP Server

ESP Workload Manager systems running at different SNA nodes can share job tracking information. Therefore, when two nodes are connected via an SNA link, an ESP Workload Manager subsystem executing at one node can be informed of the status of jobs executing at another node.

Using TP Server, you can set up internodal tracking.

Description

TP Server uses LU 6.2 protocol. The SNA nodes are APPC logical units and are not directly associated with a single JES image. An individual ESP Workload Manager subsystem and the TP Server that runs under that ESP Workload Manager subsystem own the SNA nodes. Therefore, when TP Server is used, two ESP Workload Manager master subsystems can co-exist and share job-tracking information.

Use the TPAPPL initialization parameter to initiate a connection with the partner ESP Workload Manager. If that ESP Workload Manager cannot receive VTAM data at initialization time, the job-tracking information is placed in the TP checkpoint data set until a connection is established.

Note: In a multi-access spool node, only the ESP Workload Manager master subsystem transmits the tracking data. Proxies pass tracking data only to the local master.

Transaction handler

One or more transaction handlers are registered with each TP Server. These programs each receive a particular transaction. The handler in TP Server is called TP_SERVICES_CONTROL. It starts automatically upon TP Server initialization, and receives status and control information, such as the shutdown of a partner.

Application

An application, such as ESP Workload Manager or ESP Infoserv, owns a particular TP Server.

An application has a one-to-one correspondence with a VTAM APPL LU name, as specified by the LOCAPPL initialization parameter.

A transaction (for example, ESP Workload Manager tracking data) is the individual service request TP Server processes. Any unit of work passed to TP Server for transmission contains a two-part destination address:

- Application name (for example, ESP_TORONTO_MASTER)
- Transaction name (for example, INTER_SYSTEM_JOB_TRACKING).

Data sets

TP Server checkpoint

TP Server requires a checkpoint data set. This checkpoint data set is a VSAM LINEAR data set that queues transactions until they are successfully passed to the target transaction handler or a partner server. The checkpoint data set ensures transmission integrity. The data set name is specified on the TPCKPT initialization parameter.

During normal operation between connected and operational TP Servers, the checkpoint contains only some control information. However, if a partner application is not yet active or the transmission fails, the transactions will be queued until a valid connection can be established. These transactions are referred to as held and can be manipulated using the TPAPPL command.

You can use SSCPSAME(CYBESS19) for creating the checkpoint data set.

TPPARM data set

Create your TPPARM member in the ESPPARM data set. You can use SSCPSAME(CYBESS06) as a sample TPPARM.

Configuring ESP Workload Manager

You must include the two following initialization parameters in the ESPPARM data set:

Initialization Parameter	Description
ISCOPT	Specifies the communication method (LU 6.2) used to transmit internodal job-tracking information
XMITMDL	Specifies what information to send (based on tracking models) and to which node. The node refers to the application name of the target TP Server: ESP_SYSTEM_*x*.

Configuring TP Server

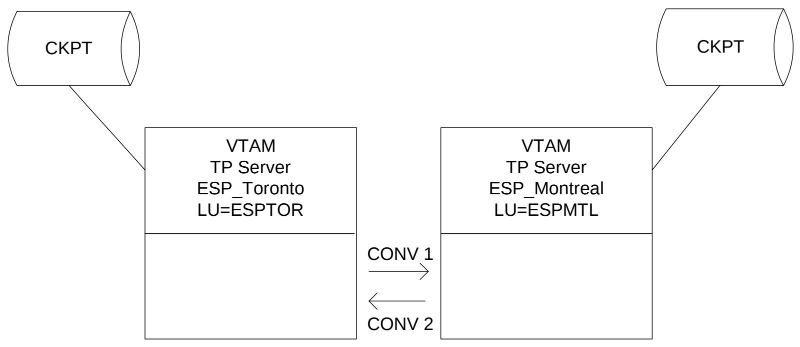
You must include the following initialization parameters in the TPPARM data set:

Initialization Parameter	Description
LOCAPPL	The name of the application using this TP Server and the basic VTAM information needed to initiate and receive session requests
TPAPPL	Manages interactions with remote partner applications
TPCKPT	The TP Server checkpoint file

Basic sample configuration

Configuration Overview

The following diagram illustrates conversations between two ESP Workload Manager subsystems:



This diagram illustrates two Applications: ESP_Toronto and ESP_Montreal.

Each is the ESP Workload Manager master subsystem at its respective complex. The VTAM LU names associated with these two applications are ESPTOR and ESPMTL.

Two conversations are allocated between the two applications and each owns a TP checkpoint data set.

Initialization parameters for ESP_Toronto

The following is an example of the TPPARM for the application ESP_Toronto:

```
LOCAPPL ESP_TORONTO LUNAME(ESPTOR) LOGMODE(CYBTPLOG)
TPCKPT DSNAME(ESP.TPTOR.CKPT)
TPAPPL ESP_MONTREAL START LUNAME(ESPMTL)
```

The following is an example of the TP Server-related initialization parameter for the TP application ESP_Toronto:

```
ISCOPT LU62 ZONECONVERT
XMITMDL ADD MODEL(XSYS) NODE(ESP_MONTREAL)
```

Initialization parameters for ESP_Montreal

The following is an example of the TPPARM for the application ESP_Montreal:

```
LOCAPPL ESP_MONTREAL LUNAME(ESPMTL) LOGMODE(CYBTPLOG)
TPCKPT DSNAME(ESP.TPMTL.CKPT)
TPAPPL ESP_TORONTO START LUNAME(ESPTOR)
```

The following is an example of the TP Server-related initialization parameter for the application ESP_Montreal:

```
ISCOPT LU62
XMITMDL ADD MODEL(XSYS) NODE(ESP_TORONTO)
```

Worksheet

The following is an example of a parameter worksheet to help you plan your TP Server configuration.

Site location: ESP_TORONTO

Item	Description	LOCAPPL statement	TPAPPL statement	TPCKPT
VTAM LU name	LUNAME initialization parameter	ESPTOR	ESPMTL	
Logon Mode	LOGMODE initialization parameter	CYBTPLOG	CYBTPLOG	
TP application name	APPLNAME initialization parameter	ESP_TORONTO	ESP_MONTREAL	
TPCKPT data set name	TP checkpoint data set			ESP.TPTOR.CKPT

Site location: ESP_MONTREAL

Item	Description	LOCAPPL statement	TPAPPL statement	TPCKPT
VTAM LU name	LUNAME initialization parameter	ESPMTL	ESPTOR	
Logon Mode	LOGMODE initialization parameter	CYBTPLOG	CYBTPLOG	
TP application name	APPLNAME initialization parameter	ESP_MONTREAL	ESP_TORONTO	
TPCKPT data set name	TP checkpoint data set			ESP.TPMTL.CKPT

Process

When an ESP Workload Manager job-tracking record initiated in ESP_Toronto is placed onto the checkpoint data set and communication is established, VTAM sends the data to the ESPMTL LU. The TP Server that ESP_Montreal owns receives the transaction into its checkpoint data set and notifies ESP_Toronto that the transmission was successful.

At this point, the data packet that ESP_Toronto owned becomes the responsibility of ESP_Montreal.

If a communication failure occurs, the transaction is retained at ESP_Toronto until it is successfully transmitted to ESP_Montreal. ESP_Montreal does not process the transaction further until the ownership transfers from ESP_Toronto.

ESP Workload Manager job-tracking records are transmitted continuously during job execution via SMF. For example, records are sent at job start, step_end, device allocations, and job_end.

Network Job Entry (NJE)

NJE transfers ESP Workload Manager job-tracking information between ESP Workload Manager subsystems, which exist on separate JES nodes, using the JES communication protocol.

Configuring ESP Workload Manager

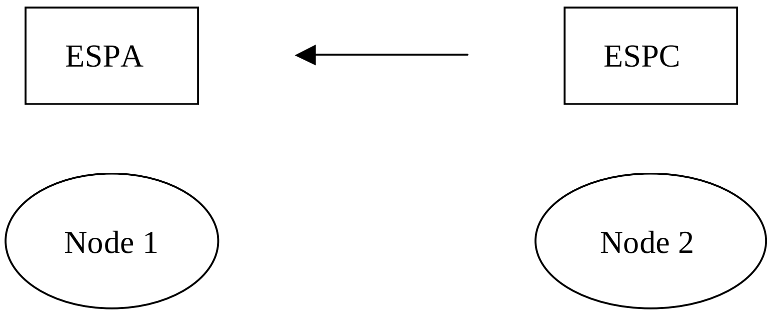
You must include the following initialization parameters in the ESPPARM data set with the exception of XMITMDL, which is required only on the sending side, and NJETOL, which is optional:

Initialization Parameter	Description
ISCOPT	Specifies the communication method (NJE) used to transmit internodal job-tracking information
ISCDEF	Specifies the current node name for internodal tracking when NJE is used as the communications method
ISCXMTR	Specifies the node name, sysout class and external writer name for each node that is to receive tracking data from the current system when NJE is used as the communications method for internodal tracking
XMITMDL	Specifies what information to send (based on tracking models) and to which node. The node refers to the JES node name specified in ISCXMTR.
NJETOL	Specifies tolerance of time differences across an NJE connection with a remote JES3 system

Examples

Example 1 — One-way transmission

This example shows two nodes, two ESP Workload Manager master subsystems and a one-way transmission of job-tracking information. The job tracking information is going to ESPA because ESPA is the system doing the scheduling.



Two nodes are directly associated with the ESP Workload Manager subsystems participating in the communication.

The ESPA subsystem does not send job-tracking data and, therefore, does not need an XMITMDL initialization parameter.

The ESPC subsystem sends all job-tracking data for tracking models whose names begin with the characters PR to the ESPA subsystem, as specified by the XMITMDL initialization parameter.

NDS

ESPA

In ESPPARM

```
ISCOPT NETWORK
```

In LOADNET

```
NETNODE ESP_SYSTEM_A
NETDEST ADD ESP_SYSTEM_C HOST(ESPC)
NETWORK START LISTENER PORT(1234)
NETWORK START CONNECTION HOST(ESPC) PORT(2222)
NETWORK START SERVICE TRACKING
```

ESPC

In ESPPARM

```
ISCOPT NETWORK
XMITMDL ADD MODEL(PR-) NETNODE(ESP_SYSTEM_A)
```

In LOADNET

```
NETNODE ESP_SYSTEM_C
NETDEST ADD ESP_SYSTEM_A HOST(ESPA)
NETWORK START LISTENER PORT (2222)
NETWORK START CONNECTION HOST(ESPA) PORT(1234)
NETWORK START SERVICE TRACKING
```

TP Server

ESPA

In ESPPARM

```
ISCOPT LU62
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_A LUNAME(ESPA) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_C START LUNAME(ESPC)
```

ESPC

In ESPPARM

```
ISCOPT LU62
XMITMDL ADD MODEL(PR-) NODE(ESP_SYSTEM_A)
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_C LUNAME(ESPC) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_A START LUNAME(ESPA)
```

NJE

ESPA

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_A) CLASS(N) RECEIVER(ESPA)
```

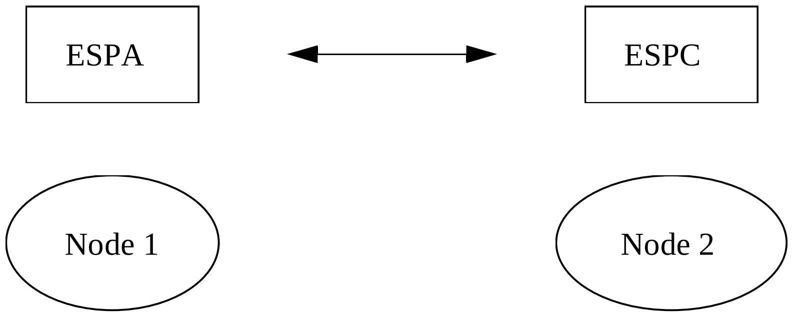
ESPC

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_C)
ISCXMTR NB NODE(ESP_SYSTEM_A) CLASS(N) RECEIVER(ESPA)
XMITMDL ADD MODEL(PR-) NODE(NB)
```

Example 2 — Two-way transmission

This example uses the same two nodes and ESP Workload Manager subsystems as the last example to illustrate how to set up a two-way transmission of job-tracking information.



The ESPA subsystem sends all job-tracking data for tracking models whose names begin with the characters PR and SYS to the ESPC subsystem, as specified by the XMITMDL initialization parameter.

The ESPC subsystem sends all job-tracking data for tracking models whose names begin with the characters PR to the ESPA subsystem, as specified by the XMITMDL initialization parameter.

NDS

ESPA

In ESPPARM

```
ISCOPT NETWORK
XMITMDL ADD MODEL(PR-,SYS-) NETNODE(ESP_SYSTEM_C)
```

In LOADNET

```
NETNODE ESP_SYSTEM_A
NETDEST ADD ESP_SYSTEM_C HOST(ESPC)
NETWORK START LISTENER PORT(1234)
NETWORK START CONNECTION HOST(ESPC) PORT(2222)
NETWORK START SERVICE TRACKING
```

ESPC

In ESPPARM

```
ISCOPT NETWORK
XMITMDL ADD MODEL(PR-) NETNODE(ESP_SYSTEM_A)
```

In LOADNET

```
NETNODE ESP_SYSTEM_C
NETDEST ADD ESP_SYSTEM_A HOST(ESPA)
NETWORK START LISTENER PORT(2222)
NETWORK START CONNECTION HOST(ESPA) PORT(1234)
NETWORK START SERVICE TRACKING
```

TP Server

ESPA

In ESPPARM

```
ISCOPT LU62
XMITMDL ADD MODEL(PR-,SYS-) NODE(ESP_SYSTEM_C)
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_A LUNAME(ESPA) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_C START LUNAME(ESPC)
```

ESPC

In ESPPARM

```
ISCOPT LU62
XMITMDL ADD MODEL(PR-) NODE(ESP_SYSTEM_A)
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_C LUNAME(ESPC) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_A START LUNAME(ESPA)
```

NJE

ESPA

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_A) CLASS(N) RECEIVER(ESPA)
ISCXMTR NB NODE(ESP_SYSTEM_C) CLASS(N) RECEIVER(ESPC)
XMITMDL ADD MODEL(PR-,SYS-) NODE(NB)
```

ESPC

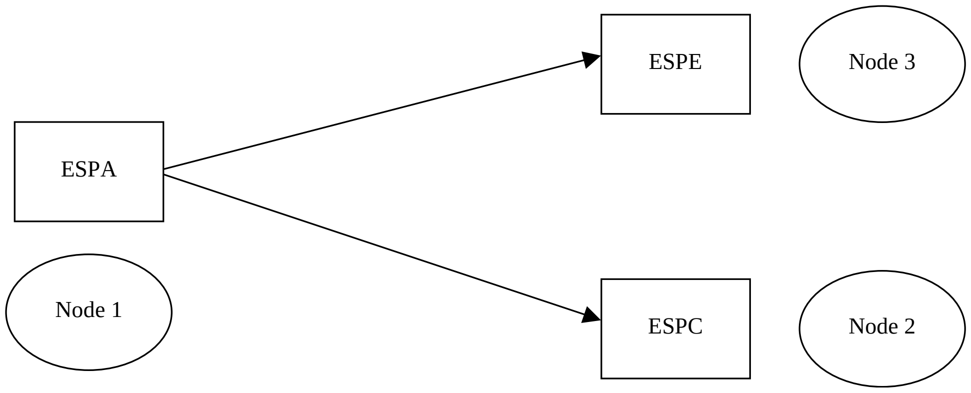
In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_C) CLASS(N) RECEIVER(ESPC)
ISCXMTR NB NODE(ESP_SYSTEM_A) CLASS(N) RECEIVER(ESPA)
XMITMDL ADD MODEL(PR-) NODE(NB)
```

Example 3 — Multi-node configuration

This example uses three nodes, three ESP Workload Manager subsystems, and a one-way transmission of job-tracking information from one of the nodes to the other two.

ESPA passes tracking data to ESPC and ESPE. ESPC requires all tracking data from ESPA and ESPE requires the tracking data associated with the model PRODMDL.



The ESPA subsystem sends all job-tracking data to the ESPC and ESPE subsystems as specified by the two XMITMDL initialization parameters.

The ESPC and ESPE subsystems do not send job-tracking data and, therefore, do not need an XMITMDL initialization parameter.

NDS

ESPA

In ESPPARM

```
ISCOPT NETWORK
XMITMDL ADD MODEL(-) NETNODE(ESP_SYSTEM_C)
XMITMDL ADD MODEL(PRODMDL) NETNODE(ESP_SYSTEM_E)
```

In LOADNET

```
NETNODE ESP_SYSTEM_A
NETDEST ADD ESP_SYSTEM_C HOST(ESPC)
NETDEST ADD ESP_SYSTEM_E HOST(ESPE)
NETWORK START LISTENER PORT(1234)
NETWORK START CONNECTION HOST(ESPC) PORT(2222)
NETWORK START CONNECTION HOST(ESPE) PORT(3333)
NETWORK START SERVICE TRACKING
```

ESPC

In ESPPARM

```
ISCOPT NETWORK
```

In LOADNET

```
NETNODE ESP_SYSTEM_C
NETDEST ADD ESP_SYSTEM_A HOST(ESPA)
NETWORK START LISTENER PORT(2222)
NETWORK START CONNECTION HOST(ESPA) PORT(1234)
NETWORK START SERVICE TRACKING
```

ESPE

In ESPPARM

```
ISCOPT NETWORK
```

In LOADNET

```
NETNODE ESP_SYSTEM_E
NETDEST ADD ESP_SYSTEM_A HOST(ESPA)
NETWORK START LISTENER PORT(3333)
NETWORK START CONNECTION HOST(ESPA) PORT(1234)
NETWORK START SERVICE TRACKING
```

TP Server

ESPA

In ESPPARM

```
ISCOPT LU62
XMITMDL ADD MODEL(-) NODE(ESP_SYSTEM_C)
XMITMDL ADD MODEL(PRODMDL) NODE(ESP_SYSTEM_E)
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_A LUNAME(ESPA) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_C START LUNAME(ESPC)
TPAPPL ESP_SYSTEM_E START LUNAME(ESPE)
```

ESPC

In ESPPARM

```
ISCOPT LU62
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_C LUNAME(ESPC) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_A START LUNAME(ESPA)
```

ESPE

In ESPPARM

```
ISCOPT LU62
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_E LUNAME(ESPE) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_A START LUNAME(ESPA)
```

NJE

ESPA

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_A)
ISCXMTR NB NODE(ESP_SYSTEM_C) CLASS(N) RECEIVER(ESPC)
ISCXMTR NB NODE(ESP_SYSTEM_E) CLASS(N) RECEIVER(ESPE)
XMITMDL ADD MODEL(-) NODE(NB)
XMITMDL ADD MODEL(PRODMDL) NODE(NB)
```

ESPC

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_C) CLASS(N) RECEIVER(ESPC)
```

ESPE

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_E) CLASS(N) RECEIVER(ESPE)
```

Example 4 — Multi-node configuration

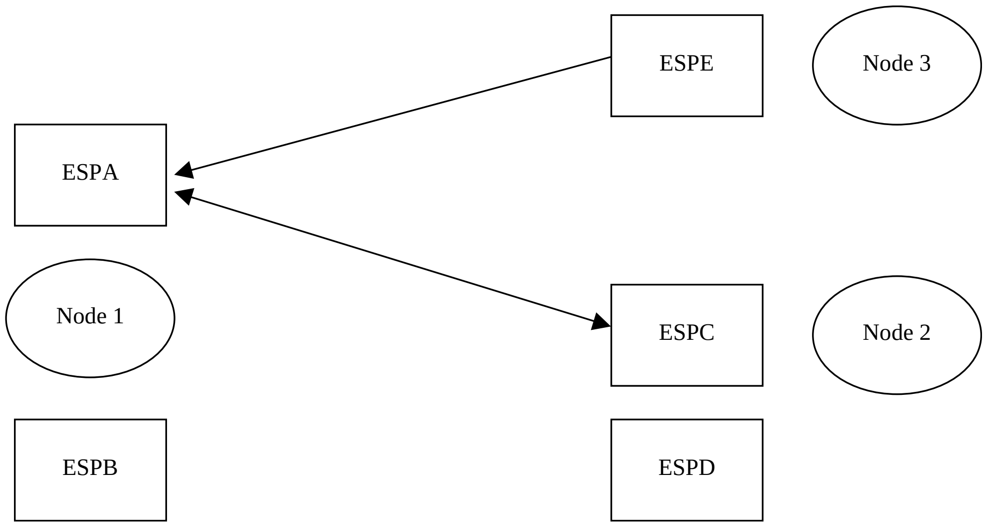
This example uses three nodes, five ESP Workload Manager subsystems, and both one-way and two-way transmissions of job-tracking information.

ESPA	• Passes tracking data to ESPC
	• Requires all tracking data from ESPC
	• Requires all tracking data from ESPE
ESPC	• Passes tracking data to ESPA
	• Requires tracking data associated with the characters PR and SYS from ESPA
ESPE	• Passes tracking data to ESPA

ESPA, ESPC, and ESPE are master subsystems. ESPB and ESPD are proxy subsystems.

The ESPA subsystem sends all job-tracking data for tracking models whose names begin with the characters PR and SYS to the ESPC subsystem, as specified by the XMITMDL initialization parameter. The ESPA subsystem does not send job-tracking data to the ESPE subsystem and, therefore, does not need a second XMITMDL initialization parameter.

The ESPC and ESPE subsystems send all job-tracking data to the ESPA subsystem, as specified by the XMITMDL initialization parameters.



NDS

ESPA

In ESPPARM

```
ISCOPT NETWORK
XMITMDL ADD MODEL(PR-,SYS-) NETNODE(ESP_SYSTEM_C)
```

In LOADNET

```
NETNODE ESP_SYSTEM_A
NETDEST ADD ESP_SYSTEM_C HOST(ESPC)
NETDEST ADD ESP_SYSTEM_E HOST(ESPE)
NETWORK START LISTENER PORT(1234)
NETWORK START CONNECTION HOST(ESPC) PORT(2222)
NETWORK START CONNECTION HOST(ESPE) PORT(3333)
NETWORK START SERVICE TRACKING
```

ESPC

In ESPPARM

```
ISCOPT NETWORK
XMITMDL ADD MODEL(-) NETNODE(ESP_SYSTEM_A)
```

In LOADNET

```
NETNODE ESP_SYSTEM_C
NETDEST ADD ESP_SYSTEM_A HOST(ESPA)
NETWORK START LISTENER PORT(2222)
NETWORK START CONNECTION HOST(ESPA) PORT(1234)
NETWORK START SERVICE TRACKING
```

ESPE

In ESPPARM

```
ISCOPT NETWORK
XMITMDL ADD MODEL(-) NETNODE(ESP_SYSTEM_A)
```

In LOADNET

```
NETNODE ESP_SYSTEM_E
NETDEST ADD ESP_SYSTEM_A HOST(ESPA)
NETWORK START LISTENER PORT(3333)
NETWORK START CONNECTION HOST(ESPA) PORT(1234)
NETWORK START SERVICE TRACKING
```

TP Server

ESPA

In ESPPARM

```
ISCOPT LU62
XMITMDL ADD MODEL(PR-,SYS-) NODE(ESP_SYSTEM_C)
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_A LUNAME(ESPA) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_C START LUNAME(ESPC)
TPAPPL ESP_SYSTEM_E START LUNAME(ESPE)
```

ESPC

In ESPPARM

```
ISCOPT LU62
XMITMDL ADD MODEL(-) NODE(ESP_SYSTEM_A)
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_C LUNAME(ESPC) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_A START LUNAME(ESPA)
```

ESPE

In ESPPARM

```
ISCOPT LU62
XMITMDL ADD MODEL(-) NODE(ESP_SYSTEM_A)
```

In TPPARM

```
LOCAPPL ESP_SYSTEM_E LUNAME(ESPE) LOGMODE(CYBLOGMD)
TPAPPL ESP_SYSTEM_A START LUNAME(ESPA)
```

NJE

ESPA

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_A) CLASS(N) RECEIVER(ESPA)
ISCXMTR NB NODE(ESP_SYSTEM_C) CLASS(N) RECEIVER(ESPC)
XMITMDL ADD MODEL(PR-,SYS-) NODE(NB)
```

ESPC

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_C) CLASS(N) RECEIVER(ESPC)
ISCXMTR NB NODE(ESP_SYSTEM_A) CLASS(N) RECEIVER(ESPA)
XMITMDL ADD MODEL(-) NODE(NB)
```

ESPE

In ESPPARM

```
ISCOPT NJE
ISCDEF NODE(ESP_SYSTEM_E)
ISCXMTR NB NODE(ESP_SYSTEM_A) CLASS(N) RECEIVER(ESPA)
XMITMDL ADD MODEL(-) NODE(NB)
```

Part 4

Customizing ESP Workload Manager

This part discusses the various customization options available to you when configuring ESP Workload Manager.

This part contains the following chapters:

- Implementing CSF Extensions
- Examples of CSF Extensions
- Setting Up User Modifications
- Specifying ESP Workload Manager User Exits

16

Implementing CSF Extensions

This chapter describes how you can extend the Consolidated Status Facility (CSF) functionality using CSF extensions. After learning how to implement the sample extensions CA provides, you can create your own CSF extensions. To see how other people have used extensions in their environments, see “Examples of CSF Extensions” on page 193.

This chapter contains the following topics:

Introduction to CSF Extensions

CSF enables you to view and control active workload across the enterprise in ESP Workload Manager. CSF extensions are a programming interface to CSF. Using CSF extensions, you can customize CSF by writing your own commands or by replacing existing commands.

For more details on CSF, see the *ESP Workload Manager Operator's Guide*.

Uses for CSF extensions

You can use CSF extensions to

- Reduce frequently performed functions to a single command.
- Access other ISPF applications from the CSF panel.

  For example, you can set up access to output distribution products and spool viewers.
- Control access to current CSF functions.

  For example, if you want to prevent users from using a command or to provide a confirmation panel, you can write a REXX exec or program to provide a front-end for an existing, internal CSF command. The exec can then decide whether to allow the internal command to proceed, reject the command or provide an alternative function.

Addition of commands

To add or replace a command, you use an ISPF table called CYBESCSU. Each CSF command invokes a REXX exec or program. These might access REXX functions, ESP Workload Manager commands, and ISPF services, and then return to CSF with a return code.

Distribution package

As part of the ESP Workload Manager distribution package, CA provides

- Sample CSF extensions.
- A general purpose REXX exec, CYBESTBG, to generate the ISPF table CYBESCSU.
- A sample CYBESCSU input table.

References to libraries

Throughout this chapter, there are references to libraries you created as part of the ESP Workload Manager installation. Whenever there is a reference to one of these libraries, such as *prefix*.SSCPSAME, you need to substitute the prefix you chose during the installation.

Installing and Testing the Sample Extensions

What you need

- Sample libraries *prefix*.SSCPSAME and *prefix*.SSCPCLST
- A REXX exec library to store the samples
- An ISPF table library, such as the ESP Workload Manager table library (*prefix*.SSCPTENU)

You must allocate the REXX exec library and the ISPF table library to a user's session to allow them to use the sample extensions. You can allocate these libraries through an exec, a CLIST or as part of a logon proc.

Installing the extensions

1. Copy the sample execs listed in the following table from library *prefix*.SSCPSAME to a REXX exec library that is allocated to a SYSEXEC DD.

REXX Exec	Command	Description
CYBESS60	CAA	Completes all generations of a user-specified Application older than a user-specified time
CYBESS61	EDJ	Edits JCL for a job (before or after submission)
CYBESS63	SD	Invokes SDSF to display a user-specified job
CYBESS64	SIM	Simulates the Event associated with a job
CYBESS65	SUB	Submits the JCL associated with a job outside the Application
CYBESS66	TSC, TST, U	Displays information retrieved by the RETSCBD() function
CYBESS67	TR	Triggers the Event associated with a job
CYBESS69	NX	Displays the next 10 execution times for an Event associated with a job

2. If you want, copy member CYBESCSU from library *prefix*.SSCPCLST to another library.

   CYBESCSU contains information related to your new commands. CYBESCSU is read by the CYBESTBG exec. You update CYBESCSU whenever you add new CSF extensions.

3. Copy the CYBESTBG REXX exec from the *prefix*.SSCPCLST library to a REXX exec library that is allocated to SYSEXEC.

   CYBESTBG reads member CYBESCSU and generates an ISPF table library member, also called CYBESCSU, that stores your commands. You execute CYBESTBG whenever you add new CSF extensions.

4. In member CYBESCSU, change the LIBRARY statement to point to a table library that is allocated to your ISPTLIB DD.

   You can specify your ESP Workload Manager table library (*prefix*.SSCPTENU) or a user table library, for example:

   ```
   LIBRARY CYB3.ESP.SSCPTENU
   ```

5. Run the CYBESTBG exec, specifying the location of input member CYBESCSU, for example:

   ```
    TSO CYBESTBG 'CYB3.ESP.SSCPCLST(CYBESCSU)'
   ```

   A message appears stating the number of rows added to ISPF table CYBESCSU.

6. Exit to the READY prompt.

   Your ISPF session is refreshed, and you can now use the new commands.

Testing the extensions

You can test the new extensions by issuing the associated commands against any test Applications listed in the CSF panel.

A good command to test is the TST command, which illustrates some of the information you can have returned as part of a CSF extension.

To test the TST command

1. Sign onto ESP Workload Manager.
2. Choose option `C` for CSF.
3. Type `TST` beside any job name and observe the results.

Implementing Your Own CSF Extensions

This section describes how to create and implement your own CSF extensions. It assumes you are familiar with some of the aspects of installing CSF extensions by installing the samples supplied.

Preparing to implement your own extensions

In addition to a REXX exec library and a table library to store your commands, you might choose to add a panel library for customized panels and a message library for customized messages.

The following table lists the libraries you can use and the corresponding DD statement:

Type of Library	DD Statement
REXX exec library	SYSEXEC
Table library	ISPTLIB
Panel library (optional)	ISPPLIB
Message library (optional)	ISPMLIB

You can allocate these libraries to a user’s session using a CLIST exec or include them in a logon proc.

To add an extension

1. Design and write the REXX exec (or program) and store it in a library allocated to the SYSEXEC DD.
2. Update your list of commands by editing the CYBESCSU member. For details, see “Adding and Customizing Commands” on page 174.
3. Run the CYBESTBG exec to update the table of commands (allocated to ISPTLIB DD). Specify the name of the library and member where you stored your list of commands, for example:

   ```
   TSO CYBESTBG 'CYB3.ESP.SSCPCLST(CYBESCSU)'
   ```

4. Re-enter ISPF to test.

Example: Adding an extension

The following is a simple, step-by-step example of adding a new CSF extension. This example can help you get started with implementing your own extensions. After working through this example, you can learn about the other features you can use.

In this example, you add a command called TR that executes a REXX exec called CSFTRIG. This exec allows a user to trigger an Event from the CSF. By entering command TR next to a job in CSF, the user triggers the Event that generated the Application the job belongs to.

To create the CSF extension

1. Write the exec and store it in a member called CSFTRIG in a REXX exec library:

```
/* REXX */
/* Command Name: TR */
/* CSF extension to trigger an Event associated with a job. */
/* Blank out Variables used in panel CYBESTRI */

KTRIG = ""
Raopt = ""
Aa2   = ""
A     = ""
Aa    = ""
B     = ""
Ff1   = ""
Ff2   = ""
Ff3   = ""
Ff4   = ""
Ff5   = ""
Ff6   = ""
Q     = "N"    /* Case sensitivity flag */

/* Display panel CYBESTRI to obtain options. */

Address ISPEXEC "DISPLAY PANEL(CYBESTRI)"
If Rc <> 0 Then
   Return

/* Construct a TRIGGER command, using user-specified options.*/

Cmd = "TRIGGER" Event() Raopt

If Aa2 <> "" Then
   Cmd = Cmd "AT('"Aa2"')"
If A = "Y" Then
   Cmd = Cmd "NOXEQ"
If B = "Y" Then
   Cmd = Cmd "ONHOLD"
If Ff1 <> "" Then
   Cmd = Cmd "USER1('"Ff1"')"
If Ff2 <> "" Then
   Cmd = Cmd "USER2('"Ff2"')"
If Ff3 <> "" Then
   Cmd = Cmd "USER3('"Ff3"')"
If Ff4 <> "" Then
   Cmd = Cmd "USER4('"Ff4"')"
If Ff5 <> "" | Ff6 <> "" Then
   Cmd = Cmd "ROOT('"Ff5 Ff6"')"
If Q   = "Y" | Q = "y" Then
   Cmd = Cmd " CaseSensitive"

/* Execute the TRIGGER command. */
/* Say Cmd */

Address ESP Cmd

Return
```

2. Update your list of commands in the sample input member CYBESCSU. For this example, add the line shown below at the end of the member.

   ```
   LC TR  EXEC(CSFTRIG) ISPF
   ```

3. To update your ISPF table library, run the CYBESTBG exec, specifying the name of the input data set you just updated. The CYBESTBG exec creates the appropriate table library member.

   ```
   TSO CYBESTBG 'CYB3.ESP.SSCPCLST(CYBESCSU)'
   ```

4. Re-enter ISPF and test your command by entering command TR beside any job listed in CSF.

Updating the exec

After you have created your exec, you can update it at any time, and the changes take effect immediately. There is no need to update any tables or refresh ISPF.

Adding and Customizing Commands

You can add or customize commands by editing the CYBESCSU member. The CYBESCSU member stores your list of commands and is issued as input to the CYBESTBG exec to create the CYBESCSU table library.

Listing of sample member CYBESCSU

```
* REF: ASEPCLST(CYBESCSU)
*
* Table of user defined commands for ESP CSF
*
* Note: You must modify this input and run the
* REXX Exec CYBESTBG to generate the ISPF table
* library member.

LIBRARY CYB3.ESP.SSCPTENU
MEMBER CYBESCSU
VARS CMDTABNM CMDTNAME CMDTPARM
KEYS CMDTABNM CMDTNAME
OPTIONS REPLACE
DATA' '
LC CAA EXEC(CYBESS60)
LC EDJ EXEC(CYBESS61)
LC NX  EXEC(CYBESS69)
LC SD  EXEC(CYBESS63)
LC SIM EXEC(CYBESS64)
LC SUB EXEC(CYBESS65)
LC TR  EXEC(CYBESS67) ISPF
LC TSC EXEC(CYBESS66) RETATR RETJTR ISPF
LC TST EXEC(CYBESS66) RETATR RETJTR
LR U   EXEC(CYBESS66) RETATR RETJTR RETJSR
```

CYBESCSU line command entries

By coding line command entries in CYBESCSU, you can

- Add new line commands.
- Replace the functionality of existing commands.
- Disable existing commands.
- Add a front-end to existing commands

When a user issues a command in CSF, ESP Workload Manager first looks for an entry in CYBESCSU to decide how to process the command.

For each new or existing command you want to process, add one line command entry after the `DATA' '` line in CYBESCSU.

Syntax

```
command-type command-name process processing-options
```

Operand	Description
command-type	The panel the command is issued from. For details, see "Command type" on page 175.
command-name	A new line command or an existing line command. Line commands are one to three characters long.
process	The process to execute for the line command specified. For details, see "Processes" on page 175.
processing-options	One or more options for the specified process. For details, see "Process-options" on page 175.

Command type

Type	Description
LC	CSF main panel line command. LC is the most common type.
LR	LR panel line command when listing resources
LX	L panel line command when listing job dependencies

Processes

Process	Description
EXEC(*rexx-exec*)	Invokes a REXX exec. The REXX exec should be in a library allocated to the SYSEXEC DD. You can pass parameters to the REXX exec using the PARMS parameter, as described in Process-options.
PGM(*program*)	Calls a program. The program should be a load module accessible to the ISPF session and should be linked with AMODE=31. You can pass parameters to the program using the PARMS parameter, as described in Process-options.
INVALID	Informs the user that the command is invalid
NOOP	Ignores the command

Process-options

Option	Parameter
PARMS(*parms*)	A parameter string that is passed to the REXX exec or program specified in Processes
ISPF	Use ISPF variable services and call the REXX exec with an ISPF SELECT function call. If you don't specify the ISPF option, CSF invokes the REXX exec directly. Use this option if you want to manipulate or examine ISPF variables from REXX.

Option	Parameter
RETJTR	Retrieve the JTR (Job Tracking Record) prior to calling the program or REXX exec. The address of the JTR is provided.
RETATR	Retrieve the ATR (Application Tracking Record) prior to calling the program or REXX exec. The address of the ATR is provided.
RETJSR	Retrieve the JSR (Job Schedule Record) prior to calling the program or REXX exec. The address of the JSR is provided.

Examples

In the following example, the line command SIM invokes a REXX exec called CSFSIM:

```
LC SIM EXEC(CSFSIM)
```

CSFSIM now overrides the standard module for SIM, CYBESS64.

In the following example, the line command TR invokes a REXX exec called CYBESS67. ISPF variable services will be available.

```
LC TR EXEC(CYBESS67) ISPF
```

In the following example, the line command LST invokes a program called LISTIT:

```
LC LST PGM(LISTIT)
```

In the following example, the line command CA, which is an existing command, is invalid:

```
LC CA INVALID
```

In the following example, the command EXT invokes a REXX exec called EXT. The EXT command is available only from the Extended Job List panel, which appears after the user enters `L` beside a job to list job dependencies.

```
LX EXT EXEC(EXT)
```

Using the Interface to the REXX Execs

If you are using a REXX exec to process a command, the exec is called directly using the REXX interface module or through an ISPF SELECT request if the ISPF option is specified. You can pass parameters to the exec using the PARMS keyword and these are available with the ARG or PARSE ARG instructions.

Using different environments

The default host environment is ESP Workload Manager, so you can invoke any ESP Workload Manager commands directly or by using ADDRESS ESP, for example:

```
ADDRESS ESP "OPER PURGSCHF NOW"
```

To access any function usually available with REXX in a TSO environment

Use ADDRESS TSO, for example:

```
ADDRESS TSO "TIME"
```

ADDRESS ISPEXEC provides all ISPF functions (such as browse or edit), for example:

```
ADDRESS ISPEXEC "DISPLAY PANEL(CSFRDC)"
```

If access to ISPF variables is required from the REXX exec, you must specify the ISPF option in the command table entry for the EXEC, for example:

```
LC ABC EXEC(MYEXEC) ISPF
```

Returning values

When the REXX exec ends, CSF examines the return value to determine how to proceed. You can specify the return value using the CSFRET() function or the REXX RETURN statement, for example:

```
X=CSFRET(value)
```

or

```
RETURN value
```

The RETURN statement is available only if the exec was invoked directly by ESP Workload Manager, and not through the ISPF SELECT function. If you use the ISPF option for the exec, you must pass back the return value through the CSFRET function.

The different methods for returning a value are illustrated below:

Using ISPF Services	Using a Direct Invocation
`If RC \=0 then X=CSFRET(REJECTM)`	`If RC \=0 then RETURN REJECTM`
`else X=CSFRET(CONTINUE)`	`else RETURN CONTINUE`

The possible return values are listed below:

Value	Action
(null) or 0	Same as NOOP, described below. This is the default.
CONTINUE	If an internal command exists with the same name, the internal command is executed. Otherwise, the same action is taken as for NOOP. This value is useful when you want to execute custom code prior to executing an existing CSF command.

Value	Action
NOOP	The command is assumed to have executed successfully. The command entry field is blanked out and processing continues with the next line command, if any.
REJECT	The command is assumed to have failed. The line command is left on the line where the error occurred, and the processing of line commands is interrupted. No message is issued because it is assumed that the command processing routine has issued an ISPF SETMSG.
REJECTM	This is the same as REJECT; however, a message is issued indicating that the command has been rejected.

Using Functions and Variables

REXX built-in functions

The CSF extensions interface provides the following REXX functions.

Function	Description	Sample Output
ADDRATR	Hexadecimal address of the ATR (Application Tracking Record) if RETATR was specified on the command description. Otherwise, ADDRATR has a value of 0.	035000AB
ADDRJSR	Hexadecimal address of the JSR (Job Schedule Record) if RETJSR was specified on the command description. Otherwise, ADDRJSR has a value of 0.	03625CF0
ADDRJTR	Hexadecimal address of the JTR (Job Tracking Record) if RETJTR was specified on the command description. Otherwise, ADDRJTR has a value of 0.	0324DC08
APPL	The Application's fully qualified name, which includes the Application name and generation number	BILLING.234
APPLGEN	The Application generation number the job belongs to	234
APPLNAME	The name of the Application the job belongs to	BILLING
CSFJOB	Information about any job that is currently available on CSF. See also function JOBONCSF. JOBONCSF and CSFJOB both accept a full name, but JOBONCSF works differently than CSFJOB depending on the length of the full name and if you code a period in the full name.	See "Using the CSFJOB Function" on page 184.
CMDT	The type of command as specified in the CYBESCSU member. CMDT has a value of LC, LR or LX.	LC
COMMAND	The command name	EDJ

Function	Description	Sample Output
CSFRET	Returns the current return value and optionally sets a new return value. The return value is a verb that tells CSF what action to take when the exec terminates.	See "Returning values" on page 177.
DATALINE	The contents of the line the command is entered on. From the main display, the job name is not included. **Note:** The contents depend on the current view the user has chosen.	234 BILLING WAITING, HC=1
EVENT	The fully qualified Event name that triggered the Application, which includes the Event's prefix and descriptive name.	CYBER. BILLING
HC	The hold count for the job when the last retrieval was made (for example when CSF was refreshed). To get an up-to-date hold count, use the RETSCBD function because the RETSCBD function forces a new access.	2
JOB	The fully qualified job name, which includes the job qualifier if one exists for the job	BILL101 or BILL101.RUN 1
JOBNAME	The job name without a qualifier	BILL101
JOBNO	The JES job number for z/OS jobs only. The number is zero if the job has not been submitted. For task and distributed jobs, ESP Workload Manager internally assigns an unique number.	034567
JOBONCSF	Information about any job that is currently available on CSF. See also function CSFJOB. JOBONCSF and CSFJOB both accept a full name, but JOBONCSF works differently than CSFJOB depending on the length of the full name and if you code a period in the full name.	See "Using the JOBONCSF Function" on page 188.
JOBQUAL	The qualifier for the job, or NULL	RUN1
MSGDEST	The output mode for ESP Workload Manager messages. MSGDEST resolves to either I for ISPF or T for TSO line mode.	I or T
PAGEMODE	Allows any number of commands to be entered and displayed in page mode	See "Using the PAGEMODE Function" on page 183.
RETSCBD	Retrieves information from the CSF scoreboard for the job and passes the data back in REXX variables	See "Using the RETSCBD Function" on page 180.

Using the built-in functions

To use a built-in function

Use the function name followed by parentheses ().

For example, the following statement retrieves the Event name by using EVENT():

```
Say 'The name of the Event is 'EVENT()
```

A built-in function does not exist for each piece of information you might need. For example, there is no built-in function for the Event's scheduled time or a job's status. To retrieve this information, use the RETSCBD function, as described in the next section.

Using the RETSCBD Function

The RETSCBD() function retrieves current data for a job and provides the results in REXX variables. This function is sometimes called the return scoreboard function because it returns all the information about a job that is stored on the CSF scoreboard.

Variables

RETSCBD returns the following variables.

Variable Name	Description	Format	Maximum Length Returned
AGENT	Agent name	string	16
APPL	Application name	string	8
APPLG	Application generation number	number	10
ASID	address space identifier	hexadecimal string	4
AVGRUNTIME	average job execution time in minutes	two-byte binary number	2
BYPASSED	job bypassed	1 for true, 0 for false	1
CLASS	JES job class	string	8
COMPLETE	job completed	1 for true, 0 for false	1
CRITICAL_ PATH	critical path job	1 for true, 0 for false	1
ENDTIME	job end time	hh.mn.ss yyyy/mm/dd	19
EVENT	Event name	string	24

Variable Name	Description	Format	Maximum Length Returned
EXECJNO	JES job number where the job executed	number	6
EXECNODE	JES execution node	string	8
EXTERNAL	external job	1 for true, 0 for false	1
FAILED	job failed	1 for true, 0 for false	1
HC	hold count	number	5
JOBN	job name	string	8
JOBNO	job number (see note below)	number	6
MANUAL	manual job	1 for true, 0 for false	1
MAXRUNTIME	maximum allowable job execution time in minutes	two-byte binary number	2
MINRUNTIME	minimum allowable job execution time in minutes	two-byte binary number	2
ORIGJNO	JES job number where job was submitted	number	6
ORIGNODE	JES submission node	string	8
OVERDUE_END	job overdue to end	1 for true, 0 for false	1
OVERDUE_ START	job overdue to start	1 for true, 0 for false	1
OVERDUE_ SUBMIT	job overdue to submit	1 for true, 0 for false	1
PGN	performance group number	number from 1 to 999 in compatibility mode, 0 on goal mode systems	3
PNODE	processing node	string	8
PRIORITY	JES execution priority	number	2
QUAL	job qualifier	string	8
RDRTIME	time the job was submitted to a JES reader	hh.mn yyyy/mm/dd	19
REQUEST	request job	1 for true, 0 for false	1
REQUESTED	requested job	1 for true, 0 for false	1
SCHED	scheduled time	hh.mn.ss yyyy/mm/dd	19
SMFID	system SMF identifier	string	4
SRVCLASS	service class	string in goal mode, blank on compatibility mode systems	8
STARTTIME	job start time	hh.mn.ss yyyy/mm/dd	19

Variable Name	Description	Format	Maximum Length Returned
STATUS	status of job	string	28
SUBAP	subApplication name	string	8
SYSNAME	sysplex member name	string	8
SYSPLEX	sysplex name	string	8
TAG	tag field	string	16
TYPE	type of job	J for job or distributed object, T for task, L for link	1
USTAT	user status	string	28
WOBTYPE	type of workload object	string	2

Note: The JOBNO value changes when a job is submitted and executed on different JES nodes. During execution, it reflects EXECJNO; otherwise, it reflects ORIGJNO.

Example: Retrieving the job name

The following example uses the RETSCBD function to retrieve information. One of the variables it returns is JOBN, representing the job name.

```
X=RETSCBD()
Say 'Jobname is 'JOBN
```

Example: Specifying a prefix

You can prefix these variables to create customized REXX variables (such as *prefix*JOBN or *prefix*JOBNO). For example, if you specify the string MY on the RETSCBD function, the function returns the variables as listed above with a prefix of MY. The variable for job name is MYJOBN instead of JOBN; the variable for the Application name is MYAPPL instead of APPL, and so on.

```
X=RETSCBD('MY')
Say 'Jobname is 'MYJOBN
Say 'Application name is' MYAPPL
```

Example: Retrieving job information

To retrieve information about a job, you can use a number of built-in functions and the RETSCBD function. You can retrieve some information, such as the Event name, using either method, for example:

```
Say 'Event name is EVENT()'
```

or

```
X=RETSCBD()
Say 'Event name is '  EVENT
```

Using the PAGEMODE Function

Page mode is an ISPF option in ESP Workload Manager that allows users to enter commands and provides scrollable output. The PAGEMODE() function allows you to enter ESP Workload Manager page mode and execute one or more commands. You remain in page mode until you enter the END command or using the END function key.

Note: You cannot use the PAGEMODE() function when the ISPF SELECT function invokes the exec.

Example: Issue a status command

The following example issues a STATUS command to ESP Workload Manager and displays the results in page mode:

```
X=PAGEMODE('oper status')
```

Example: Stacking commands

You can stack commands by separating them with a semi-colon (;), as shown below:

```
X=PAGEMODE('oper status;ldsn')
```

Example: Concatenate commands

You can concatenate commands by building a string of commands, as shown below:

```
CMD = "oper status"
CMD = CMD ";ldsn"
X = PAGEMODE(CMD)
```

Example: Simulate an Event

The following example uses the EVENT() function to retrieve the current Event name and then issues the ESP Workload Manager SIMULATE command to simulate the Event’s next occurrence. Output appears in page mode.

```
X=PAGEMODE('SIMULATE EV('EVENT()')')
```

Example: Display next execution time

The following example uses the EVENT() function to retrieve the current Event name and then issues the ESP Workload Manager NEXT command to display the Event’s next 10 scheduled occurrences. Output appears in page mode.

```
X =  Pagemode("NEXT 10" Event())
```

Using the CSFJOB Function

The CSFJOB function returns information about any job that is available on CSF. CSFJOB returns data in the form of REXX stemmed variables. Unlike the RETSCBD function that returns data for a single job, the CSFJOB function can return data for all jobs that match a job name mask, such as PAY-. This function allows you to

- Retrieve all occurrences of a particular job name.
- Retrieve all jobs matching a job name mask.
- Retrieve all jobs currently available on CSF.

Differences between CSFJOB and JOBONCSF

JOBONCSF and CSFJOB both accept a full name, but JOBONCSF works differently than CSFJOB depending on the length of the full name and if you code a period in the full name. The differences are

- If you specify a one-to-eight character full name, CSFJOB returns jobs matching the full name, but it does not include jobs with qualifiers or jobs with a period followed by other characters. For example,

  ```
  CSFJOB('MYJOB','X')
  ```

  returns MYJOB and

  ```
  JOBONCSF('MYJOB','X')
  ```

  returns MYJOB, MYJOB.QUAL, MYJOB.OTHERCHARACTERS

- If you specify a period in positions 2 through 7 of the full name, JOBONCSF does not work. For example, of the following functions, only the last one works.

  ```
  JOBONCSF('A.-','X')
  JOBONCSF('ABCDEF.-','X')
  JOBONCSF('ABCDEFG.-','X')
  ```

Syntax

n=CSFJOB('*fullname*','*varprefix*')

Operand	Description
n	Number of matching entries returned
fullname	The full name of the job. You can use the asterisk and hyphen wildcard characters (see “Wildcards and masking” on page xiv).
varprefix	A character string used as the prefix to a series of stemmed variables the CSFJOB function returns

If data is returned

If the CSFJOB returns data (the result is larger than 0), it generates REXX stemmed variables in the form *ppppvvvvv.i*, where *pppp* is the stemmed variable prefix you specified when you used the function, *vvvvv* is the variable name, and *i* is the numeric stem. For example, XJOBN.1, XJOBN.2, ... represent the first occurrence of the job, the second occurrence of the job, and so on.

Variables returned

CSFJOB returns variables similar to the RETSCBD variables, as shown below. However, each variable starts with the prefix you specified when you invoked the CSFJOB function.

Variable Name	Description	Format	Maximum Length Returned
AGENT	Agent name	string	16
APPL	Application name	string	8
APPLG	Application generation number	number	10
ASID	address space identifier	hexadecimal string	4
AVGRUNTIME	average job execution time in minutes	two-byte binary number	2
BYPASSED	job bypassed	1 for true, 0 for false	1
CLASS	JES job class	string	8
COMPLETE	job completed	1 for true, 0 for false	1
CONDCODE	completion code	See “CONDCODE format” on page 191.	6
CRITICAL_ PATH	critical path job	1 for true, 0 for false	1
ENDTIME	job end time	hh.mn.ss yyyy/mm/dd	19
EVENT	Event name	string	24
EXECJNO	JES job number where the job executed	number	6
EXECNODE	JES execution node	string	8
EXTERNAL	external job	1 for true, 0 for false	1
FAILED	job failed	1 for true, 0 for false	1
FULLNAME	full name of the job	string	64
HC	hold count	number	5
JOBN	job name	string	8
JOBNO	job number (see note below)	number	6

Variable Name	Description	Format	Maximum Length Returned
MANUAL	manual job	1 for true, 0 for false	1
MAXRUNTIME	maximum allowable job execution time in minutes	two-byte binary number	2
MINRUNTIME	minimum allowable job execution time in minutes	two-byte binary number	2
ORIGJNO	JES job number where the job was submitted	number	6
ORIGNODE	JES submission node	string	8
OVERDUE_ END	job overdue to end	1 for true, 0 for false	1
OVERDUE_ START	job overdue to start	1 for true, 0 for false	1
OVERDUE_ SUBMIT	job overdue to be submitted	1 for true, 0 for false	1
PGN	performance group number	number from 1 to 999 in compatibility mode, 0 on goal mode system	3
PNODE	processing node	string	8
PRIORITY	JES execution priority	number	2
QUAL	job qualifier	string	8
RDRTIME	time the job was submitted to a JES reader	hh.mn yyyy/mm/dd	19
REQUEST	request job	1 for true, 0 for false	1
REQUESTED	requested job	1 for true, 0 for false	1
SCHED	scheduled time	hh.mn.ss yyyy/mm/dd	19
SMFID	system SMF identifier	string	4
SRVCLASS	service class	string in goal mode, blank on compatibility mode systems	8
STARTTIME	job start time	hh.mn.ss yyyy/mm/dd	19
STATUS	status of job	string	28
SUBAP	subApplication name	string	8
SYSNAME	sysplex member name	string	8
SYSPLEX	sysplex name	string	8
TAG	tag field	string	16

Variable Name	Description	Format	Maximum Length Returned
TYPE	type of job	J for job or distributed object, T for task, L for link	1
USTAT	user status	string	28
WOBTYPE	type of workload object for distributed work	string	2

Note: The JOBNO value changes when a job is submitted and executed on different JES nodes. During execution, JOBNO reflects EXECJNO; otherwise, it reflects ORIGJNO.

CONDCODE format

Condition	Format	Legend	
System abend	`ƀƀSxxx`	ƀ	Blank
User abend	`ƀUnnnn`	x	Hexadecimal digit
Overflow	`nnnnnn`	n	Decimal digit
Overflow with value of more than 6 digits	`ƀƀ****`	z	Zero-suppressed decimal digit (leading zeroes not shown)
Null	`ƀƀƀƀƀ-`		All other characters represent themselves
Other (numeric value up to six digits. Values larger than six digits use the overflow format.)	`zzzzzn`		

Example: Find multiple occurrences of job

A job might exist in different Applications, in different generations of the same Application or multiple times within the same generation of an Application.

This example uses the CSFJOB function to find all occurrences of job PAY01. For each occurrence of the job, the function generates variables with the prefix X. The CSFJOB function sends information back to your terminal indicating which Applications (applname.generation) the job belongs to.

```
/* REXX * /
NUM=CSFJOB('PAY01','X')
SAY 'THERE ARE ' NUM 'OCCURRENCES OF JOB PAY01'
DO I=1 TO NUM
   SAY JOBNAME() 'IS IN APPL' XAPPL.I'.'XAPPLG.I
END
```

Sample output might look like this:

```
THERE ARE 3 OCCURRENCES OF JOB PAY01
PAY01 IS IN APPL PAYROLL .293
PAY01 IS IN APPL PAYROLL .294
PAY01 IS IN APPL HR      .43
```

Using the JOBONCSF Function

The JOBONCSF function returns information about any job that is available on CSF. JOBONCSF returns data in the form of REXX stemmed variables. Unlike the RETSCBD function that returns data for a single job, the JOBONCSF function can return data for all jobs that match a job name mask, such as PAY-. This function allows you to

- Retrieve all occurrences of a particular job name.
- Retrieve all jobs matching a job name mask.
- Retrieve all jobs currently available on CSF.

Differences between CSFJOB and JOBONCSF

JOBONCSF and CSFJOB both accept a full name, but JOBONCSF works differently than CSFJOB depending on the length of the full name and if you code a period in the full name. The differences are

- If you specify a one-to-eight character full name, CSFJOB returns jobs matching the full name, but it does not include jobs with qualifiers or jobs with a period followed by other characters. For example,

  ```
  CSFJOB('MYJOB','X')
  ```

  returns MYJOB and

  ```
  JOBONCSF('MYJOB','X')
  ```

  returns MYJOB, MYJOB.QUAL, MYJOB.OTHERCHARACTERS

- If you specify a period in positions 2 through 7 of the full name, JOBONCSF does not work. For example, of the following functions, only the last one works.

  ```
  JOBONCSF('A.-','X')
  JOBONCSF('ABCDEF.-','X')
  JOBONCSF('ABCDEFG.-','X')
  ```

Syntax

n=JOBONCSF('*fullname*','*varprefix*')

Operand	Description
n	Number of matching entries returned
fullname	Full name. You can use the asterisk and hyphen wildcard characters (see “Wildcards and masking” on page xiv).
varprefix	A character string used as the prefix to a series of stemmed variables the JOBONCSF function returns

If data is returned

If the JOBONCSF returns data (the result is larger than 0), it generates REXX stemmed variables in the form *ppppvvvvv.i*, where *pppp* is the stemmed variable prefix you specified when you used the function, *vvvvv* is the variable name, and *i* is the numeric stem. For example, XJOBN.1, XJOBN.2, ... represent the first occurrence of the job, the second occurrence of the job, and so on.

Variables returned

JOBONCSF returns variables similar to the RETSCBD variables, as shown below. However, each variable starts with the prefix you specified when you invoked the JOBONCSF function.

Variable Name	Description	Format	Maximum Length Returned
AGENT	Agent name	string	16
APPL	Application name	string	8
APPLG	Application generation number	number	10
ASID	address space identifier	hexadecimal string	4
AVGRUNTIME	average job execution time in minutes	two-byte binary number	2
BYPASSED	job bypassed	1 for true, 0 for false	1
CLASS	JES job class	string	8
COMPLETE	job completed	1 for true, 0 for false	1
CONDCODE	completion code	See “CONDCODE format” on page 191.	6
CRITICAL_ PATH	critical path job	1 for true, 0 for false	1
ENDTIME	job end time	hh.mn.ss yyyy/mm/dd	19
EVENT	Event name	string	24

Variable Name	Description	Format	Maximum Length Returned
EXECJNO	JES job number where the job executed	number	6
EXECNODE	JES execution node	string	8
EXTERNAL	external job	1 for true, 0 for false	1
FAILED	job failed	1 for true, 0 for false	1
FULLNAME	full name of the job	string	64
HC	hold count	number	5
JOBN	job name	string	8
JOBNO	job number (see note below)	number	6
MANUAL	manual job	1 for true, 0 for false	1
MAXRUNTIME	maximum allowable job execution time in minutes	two-byte binary number	2
MINRUNTIME	minimum allowable job execution time in minutes	two-byte binary number	2
ORIGJNO	JES job number where the job was submitted	number	6
ORIGNODE	JES submission node	string	8
OVERDUE_ END	job overdue to end	1 for true, 0 for false	1
OVERDUE_ START	job overdue to start	1 for true, 0 for false	1
OVERDUE_ SUBMIT	job overdue to be submitted	1 for true, 0 for false	1
PGN	performance group number	number from 1 to 999 in compatibility mode, 0 on goal mode system	3
PNODE	processing node	string	8
PRIORITY	JES execution priority	number	2
QUAL	job qualifier	string	8
RDRTIME	time the job was submitted to a JES reader	hh.mn yyyy/mm/dd	19
REQUEST	request job	1 for true, 0 for false	1
REQUESTED	requested job	1 for true, 0 for false	1
SCHED	scheduled time	hh.mn.ss yyyy/mm/dd	19
SMFID	system SMF identifier	string	4
SRVCLASS	service class	string in goal mode, blank on compatibility mode systems	8
STARTTIME	job start time	hh.mn.ss yyyy/mm/dd	19

Variable Name	Description	Format	Maximum Length Returned
STATUS	status of job	string	28
SUBAP	subApplication name	string	8
SYSNAME	sysplex member name	string	8
SYSPLEX	sysplex name	string	8
TAG	tag field	string	16
TYPE	type of job	J for job or distributed object, T for task, L for link	1
USTAT	user status	string	28
WOBTYPE	type of workload object for distributed work	string	2

Note: The JOBNO value changes when a job is submitted and executed on different JES nodes. During execution, JOBNO reflects EXECJNO; otherwise, it reflects ORIGJNO.

CONDCODE format

Condition	Format	Legend	
System abend	`ƀƀSxxx`	ƀ	Blank
User abend	`ƀUnnnn`	x	Hexadecimal digit
Overflow	`nnnnnn`	n	Decimal digit
Overflow with value of more than 6 digits	`ƀƀ****`	z	Zero-suppressed decimal digit (leading zeroes not shown)
Null	`ƀƀƀƀƀ-`		All other characters represent themselves
Other (numeric value up to six digits. Values larger than six digits use the overflow format.)	`zzzzzn`		

Example: Find multiple occurrences of job

A job might exist in different Applications, in different generations of the same Application or multiple times within the same generation of an Application.

This example uses the JOBONCSF function to find all occurrences of job PAY01. For each occurrence of the job, the function generates variables with the prefix X. The

JOBONCSF function sends information back to your terminal indicating which Applications (applname.generation) the job belongs to.

```
/* REXX * /
NUM=JOBONSCF('PAY01','X')
SAY 'THERE ARE ' NUM 'OCCURRENCES OF JOB PAY01'
DO I=1 TO NUM
   SAY JOBNAME() 'IS IN APPL' XAPPL.I'.'XAPPLG.I
END
```

Sample output might look like this:

```
THERE ARE 3 OCCURRENCES OF JOB PAY01
PAY01 IS IN APPL PAYROLL .293
PAY01 IS IN APPL PAYROLL .294
PAY01 IS IN APPL HR      .43
```

Handling Errors

When working with CSF extensions, the following are some common errors that might occur:

- Invalid Select Code
- REXX exec Load file SYSEXEC does not contain exec member
- Inconsistent date parameters

Invalid select code

This message indicates that the command is not valid. Check the following:

- Have you updated your ISPF table with the new command?
- Is your table library allocated to your ISPF session?
- Has your ISPF session been refreshed?

REXX exec Load file SYSEXEC...

This message indicates the command is valid, but the REXX exec could not be located. Check that your SYSEXEC DD points to the REXX library where you stored your exec.

Inconsistent date parameters

Some execs use a date in a particular format. Check the DATEFORM setting in the ESP Workload Manager initialization parameter data set. Date formats should be specified using this format. The default is YMD (year/month/day).

17

Examples of CSF Extensions

This chapter contains examples of CSF extensions you can use to perform the following functions at your site:

- Adding Help Messages
- Accessing SDSF
- Editing a Data Set
- Displaying Job Information
- Generating a History Report
- Adding Security to Commands
- Displaying the Script Name
- Adding a Confirmation Panel
- User Request Panel
- Turning Off a Command with a Message
- Adding a Help Panel

Adding Help Messages

When you start using your own CSF extensions, you should give other users a way to access a list of available commands and their functions. You can do this by writing a CSF extension that displays a list of commands.

REXX exec

The following is a REXX exec that contains a series of SAY commands to list the CSF commands you added. You can execute this REXX exec as part of a CSF extension:

```
/* REXX */
SAY 'THE FOLLOWING ARE THE CSF EXTENSIONS CURRENTLY TURNED ON:'
SAY '***********************************************************'
SAY ''
SAY 'CAA --> COMPLETE ALL GENERATIONS OF AN APPLICATION'
SAY 'EDJ --> EDIT THE JCL FOR A JOB'
SAY 'SD  --> INVOKE SDSF'
SAY 'SIM --> SIMULATE AN EVENT'
SAY 'SUB --> SUBMIT JCL'
SAY 'TR  --> TRIGGER AN EVENT'
SAY 'NX  --> DISPLAY NEXT 10 EVENT EXECUTIONS'
SAY 'X   --> DISPLAY LIST OF CUSTOMER WRITTEN EXTENSIONS'
```

Accessing SDSF

One of the sample CSF extensions CA supplies is the following REXX exec. It invokes SDSF from CSF to access information for a job.

REXX exec

Any current active SDSF filtering, such as PREFIX or OWNER, will affect the SDSF output.

```
/* REXX    */
/* Command Name: SD   */
Address ISPEXEC,
"SELECT PGM(ISFISP) NOCHECK NEWAPPL(ISF)-
PARM('ST" Jobname() "')"
Return NOOP
```

You can use a similar technique for accessing other ISPF applications, such as IOF, $AVRS, RMDS, and so on.

Editing a Data Set

Some installations might have libraries, such as job documentation libraries, set up in PDS format where the member name is the same as the job name. To access this information directly from CSF, you can use a CSF extension that takes you into a job's correct PDS member.

REXX exec

This example uses the JOBNAME function as the PDS member name and takes the user into ISPF edit for this member of the ABC.PROD.DOCS library:

```
/* REXX */
"ISPEXEC EDIT DATASET('ABC.PROD.DOCS("JOBNAME()")'"
```

CA supplies the ED and BD commands to edit job documentation and browse job documentation respectively, but the commands are only valid if you are using the ESP Workload Manager job documentation facility. You can either replace these commands (by using the same names) or create your own command names.

Displaying Job Information

The LA line command in CSF displays information about all jobs in an Application. To limit the display of information to a particular job, you can use a CSF extension that issues an ESP LAP (or LISTAPPL) command with the JOB keyword.

If you enter LAJ next to a job in CSF, CSF displays information about that job, such as predecessors, successors, time dependencies, and other requirements.

REXX exec

This example issues an LAP command for a specific job in an Application. The command uses the APPL() function that resolves to the Application's fully qualified name and the JOB() function that resolves to the current job's fully qualified name:

```
/* REXX */
/* COMMAND NAME: LAJ */
*/
/* CSF EXTENSION TO LIST A SPECIFIC JOB IN AN APPLICATION */
*/
address ESP "LAP " APPL()  "JOB(" JOB()")"
```

Generating a History Report

This example executes an ESP Workload Manager history report based on the Application name (APPLNAME function) for the current job. It contains a number of ESP Workload Manager history-reporting commands that are concatenated into a REXX variable called CMD.

REXX exec

The exec displays the results in page mode:

```
/* REXX */
/* COMMAND NAME: REP */
CMD = "REPORT"
CMD = CMD ";SETWIDTH 80"
CMD = CMD ";FROM 8AM TODAY"
CMD = CMD ";CRITERIA APPLSYS EQ " APPLNAME()
CMD = CMD ";DISPLAY JOBNAME JOBNO EXECST CPUTIME"
CMD = CMD ";ENDR"
X = PAGEMODE(CMD)
```

Adding Security to Commands

This example shows you how to restrict command access based on the user ID issuing the command. You can use this technique with any command.

REXX exec

The REXX exec uses the REXX USERID() function to retrieve the user ID. If the first five characters are CYBER, the command is rejected and a message returned; otherwise, the CSF extension continues by executing the built-in command with the same name:

```
/* REXX */
IF LEFT( USERID(), 5 ) = CYBER then
 do
    say 'You are not allowed to use this CSF command'
    RETURN REJECTM
 end
X=CSFRET(CONTINUE)
```

Note: If you use the ESP Workload Manager SAF interface, you can also protect CSF commands using your existing host security product.

Displaying the Script Name

If you use ESP Workload Manager to schedule work on a distributed platform such as UNIX, you might want to obtain the name of the script associated with a workload object. To obtain this information, you trap the output from an ESP SIMULATE command for a specific workload object.

REXX exec

As part of the simulation, two lines relating to the message are sent to the Agent running on the distributed platform. The script name appears as part of this message. Sample output looks like one of the following, depending on the options used when defining the workload object:

```
Agentmsg 19970318 08194240+0500  SUN * OFFMVS1/PAY.0/MAIN RUN +
. Data(Script=/export/home/scripts/dw,ExitCode=(0-8,s))
```

or

```
Agentmsg 19970318 08194240+0500  AIX * OFFMVS2/PAY.0/MAIN RUN +
. Data(Script=/u1/dcooper/scripts/dw)
```

The following exec parses the output from the SIMULATE command. The Event() function retrieves the Event name, the Sched variable from the RETSCBD function retrieves the scheduled time for this Event, and the Job() function retrieves the current job name. The exec parses the data in two ways to accommodate differences in the output, obtains the script name, and sends this information to the user issuing the following command:

```
/* REXX                           */
/* Command Name: LX              */
X = Retscbd()
Z=outtrap("line.")
 "SIM EV(" || Event() || ") SCHED('"Sched"') ROOT("Job()")"
Z=outtrap("off")
parse var line.2  something '=' script1 ','
parse var line.2  something '=' script2 ')'
if length(script1) < length(script2) then scriptname=script1
  else scriptname=script2
if scriptname = '' then say 'no script for this object'
  else say 'scriptname is ' scriptname
```

Adding a Confirmation Panel

The RD command is a CSF command that readies a job, removing any submit time and predecessor dependencies for that job. ESP Workload Manager does not provide the user with a confirmation panel for this command. The following example adds a confirmation panel for the RD command.

REXX exec

The following REXX exec presents a panel called CSFRDC (which must be stored in your ISPPLIB data set) to the user. If the user presses Enter, the return code (RC) is 0 and the extension continues with the normal RD command. If the user presses End, the return code (RC) is 8 and the extension rejects the command with an error message.

```
/* REXX */
ISPEXEC "DISPLAY PANEL(CSFRDC)"
IF RC=0 THEN RETURN CONTINUE
  ELSE RETURN REJECTM
```

To develop the CSFRDC panel, you can use panel CYBESSCR from your ESP Workload Manager panel library (*prefix*.SSCPPENU) as a model confirmation panel. Here is an example of a confirmation panel:

```
)ATTR
  ! TYPE(OUTPUT) INTENS(HIGH)
)BODY EXPAND(//)
%ESP /-/ Job Ready Request Confirmation /-/ ESP+
%COMMAND ===>_ZCMD              +
+    Confirm request to ready job
+
     Press%ENTER+to proceed with the ready request
     Press%END+to cancel the request
)PROC
)END
```

When you add the RD command to your CYBESCSU member, you do not need to specify the ISPF option because the exec does not use ISPF variables. However, if you specify ISPF (for example, `LC RD EXEC(NEWRD) ISPF`), an ISPF SELECT function invokes the exec and you need to use the CSFRET function to return values, for example:

```
/* REXX */
ISPEXEC "DISPLAY PANEL(CSFRDC)"
IF RC=0 THEN X=CSFRET(CONTINUE)
  ELSE X=CSFRET(REJECTM)
```

User Request Panel

The following is an example of a customized user panel you can present to users to handle simple job submissions. When the user enters the command you assign to this extension, ESP Workload Manager presents the user with a panel that shows a job's name and, optionally, predecessors and successors. After the user enters the data, ESP Workload Manager builds an AJ command to insert the job into the current Application.

Panel definition

The following is the sample panel definition that must be stored in ISPPLIB:

```
)BODY EXPAND(#@)
%ESP #-@ USER REQUEST PANEL #-@ ESP+
%COMMAND ===>_ZCMD +
 Fill in appropriate fields then press%ENTER.+
 Use a comma or a space to separate multiple
 predecessors or successors.
 To cancel your request press%PF3. +
   Job information:
       JOBNAME      %===>_Job             +
       PREDECESSORS %===>_Pred                  +
       SUCCESSORS   %===>_Succ                  +
)INIT
  .CURSOR=Job
)PROC
  VER (&Job,NB)
)END
```

REXX exec that presents the panel

The following is the REXX exec that presents the panel and builds an ESP Workload Manager command to insert the job into the current Application:

```
/* REXX */
/* Command Name: NP  */
/* Blank out Variables used in panel NEWPAN */
 Job  = ""
 Pred = ""
 Succ = ""
/* Display panel NEWPAN to obtain options */
ADDRESS ISPEXEC "DISPLAY PANEL(NEWPAN)"
If Rc <> 0 Then
   Return
/* Construct AJ command */
Cmd = "AJ " Job "INSERT APPL("APPL()")"
IF Pred <> "" THEN Cmd = Cmd || " PREDECESSOR("Pred")"
IF Succ <> "" THEN Cmd = Cmd || " SUCCESSOR("Succ")"
/* Execute the AJ command. */
ADDRESS ESP CMD
```

CYBESCSU member

When you add this command to your CYBESCSU member, you must specify the ISPF option because the exec uses ISPF variables, for example:

```
LC  NP  EXEC(ADHOC)  ISPF
```

Turning Off a Command with a Message

You use the CA command to mark an entire Application complete. The CSF extension in this example replaces the existing CA command and rejects the command.

When a user enters the CA command beside a workload object in CSF, the CSFCANO exec executes. This exec issues a customized message (CSFX001), stored in member CSFX00 of your ISPMLIB data set, and rejects the command.

CYBESCSU member

Update your CYBESCSU member and specify CA as the command name:

```
LC CA EXEC(CSFCANO)
```

REXX exec

The following is the REXX exec that sets a customized message and rejects the command:

```
/* REXX */
/* SET CUSTOMIZED MESSAGE AND REJECT COMMAND */
ISPEXEC "SETMSG MSG(CSFX001)"
RETURN REJECT
```

Member CSFX00 in your message library

The following member in your message library (ISPMLIB) contains a sample customized message:

```
CSFX001 'CA command not allowed' .ALARM=YES
'The CA command has been disallowed by my extension'
```

Adding a Help Panel

The following REXX exec displays a panel (CSFHELP) to list the CSF commands you added. You can execute this REXX exec as part of a CSF extension.

REXX exec

```
/* REXX             */
/* CSF HELP PANEL   */
ADDRESS ISPEXEC "CONTROL ERRORS RETURN"
ADDRESS ISPEXEC "DISPLAY PANEL(CSFHELP)"
IF RC <> 0 THEN RETURN NOOP
```

Panel definition (CSFHELP)

The following is a sample CSF Help Panel:

```
)BODY EXPAND(@@) WIDTH(&ZSCREENW)
%COMMAND ===>_ZCMD+
+              %CSF HELP PANEL                +
+                                             +
%**********************************************
%*    +  List of CSF Extension Commands       %*
%*                                            %*
%*    +  ED  - Edit Dataset                   %*
%*    +  XH  - Help Panel                     %*
%*    +  TR  - Trigger an Event               %*
%*    +  SIM - Simulate an Event              %*
%*                                            %*
%**********************************************
+
%
%
)PROC
)END
```

18 Implementing Application Monitor Extensions

This chapter describes how you can extend the functionality of Application Monitor by using Application Monitor extensions.

This chapter contains the following topics:

- Introduction to Application Monitor Extensions
- Installing and Testing the Sample Extensions
- Implementing Your Own Application Monitor Extensions
- Adding and Customizing Commands
- Example: Adding help messages

Introduction to Application Monitor Extensions

Application Monitor enables you to view and control active Applications in ESP Workload Manager. Application Monitor extensions are a programming interface that allow you to customize Application Monitor by writing your own commands or by replacing existing commands.

For more details on Application Monitor, see the *ESP Workload Manager Operator's Guide*.

Uses for Application Monitor extensions

You can use Application Monitor extensions to

- Reduce frequently performed functions to a single command.
- Access other ISPF applications from the Application Status panel.

  For example, you can set up access to output distribution products and spool viewers.
- Control access to current Application Monitor functions.

  For example, if you want to prevent users from using a command or to provide a confirmation panel, you can write a REXX exec or program to provide a front-end for an existing, internal Application Monitor command. The exec can then decide whether to allow the internal command to proceed, reject the command or provide an alternative function.

Addition of commands

To add or replace a command, you use an ISPF table called CYBESCSA. Each Application Monitor command invokes a REXX exec or program. These might access REXX functions, ESP Workload Manager commands, and ISPF services, and then return to Application Monitor with a return code.

Distribution package

As part of the distribution package of ESP Workload Manager, CA provides

- Sample Application Monitor extensions.
- A general purpose REXX exec, CYBESTBG, to generate the ISPF table CYBESCSA.
- Sample member CYBESCSA, which is input to REXX exec CYBESTBG.

References to libraries

Throughout this chapter, there are references to libraries you created as part of the ESP Workload Manager installation. Whenever there is a reference to one of these libraries, such as *prefix*.SSCPSAME, you need to substitute the prefix you chose during installation.

Installing and Testing the Sample Extensions

What you need

- Sample libraries *prefix*.SSCPSAME and *prefix*.SSCPCLST
- A REXX exec library to store sample Application Monitor extensions
- An ISPF table library, such as the ESP Workload Manager table library (*prefix*.SSCPTENU)

You must allocate the REXX exec library and the ISPF table library to a user's session to allow them to use the sample extensions. You can allocate these libraries through an exec, a CLIST or as part of a logon proc.

Installing the extensions

1. Copy the sample REXX execs listed in the following table from library *prefix*.SSCPSAME to a REXX exec library that is allocated to a SYSEXEC DD.

REXX Exec	Command	Description
CYBESS60	CAA	Completes all generations of a user-specified Application older than a user-specified time
CYBESS67	TR	Triggers the Event associated with a job
CYBESS69	NX	Displays the next 10 execution times for an Event associated with a job
CYBESS84	SIM	Simulates the Event associated with a job

2. If you want, copy member CYBESCSA from library *prefix*.SSCPCLST to another library.

   CYBESCSA contains information relating to your new commands. CYBESCSA is read by the CYBESTBG exec. You update CYBESCSA whenever you add new Application Monitor extensions.

3. Copy the CYBESTBG REXX exec from the *prefix*.SSCPCLST library to a REXX exec library that is allocated to SYSEXEC.

   CYBESTBG reads member CYBESCSA and generates an ISPF table library member, also called CYBESCSA, that stores your commands. You execute CYBESTBG whenever you add new Application Monitor extensions.

4. In member CYBESCSA, change the LIBRARY statement to point to a table library that is allocated to your ISPTLIB DD.

   You can specify your ESP Workload Manager table library (*prefix*.SSCPTENU) or a user table library, for example:

   ```
   LIBRARY CYB3.ESP.SSCPTENU
   ```

5. Run the CYBESTBG exec, specifying the location of input member CYBESCSA, for example:

   ```
    TSO CYBESTBG 'CYB3.ESP.SSCPCLST(CYBESCSA)'
   ```

   A message appears stating the number of rows added to ISPF table CYBESCSA.

6. Exit to the READY prompt.

   Your ISPF session is refreshed, and you can now use the new commands.

Testing the extensions

You can test the new extensions by issuing the associated commands against any test Applications listed in the Application Status panel.

A good command to test is the SIM command, which simulates the functions of an Application.

To test the SIM command

1. Sign onto ESP Workload Manager.
2. Choose option A for Application Monitor.
3. Type `SIM` beside any Application name and observe the results.

Implementing Your Own Application Monitor Extensions

This section describes how to create and implement your own Application Monitor extensions. It assumes you are familiar with some of the aspects of installing Application Monitor extensions by installing the samples supplied.

Note: ESP Workload Manager does not provide any REXX built-in functions, such as the RETSCBD and JOBONCSF functions available in CSF, for creating Application Monitor extensions.

Preparing to implement your own extensions

In addition to a REXX exec library and a table library to store your commands, you might choose to add a panel library for customized panels and a message library for customized messages.

The following table lists the libraries you can use and the corresponding DD statements:

Type of Library	DD Statement
REXX exec library	SYSEXEC
Table library	ISPTLIB
Panel library (optional)	ISPPLIB
Message library (optional)	ISPMLIB

You can allocate these libraries to a user's session using a CLIST exec or include them in a logon proc.

To add an extension

1. Design and write the REXX exec (or program) and store it in a library allocated to the SYSEXEC DD.
2. Update your list of commands by editing the CYBESCSA member. For details, see "Adding and Customizing Commands" on page 208.
3. Run the CYBESTBG exec to update the table of commands (allocated to ISPTLIB DD). Specify the name of the library and member where you stored your list of commands, for example:

   ```
   TSO CYBESTBG 'CYB3.ESP.SSCPCLST(CYBESCSA)'
   ```
4. Re-enter ISPF to test.

Example: Adding an extension

The following is a simple, step-by step example of adding a new Application Monitor extension. This example can help you get started with implementing your own extensions.

In this example, you add a command called HI that executes a REXX exec called APPHELLO. This exec sends the message "HELLO THERE" to the user who enters the HI command beside any Application listed in Application Monitor.

To create the extension

1. Write the exec and store it in a member called APPHELLO in a REXX exec library:

   ```
   /* REXX */
   /* Command name HI */
   SAY 'HELLO THERE'
   ```
2. Update your list of commands in the sample input member CYBESCSA. For this example, add the line shown below at the end of the member.

   ```
   LC  HI  EXEC(APPHELLO)
   ```

3. To update your ISPF table library, run the CYBESTBG exec, specifying the name of the input data set you just updated. The CYBESTBG exec creates the appropriate table library member.

   ```
   TSO CYBESTBG 'CYB3.ESP.SSCPCLST(CYBESCSA)'
   ```

4. Re-enter ISPF and test your command by entering HI beside any Application listed in Application Monitor.

Updating the exec

After you have created your exec, you can update it at any time, and the changes take effect immediately. There is no need to update any tables or refresh ISPF.

For example, if you want to update the exec in the previous example to say HELLO followed by the user ID of the user issuing the HI command, make the following change:

```
/* REXX */
/* Command name HI */
SAY 'HELLO' USERID()
```

Adding and Customizing Commands

You can add or customize commands by editing the CYBESCSA member. The CYBESCSA member stores your list of commands and is used as input to the CYBESTBG exec to create the CYBESCSA table library.

Listing of sample member CYBESCSA

```
* Table of REXX extension commands for ESP Application Monitor
*
*     Note: You must modify this input and run the REXX Exec
*       'CYBESTBG' to generate the ISPF table library member:
*       "TSO CYBESTBG prefix.SSCPCLST(CYBESCSA)"
*
* REF: ASEPCLST(CYBESCSA), ASEPSAME(CYBESSxx) SEP5500 - 01Apr2006

LIBRARY CYB3.ESP.SSCPTENU
MEMBER CYBESCSA
VARS CMDTABNM CMDTNAME CMDTPARM
KEYS CMDTABNM CMDTNAME
OPTIONS REPLACE
DATA' '
LC CAA EXEC(CYBESS60)
LC NX  EXEC(CYBESS69)
LC SIM EXEC(CYBESS84)
LC TR  EXEC(CYBESS67) ISPF
```

CYBESCSA line command entries

By coding line command entries in CYBESCSA, you can

- Add new line commands.
- Replace the functionality of existing commands.
- Disable existing commands.
- Add a front-end to existing commands.

When a user issues a command in Application Monitor, ESP Workload Manager looks first for an entry in CYBESCSA to decide how to process the command.

For each new or existing command you want to process, add one line command entry after the `DATA'  '` line in CYBESCSA.

Syntax

```
LC command-name process processing-options
```

Operand	Description
LC	Signifies that the entry is for a line command issued in the Application Status panel
command-name	A new line command or an existing line command. Line commands are one to three characters long.
process	The process to execute for the line command specified. For details, see "Processes" on page 209.
processing-options	One or more options for the specified process. For details, see "Process-options" on page 210.

Processes

Process	Description
EXEC(*rexx-exec*)	Invokes a REXX exec. The REXX exec should be in a library allocated to the SYSEXEC DD. You can pass parameters to the REXX exec using the PARMS parameter, as described in Process-options.
PGM(*program*)	Calls a program. The program should be a load module accessible to the ISPF session and should be linked with AMODE=31. You can pass parameters to the program using the PARMS parameter, as described in Process-options.
INVALID	Informs the user that the command is invalid
NOOP	Ignores the command

Process-options

Option	Parameter
PARMS(*parms*)	A parameter string that is passed to the REXX exec or program specified in Processes
ISPF	Use ISPF variable services and call the REXX exec with an ISPF SELECT function call. If you don't specify the ISPF option, Application Monitor invokes the REXX exec directly. Use this option if you want to manipulate or examine ISPF variables from REXX.
RETATR	Retrieve the ATR (Application Tracking Record) prior to calling the program or REXX exec. The address of the ATR is provided.

Examples

In the following example, the line command SIM invokes a REXX exec called APPSIM:

```
LC SIM EXEC(APPSIM)
```

APPSIM now overrides the standard module for SIM, CYBESS84.

In the following example, the line command TR invokes a REXX exec called CYBESS67. ISPF variable services will be available.

```
LC TR EXEC(CYBESS67) ISPF
```

Example: Adding help messages

Once you start using your own Application Monitor extensions, you should provide a method for users to access a list of available commands and their functions. For example, you can write an Application Monitor extension that displays a list of commands.

The following REXX exec contains a series of SAY commands to list the Application Monitor commands you have added. You can execute this REXX exec as part of an Application Monitor extension:

```
/* REXX */
SAY 'APPLICATION MONITOR EXTENSIONS THAT ARE TURNED ON:'
SAY '*****************************************************'
SAY ''
SAY 'CAA --> COMPLETE ALL GENERATIONS OF AN APPLICATION'
SAY 'SIM --> SIMULATE AN EVENT'
SAY 'TR  --> TRIGGER AN EVENT'
SAY 'NX  --> DISPLAY NEXT 10 EVENT EXECUTIONS'
```

19

Setting Up User Modifications

The USERMOD feature provides facilities for customizing ESP Workload Manager. These facilities are assigned unique identifying numbers, each an integer ranging between 1 and 255. You specify which USERMOD, if any, you want ESP Workload Manager to activate. You can also deactivate any previously activated USERMODs.

This chapter contains the following topics:

- Activating USERMODs
- ESP Workload Manager USERMODs

Activating USERMODs

You can activate a USERMOD upon startup or while ESP Workload Manager is running. The activity status of a USERMOD specified upon ESP Workload Manager startup overrides the status of the same USERMOD retained from a previous ESP Workload Manager run.

The status of a USERMOD is retained past the termination of ESP Workload Manager, and remains in effect when ESP Workload Manager restarts. Upon IPL, however, the activity status of all USERMODs is lost, unless USERMODs are set in the ESP Workload Manager initialization parameters.

Important: When setting up USERMODs in an environment with multiple ESP Workload Manager subsystems, ensure all your subsystems have the same USERMODs set. This includes the ESP Workload Manager master, proxies, and shadow managers.

Activating USERMODs on initialization

To activate a single USERMOD upon initialization

Code the following ESP Workload Manager initialization parameter:

```
USERMOD SET(n)
```

where *n* represents the USERMOD number.

To activate a contiguous range of USERMODs upon initialization

Code the following ESP Workload Manager initialization parameter:

```
USERMOD SET(min:max)
```

where *min* represents the smallest USERMOD number in the range and *max* represents the largest USERMOD number.

To activate multiple USERMODs, you can also specify several USERMOD initialization parameters.

Activating USERMODs while ESP Workload Manager is running

While ESP Workload Manager is running, you can activate or deactivate a USERMOD via the USERMOD command. The following is an example of activating USERMOD 33:

```
OPER USERMOD SET(33)
```

To obtain a list of active USERMODs

Issue the following USERMOD command:

```
OPER USERMOD LIST
```

For the syntax of the USERMOD command, see the *ESP Workload Manager Reference Guide.*

Activating or deactivating USERMODs temporarily

While ESP Workload Manager is running, you can activate or deactivate USERMODS temporarily. To ensure consistency at startup in installations where USERMODs are temporarily activated, code the following initialization parameters:

```
USERMOD RESET(1:255)
USERMOD SET(n1)
USERMOD SET(n2)
        .
        .
        .
USERMOD SET(nk)
```

where *n1*, *n2*, …, *nk* represent permanent USERMOD numbers or ranges.

ESP Workload Manager USERMODs

USERMOD 1

When USERMOD 1 is active, ESP Workload Manager treats the flush of the last step of a job as a job failure.

USERMOD 2

USERMOD 2 allows you to use the ESP PURGSCHF command to purge all completed jobs, even those belonging to incomplete Applications. This USERMOD is useful if the scoreboard has grown so large that CSF performance is suffering.

When USERMOD 2 is not active, you can use the ESP PURGSCHF command only to purge completed jobs in completed Applications.

USERMOD 3

When USERMOD 3 is active, ESP Workload Manager allows procedures with improperly nested DO/ENDDO blocks to run without any DO/ENDDO checking. This USERMOD is useful for customers who have old ESP Workload Manager Procedures with missing ENDDO statements.

USERMOD 4

When USERMOD 4 is active, ESP Workload Manager rebuilds the REXX parameter block model for each execution of an Event.

When USERMOD 4 is not active, ESP Workload Manager builds a REXX parameter block model upon startup.

USERMOD 5

When USERMOD 5 is active, ESP Workload Manager foregoes the cross-memory TPUT and always issues the SEND command.

When USERMOD 5 is not active, ESP Workload Manager attempts a cross-memory TPUT when notifying a TSO user. If this fails, ESP Workload Manager issues a SEND command. If the NOW keyword is specified on the SEND statement and the TPUT fails, ESP Workload Manager does not send the message.

USERMOD 6

When USERMOD 6 is active, the ESP(*userid*) identification tag is not appended. Installations that activate this USERMOD prefer readability of multi-line console messages over security concerns.

An identification tag (the string ESP(*userid*)) is appended to the WTO console message if the following three conditions are met:

- USERMOD 6 is not activated.
- The user codes a SEND command in an Event.
- The output is routed to the console.

USERMOD 7

When USERMOD 7 is active, NOPOST_UNTIL_READY is ignored for all non-MANUAL jobs. That means that job posts for non-MANUAL jobs are accepted even if they are not READY.

When USERMOD 7 is not active, if NOPOST_UNTIL_READY is set for non-MANUAL jobs, a job post is not accepted if the job is not READY. Under normal circumstances, this is satisfactory. However, some Applications cannot use this facility because of manual job processing.

USERMOD 8

USERMOD 8 streamlines the work in data centers that experience a high volume of job submissions, such as 1000 jobs per hour. When this USERMOD is active, ESP Workload Manager allows the Application Manager to voluntarily relinquish execution after four jobs have been scheduled for submission. This allows the Event processor to begin submitting jobs before the Application Manager schedules more.

USERMOD 9

When USERMOD 9 is active, ESP Workload Manager generates an audit log message each time the Application Manager is notified that one of its jobs has executed.

USERMOD 11

USERMOD 11 improves performance in data centers that experience a high volume of job submissions, such as 1000 jobs per hour. This is done by not sending step-end information to CSF for all the jobs, thus reducing access to the Application file.

One side-effect is that job status information is not updated on CSF while a job is running. Instead of showing EXECUTING, STEP *nn* in the STATUS column, ESP Workload Manager shows EXECUTING, STEP 1 until the job ends. An LJ command shows the true status and step-level monitoring can be used.

USERMOD 12

When USERMOD 12 is active, ESP Workload Manager does not require UPDATE authority for the resubmission of a job from a different JCLLIB.

When USERMOD 12 is not active, UPDATE authority is required to resubmit a job from a library other than its original JCLLIB.

USERMOD 13

When USERMOD 13 is active, variables are not padded to their full length.

When USERMOD 13 is not active, ESP Workload Manager processes as documented variables that are defined to the full length. For example, when coding

```
%variable(length)
```

and *length* is longer than the length of the string assigned to `variable`, the data are padded up to the full length.

USERMOD 14

USERMOD 14 provides timely diagnostic information in circumstances where certain in-memory queues belonging to ESP Workload Manager become corrupted. In some cases, it also allows processing to continue in a somewhat degraded manner while the problem is being diagnosed.

USERMOD 14 operates by checking for existing duplicates each time an entry is added to one of the queues in question, and by checking for circularity in these queues at that time. If an existing entry is found that duplicates the new entry to be added, an SVC dump is produced, with the title ESP APMW Loop; processing then continues, although the Application causing the problem might not process completely. The SVC dump is produced only the first time this problem occurs during the lifetime of an ESP Workload Manager execution. If a circular queue is detected, ESP Workload

Manager immediately terminates with a User ABEND code 700 because it is impossible to recover from such a situation.

USERMOD 14 adds significant processing overhead to ESP Workload Manager operation, and should be activated only at the request of CA Technical Support.

USERMOD 15

When USERMOD 15 is active, ESP Workload Manager stores a job's execution class in the job-tracking record, in the field where the Network ID is usually stored. This enables users who do not care about the NET ID to include the execution class in job history reports by referring to the NETID field. The execution class is also passed to monitors, as the value of the MNNETID variable.

USERMOD 16

When USERMOD 16 is active, ESP Workload Manager writes a message to the audit log for every job the PURGSCHF command purges from CSF.

USERMOD 17

USERMOD 17 is no longer available.

USERMOD 18

When USERMOD 18 is active, an attempt is made to distinguish, for the purpose of job tracking, between jobs with the same name running in rapid succession, when ESP Workload Manager running under JES2 transmits the jobs to be executed under a different JES2 node. This is done by manipulating the difference between the time stamps of the two JES subsystems. Without this USERMOD, such a manipulation is performed only if the remote node is a JES3 node.

USERMOD 20

When a job name is unavailable because of a JCL error in the JOB statement, JES2 substitutes the name JOBFAIL for the missing job name. This job name identifies the job when JES2 issues a WTO for a diagnostic message on the operator's console. ESP Workload Manager detects this WTO message. ESP Workload Manager WTO intercept routine also inserts the name JOBFAIL for certain IRR series messages issued for security failures.

When active, USERMOD 20 causes ESP Workload Manager to ignore WTO messages of JCL errors if JOBFAIL has been substituted for the job name.

USERMOD 21

When USERMOD 21 is active, the Application Manager assumes there are no valid ESP Workload Manager statements in the documentation library members. Given this information, it does not need to allocate, open, search for and read members of that library. The data set and member name information is simply preserved with the job data, so that it can be referenced later via CSF.

This behavior can avoid significant processing overhead in cases where a documentation library contains a very large number of members, with a directory spanning possibly several cylinders. It is only useful when the library is used solely for documentation purposes, and indeed does not contain any ESP Workload Manager statements.

USERMOD 22

Under normal circumstances, if an ESP Workload Manager Application definition contains an invalid ESP Workload Manager command, that Application generation is abandoned. Similarly, if an Event definition contains an invalid ESP Workload Manager statement, that Event does not execute.

When active, USERMOD 22 causes ESP Workload Manager to reject the statement in error but still process the remaining statements in the Application or Event definition.

USERMOD 25

When USERMOD 25 is activate, ESP Workload Manager insists that, for job execution to occur, the following criteria must be met:

- Upon submission from TSO, or directly by a SUBMIT command from an ESP Workload Manager Event, the submission member name and the job name in the JCL JOB statement are the same. Otherwise, the job is not executed and ESP Workload Manager issues the diagnostic message ESP2828E.
- Upon submission from an ESP Workload Manager Application, the job name in the Application JOB statement and the job name in the JCL JOB statement are the same. Otherwise, the job is not executed and ESP Workload Manager issues the diagnostic message ESP2827E.

When USERMOD 25 is not active, job execution occurs even if multiple jobs are included in the submission member, and regardless of discrepancies in the job name. However, in case of discrepancy where there is no match for any job name, ESP Workload Manager does not track the job properly; furthermore, if the discrepancy occurs in an Application, a Submit Error message, JCL missing appears.

USERMOD 27

When USERMOD 27 is active, ESP Workload Manager ignores any attempt you make to set an explicit owner different from the defining user when you define an Event, which would force the owner of the Event to be the defining user. This happens whether the defining owner is authorized to assign explicit ownership of Events or not.

USERMOD 29

When USERMOD 29 is active, a SADUPD run does not to store the actual start and end times for a rerun execution of a job.

USERMOD 30

When USERMOD 30 is active, it disables a new functionality introduced with ESP Workload Manager R5.1.0 for users who rely on the previous behavior.

- For z/OS jobs

  If USERMOD 30 is not active, the job is forced complete in the remote Application. This happens whether the job has actually been submitted or not.

  If USERMOD 30 is active, the job is forced complete in the remote Application only if it has actually been submitted (that is, the JES internal reader successfully processed the remote Application). This was the default behavior of ESP Workload Manager prior to R5.1.

- For other jobs

  USERMOD 30 has no effect. The job is always forced complete in the remote Application, regardless of its actual submission status.

USERMOD 31

When USERMOD 31 is active, ESP Workload Manager treats all data sets with DISP=(NEW,*xxxx*), where *xxxx* is any value but CATLG, as DISP=(NEW,CATLG).

When you request CLEANUP, ESP Workload Manager uncatalogs tape data sets with DISP=(NEW,CATLG) and deletes (uncatalogs and scratches) disk data sets with DISP=(NEW,CATLG).

Important: Use caution when activating USERMOD 31. While it gives you more flexibility when coding JCL, ESP Workload Manager might inadvertently uncatalog or delete some data sets. See the following example.

Example — Inadvertent data set deletion with USERMOD 31

1. Data set ABC exists and is cataloged.
2. A job under ESP Encore creates temporary data set ABC and never catalogs it.

If USERMOD 31 is active and

- The cataloged data set is on tape, ESP Workload Manager uncatalogs the data set.
- The cataloged data set is on disk, ESP Workload Manager deletes (uncatalogs and scratches) the data set.

If USERMOD 31 is not active, the preceding problem does not occur.

USERMOD 32

When USERMOD 32 is active, any user can define and modify Events whose prefix is identical to that user's TSO user ID. Without this USERMOD, a user must satisfy all security checks before being allowed to access any Event.

USERMOD 33

USERMOD 33 negates the effect of the following change:

- As of ESP Workload Manager R5.1.0, advanced due-out warning is enhanced by propagating due-out times back up an Application's job hierarchy from any job with a DUEOUT or LATESUB specification. This facilitates the prediction of a situation that might result in a critical job becoming late at the earliest possible moment, for example, as soon as any predecessor fails to meet a due-out time.

USERMOD 33 completely bypasses this backward propagation of due-out times.

USERMOD 34

USERMOD 34 negates the effect of the following changes:

- As of PTF SEPS081, the execution node ID in a job's JTR starts with the SMF ID in the type 30-1 SMF record for that job. ESP Workload Manager ignores any type 30-4 or type 30-5 SMF record for the job with an execution node that does not match this value. This mechanism avoids some subtle timing problems with jobs transmitted between nodes when records might arrive in an unexpected sequence.
- This execution node ID is now not overwritten by the JESplex member ID from the type 26 (purge) SMF record, as was previously the case.

You can use USERMOD 34 to satisfy existing report selection logic.

USERMOD 35

When USERMOD 35 is active, you can use the APPLJOB (AJ) command's operator version, which allows you to mark jobs, subApplications, and Applications complete.

USERMOD 36

When USERMOD 36 is active, JOBEND and STEPEND monitors honor the Event class of the initiating Event and run in an Event initiator of the appropriate class. Without this USERMOD, all JOBEND and STEPEND monitors run in a class 0 Event initiator.

USERMOD 38

When USERMOD 38 is active, ESP Encore ignores I/O errors on the EXH file and continues to process.

USERMOD 42

USERMOD 42 allows you to use CCCHK and CCFAIL in the same job.

USERMOD 44

USERMOD 44 causes ESP Workload Manager to issue the informational message 2779I when two or more jobs have the same name and are of the same type.

USERMOD 49

USERMOD 49 deactivates ESP Encore simulation.

USERMOD 52

When active, USERMOD 52 suppresses Type 30 SMF data that OMVS created.

USERMOD 56

When USERMOD 56 is active, ESP Workload Manager disregards ESP group and allows jobs with higher priority to get resources before jobs with lower priority. For details, see the PRIORITY statement in the *ESP Workload Manager Reference Guide*.

USERMOD 58

Usermod 58 suppresses warning message 2069W. This message can appear when

- The long job name does not conform to the syntax rules for a full name.
- The long job name is the same as the long job name or job name of another job in this Application.
- The long job name differs from the long job name for this job specified earlier in this Application.

USERMOD 59

Usermod 59 disables the ability for long name to become the full name for a job.

USERMOD 60

Usermod 60 changes characters specified in the *value* operand of the CRITERIA command to upper case before doing the comparison.

USERMOD 75

USERMOD 75 prevents you from setting or changing the available quantity of a resource using the RESDEF command. You can still set the MAX quantity that, in turn, changes the available quantity.

USERMOD 79

The CCCHK initialization parameter or statement adds step ESPCCCHK to the top of the job. USERMOD 79 sets the SYSPRINT DD sysout class in ESPCCCHK to Z as follows:

```
//SYSPRINT DD SYSOUT=(Z)
```

Without USERMOD 79, the SYSPRINT DD sysout class is set to comma as follows:

```
//SYSPRINT DD SYSOUT=(,)
```

USERMOD 128

When USERMOD 128 is active, ESP Workload Manager provides improved diagnostic information to NJE users when they are cross-node tracking between JES3 nodes. An SVC dump is taken for the first occurrence of each of the messages ESP2006E and ESP2007E. A new message, ESP2009E, is produced for each bad transaction, showing the bad record in characters and as hexadecimal data. Only one dump of each type is taken for every ESP Workload Manager startup.

When USERMOD 128 is not active, the ISC receiver discards erroneous data when it detects them.

USERMOD 129

USERMOD 129 causes a 10-second delay after a job message is sent to an Agent before the corresponding Agent Notified message is sent to the Application Manager. This is a diagnostic USERMOD, intended to help track down timing problems when communicating with distributed Agents.

USERMOD 182

USERMOD 182 changes the date ESP Workload Manager returns when you specify criteria that resolves to a non-existing date.

For example, if USERMOD 182 is not active, the schedule criteria “31 May less one month” returns “31 May” because there is no 31 April. If USERMOD 182 is active, the same schedule criteria returns “30 April”.

USERMOD 183

When USERMOD 183 is active, ESP Workload Manager bypasses the BLDL SVC and searches the PDS directory directly using EXCP when searching for a member of a PDS.

USERMOD 184

When USERMOD 184 is active, ESP Workload Manager ignores certain date errors when you issue the TRIGGER command. For example, if USERMOD 184 is not activate, a schedule criteria of “51 june 2003 18:00” causes an error.

USERMOD 185

USERMOD 185 causes the absence of a right parenthesis to be treated as an error, contrary to standard TSO practice.

USERMOD 187

Usermod 187 displays all job names in upper case. By default, all job names display in their original case in ESP Workload Manager version 5.5.

20

Specifying ESP Workload Manager User Exits

ESP Workload Manager provides a series of user exit points to help you customize some ESP Workload Manager functions. Review the list of user exits and complete the specification and design of any required exit modules.

This chapter contains the following topics:

Activating and Controlling a Supplied User Exit

You can activate a user exit supplied by ESP Workload Manager, to customize certain ESP Workload Manager functions. You can also control the status of a user exit, for example, you can disable a user exit or unload the load module associated with the user exit.

To activate and control the status of a user exit

- Code the EXIT initialization parameter or issue the EXIT command.

For example, to specify the STARTUP user exit, specify the following:

```
EXIT FUNCTION(STARTUP) MODULE(usercodename)
```

User Exits Shipped with ESP Workload Manager

The ESP Workload Manager package includes several user exits you can use. The following is a list of the user exits shipped with ESP Workload Manager:

- COLDSTART
- CYBESU40
- CYBXPC01
- DUEOUT
- EVENTDEF
- EVENTEXEC
- EVENTSAF
- JCLSCAN
- JOBPSWD
- JOBENDNOTIFY
- SHUTDOWN
- STARTUP
- USERSEND
- VSCMD
- WARMSTART

For information about user exits. see the *ESP Workload Manager System Programmer's Guide*.

Write a Data Set Processing Exit for ESP Encore

For complex naming conventions that ESP Workload Manager's wildcard capability does not support, you can write an exit routine that can decide how ESP Encore handles each data set in a job.

ESP Encore invokes a user-defined exit routine while processing each data set in a job. You specify the exit by adding the following statement to the ESP Workload Manager initialization parameters or the ESP Workload Manager procedure that submits the job. The module name is the member name of the load module that contains the exit. This load module must be present in the ESP Workload Manager load library.

```
ENCPARM MODIFY  DSEXIT(module)
```

ESP Encore invokes the exit with standard BALR linkage in 31-bit addressing mode. The register contents upon entry to the exit are as follows:

```
R1:  address of a list of parameter addresses
R13: address of an 18-word save area
R14: the return address
R15: the entry point address of the exit
```

The calling sequence used when invoking the exit is

```
Parm 1:  44-byte data set name
Parm 2:  6-byte serial of first volume
Parm 3:  1 byte of description flags:
         X'80': data set is cataloged
         X'40': data set is tape or cartridge
         X'20': data set is VSAM
Parm 4:  1 byte of action flags:
         X'80': ESP Encore wants to delete this data set
         X'40': ESP Encore wants to only scratch it
         X'20': ESP Encore wants to only uncatalog it
Parm 5:  1 byte of result flags
         X'80': ESP Encore should ignore this data set
```

The exit routine might inspect the first four parameters and might modify the fifth. The exit routine will ignore any changes to the first four parameters. If the exit routine sets the flag in the fifth parameter, ESP Encore will not perform actions on the data set.

Note: There is an example of a data set processing exit in your sample library (*prefix*.SSCPSAME) in member CYBRMDSX.

Part 5

Installation Reference

This part provides detailed information about each of the ESP Workload Manager initialization parameters, to help you configure your ESP Workload Manager systems.

This part consists of the following chapter and subsequent sections detailing each initialization parameter:

- About ESP Workload Manager Initialization Parameters
- ESP Workload Manager Initialization Parameters

21

About ESP Workload Manager Initialization Parameters

Initialization parameters set the conditions under which ESP Workload Manager runs and are read during ESP Workload Manager startup.

This chapter contains the following topics:

- Initialization Parameter Summary
- Entering Initialization Parameters
- Using Symbolic Variables in the Initialization Parameters
- Choosing the Subsystem Type
- Choosing the Initialization Data Set

A detailed description of each initialization parameter, listed alphabetically, follows this chapter.

Initialization Parameter Summary

This section summarizes the initialization parameters and their functions. Some initialization parameters are also available as commands. The initialization parameters that are not available as commands are noted with an asterisk (*).

Parameter	Description
ACF2EXIT *	Specifies the name of the ACF2 MUSASS initialization exit and, optionally, requests //*JOBFROM card inclusion
AGENT*	Defines ESP Agents to ESP Workload Manager in the Agent definition data set
AGENTRCV	Defines and displays the ESP Workload Manager's receiver function for communication with ESP Agents
ALERTDEF	Defines Alert definitions.
APPLBUF *	Defines the maximum supported size of buffers the Application Manager and ESP Encore's EXHFILE use.
APPLFILE *	Specifies the name and other attributes of the Application data set
ARMELEM*	Registers ESP Workload Manager to the z/OS Automatic Restart Management function
AUDITLOG*	Sets audit log characteristics
AUTHSTR *	Specifies the source of the job-related data used as an authorization string with authorization tables
BKUPINDX	Sets a backup schedule the regular index data set
BKUPJNDX	Sets a backup schedule the regular jobindex data set
BKUPJSTS	Backs up the jobstats data set
BKUPUDEF	Sets a backup schedule the regular user definition data set
CCCHK	Identifies specific completion codes for specific job steps and specifies the action ESP Workload Manager should take
CCFLOPT	Identifies condition code failure options
CKPT *	Specifies the name of the checkpoint data set
CKPTRACE	Traces activity in the checkpoint data set
CMDPRFX	Specifies a substitution character for the started task name
COMMQ *	Specifies the name of the ESPCOM checkpoint data set used for Agent communications
CONSOLE	Sets or displays the owning or primary console
CPU	Defines a member of Job Entry System (JES) node to a CPU for resource scheduling
CRITPATH *	Requests critical path analysis
CUSTCTBL*	Allows you to load custom ASCII-EBCDIC and EBCDIC-ASCII character-conversion tables
DATEFORM	Specifies the date format

Parameter	Description
DELAYINT	Sets and alters the time between retries if the JCL library has contention
DFLTDSN	Specify Default Names for Data Sets Used by ESP Workload Manager
DSTRDLY	Sets data set trigger delay time
DSTREXCL	Specifies the names of one or more programs from which data set triggers are to be excluded
DSTRIG	Initializes the data set trigger cells
EICLASS	Controls and manipulates Event initiator class definitions
EMAIL *	Specifies an email address in a mailbox
ENCPARM ABENDER	Specifies the name of a program that abends on behalf of another program when using ESP Encore
ENCPARM AUTOREST	Specifies that ESP Encore will automatically recover missing data sets during job restart
ENCPARM CLEANUP	Prevents NOT CATLGD 2 and DUPLICATE NAME ON DASD errors when using ESP Encore
ENCPARM CONDCODE	Establishes customized condition codes for the ESP Encore step
ENCPARM DIAG	Causes the diagnostic sections of the ESP Encore report to be printed
ENCPARM FORCE	Causes the job to run even when ESP Encore predicts errors within the job
ENCPARM GDGADJ	Specifies how ESP Encore should deal with relative GDG generations when a job is being restarted
ENCPARM HONORCC	Specifies whether or not ESP Encore should honor past condition codes when restarting a job
ENCPARM IGNOREDS	Specifies which data sets ESP Encore ignores
ENCPARM MODIFY	Modifies ESP Encore internal processing functions
ENCPARM PREDICT	Specifies what type of errors ESP Encore predicts
ENCPARM PRINT	Specifies which sections of the ESP Encore report are to be printed
ENCPARM PURGE	Causes ESP Encore to purge job history records automatically from the EXH data set
ENCPARM TAPESCR	Specifies how ESP Encore scratches tape data sets
ENCPARM TRACE	Specifies the name or prefix of the module ESP Encore is to trace
ENCPARM VOLUME	Specifies ESP Encore special processing options for selected volumes
ENCPARM WARNING	Causes warning messages to be issued from ESP Encore Restart Analysis panel
ENCRYPT *	Enables encrypted communications between the ESP Agent or Workstation Server and the ESP Manager using a globally defined encryption key
ENQSELF	Sets an enqueue on every job
ERMSTEP	Identifies the JCL step to be added for ESP Encore to use
ESPGROUP *	Specifies the name of the ESP Workload Manager master and proxy group for CCCHK and ESP Encore
EVENTOPT	Allows or disallows the use of z/OS commands from within an Event. Also allows or disallows the use of WTO. Replaces EVENTWTO.
EVENTSET	Identifies the Event data set

Parameter	Description
EXHFILE *	Specifies the name of the EXH data set ESP Encore uses
EXIT	Activates a user exit and specifies processing options
EXPDSTRG	Specifies the conditions under which an explicit data set trigger notification Event is accepted
EXPEDITE	Defines an Expedite policy, which specifies criteria that must be met for a job in an Application to be expedited
FTP	Defines if ESP Workload Manager monitors SMF record types 118 or 119 for FTP data set triggering
GMTCHECK *	Checks GMT on the hardware clock and issues message if GMT and local time differ
HISTFILE	Specifies the name and other attributes of a job-history recording data set
HOME *	Specifies the IP address of a Workstation Server
INDEX *	Specifies the name and other characteristics of the index data set.\
INET	Sets the sysout class for a TCP/IP trace and is used to start the TCP/IP trace
INFOCOMM	Controls transaction servers used in sending Infoserv requests
ISCDEF	Identifies the current node name
ISCOPT *	Specifies whether Network Delivery Services (NDS), NJE or the LU 6.2 TP Server is used as the communications method for intersystem job tracking
ISCXMTR	Specifies the node name, sysout class, and external-writer name for nodes receiving tracking data
JESCOMCH	Specifies the character used to prefix Job Entry System (JES) commands
JESNAME	Specifies the name of the Job Entry System for whom tracking is kept
JESTYPE	Specifies the type of Job Entry System (JES) your installation uses
JOBINDEX *	Specifies the name and other characteristics of the jobindex data set. Required only when tracking is used.
JOBPROF	Defines a Job Profile Name
JOBSTATS	Specifies the name and other characteristics of the jobstats data set.
JTPEXCL	Specifies the name of a program to be excluded from job tracking
LOADAGDF	Loads the Agent definition data set
LOADJTDT	Loads the job-tracking definition table
LOADNET *	Loads the LOADNET data set that contains all the Network Delivery Services (NDS) initialization parameters
LOADNL	Loads the MAILLIST data set
LOADSCHF	Copies schedule information from a sequential work data set that a SADGEN batch job initializes to the schedule data set (SCHDFILE) that you can view from CSF
LOADUPDT	Loads the user profile definition entries
LOCAPPL *	Defines the local Application and defines the basic VTAM information necessary to initiate and receive session information
LOG *	Controls whether messages to a mailbox are written to the mail log

Parameter	Description
LOGTRACE	Consolidates certain trace data into a sysout data set
MAILBOX *	Specifies the name of a mailbox
MAILLOG	Sets the attributes of the mail log sysout
MANAGER *	Describes the ESP Workload Manager subsystem controlling the ESP Agents
MAPUSER *	Maps distributed system user IDs to authorized mainframe user IDs
MAXCXME	Sets the maximum amount of space, in bytes, to be used for XMEs in the checkpoint data set.
MAXDORM	Sets the maximum dormancy interval before an ESP Workload Manager subsystem forces a read of the QUEUE data set in a shared environment
MAXLRECL *	Obsolete.
MAXQXME	Sets the maximum amount of space, in bytes, to be used for XMEs in the QUEUE data set
MCS	Controls the ESP Workload Manager MCS (Multiple Console Support) extended console facility
MGRADDR	Indicates ESP Workload Manager will notify Agents of the TCP/IP address and optionally the port number of the Agent receiver to which the Agent should connect
MINDORM	Sets the minimum interval at which an ESP Workload Manager subsystem attempts to access the QUEUE data set
MINHOLD	Sets the minimum time interval that an ESP Workload Manager subsystem retains control of the QUEUE data set
MODE *	Specifies a Workstation Server's he operating mode
MSG	Adds or deletes routing or descriptor codes to a message or range of messages
MSGLIMIT	Allows an installation to limit the number of console messages a single Event generates
MSGPRFX	Specifies an alternate message ID prefix
MSGTYPE	Adds or deletes routing or descriptor codes to groups of messages based on their message type
NETAUTH	Defines authorization profiles for remote hosts that are allowed to connect to the local Network Delivery Services (NDS) network
NETDEST	Defines a Network Delivery Service (NDS) destination profile
NETNODE *	Names a local ESP Workload Manager master subsystem in the context of a NETWORK Delivery Services (NDS) network
NETQUEUE *	Specifies the name of the NETQUEUE data set
NETWORK	Sets up and starts Network Delivery Services (NDS) connections
NJETOL	Sets ESP Workload Manager tolerance of time differences across an NJE connection in a JES3 environment
NODE	Defines a node to ESP Workload Manager for resource scheduling
NTRCLASS	Specifies a job execution class. ESP Workload Manager does not execute jobs in this class.

Parameter	Description
NULLPSWD	Specifies a password to be compared to the PASSWORD parameter of a job card
PCACHE	Specifies if ESP Workload Manager uses ESP Procedure caching and the size of the cache
PLEXID	See “ESPGROUP: Specify ESP complex ID” on page 341 for information about this initialization parameter.
PORT*	Specifies the port that a Workstation Server listen to
PREALLOC	Identifies a pre-allocated data set
PROFILE *	Identifies the Event prefix and the EVENTSET that ESP Workload Manager uses to store Events
QTRACE *	Causes all allocations and freeing of space in the QUEUE data set to be traced
QUEUE *	Specifies the name and other characteristics of the QUEUE data set
RACROUTE	Specifies whether ESP Workload Manager issues a RACINIT prior to executing Events
REMOTE *	Specifies the access information of Workstation Server that the Consolidated Workstation Server (CWSS) connects to
RESDEF	Defines resource definitions
RESDFLT	Specifies a list of default resources that will be assigned to jobs
RESFILE *	Specifies the name and other attributes of the resource data set used to store resource information
ROUTING *	Routes ESP Workstation requests
RSVLOGIC	Identifies how ESP Workload Manager serializes its data sets
RUNMODE *	Allows you to restrict ESP Workload Manager batch usage
SADLINK	Identifies an external scheduled activity data set (SAD file) to ESP Workload Manager
SAFCLASS	Identifies the host security resource class to which ESP Workload Manager objects will be defined
SAFOPTS	Activates host security processing options
SCHDFILE	Identifies the schedule data set and work data set that the LOADSCHF command uses
SHADGOAL	Identifies a set of actions that a shadow manager will take when the ESP Workload Manager master terminates
SMFINT *	Specifies whether the SMF interface routine should be initialized
SMFREC *	Specifies an SMF record ID that ESP Workload Manager can use when journaling updates to specified data sets
SMTPPARM *	Specifies the Job Entry System (JES) class used to send messages and the name of your SMTP server
SORTNAME	Specifies the name of the installation SORT/MERGE utility. Use SORTNAME for the reporting feature
SORTUNIT	Specifies the unit name to be used for the dynamic allocation of SORT/ MERGE work files

Parameter	Description
STRACE *	Allocates space and activates trace.
SUBSYS (ESP Workload Manager) *	Specifies the subsystem name if different from the started task name.
SUBSYS (ESP Workstation Server) *	Specifies the subsystem that ESP Workstation will communicate with.
SVC *	Indicates which SVC number the ESP Workload Manager SVC uses.
SYMBOL	Specifies the symbolic variable and control statement introducing character, which defaults to percent (%).
SYSID *	Specifies the system identifier.
SYSMSGS	Enables or disables system message interception for the specified ID.
SYSPLEX *	Provides support for Sysplex Communication Services, that is, the Cross-System Coupling Facility (XCF).
TAPETRAK	Includes or excludes any tape device, real or virtual.
TCELL	Defines global storage cell pools for passing data between a job and ESP Workload Manager when the tracking feature is used.
TCPIP	Specifies which vendor's TCP/IP to access.
TIMECHK *	Issues a warning message if more than a specified time has elapsed since ESP Workload Manager was last up or if the system clock changes by more than the specified time
TIMEZONE *	Defines time zones.
TPAPPL	Manages interactions with remote partner applications
TPCKPT *	Allocates the TP Server checkpoint data set.
TPRETRY	Sets and displays the interval at which attempts will be made to contact an LU that is down
TRACE	Activates the trace facility and allows trace options to be set. For diagnostic purposes.
TRACEDEF	Identifies the data sets to be used to record information the TRACE facility collects for diagnostic purposes
TRACKDEF	Specifies the tracking information in a job-tracking definition table
TRACKING	Controls the ESP Workload Manager tracking facility
TRACKOPT	Sets tracking options
TRAKFILE *	Specifies the name and other characteristics of the tracking data set
TRDFLT	Specifies the default to be used when an Event is triggered manually
TSOSEND	Controls the sending of TSO messages to users
TSOUSER *	Includes a user ID in a mailbox
USERDEF *	Specifies the name and other characteristics of the user definition data set
USERMOD	Specifies which ESP Workload Manager user modifications are to be implemented
USERVFYX	Identifies the user verification exit
WILDCARD	Defines wildcards for the job-tracking definition table

Parameter	Description
WKSTART	Specifies the day on which the week should start. The default is Sunday.
WOBDEF	Identifies the program modules ESP Workload Manager must load to support distributed job types
WORKDAYS	Changes or adds to the default workdays specified, which are any weekdays that are not holidays. The default is Monday through Friday, inclusively.
WORKMGR	Allows ESP Workload Manager to support enclaves by connecting to IBM Workload Management
WSSCTL	Sets the sysout class for a Workstation Server messages trace and starts that trace
WSSSET *	Specifies the WSSPARM parameters required to run a Workstation Server. WSSSET is included in a WSSPARM file.
XCF SET SERVICE	Specifies a fixed interval an XCF Service is to stay suspended after a problem. When the suspended time has elapsed, the XCF Service will automatically resume.
XCF SET TRACE	Defines a sysout class for trace data output
XCF START SERVICE	Allows the registration and activation of XCF Services. To enable the XCF connection, the XCF START SERVICE *xxxxx* command must be coded in the initialization parameter data set.
XDAB	Traces VSAM I/O events
XFRB	Traces slot I/O events
XMITMDL	Instructs ESP Workload Manager which tracking models are to be transmitted and to which node

Entering Initialization Parameters

Note: This section also applies to the commands and statements contained in a data set.

Continuation

To continue a line of input, type either a hyphen or a plus sign as the last non-blank character on a line.

- The hyphen attaches the next line, including any leading blank position, to the previous line.
- The plus sign strips leading blanks from a continuing line.

Blanks preceding the hyphen or the plus sign are retained. Statements cannot extend beyond column 72. The continuation character can be placed in column 72 or before.

Note: A hyphen can also be used as a wildcard character. If the wildcard hyphen is the last character on the line, it will be interpreted as a continuation character and not as a wildcard. For the hyphen to be interpreted as a wildcard, follow it with a semicolon or something else on the line, such as a comment: (/* */).

Example

```
---+---1---+---2---+---3---+---4---+---5---+---6---+---7---

AGENT NT_Agent_DES ADDRESS(XXX.XX.XX.XXX) PORT(XXXX) -
   UNIX ASCII TCPIP PREFIXING ENCRYPT(X'XXXXXXXXXXXXXXXX')
```

Wildcards and Masking

Many statements, commands, or initialization parameters permit the use of the following wildcard characters (also called masking):

- An asterisk (*) matches a specific single character.
- A hyphen (-) matches zero or more characters. You must use the hyphen as the operand's last character. If the wildcard hyphen is the last character on the line, it will be interpreted as a continuation character and not as a wildcard. For the hyphen to be interpreted as a wildcard, follow the hyphen with a semicolon or something else on the line, such as a comment: (/* */).

Comments

Enclose comments between /* and */. You can write comments anywhere in an initialization data set.

Data sets

Use LIB- or PAN- prefixes to identify Librarian and Panvalet data sets.

Delimiters

Use single quotation marks when you want to denote character strings and literal data in expressions, assignment statements, and built-in functions. You must include single quotation marks around a string that contains blanks.

Indentation

You can use indentation to improve readability.

Using Symbolic Variables in the Initialization Parameters

You can use symbolic variables in your initialization parameter data set. They allow you to take different actions based on how ESP Workload Manager starts. The following symbolic variables are supported only in the initialization parameters:

- COLD
- QFORM
- QUIESCE
- RELOAD
- RESFORM
- SHADOW

Note: SHADOW requires the ESP Workload Manager High Availability Option (HAO).

Each symbolic variable, with the exception of SHADOW, corresponds to a parameter that can be specified when ESP Workload Manager starts. Each has a value of 1 (parameter specified) or 0 (parameter not specified). For example, if a COLD start is requested, the COLD symbolic variable is set to 1.

SHADOW has a value between 0 and 31. It contains the value assigned to the master and the shadow managers. By default, the value assigned to the master is 0. The SHADOW symbolic variable is created only if the SHADOW operand is used in the START parameters or JCL for the ESP Workload Manager subsystem.

To ensure the ESP Workload Manager master and shadow managers have different XCF member names when they read the same initialization parameter data set, use the SHADOW symbolic variable.

For information about these and other symbolic variables, see the *ESP Workload Manager Advanced User's Guide*.

Example — Defining topology and resources when RESFORM is requested

The following shows you how to define your topology and the resources when the resource data set has been formatted:

```
IF RESFORM THEN DO
    NODE NJE1 ADD ROUTEJCL('/*ROUTE XEQ NJE1')
    CPU CPU1 ADD ROUTEJCL('/*JOBPARM SYSAFF=CPU1')
    RESDEF TAPE ADD RENEWABLE LOCAL MAX(10) AVAIL(10)
ENDDO
```

Choosing the Subsystem Type

Some initialization parameters are valid only on a master system, a proxy system, or both. The valid system role is specified in the "Where defined" section of each initialization parameter description.

Subsystem type	Explanation
Master only	The initialization parameter must not be specified on a proxy.
Master	The initialization parameter can be specified on a proxy, but it has no effect.
Master and proxy	The initialization parameter must be specified on both a master and a proxy.

Depending on the initialization parameter, the "Where defined" section can also indicate whether the initialization parameter should be specified on the shadow subsystem or on the tracking subsystem, regardless of whether it is the master or proxy.

Choosing the Initialization Data Set

When ESP Workload Manager starts, much of its operating environment is pre-coded in one or more data sets. These data sets are read only at initialization time and contain initialization parameters that provide the definitions. These initialization parameters define the ESP Workload Manager environment.

You need to decide whether the initialization parameter should be read at every startup, or just at cold or warm startups. In addition, some initialization parameter types, such as TP Server initialization parameters, apply only to certain data sets, such as the TPPARM data set. These data sets are listed in the "Where defined" section of the initialization parameter description.

Startup initialization parameter data set (ESPPARM)

The ESPPARM data set is mandatory and read at every startup. Unless otherwise indicated, specify the initialization parameters in this data set. If another initialization data set must be used, it is specified in the "Where defined" section of the initialization parameter description.

Any initialization parameters can be specified in ESPPARM.

Cold and warm initialization data sets (ESPCOLD and ESPWARM)

The ESPCOLD data set is mandatory and read only during a cold start. A cold start formats the checkpoint file. The first time you start ESP Workload Manager, you do so with a cold start.

The ESPWARM data set is optional and read only during a warm start.

ESPWARM and ESPCOLD contain only initialization parameters that are also valid as commands. Initialization parameters only valid as initialization parameters appear in ESPPARM.

Note: Initialization parameters valid only as initialization parameters are marked with an asterisk in the "Initialization Parameter Summary" on page 230. If an initialization parameter is also valid as a command, the description for that parameter will indicate this.

Workstation Server initialization parameter data set (WSSPARM)

Workstation Server provides communication between ESP Workload Manager and ESP Workstation. Related initialization parameters must be defined in the Workstation Server parameter file.

For more information about the Workstation Server initialization parameter data set, see "Initialization parameters for Workstation Server" on page 96.

Consolidated Workstation Server initialization parameter data set (WSSREMOT)

A Consolidated Workstation Server is a Workstation Server with the ability to connect to other Workstation Servers. It requires an additional initialization parameter data set.

For more information about the Workstation Server initialization parameter data set, see "Initialization parameters for Consolidated Workstation Server" on page 99.

Agent Definition data set (AGENTDEF)

ESP Workload Manager uses an Agent definition data set to define the remote systems that form ESP Workload Manager's span of control. The AGENTDEF data set identifies the ESP Workload Manager master subsystem and any ESP Agents in the network. An AGENTDEF might be a sequential data set or a member of a partitioned data set.

The LOADAGDF initialization parameter loads the Agent definition data set during ESP Workload Manager initialization.

The initialization parameters that make up an AGENTDEF are

- "MANAGER: Specify the Subsystem Controlling the ESP Agents" on page 400
- "ENCRYPT: Specify encryption key for communications" on page 335
- "MAPUSER: Map distributed system user IDs to authorized mainframe user IDs" on page 403
- "AGENT: Define Agent" on page 244

Example of an AGENTDEF

The following is an example of an Agent definition data set:

```
MANAGER NAME(CM_CENTRAL) TCPIP
ENCRYPT KEY(X'xxxxxxxxxxxxxxxx')
MAPUSER JDOE TO(CYBJD01) AGENT(TORSUN1)
AGENT TORSUN1 ADDRESS(xxx.xx.xx.xx) PORT(9999) UNIX ASCII +
TCPIP PREFIXING
AGENT TORNT1 ADDRESS(xxx.xx.xx.xx) PORT(9998) NT ASCII +
TCPIP PREFIXING ENCRYPT
AGENT TORAS4001 ADDRESS(xxx.xx.xx.xx) PORT(9997) AS400 EBCDIC +
TCPIP PREFIXING ENCRYPT(X'xxxxxxxxxxxxxxxx')
```

Network Delivery Service data set (LOADNET)

The initialization parameters related to Network Delivery Service (NDS) are included in the LOADNET data set.

Detailed information about the LOADNET data set is included in the chapter "Setting Up Communications Between Two or More Masters" on page 145.

Notification list data set (MAILLIST)

The mailbox feature enables ESP Workload Manager to send messages to all TSO users and email addresses in a mailbox. To use the mailbox feature, you need to set the mailboxes in the MAILLIST data set.

To define the MAILLIST data set, see "PROFILE: Identify the Event prefix and data set" on page 447. For more information about other MAILLIST data set parameters and specifications, see "MAILLIST data set" on page 15.

TP Server initialization data set (TPPARM)

TP Server initialization parameters must be defined in the TPPARM data set. For more information, see "Configuring TP Server" on page 151 and "TPPARM data set" on page 151.

User profile definition table (UPDT)

The UPDT associates an Event prefix with an Event data set. It contains one or more PROFILE initialization parameters.

For more information, see “PROFILE: Identify the Event prefix and data set” on page 447.

Job-tracking definition table (JTDT)

A JTDT identifies the characteristics of the jobs you want tracked. For more information, see “Using Job-Tracking Definition Tables” on page 65.

ACF2EXIT: Identify ACF2 MUSASS

Purpose

The ACF2EXIT initialization parameter requests that an ACF2 MUSASS initialization exit be invoked.

Where defined

Master

Syntax

```
ACF2EXIT exitname
```

Operand	Description
exitname	The name of the initialization exit load module or IEFBR14

Example

The following example requests that a null ACF2 initialization exit be invoked:

```
ACF2EXIT IEFBR14
```

AGENT: Define Agent

Purpose

The AGENT initialization parameter defines an ESP Agent to ESP Workload Manager. An ESP Agent is defined in an Agent definition data set, typically called AGENTDEF. The AGENT initialization parameter is only one of five Agent definition initialization parameters that make up AGENTDEF. For more information on creating an Agent definition data set, see “Agent Definition data set (AGENTDEF)” on page 240.

Where defined

Master

AGENTDEF data set

Syntax

```
AGENT name {ADDRESS(ip_address)}
           {PORT(port)}
           [platform]
           {ASCII|EBCDIC}
           {TCPIP|APPC}
           [PREFIXING|NOPREFIXING]
           [ENCRYPT|ENCRYPT(key)|KEYNAME(keyname)|NOENCRYPT]
```

Operand	Description
name	The Agent name used in ESP Workload Manager Applications. The name should be as descriptive as possible. For example, it could include the location or type of product workload. The name can be up to 16 characters long. It must start with an alphabetic character. The remaining characters can be any combination of alphanumeric characters, including the underscore character. The name cannot contain spaces.
ADDRESS(*ip_address*)	The TCP/IP address of the Agent. The address can be in dotted decimal notation or host name format.
PORT(*port*)	The TCP/IP port number on the Agent platform. This port definition must match the port specified on the Agent.
platform	The Agent platform. Specify one of the following platforms: • NT • OPENVMS • UNIX
ASCII	The remote system character code. Specify ASCII for all Agents except • AS400 • IBM UNIX System Services

Operand	Description
EBCDIC	The remote system character code. Specify EBCDIC for the following systems: • AS400 • IBM UNIX System Services
TCPIP	The network protocol used to connect to the Agent.
APPC	The network protocol used to connect to the Agent.
PREFIXING	Inform the Agent of how many TCP/IP packets to expect and the message prefixing to avoid potential data loss
<u>NOPREFIXING</u>	Do not inform the Agent of how many TCP/IP packets to expect.
ENCRYPT	Enable encryption for the Agent using the key specified in the ENCRYPT initialization parameter.
ENCRYPT(*key*)	Enable encryption for the Agent with the specified key. The key can be up to 16 hexadecimal numbers in the form X'*nnnnnnnnnnnnnnnn*', where *n* is a number.
KEYNAME(*keyname*)	Enable encryption for the Agent using the key referenced by *keyname*. The key name is defined with the CRYPTKEY command.
NOENCRYPT	Disable encryption for the Agent.

Example

The following example shows an AGENT initialization parameter in an AGENTDEF:

```
AGENT TORSUN ADDRESS(123.45.67.89) PORT(9999) UNIX ASCII +
TCPIP PREFIXING ENCRYPT(X'5436734268479015')
```

Related information

The following initialization parameters are used for ESP Workload Manager distributed workload:

- "WOBDEF: Define workload objects" on page 540
- "LOADAGDF: Load Agent definitions" on page 385
- "AGENTRCV: Define Agent receiver" on page 246
- "MGRADDR: Identify manager to Agent" on page 410
- "COMMQ: Define ESPCOM checkpoint data set" on page 270
- "INET: Start or set TCP/IP tracing" on page 364

AGENTRCV: Define Agent receiver

Purpose

Use the AGENTRCV initialization parameter to define the ESP Workload Manager receiver for communication with ESP Agents.

Note: You can also use AGENTRCV as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Where defined

Master

Syntax

```
AGENTRCV
          [DEFINE] [NAME(name)]
          [PORT(portnumber)[HOME(host)]]
          [SYMDEST(symdest)]
          [LU(luname) TPNAME(tpname) LOGMODE(logmode)]
          [POLLINGINTERVAL(interval1[,interval2])]
          [DUMPMSGS|NODUMPMSGS]
          [ENCRYPTEDONLY|NOENCRYPTEDONLY]
```

Operand	Description
DEFINE	Defines an Agent receiver to the ESP Workload Manager subsystem and starts it.
NAME(*name*)	Indicates an Agent receiver name. The receiver name can be up to 8 characters long; the first character must be alphabetic or a national character.
PORT(*portnumber*)	Indicates the TCP/IP port number of a TCP/IP Agent receiver. This operand is mutually exclusive with the SYMDEST, LU, TPNAME, LOGMODE, and POLLINGINTERVAL operands.
HOME(*host*)	Indicates a DNS host name or TCP/IP address. The Agent receiver will bind to the TCP/IP address corresponding to the HOME(*host*) operand. ESP Agents can then only connect to the Agent receiver through the specified TCP/IP address. Use this operand when there is a need to limit ESP Agent connections to a single TCP/IP address. This operand can only be specified when the PORT(*portnumber*) operand is specified.

Operand	Description
SYMDEST(*symdest*)	The symbolic destination name in the APPC side information data set that represents the Agent LU (Logical Unit) and TP (Transaction Program) name, and the LOGMODE name of the session on which the APPC conversation is carried. This operand is mutually exclusive with the PORT operand. The *symdest* operand is where Agent messages are received in an APPC/APPN network.
LU(*luname*)	The LU (Logical Unit) name of an APPC Agent receiver. When you specify LU with the DEFINE option and omit the SYMDEST operand, you must also specify the LOGMODE and TPNAME operands. This operand is mutually exclusive with the PORT operand.
TPNAME(*tpname*)	The TP (Transaction Program) name of the APPC Agent. If you specify TPNAME with the DEFINE option and omit the SYMDEST operand, you must also specify the LOGMODE and LU operands. This operand is mutually exclusive with the PORT operand.
LOGMODE(*logmode*)	The APPC logmode name designating the network properties for the session to be allocated for the APPC conversation with the Agent. If you specify LOGMODE with the DEFINE option and omit the SYMDEST operand, you must also specify the LU and TPNAME operands. This operand is mutually exclusive with the PORT operand.
POLLINGINTERVAL (*interval1*,*interval2*)	The *interval1* operand is the polling interval in seconds for an APPC Agent receiver. If omitted, the default is 5 seconds. The *interval2* operand is the polling interval in seconds for an APPC Agent receiver if an error occurs. This value should be higher than *interval1*. If omitted, *interval1* is used. This operand is mutually exclusive with the PORT operand.
DUMPMSGS	All messages from the specified Agent receiver are written to the console in a dump format. You may specify this operand only with the DEFINE and SET options.

Operand	Description
NODUMPMSGS	Messages are not written to the console. NODUMPMSGS is the default.
ENCRYPTEDONLY	Only encrypted data is accepted.
NOENCRYPTEDONLY	Both encrypted and unencrypted data can be received. NOENCRYPTEDONLY is the default.

Examples

Defining and starting a TCP/IP Agent receiver

The following example shows how to define and start an Agent receiver named CYBTCPIP that listens for incoming messages on TCP/IP port 6664:

```
AGENTRCV DEFINE NAME(CYBTCPIP) PORT(6664)
```

Starting an Agent receiver

The following example shows how to start a TCP/IP Agent receiver named CYBTCPIP that listens for incoming connections on TCP/IP port 6665, but accepts only encrypted data:

```
AGENTRCV DEFINE NAME(CYBTCPIP) PORT(6665) ENCRYPTEDONLY
```

Define and start an Agent receiver, and restrict connections to a single TCP/IP address

In the following example, an Agent receiver named CYBTCPIP starts on TCP/IP port 6666, and all ESP Agent connections will use TCP/IP address 10.1.15.1 only:

```
AGENTRCV DEFINE NAME(CYBTCPIP) PORT(6666) HOME(10.1.15.1)
```

Defining and starting an APPC Agent receiver

The following example shows how to define and start an APPC Agent receiver named CYBAPPCI that polls an Agent on LU AIX001, identified by TP name CYBAGT1. CYBAPPCI runs under session mode name CYBLOGMD and polls every 10 seconds under normal conditions and every 60 seconds if an error occurs:

```
AGENTRCV DEFINE NAME(CYBAPPCI) LU(AIX001) TPNAME(CYBAGT1) +
   LOGMODE(CYBLOGMD) POLLINGINTERVAL(10,60)
```

ALERTDEF: Define an Alert

Note: You can also issue the ALERTDEF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The ALERTDEF initialization parameter allows you to define Alert definitions. An Alert definition refers to an Event that ESP Workload Manager automatically triggers when it processes the Alert as part of a NOTIFY statement.

Where defined

Master only

Syntax

```
ALERTDEF ADD
         [ID(alertid)]
         [EVENT(eventid)]
```

Operand	Description
ADD	Add an Alert definition
xxxx	The Alert definition's four-character logical identifier. You specify this Alert identifier in an ESP Procedure's NOTIFY statement.
eventid	Event name. When a job reaches a particular processing stage, as specified in the NOTIFY statement, the Alert triggers this Event.

Usage notes

The Alert feature enables you to trigger an Event at different monitor points, such as job submission, job start, when a job becomes overdue at any processing stage or when a job terminates too early. These monitor points correspond to the different processing points the NOTIFY statement supports. You can trigger an Event for all jobs, or specific jobs, in an ESP Application.

To use the Alert feature

1. Use the NOTIFY statement in an ESP Application to identify when the Alert triggers the Event.
2. Define the Alert using the ALERTDEF initialization parameter or command.
3. Define the Event that the Alert triggers.

Example

The following ALERTDEF initialization parameter defines an Alert and associates that Alert with an Event name:

```
ALERTDEF ADD ID(INFO) EVENT(CYBER.CREATE_RECORD)
```

In the above example, an Alert with a logical identifier of INFO is defined and associated with an Event called CYBER.CREATE_RECORD, which is automatically triggered when the following NOTIFY statement appears in an ESP Application:

```
NOTIFY ABEND ALERT(INFO)
```

APPLBUF: Define Application Manager buffers

Purpose

The APPLBUF initialization parameter defines the maximum supported size of buffers that the Application Manager and the ESP Encore EXHFILE use.

Where defined

Master and proxy

Syntax

```
APPLBUF number
```

Operand	Description
number	Maximum supported buffer size. It must be a number from 16376 to 8388600. The default varies with the APPLFILE slot size: • 4k slots=1048568 • 8k slots=2093048 • 12k slots=3137528 • 16k slots=4182008

Usage notes

Setting this initialization parameter to its maximum buffer size enables the ESP Workload Manager to generate an Application that contains approximately 10,000 jobs. If APPLBUF is too low, ESP Workload Manager fails when it tries to generate an Application requiring a larger buffer size.

Note: Applications bigger than 4 megs also require a slot size of at least 8 k. See the SLOTSIZE operand in “APPLFILE: Define Application data set” on page 252.

If you are using ESP Encore, APPLBUF must not be less than 1044480 bytes.

We recommend a minimum Application buffer size of 1 megabyte minus 8 bytes (1048568).

Example

```
APPLBUF 1048568
```

APPLFILE: Define Application data set

Purpose

The APPLFILE initialization parameter defines the name of an Application data set.

Where defined

Master and proxy

Syntax

```
APPLFILE MAIN DSN(dsname) [NOSHR|SHR]
                          [NOREADONLY|READONLY]
                          [SLOTSIZE(nn)]
                          [CACHE(nn|1)]
```

Operand	Description
MAIN	The logical identifier of the data set. This operand cannot be changed.
DSN(*dsname*)	The name of the data set
NOSHR	The data set is not shared. NOSHR is the default.
SHR	The data set is shared.
NOREADONLY	Allows ESP Workload Manager to update the data set. NOREADONLY is the default.
READONLY	Prevents ESP Workload Manager from updating the APPLFILE. The READONLY operand should be used only on proxy systems. It must not be used on the master system. READONLY allows full sharing of ESP Workload Manager control data sets in ESP Workload Manager master and proxy relationships where the use of usual VSAM share options might pose operational difficulties and data set integrity must be preserved.
SLOTSIZE(*nn*)	The number of bytes from 4096 to 16384. Use this for very large Applications. The value must be a multiple of 4096. The default is 4096. If you change this operand, ensure the buffer size specification in the APPLBUF initialization parameter is still valid.
CACHE(*nn*)	Indicates the number of megabytes of memory for caching. The default value is 1. Cache is not enabled by default. The Application tracking record uses this cache. **Note:** The CACHE operand is ignored on proxy systems.

Example

The following example defines ESP.APPLFILE as the Application data set. It is shared and read only.

```
APPLFILE MAIN DSN(ESP.APPLFILE) SHR READONLY
```

ARMELEM: Set up Automatic Restart Management

Purpose

The ARMELEM initialization parameter registers the ESP Workload Manager subsystem to the z/OS Automatic Restart Management (ARM) function.

Applicability

HAO only

Where defined

Master and proxy

Note: The system must be connected to an ARM couple data set.

Syntax

```
ARMELEM {NAME(element)} {TYPE(ESP|elemtype)} +
        {RESTART(ALL_FAILURES|ELEMENT_FAILURE)}
```

Operand	Description
NAME(*element*)	The name of the element that ARM will register **Note:** The element name must be unique in the sysplex. The name must be 1 to 16 characters long. The first character must be alphabetic or a national character, such as dollar sign ($), number sign ($) or commercial at (@). The remaining characters must be alphanumeric, national or underscore (_). If omitted, the element name defaults to the ESP SYSID (value specified by the SYSID statement in the initialization parameter data set).
TYPE(*elemtype*)	The type of element that ARM is to register. The name of the type must be one to eight characters long. The first character must be alphabetic or a national character, such as dollar sign ($), number sign ($) or commercial at (@). The remaining characters must be alphanumeric or national. The default is ESP.
RESTART(ALL_ FAILURES)	The element should be restarted when the ESP Workload Manager that registered with ARM fails, and when the system that the registered ESP Workload Manager is running on fails. If the system fails, the registered ESP Workload Manager will restart on another system in the sysplex. The default is ALL_FAILURES for shadow-disabled ESP Workload Manager masters, and ELEMENT_FAILURE for shadow-enabled ESP Workload Manager masters and ESP Workload Manager proxies.

Operand	Description
RESTART(ELEMENT_ FAILURE)	The element should be restarted only when the ESP Workload Manager that registered with ARM fails, but not when the system the registered ESP Workload Manager is running on fails. The default is ALL_FAILURES for shadow-disabled ESP Workload Manager masters, and ELEMENT_FAILURE for shadow-enabled ESP Workload Manager masters and ESP Workload Manager proxies.

Usage notes

ARM will restart a registered ESP Workload Manager that fails if an ARM policy (either the IBM default or a user-defined policy) is active on the system where the failure occurs and on the system where the restart is to occur.

For shadow-disabled ESP Workload Manager, a master restart on another system is highly desirable if it is registered with ARM and the system it is running on fails. Therefore, the default value for the RESTART keyword of the ARMELEM statement is ALL_FAILURES.

For shadow-enabled ESP Workload Manager masters and proxies, one proxy will run on each system in the sysplex. Therefore, if a proxy is registered with ARM, the default value for the RESTART keyword of the ARMELEM initialization parameter is ELEMENT_FAILURE, so that ARM restarts are limited to the system the proxy runs on.

ARM and shadow manager

If your installation uses shadow manager, you might consider ARM unnecessary for the master. However, ARMELEM co-exists well with shadow manager. Note that the default RESTART keyword value for shadow-enabled ESP Workload Manager masters has deliberately been set to ELEMENT_FAILURE.

You can use ARM or shadow manager for ESP Workload Manager master restarts. Note that shadow manager does not require an ARM couple data set and ARM policy. Also, shadow manager can automatically take over as the ESP Workload Manager master if the current ESP Workload Manager master terminates. For more information, see “SHADGOAL: Define shadow manager options” on page 482.

When an ESP Workload Manager shadow-enabled master is used with shadow managers, the default ARMELEM initialization parameter element name (ESP SYSID) might not be unique and, therefore, you must specify the NAME keyword. The following example shows using the ARMELEM initialization parameter in such a case:

```
ARMELEM NAME(ESPM%SHADOW)
```

The SHADOW value assigned to the ESP Workload Manager master is unique.

AUDITLOG: Audit Log

Purpose

The AUDITLOG initialization parameter sets the print characteristics of the ESP Workload Manager audit log, CPEENQLG audit log, and the ENCORELG audit log. The ESP Workload Manager audit log records ESP Workload Manager activities, such as those related to Events, Applications, and commands. For details on the CPEENQLG and ENCORELG audit logs, see "ESP Encore audit logs CPEENQLG and ENCORELG" in the *ESP Workload Manager System Programmer's Guide*.

There is no default for AUDITLOG.

Where defined

Master and proxy

Syntax

```
AUDITLOG class [FORM(xxx)] [EXTWTR(a1234567)] [AUDCOUNT(0|nn)]
```

Operand	Description
class	The sysout class for the audit log
FORM(*form_name*)	The form to be used for printing the audit log
EXTWTR(*a1234567*)	The name of an external writer. The name must start with an alphabetic character and is a maximum of eight characters long.
AUDCOUNT(*nn*)	The maximum number of lines before ESP Workload Manager automatically spins off the audit log. The default is 0. The value must be set to 5 or above for ESP Workload Manager to automatically spin off the audit log.

Usage notes

If you do not use the AUDCOUNT operand or you want to spin off the audit log before the number of lines specified is reached, use the SPINLOG command to spin the audit log data set to the sysout class. This initialization parameter overrides the AUDITLOG DD statement in the ESP Workload Manager started task procedures, and causes the log data set to be dynamically allocated.

Example

The following example specifies an external writer called MYWRT and creates an audit log that will be spun very 10,000 lines to class X:

```
AUDITLOG X FORM(STD) EXTWTR(MYWRT) AUDCOUNT(10000)
```

AUTHSTR: Define authorization string

Purpose

The AUTHSTR initialization parameter specifies one of five job-related fields used to uniquely identify a job in a case of multiple occurrences with the same job name.

Where defined

Master

Syntax

```
AUTHSTR [ACCOUNT1|ACCOUNT2|ACCOUNT3|ACCOUNT4|RACUSER]
```

Operand	Description
ACCOUNT1	The first account number field is used as an authorization string.
ACCOUNT2	The second account number field is used as an authorization string.
ACCOUNT3	The third account number field is used as an authorization string.
ACCOUNT4	The fourth account number field is used as an authorization string.
RACUSER	The user ID under which the job executes is used as an authorization string. RACUSER is the default.

Usage notes

Use an authorization string to control External and Manual job postings. The AUTHSTR operand on the JOB statement determines the authorization string.

Example

The following example shows that the user ID under which the job runs uniquely identifies a job in a case of multiple occurrences with the same job name:

```
AUTHSTR RACUSER
```

BKUPINDX: Back up INDEX data set

Note: You can also issue the BKUPINDX initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

Use the BKUPINDX initialization parameter to set the backup schedule for the index data set (INDEX).

Where defined

Master

ESPCOLD data set

Syntax

```
BKUPINDX schedule
```

Operand	Description
schedule	The regular backup schedule

Usage notes

The BKUPINDX initialization parameter sets the time schedule for the regular index data set backup. The BACKUPDATASET operand in the INDEX initialization parameter identifies the data set that should be backed up. For more information, see “INDEX: Identify index data set” on page 362.

Any time you need to change the schedule, you can issue the BKUPINDX command.

Example

Schedule backup at 6am daily

In the following example, the index data set is scheduled for backup at 6am every day:

```
BKUPINDX DAILY AT 6AM
```

Related information

For information on defining the index data set, see “Index Data Set (INDEX)” on page 24.

BKUPJNDX: Back up JOBINDEX data set

Note: You can also issue the BKUPJNDX initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the BKUPJNDX initialization parameter to set the backup schedule for the job index data set (JOBINDEX).

Where defined

Master

ESPCOLD data set

Syntax

```
BKUPJNDX schedule
```

Operand	Description
schedule	The regular backup schedule

Usage notes

The BKUPJNDX initialization parameter sets the time schedule for the job index data set backup. The BACKUPDATASET operand in the JOBINDEX initialization parameter identifies the data set that should be backed up. For more information, see "JOBINDEX: Jobindex data set" on page 377.

Any time you need to change the schedule change, you can issue the BKUPJNDX command.

Example

Schedule backup at 6am daily

In the following example, the job index data set is scheduled for backup at 6am every day:

```
BKUPJNDX DAILY AT 6AM
```

Related information

For information on defining the job index data set, see the "Jobindex Data Set (JOBINDEX)" on page 27.

BKUPJSTS: Back up the jobstats data set

Purpose

Use the BKUPJSTS initialization parameter to set the time schedule for the jobstats data set backup.

Note: You can also use BKUPJSTS as a command to take an immediate backup, display the time of the next scheduled backup, or to set a time schedule. For details about syntax, see the BKUPJSTS command in the *ESP Workload Manager Reference Guide*.

Where defined

Master

Syntax

```
BKUPJSTS schedule
```

Operand	Description
schedule	Sets the time schedule for the jobstats data set backup.

Usage notes

The backup is made to the data set identified by the BACKUPDATASETNAME operand in the JOBSTATS initialization parameter. For information, see "JOBSTATS: Jobstats data set" on page 382.

Example

The following initialization parameter backs up the jobstats data set at 6AM every day:

```
BKUPJSTS DAILY AT 6AM
```

BKUPUDEF: Back up user definition data set

Note: You can also issue the BKUPUDEF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the BKUPUDEF initialization parameter to set a backup schedule for the user definition data set (USERDEF).

Where defined

Master

ESPCOLD data set

Syntax

```
BKUPUDEF schedule
```

Operand	Description
schedule	The regular backup schedule

Usage notes

The BKUPUDEF initialization parameter sets the time schedule for the regular user definition data set backup. The BACKUPDATSET operand in the USERDEF initialization parameter identifies the data set that should be backed up. For more information, see “USERDEF: Identify user definition data set” on page 533.

Any time you need to change the schedule, you can issue the BKUPUDEF command. If the system is down when a backup is to occur, the backup starts as soon as ESP Workload Manager restarts.

Example

Schedule backup at 6am daily

In the following example, the user definition data set is scheduled for backup at 6am every day:

```
BKUPUDEF DAILY AT 6AM
```

Related information

For information on defining the user definition data set, see the “User Definition Data Set (USERDEF)” on page 38.

CCCHK: Specify Job Processing Based on Condition Codes

Note: You can also issue the CCCHK initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The CCCHK initialization parameter allows you to specify the action taken if a job produces a specified condition code.

CCCHK

- Sets the job to a given state (OK or FAILED).
- Specifies whether further processing is done.

You could use CCCHK to immediately stop a failing job regardless of the COND operands in the JCL.

Where defined

Master and proxy

Syntax

```
CCCHK [JOB(jobname)]
      [STEP(stepname)]
      [PROC(procstepname)]
      [PROGRAM(progname)]
      RC(num)|RC(num1:num2)|RC(Sccc)|RC(Unnnn)
      [OK|FAIL]
      [CONTINUE|STOP|ASK]
```

Operand	Description
JOB(*jobname*)	A string that matches a job name. You can use wildcard characters. If you omit this operand, CCCHK matches any job name.
STEP(*stepname*)	A stepname in a job. You can use wildcard characters. If you omit this operand, CCCHK matches any step name.
PROC(*procstepname*)	A procstepname in a cataloged procedure or instream procedure. You can use wildcard characters. If you omit this operand, CCCHK matches any procstepname.
PROGRAM(*progname*)	The name of the program that the step executes, as found in the PGM operand on the job's EXEC statement. You can use wildcard characters. If you omit this operand, CCCHK matches any program name.
RC(*num*)	A condition code of *num*, where *num* is an integer between 0 and 4095 inclusively. This is not a user abend code.

Operand	Description
RC(*num1*:*num2*)	A condition code between *num1* and *num2* inclusively, where *num1* and *num2* are integers between 0 and 4095. The value *num2* must be greater than or equal to the value of *num1*. These are not user abend codes.
RC(S*ccc*)	A system abend code, such as S0C1 or SB37. *ccc* must be three hexadecimal digits.
RC(U*nnnn*)	A user abend code, such as U0001 or U0462. *nnnn* must be four decimal digits and cannot exceed 4095.
OK	If the condition code is matched, consider the job as not failed. OK is the default.
FAIL	If the condition code is matched, consider the job as failed. CONTINUE/STOP/ASK determines whether the job continues.
CONTINUE	If the condition code is matched, the job continues with the next step, whether the job is considered as failed or not failed. CONTINUE is the default.
STOP	If the condition code is matched, stop the job, flush it, and issue message 453I. This message indicates the job failed with a JCL error, although there actually is no JCL error.
ASK	If the condition code is matched, display message 696A and prompt the operator with the following options for handling the condition code: • C — FAIL and CONTINUE • S — FAIL and STOP • O — OK and CONTINUE The operands in the operator prompt are documented earlier in this table.

Usage notes

We recommend that you specify the CCCHK initialization parameter with the FAIL operand, preventing jobs that do not contain a CCCHK at the Application or job level from completing with a non-zero return code.

If CCCHK is not specified in the initialization parameter, at the Application level or at the job level, ESP Workload Manager sets the status of jobs completing with a non-zero return code to COMPLETED and continues to execute the Application.

Using CCCHK with CCFAIL

You can control job condition code checking with the CCCHK initialization parameter or statement or the CCFAIL statement. CCCHK replaces CCFAIL.

If you use CCCHK and CCFAIL together, both checks are performed. If either check indicates a failure, the job is considered a failure for reason CCFAIL. The CCFAIL step (named ESPCCFCK) is not added to any job that uses CCCHK.

Using multiple CCCHK statements

For a single job, you can specify up to 254 CCCHK statements. When several CCCHKs are active for a single job, their effect is cumulative (unlike the CCFAIL statement). For example, one CCCHK statement might indicate a particular action, and other statements might indicate some exceptional cases.

Understanding how CCCHK works

CCCHK initialization parameters and statements can detect specified condition codes from specific jobs, steps, procsteps or programs. For example, a CCCHK statement can detect a specified condition code whenever the IEBGENER program runs.

The statement search order and statement scope

When each step in a job or procedure ends and produces a condition code, CCCHK initialization parameters and statements are searched for a match in the following order:

1. CCCHK statements in a job definition (in the order they occur)

   A matched statement affects the job that the statement is in.
2. CCCHK statements in an Application definition (in the order they occur)

   A matched statement affects all jobs in the Application the statement is in.
3. CCCHK initialization parameters (in the order they occur)

   A matched statement affects all jobs and Applications in the ESP Workload Manager subsystem (including proxies in a master and proxy system).

When the first match is found, all other statements are ignored and the matching CCCHK is processed for the step.

Understanding the OK and FAIL operands

If the first matching CCCHK from any step indicates FAIL, a job is considered failed, even if a subsequent matching CCCHK for that step specifies OK.

If the first matching CCCHK from any step indicates OK, the job is considered OK, even if a subsequent matching CCCHK for that step specifies FAIL. However, subsequent steps could still cause the job to be considered failed.

Understanding the CONTINUE, STOP, and ASK operands

If a step abends and it has matching CCCHK initialization parameters or statements then

- If the first matching CCCHK specifies CONTINUE, the job proceeds as usual. Subsequent steps are still subject to matching CCCHK statements and JCL COND parameters.
- If the first matching CCCHK specifies STOP, the job stops and is flushed, even if any steps include COND=EVEN or COND=ONLY.

Setting ESPGROUP for CCCHK in a master/proxy environment

In a master and proxy environment, CCCHK uses the ESPGROUP initialization parameter (also known as PLEXID). If you issue CCCHK in a master and proxy environment, specify the same ESPGROUP name for all ESP Workload Manager subsystems that are part of a tracking group.

Ignoring an abend

You can use CCCHK to ignore an abend. A history is maintained of each job step that had a completion code (not a condition code), and of matching CCCHK initialization parameters or statements with OK specified. At the end of the job, ESP Workload Manager treats the job as successful if

- No other failure conditions exist.
- All matching CCCHK initialization parameters or statements specified OK.
- There were no system completion codes in the format X22.

This processing has no effect on how z/OS and JES handle a job that had a step abend. Step-level monitors still see an abend as it appeared before. Use the COND operands EVEN or ONLY to run subsequent steps. LJ, LTJ, LAP, and history reporting show the abend. Alert processing considers the job successful.

Memory required by the ASK operand

The ASK operand uses a temporary X'180'-byte work area for the operator prompt and reply. In the unlikely event that the memory allocation fails, ESP Workload Manager issues message 697E and processes the job as if you had issued CCCHK using the FAIL and STOP operands.

Examples

Failure based on condition code

In this example, any step in any job is considered to have failed if it produces a condition code greater than zero:

```
CCCHK RC(1:4095) FAIL
```

Failure and stopping processing based on condition code

In this example, any step in any job is considered to have failed if it produces a condition code greater than zero, but only a condition code greater than 8 stops processing:

```
CCCHK RC(1:8) FAIL CONTINUE
CCCHK RC(9:4095) FAIL STOP
```

CCFLOPT: Set condition code failure options

Note: You can also issue the CCFLOPT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The CCFLOPT initialization parameter identifies the action ESP Workload Manager takes when it detects a condition code failure during job execution.

Where defined

Master and proxy

Syntax

```
CCFLOPT 66660044|20200000
```

Operand	Description
66660044	When a failure is detected, a JCL error is forced. DJC treats the job as a failure and successor jobs do not run. If you use nets, specify 66660044.
20200000	ESP Application Manager is notified of the failure, but does not signal the failures to the initiator. If you do not use DJC or JES3 job nets, 20200000 is recommended.

CKPT: Specify checkpoint data set

Purpose

The CKPT initialization parameter specifies the name and, optionally, where the data set is stored.

Where defined

Master and proxy

Syntax

```
CKPT dsname [VOLUME(volser)|UNIT(unitname)]
```

Operand	Description
dsname	The name of the checkpoint data set
VOLUME(*volser*)	The volume on which the data set resides
UNIT(*unitname*)	The unit type

Usage notes

Each job in the CKPT data set requires approximately 64 bytes.

Each master or proxy system needs a unique CKPT data set.

The size of the CKPT data set is limited to one extent. When ESP Workload Manager needs to perform I/O to this file, it uses a single-channel program. A single-channel program can only read or update one extent at a time. Size is determined automatically when the data set is formatted.

Example

The following example specifies ESP.CKPT as the checkpoint data set:

```
CKPT ESP.CKPT
```

CKPTRACE: Trace checkpoint data set

Note: You can also issue the CKPTRACE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The CKPTRACE initialization parameter traces the allocation and freeing of storage in the checkpoint data set CKPT. This is used for diagnostic purposes only.

Where defined

Master and proxy

Syntax

```
CKPTRACE
```

Usage notes

Use the CKPTRACE initialization parameter only when requested by CA Technical Support.

CMDPRFX: Set command prefix

Note: You can also issue the CMDPRFX initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The CMDPRFX initialization parameter allows you to substitute the string 'F *stcname*,' (where *stcname* is the started task name for ESP Workload Manager) with a single character.

Where defined

Master and proxy

Syntax

```
CMDPRFX char 'F started_task.name,'
```

Operand	Description
char	The prefix character. Use a single character from the special character set.
started_task.name	The name of the started task. The started task name for ESP Workload Manager is usually ESP.

Example

In the following example, the number sign (#) is substituted for `'F ESP,'`:

```
CMDPRFX # 'F ESP,'
```

To issue a command at the system console, type

```
# command text
```

COMMQ: Define ESPCOM checkpoint data set

Purpose

The COMMQ initialization parameter specifies the name of the checkpoint data set that ESPCOM uses. ESPCOM provides ESP Workload Manager with generic outbound communications that ensure message persistence. When ESP Workload Manager is requested to send a message to an Agent, ESPCOM takes the parameters for the message from the Agent definition data set and sends the message. ESPCOM notifies ESP Workload Manager of the result.

If your installation uses ESP Agents, then you must define an ESPCOM checkpoint data set.

Where defined

Master

Syntax

COMMQ *data_set_name*

Operand	Description
data_set_name	The name of the VSAM linear data set used to store information about Agent communications

Example

The following example uses the full initialization parameter syntax:

```
COMMQ CYBER.ESPCOM.DATASET
```

CONSOLE: Set primary console

Note: You can also issue the CONSOLE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The CONSOLE initialization parameter sets the owning or primary console. All general message traffic is directed to the owning console.

Where defined

Master and proxy

Syntax

```
CONSOLE [*|ucmid|0|name]
```

Operand	Description
*	The console issuing the command is the primary console
ucmid	The two-digit console UCMID
0	No console. Messages are routed by routing code on to the active master console.
name	The one-to-eight-character name of an active console in the ESP Workload Manager complex

Usage notes

A primary console must be an active one, with the capability for both input and output. If no operands are specified, the current primary console information appears. The console must be active when the request is issued.

Examples

Specific console UCMID

To set console 11 as the primary console, type

```
CONSOLE 11
```

Issuing console

To set the issuing console as the primary console, type

```
CONSOLE *
```

Specified console name

To set the console named BOSS as the primary console, type

```
CONSOLE BOSS
```

CPU: Central Processing Unit

Note: You can also use the CPU initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the CPU initialization parameter with the NODE initialization parameter to define your network's resource topology. The CPU command defines each CPU within a node.

Note: You do not need a RESFILE if you do not use resources.

Where defined

Master only

Syntax

```
CPU cpuname [ADD|DEL|LIST|SET]NODE(nodename)
            [ROUTEJCL('/*JOBPARM SYSAFF=membername') -
            |AGENT(agentname)]
            [ORDER(nn)]
            [CURRENT]
            [ACTIVE|INACTIVE]
            [SYSNAME(sysname)]
```

Operand	Description
cpuname	The name that identifies this CPU, which must be a unique name in a node. When used with the NODE operand, you might have the name correspond to, for example, an existing JES member name, but this is not mandatory. You can use the wildcard characters asterisk and hyphen for masking the name with all parameters except ADD. If you intend to use Agent workload balancing or route to an Agent based on resource availability, use the CPU command to refer to an Agent.
ADD	Adds this definition to the existing definitions
DEL	Deletes one or more existing definitions
LIST	Displays a list of one or more existing definitions. LIST is the default.
SET	Modifies attributes of existing definitions
NODE(*nodename*)	The node to which the CPU belongs

Operand	Description
ROUTEJCL	Inserts a JOBPARM statement into the job JCL to route the job to the appropriate z/OS system
AGENT(*agentname*)	Specifies an ESP Agent that the CPU is associated with. The Agent name is the unique name specified for an Agent in the AGENTDEF data set. Use this operand when you want ESP Workload Manager to use Agent workload balancing or when you want to route to an Agent that has sufficient resources.
ORDER(*nn*)	Specifies a CPU search sequence. When ESP Workload Manager tries to schedule a job, it scans for resource availability by CPU, searching the CPUs in a sequence defined by the ORDER value *nn*. If two CPUs have the necessary resources to execute a job, ESP Workload Manager schedules the job to the one with the lowest ORDER value.
CURRENT	Overrides the ROUTE JCL operand. If you specify CURRENT, *cpuname* identifies the CPU executing the ESP Workload Manager subsystem that submits jobs. For the CPU command on which you code CURRENT, you don't need to code ROUTE JCL.
ACTIVE	Declares the CPU as active. ACTIVE is the default.
INACTIVE	Declares the CPU as inactive. ESP Workload Manager does not attempt to schedule a job to that CPU while it is inactive.
SYSNAME(*sysname*)	The system name for a CPU. Use *sysname* for Workload Manager resource balancing. **Note:** This operand is only relevant if ESP Workload Manager Service Governor is installed.

Usage notes

You must define NODE prior to defining CPU. For more information, see "NODE: Define JES node" on page 436.

You must code a CPU initialization parameter for every JES member you want to define resources on, explicitly or implicitly. If you use real resources, you should specify the CURRENT operand in only one CPU. Use conditional logic if you expect the master to move.

If you use ESP Agents and if you want to use Agent workload balancing for jobs running on CPUs associated with ESP Agents, you must code a CPU initialization parameter for each of those CPUs. The CPU initialization parameter must include an AGENT operand.

We recommend that you use the CPU and NODE names that JES knows (except for CPUs associated with ESP Agents).

If you use resources, see “RESFILE: Specify the resource data set” on page 468.

Examples

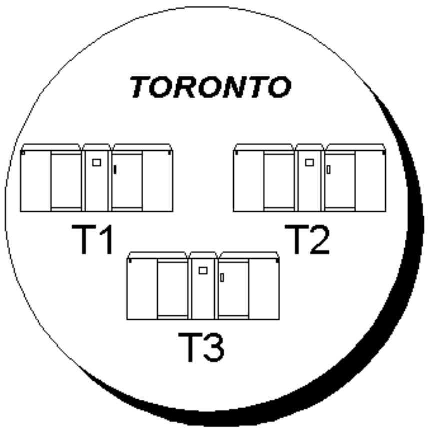
Defining a single node and a single CPU

This example defines a single node called TORONTO and a single CPU called MVSA:

```
NODE TORONTO ADD
CPU MVSA ADD NODE(TORONTO) CURRENT
```

Recommended implementation of multiple CPUs

The following example defines the TORONTO node with multiple CPUs:



ESPPARM looks like this:

```
RESFILE CYBER.RESFILE SIZE(50000)
IF RESFORM THEN DO
  NODE TORONTO ADD
   CPU T1 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T1') -
ORDER(1) CURRENT
   CPU T2 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T2') -
ORDER(2)
   CPU T3 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T3') -
ORDER(3)
  RESDEF FRED ADD GLOBAL RENEWABLE MAX(1) AVAIL(1)
ENDDO
```

When you use the CPU initialization parameter in the IF THEN DO and ENDDO statements and ESP Workload Manager starts with the RESFORM option that reformats the RESFILE, the NODE and CPU definitions are re-created from this information.

Defining NODE and CPU for IBM WLM workload balancing

The following example defines two CPUs that belong to the Toronto node:

```
NODE TORONTO ADD SYSPLEX
CPU T1 ADD NODE(TORONTO) CURRENT
CPU T2 ADD NODE(TORONTO)
```

All the CPUs that have the same NODE operand in the CPU initialization parameter are linked. If you have Service Governor installed, ESP Workload Manager can balance the workload among members belonging to one node if the SYSPLEX operand is coded in the NODE initialization parameter. For information on how to use WLM workload balancing, see the *ESP Workload Manager User's Guide.*

Defining NODE and CPU for Agent workload balancing

The following example defines

- Three Windows NT CPUs associated with Agents called AGwin1, AGwin2, and AGwin3.
- Four UNIX CPUs associated with Agents called AGunix1, AGunix2, AGunix3, and AGunix4.

```
NODE WIN ADD
NODE UNIX ADD
CPU WIN1 ADD NODE(WIN)AGENT(AGwin1)
CPU WIN2 ADD NODE(WIN)AGENT(AGwin2)
CPU WIN3 ADD NODE(WIN)AGENT(AGwin3)
CPU UNIX1 ADD NODE(UNIX)AGENT(AGunix1)
CPU UNIX2 ADD NODE(UNIX)AGENT(AGunix2)
CPU UNIX3 ADD NODE(UNIX)AGENT(AGunix3)
CPU UNIX4 ADD NODE(UNIX)AGENT(AGunix4)
```

All the CPUs that have the same NODE operand in the CPU initialization parameter are linked. If you have Service Governor installed, ESP Workload Manager can balance the workload among Agents belonging to one node. For information on how to use Agent workload balancing, see the *ESP Workload Manager User's Guide*.

CRITPATH: Request critical path analysis

Purpose

The CRITPATH initialization parameter enables or disables the analysis of an Application's critical path. ESP Workload Manager determines the longest path to critical jobs in terms of execution time, based on historical averages, and identifies this path as the critical path to the job.

Where defined

Master

Syntax

```
CRITPATH [ENABLE|DISABLE|ON]
```

Operand	Description
ENABLE	Critical path analysis is enabled. Specify CRITPATH ON in the Applications where you want critical path analysis performed. ENABLE is the default.
DISABLE	Critical path analysis is disabled. You cannot use critical path analysis on this system when it is disabled.
ON	Critical path analysis is turned on for all Applications. You can turn it off in an Application by specifying CRITPATH OFF.

Examples

No critical path analysis

If you do not require critical path analysis for any Applications, in your initialization parameters, type

```
CRITPATH DISABLE
```

or allow it to default to CRITPATH DISABLE.

Note: If you disable critical path analysis in your initialization parameters, you cannot turn it on in an Application.

Critical path analysis for a single Application

This example shows you how to request critical path analysis for a single Application.

In your initialization parameters, type

```
CRITPATH ENABLE
```

In the Application, type

```
APPL PAYROLL
  CRITPATH ON
    .
    .
    .
```

Critical path analysis for all Applications

If you require critical path analysis for all Applications, specify the following in your initialization parameters:

```
CRITPATH ON
```

No other coding is required in any Application.

Critical path analysis for all Applications but one

If you require critical path analysis for all Applications except one, specify the following in your initialization parameters:

```
CRITPATH ON
```

Specify the following in the Application you do not want analyzed:

```
APPL PAYROLL
  CRITPATH OFF
    .
    .
    .
```

CUSTCTBL: Load custom ASCII-EBCDIC conversion tables

Purpose

The CUSTCTBL initialization parameter allows you to load custom ASCII-EBCDIC and EBCDIC-ASCII character-conversion tables. These tables can be configured for ESP Workload Manager, ESP z/OS Agent, and ESP Workstation Server. ESP Workstation sends ASCII data to the Workstation Server and receives ASCII data from the Workstation Server. The Workstation Server expects to receive ASCII data and converts it to EBCDIC; the Workstation Server converts its responses from EBCDIC to ASCII before sending them back to ESP Workstation.

Syntax

`CUSTCTBL` *`dsname`*`[(`*`member`*`)]`

Operand	Description

dsname	The name of a sequential data set or a member of a partitioned data set containing the ASCII-EBCDIC, EBCDIC-ASCII character-conversion tables saved in binary format. No quotation marks required.
	
	A valid input data set contains three 256-character records:
	1. Title: `“*TCP/IP translate tables”`
	2. ASCII-EBCDIC table
	3. EBCDIC-ASCII table
	The CUSTCTBL statement runs when ESP Workload Manager starts. When the CUSTCTBL statement successfully runs, informational message 2731 is issued:
	`Custom character conversion table loaded:`
	`ESP.CUSTOM.TRTBLBIN`
	
	If a problem arises, error message 2732 is issued:
	`CUSTCTBL error: <reason>`
	
	If the <reason> string is not self-explanatory (for example, “Access failure”), the system log might contain a separate message that contains different information, for example
	`IKJ56228I Data set ESP.CUSTOM.TRTBLBIN not in`
	`catalog or catalog can not be accessed.`

Example

Default ASCII to EBCDIC table

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
       -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
00 -   00 01 02 03 37 2D 2E 2F 16 05 25 0B 0C 0D 0E 0F  -  00
10 -   10 11 12 13 3C 3D 32 26 18 19 3F 27 1C 1D 1E 1F  -  10
20 -   40 5A 7F 7B 5B 6C 50 7D 4D 5D 5C 4E 6B 60 4B 61  -  20
30 -   F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 7A 5E 4C 7E 6E 6F  -  30
40 -   7C C1 C2 C3 C4 C5 C6 C7 C8 C9 D1 D2 D3 D4 D5 D6  -  40
50 -   D7 D8 D9 E2 E3 E4 E5 E6 E7 E8 E9 BA E0 BB 5F 6D  -  50
60 -   79 81 82 83 84 85 86 87 88 89 91 92 93 94 95 96  -  60
70 -   97 98 99 A2 A3 A4 A5 A6 A7 A8 A9 C0 4F D0 A1 07  -  70
80 -   D8 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F  -  80
90 -   3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F 3F  -  90
A0 -   3F AA 3F B1 9F B2 6A B5 3F B4 9A 8A 3F 3F AF BC  -  A0
B0 -   90 8F EA FA BE A0 B6 B3 9D DA 9B 8B B7 B8 B9 AB  -  B0
C0 -   64 65 62 66 63 67 9E 68 74 71 72 73 78 75 76 77  -  C0
D0 -   AC 69 ED EE EB EF EC BF 80 FD FE FB FC 3F 3F 59  -  D0
E0 -   44 45 42 46 43 47 9C 48 54 51 52 53 58 55 56 57  -  E0
F0 -   8C 49 CD CE CB CF CC E1 70 DD DE DB DC 8D 8E DF  -  F0
       -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

Row X'60', column X'0A' shows that ASCII X'6A' (C'j') translates into EBCDIC X'91' (C'j').

Default EBCDIC to ASCII table

```
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
       -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
00 -   00 01 02 03 1A 09 1A 7F 1A 1A 1A 0B 0C 0D 0E 0F  -  00
10 -   10 11 12 13 1A 1A 08 1A 18 19 1A 1A 1C 1D 1E 1F  -  10
20 -   1A 1A 1A 1A 1A 0A 17 1B 1A 1A 1A 1A 1A 05 06 07  -  20
30 -   1A 1A 16 1A 1A 1A 1A 04 1A 1A 1A 1A 14 15 1A 1A  -  30
40 -   20 1A E2 E4 E0 E1 E3 E5 E7 F1 5B 2E 3C 28 2B 7C  -  40
50 -   26 E9 EA EB E8 ED EE EF EC DF 21 24 2A 29 3B 5E  -  50
60 -   2D 2F C2 C4 C0 C1 C3 C5 C7 D1 A6 2C 25 5F 3E 3F  -  60
70 -   F8 C9 CA CB C8 CD CE CF CC 60 3A 23 40 27 3D 22  -  70
80 -   D8 61 62 63 64 65 66 67 68 69 AB BB F0 FD FE B1  -  80
90 -   B0 6A 6B 6C 6D 6E 6F 70 71 72 AA BA E6 B8 C6 A4  -  90
A0 -   B5 7E 73 74 75 76 77 78 79 7A A1 BF D0 5B 1A AE  -  A0
B0 -   5E A3 A5 B7 A9 A7 B6 BC BD BE 5B 5D AF 5D B4 D7  -  B0
C0 -   7B 41 42 43 44 45 46 47 48 49 1A F4 F6 F2 F3 F5  -  C0
D0 -   7D 4A 4B 4C 4D 4E 4F 50 51 52 B9 FB FC F9 FA FF  -  D0
E0 -   5C F7 53 54 55 56 57 58 59 5A B2 D4 D6 D2 D3 D5  -  E0
F0 -   30 31 32 33 34 35 36 37 38 39 B3 DB DC D9 DA 1A  -  F0
       -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
       00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
```

Row X'F0', column X'01' shows that EBCDIC X'F1' (C'1') translates into ASCII X'31' (C'1').

Usage notes

To use the custom ASCII-EBCDIC character-conversion tables, you require ESP Workload Manager v.5.5, CPE v.5.1, and PTF SU02486. The SSCPSAME member CYBESS76 outlines a customization procedure and provides sample character tables.

IBM's CONVXLAT utility converts the tables from editable text to binary. CONVXLAT offers a ready-made tool for defining character tables and is consistent with the IP customization technique IBM supports. Familiarity with CONVXLAT is not required, but for more information see IBM's *z/OS Communications Server: IP Configuration Reference* manual. An ISPF user can obtain information by entering `TSO HELP CONVXLAT` on the command line; if QuickRef is available, enter `QW CONVXLAT`.

To set up custom tables

1. Edit the tables in sample member CYBESS76 and save the result.
2. Use the CONVXLAT utility to convert the tables from editable text to binary. CONVXLAT writes the output to a sequential data set, which CONVXLAT creates if the data set does not already exist.

   Tip: Optionally, you can copy the binary table data set to a partitioned data set that has RECFM=F and LRECL=256. The CUSTCTBL statement accepts a PDS member name as input.

TSO command

```
CONVXLAT 'ESP.SSCPSAME(CYBESS76)' 'ESP.CUSTOM.TRTBLBIN'
```

JCL

```
//CONVXLAT JOB  CYB1,'JANE SMITH',
//         CLASS=J,MSGCLASS=X,NOTIFY=&SYSUID
//*---------------------------------------------
//STEP10   EXEC PGM=CONVXLAT,
//         PARM=('''ESP.SSCPSAME(CYBESS76)''',
//         ' ''ESP.CUSTOM.TRTBLBIN''')
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY,BLKSIZE=80
//
```

3. Add the following statement to the initialization parameter data set of the address space you want to configure:

   ```
   custctbl esp.custom.trtblbin
   ```

DD Statement	Sample member	Address space
WSSPARM	SSCPSAME(CYBESS32)	ESP Workstation Server
ESPPARM	SSCPSAME(CYBESS03)	ESP Workload Manager or ESP z/OS Agent

   This change takes effect the next time the address space starts.

Note: Configuring ESP Workload Manager, ESP z/OS Agent or ESP Workstation Server to use a custom character table affects all commands and data that address space receives or sends.

Tailoring the character conversion on the mainframe side will not update data already saved in data sets and will not change the character tables that client programs on Windows personal computers or on other platforms use.

DATEFORM: Set date format

Note: You can also issue the DATEFORM initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The DATEFORM initialization parameter sets the date format for schedule criteria in the format xx/xx/xx.

Where defined

Master and proxy

Syntax

```
DATEFORM [YMD|MDY|DMY]
```

Operand	Description
YMD	The date format is YY/MM/DD. YMD is the default.
MDY	The date format is MM/DD/YY.
DMY	The date format is DD/MM/YY.

Example

In this example, the date format is set to MDY:

```
DATEFORM MDY
```

In the above example, a schedule statement 04/12/06 is interpreted as April 12, 2006.

DELAYINT: Delay interval between retries

Note: You can also issue the DELAYINT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The DELAYINT initialization parameter allows you to set and alter the time between retries if a required JCL or ESP Workload Manager Procedure library is not available because of data set contention or because the data set was migrated and needs to be recalled.

Where defined

Master and proxy

Syntax

```
DELAYINT [nn|5]
```

Operand	Description
nn	The delay interval in minutes. The value range is from 1 to 15 minutes. The default is 5 minutes.

Usage notes

This setting is in effect until the next restart, unless you use the DELAYINT command to override the setting.

The libraries or procedures to be retried are

- INVOKE (for example, Procedure library)
- SUBMIT (for example, JCL library)
- JCLLIB
- TEMPLIB
- SYMLIB
- COPYJCL
- DOCLIB

Example

This example causes ESP Workload Manager to retrigger an Event in two minutes should a required data set, such as a Procedure library, not be available:

```
DELAYINT 2
```

DFLTDSN: Specify Parameters for Data Sets Used by ESP Workload Manager

Purpose

DFLTDSN sets the parameters for the data sets that ESP Workload Manager uses.

Where defined

Master and proxy

Syntax

```
DFLTDSN prefix [CACHE(nn|10)] [SHR]
```

Operand	Description
prefix	The data set name prefix. This operand applies to the APPLFILE, CKPT, COMMQ, EVENTSET, INDEX, JOBINDEX, JOBSTATS, QUEUE, RESFILE, and TRAKFILE data sets.
CACHE(*nn*)	The number of megabytes of memory for caching (*nn*). The default is 10. This operand applies to the APPLFILE and TRAKFILE data sets.
SHR	Allows the data set to be shared. This operand applies to the APPLFILE, EVENTSET, INDEX, JOBINDEX, QUEUE, and TRAKFILE data sets.

Usage notes

Use the DFLTDSN initialization parameter to specify default parameters for data sets that have not been specified explicitly. You can override the default parameters of a particular data set by coding the initialization parameter for that data set.

The DFLTDSN initialization parameter specifies the COMMQ data set, the JOBSTATS data set, and the RESFILE data set only on the master, not on the proxies.

For a description of the data sets used by ESP Workload Manager, see "Data Set Summary" on page 14.

Example 1

Adding DFLTDSN PA.ESPWLM to the initialization parameter data set specifies default data set names as follows:

Default Data Set Name	Notes
`PA.ESPWLM.APPLFILE`	
`PA.ESPWLM.`*`sysid`*`.CKPT`	The *sysid* qualifier is set to the value specified in the SYSID initialization parameter.
`PA.ESPWLM.COMMQ`	
`PA.ESPWLM.EVENT1`	
`PA.ESPWLM.INDEX`	
`PA.ESPWLM.JOBINDEX`	
`PA.ESPWLM.JOBSTATS`	
`PA.ESPWLM.QUEUE`	
`PA.ESPWLM.RESFILE`	
`PA.ESPWLM.TRAKFILE`	

Example 2

Adding DFLTDSN PA.ESPWLM CACHE(20) SHR to the initialization parameter data set specifies default data set names as follows:

Default Data Set Name	Notes
`PA.ESPWLM.APPLFILE`	• 20 MB of memory are used for caching. • The data set is shared.
`PA.ESPWLM.`*`sysid.`*`CKPT`	The *sysid* qualifier is set to the value specified in the SYSID initialization parameter.
`PA.ESPWLM.COMMQ`	
`PA.ESPWLM.EVENT1`	The data set is shared.
`PA.ESPWLM.INDEX`	The data set is shared.
`PA.ESPWLM.JOBSTATS`	
`PA.ESPWLM.JOBINDEX`	The data set is shared.
`PA.ESPWLM.QUEUE`	The data set is shared. RESERVE/DEQ is performed.
`PA.ESPWLM.RESFILE`	
`PA.ESPWLM.TRAKFILE`	• 20 MB of memory are used for caching. • The data set is shared. RESERVE/DEQ is performed.

DSTRDLY: Delay data set triggering

Note: You can also issue the DSTRDLY initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The DSTRDLY initialization parameter delays data set-triggered Events when a trigger occurs.

Where defined

Master and proxy

Syntax

```
DSTRDLY [nnnn|0]
```

Operand	Description
nnnn	The number of seconds to delay. The default is 0.

Usage notes

Data set-triggered Events might trigger too soon after a data set closes. When data set triggering is detected, the DSTRDLY initialization parameter globally delays data set-triggered Events.

Example

The following example specifies a delay of 30 seconds when a data set trigger is detected:

```
DSTRDLY 30
```

DSTREXCL: Exclude programs from data set triggering

Note: You can also issue the DSTREXCL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The DSTREXCL initialization parameter specifies the names of programs that are ineligible for data set triggering. A data set updated by one of these ineligible programs will not cause a data set trigger.

Where defined

Master and proxy

Syntax

```
DSTREXCL {name}
         {(name[,name]...)}
```

Operand	Description
name	The program name. It contains up to eight alphanumeric characters. You can use the hyphen as a wildcard character. For more information, see "Representing characters with wildcards" on page 64.

Usage notes

Use this initialization parameter to avoid generating some data set triggers. Data set triggers occur when a data set is modified. You might want to prevent updates caused by certain programs from being considered for data set triggering. The following programs are examples of candidates for exclusion:

- Automatic compress utilities
- Data set archival and retrieval programs
- Dump and restore utilities.

If several programs begin with the same prefix, you can specify only the prefix followed by a hyphen. You can specify multiple initialization parameters in the initialization data set.

Note: If you need to delete a data set trigger exclusion entry, use the DSTREXCL command. For more information, see the *ESP Workload Manager Reference Guide.*

Example

The following example prevents data set triggering for all data sets updated by any program beginning with the characters DMPRST and by the program COMPRESS.

```
DSTREXCL (DMPRST-,COMPRESS)
```

DSTRIG: Initialize data set trigger cells

Note: You can also issue the DSTRIG initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The DSTRIG initialization parameter initializes the data set trigger cells.

Where defined

Master and proxy

Syntax

```
DSTRIG [CELLS(cellcount)]
       [EXP(maxexp)]
```

Operand	Description
CELLS(*cellcount*)	The initial data set trigger cell count. The default is 10. The maximum is 1000.
EXP(*maxexp*)	The maximum expansion limit, beyond which no more cells are GETMAINed. This value is added to the *cellcount* value. The default is 50. The maximum is 1000.

Usage notes

A data set trigger cell passes information from the triggering job to ESP Workload Manager upon closure of a data set when the data set matches the description of a data set on a DSTRIG statement in an Event definition or DSNAME in an Application definition. The cell is queued to ESP Workload Manager, which copies the information out. It then replaces the cell on the free queue.

Normally, very few elements are needed because they are used only when a data set trigger is to occur. These elements are freed as soon as ESP Workload Manager has examined them. However, if ESP Workload Manager is down for an extended period, these elements will remain on a request queue until ESP Workload Manager is brought up again. When no more cells are available in the free queue, more will be obtained, one at a time, by use of a GETMAIN. When the cell expansion limit is reached, no more cells are obtained, and data set triggering information is lost. When ESP Workload Manager restarts, it examines all data set trigger elements, and FREEMAINs any cells that were explicitly GETMAINed.

Example

This example sets the initial data set trigger count to 10 and the expansion limit to 12, beyond which no more cells are GETMAINed.

```
DSTRIG CELLS(10) EXP(12)
```

EICLASS: Control Event initiator class

Note: You can also issue the EICLASS initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The EICLASS initialization parameter allows you to prioritize workload submission when multiple Events are triggered at a given time. You use the EICLASS initialization parameter to create multiple Event initiator classes. The EICLASS initialization parameter specifies the class and the number of initiators assigned to that class. If you do not specify any values for EICLASS, ESP Workload Manager uses one class 0 initiator.

Where defined

Master and proxy

Syntax

```
EICLASS [DISPLAY|SET|DELETE] CLASS(nnn) MPL(nn)
```

Operand	Description
DISPLAY	Displays the specified Event initiator class settings. DISPLAY is the default. This operand has an alias of LIST.
SET	Alters the specified class settings
DELETE	Deletes the specified class. All TDRs queued to the deleted class are moved to class 0.
CLASS(*nnn*)	The target class. *nnn* must be a number from 0 to 254. If DELETE is specified, *nnn* cannot be 0. If SET or DELETE is specified, CLASS is required.
MPL(*nn*)	The maximum number of initiators to be defined for the specified class. *nn* is a number from 0 to 16. If SET is specified, MPL must be specified. If DISPLAY is specified, MPL is ignored.

Examples

Displaying current class

To display current classes, type

```
EICLASS DISPLAY
```

Defining new class

To define a new or alter an existing class MPL, type

```
EICLASS SET CLASS(4) MPL(5)
```

The above example sets class 4 MPL level to 5.

Deleting existing class

To delete an existing class, type

```
EICLASS DELETE CLASS(6)
```

The above example deletes class 6. All requests for class 6 are moved to class 0.

EMAIL: Include email address in mailbox

Purpose

The EMAIL initialization parameter specifies an email address in a mailbox.

Where defined

Master and proxy

MAILLIST data set

Syntax

```
EMAIL user@yourisp.com
```

Operand	Description
user@yourisp.com	The email address that receives the messages. An email address can be the address of an individual or a group.

Usage notes

If you have a complex distribution list, we recommend that you use email group addresses for each distribution point. Each group can include a number of different individual email addresses and each individual recipient can belong to any number of groups.

For information on the MAILLIST data set, see “MAILLIST data set” on page 15.

Examples

The EMAIL initialization parameter is included in the MAILLIST data set as follows:

```
SMTPPARM CLASS(A) JOBNAME(SMTP)
MAILBOX  PAYROLL MAXLINES(0)
TSOUSER CYBPAY1 SYSID(SYSC)
EMAIL paymaster@company.com
EMAIL payservice@payservice.com
MAILBOX CYBACCOUNTING MAXLINES(300)
TSOUSER (CYBACC1 CYBACC2) SYSID(SYSA)
```

ENCPARM ABENDER: Specify a Program That Abends on Behalf of Another Program

Purpose

Some job steps run a program that abends if a prior step ends with a specified completion code. You can issue ENCPARM ABENDER to have ESP Encore ignore the abend program step and restart on the step that caused the abend program step to run.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM ABENDER program_name
```

Operand	Description
program_name	The name of the program that abends on behalf of another program

Usage notes

You can specify only one program name in an ENCPARM ABENDER command. However, you can specify several ENCPARM ABENDER commands in the initialization parameters. ENCPARM ABENDER is usually coded in the ESP Workload Manager initialization parameter data set but can be overridden in an ESP Procedure using the ENCPARM ABENDER command.

Examples

In the following example, STEP3 executes when STEP2 ends with a completion code of 16. Program ABEND806 in STEP3 abends on behalf of STEP2 and causes the rest of the job to be flushed.

```
//...
//STEP1    EXEC PGM=P1
//STEP2    EXEC PGM=P2
//STEP3    EXEC PGM=ABEND806,COND(16,NE,STEP2)
//STEP4    EXEC PGM=P3
//...
```

You can use the following ENCPARM ABENDER command to avoid restarting the job in STEP3, the abend step, and the restart in STEP2:

```
ABENDER ABEND806
```

ENCPARM AUTOREST: Recover Data Sets Automatically For a Job Restart

Purpose

The ENCPARM AUTOREST initialization parameter controls automatic data set recovery during job restart.

During a job restart, ESP Encore can automatically recover

- Missing data sets to avoid DATA SET NOT FOUND errors.
- Data sets that are updated by an abending step.

  If a step is updating a data set and that step abends, ESP Encore considers the data set's content to be invalid. ESP Encore then looks for an earlier step that creates the data set. If it finds such a step, it deletes the data set so that the earlier step can recreate it.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM AUTOREST [YES|ALWAYS|NO
                 [TEMPONLY([YES|NO])
                 [NEWPASS(YES|NO)]]]
```

Operand	Description
YES	ESP Encore recovers data sets if it can to do so; otherwise, the restart continues only if specific conditions are met (see below). YES is the default. To recover data sets, ESP Encore must run any steps preceding the FROMSTEP that recreate or update the data sets. If these steps have changed from the previous run and the changes are disallowed (see “Limits to JCL changes before restarting a job” in the *ESP Encore User’s Guide*), ESP Encore does not rerun earlier steps and does not recover the missing data sets. If ESP Encore cannot recover missing data sets, continuation of the restart is determined as follows: • If there is no DATA SET NOT FOUND error or the FORCE YES initialization parameter is specified, the ESP Encore step completes normally and the restart continues to run. • If there is a DATA SET NOT FOUND error and the FORCE YES initialization parameter is not specified, the ESP Encore step fails and the restart job is flushed.
ALWAYS	ESP Encore must always be able to recover data sets; otherwise, the restart is abandoned. To recover data sets, ESP Encore must run any steps preceding the FROMSTEP that recreate or update the missing data sets. If these steps have changed from the previous run and the changes are disallowed (see “Limits to JCL changes before restarting a job” in the *ESP Encore User’s Guide*), ESP Encore flushes the restart job and returns a condition code of 40.
NO	ESP Encore does not perform automatic data set recovery. However, it predicts DATA SET NOT FOUND errors unless the FORCE YES initialization parameter has been specified.
TEMPONLY(YES)	ESP Encore performs automatic recovery of temporary data sets only. This operand is ignored when you specify ENCPARM AUTOREST NO.
TEMPONLY(NO)	ESP Encore performs automatic recovery of both permanent and temporary data sets. TEMPONLY(NO) is the default.
NEWPASS(YES)	ESP Encore performs automatic recovery of data sets created with DISP=(NEW,PASS). This operand is ignored when you specify ENCPARM AUTOREST NO. When you specify NEWPASS(YES), you must also specify TEMPONLY(YES).
NEWPASS(NO)	ESP Encore does not perform automatic recovery of data sets created with DISP=(NEW,PASS).

Usage notes

The following chart shows the relationship between the ENCPARM AUTOREST operands and the types of data sets that are restored:

AUTOREST	TEMPONLY	NEWPASS	Restores Permanent data sets	Restores && data sets	Restores NEW, PASS data sets
NO	YES, NO	YES, NO	No	No	No
YES, ALWAYS	NO	YES, NO	Yes	Yes	Yes
YES, ALWAYS	YES	NO	No	Yes	No
YES, ALWAYS	YES	YES	No	Yes	Yes

Example

ENCPARM AUTOREST is issued in the following segment of a procedure:

```
JOB AUTOR027
  ENCPARM AUTOREST YES TEMPONLY(YES) NEWPASS(YES)
  RUN DAILY
  ...
ENDJOB
```

According to the preceding ENCPARM AUTOREST settings, ESP Encore will perform automatic recovery only of temporary data sets, including those created with DISP=(NEW,PASS).

The following job AUTOR027 runs with the preceding ENCPARM AUTOREST:

```
//AUTOR027 JOB (CYB1000),...
//*
//STEP010  EXEC PGM=IEBGENER,COND=(0,NE)
//
...
//SYSUT2   DD DSN=&&TEMP1A,DISP=(NEW,KEEP),
//         SPACE=(10,5),UNIT=SYSDA,
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=FB)
//*
//STEP020  EXEC PGM=PATEST,COND=(0,NE)
//
...
//SYSUT2   DD DSN=PA.TEST.REST1A,DISP=(NEW,CATLG,KEEP),
//         SPACE=(10,5),UNIT=TEST,
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=FB)
//*
//STEP030  EXEC PGM=IEBGENER,COND=(0,NE)
//
...
//SYSUT2   DD DSN=&&TEMP1B,DISP=(NEW,PASS),
//         SPACE=(10,5),UNIT=SYSDA,
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=FB)
//*
//STEP040  EXEC PGM=IEBGENER,COND=(0,NE)
//
...
//SYSUT2   DD DSN=PA.ENCQA.REST1A,DISP=(NEW,PASS,DELETE),
//         ...
//*
//STEP050  EXEC PGM=IEBGENER,COND=(0,NE)
//
...
//SYSUT2   DD DSN=PA.ENCQA.REST1B,DISP=(NEW,PASS,DELETE),
//         ...
//*
//STEP060  EXEC PGM=CONDCODE,PARM=99999,COND=(0,NE)
//*
//STEP070  EXEC PGM=IEBGENER,COND=(0,NE)
...
```

Job AUTOR027 abends in step STEP060. ESP Encore restarts at STEP060 but, because of the settings of AUTOREST, ESP Encore first restores temporary data sets as follows:

- Data sets &&TEMP1A and &&TEMP1B are restored by rerunning STEP010 and STEP030. This is done because TEMPONLY is set to YES.
- Data sets PA.ENCQA.REST1A and PA.ENCQA.REST1B are restored by rerunning STEP030 and STEP040. This is done because NEWPASS is set to YES.

ESP Encore does not restore the permanent data set PA.TEST.REST1A in STEP020.

ENCPARM CLEANUP: Delete Data Sets That a Restart Job Will Allocate

Purpose

The ENCPARM CLEANUP initialization parameter deletes the data sets that this job will allocate, preventing NOT CATLGD 2 and DUPLICATE NAME ON DASD errors.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM CLEANUP [INITIAL(YES|NO)]
                [RESTART(YES|NO)]
                [DYNALLOC(YES|NO])]
```

Operand	Description
INITIAL(YES)	ESP Encore performs data set cleanup during the job's initial run.
INITIAL(NO)	ESP Encore does not perform data set cleanup during the job's initial run. INITIAL(NO) is the default.
RESTART(YES)	ESP Encore performs data set cleanup when the job restarts. RESTART(YES) is the default.
RESTART(NO)	ESP Encore does not perform data set cleanup when the job restarts.
DYNALLOC(YES)	When a job restarts, ESP Encore deletes any data set that the job's previous run dynamically allocated and created. DYNALLOC(YES) is the default.
DYNALLOC(NO)	When a job restarts, ESP Encore does not delete data sets that the job's previous run dynamically allocated and created.

Usage notes

During a restart from the ESP Encore ISPF interface or ESP Workstation interface, ESP Encore looks at the previous run's JCL.

During a batch restart, ESP Encore looks at the JCL to be restarted, not the previous run's JCL.

Example

The following command indicates that ESP Encore deletes all data sets (including dynamically allocated data sets) that were created during the initial run or before a restart run. During a restart run, ESP Encore only deletes dynamically allocated data sets created by steps that are being restarted.

```
ENCPARM CLEANUP  INITIAL(YES)  RESTART(YES)  DYNALLOC(YES)
```

ENCPARM CONDCODE: Specify the Condition Codes That the ESP Encore Job Step Produces

Purpose

Use the ENCPARM CONDCODE initialization parameter to establish customized condition codes for the ESP Encore step. Usually, the ESP Encore step finishes with a condition code of 0, unless errors have been predicted. Your JCL might need to distinguish between a restart and an initial run. You can have the ESP Encore step produce a specified condition code for an initial run and a different specified condition code for a restart run.

Applicability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM CONDCODE  [INITIAL(n1|0)]
                  [RESTART(n2|0)]
                  [FAIL(CC|ABEND)]
```

Operand	Description
INITIAL(*n1*|0)	The completion code that the ESP Encore step issues when a job runs as TYPE INITIAL and there are no errors detected or predicted. The default is 0.
RESTART(*n2*|0)	The completion code that the ESP Encore step issues when a job runs as TYPE RESTART and there are no errors detected or predicted. The default is 0.
FAIL(CC)	ESP Encore completes with the appropriate non-zero condition code when an error is predicted or when an environmental problem occurs (see “ESP Encore condition codes” in the *ESP Encore User’s Guide*). FAIL(CC) is the default operand. CC is the default.
FAIL(ABEND)	ESP Encore issues a user abend with the same value as the appropriate non-zero condition code. For example, if ESP Encore predicts a DATA SET NOT FOUND error, it completes with a U0146 abend (see “ESP Encore condition codes” in the *ESP Encore User’s Guide*).

Example

The following ENCPARM CONDCODE initialization parameter specifies that the ESP Encore step ends with RC=24 instead of 0 for initial runs and RC=48 instead of 0 for restarts. It also specifies that ESP Encore issues a user abend when a pre-substitution, non-zero condition code is predicted.

```
ENCPARM CONDCODE INITIAL(24) RESTART(48) FAIL(ABEND)
```

ENCPARM DIAG: Produce All Diagnostic Sections of the ESP Encore Job Run Report

Purpose

The ENCPARM DIAG initialization parameter causes the diagnostic sections of the report to be printed.

Applicability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM DIAG [YES|NO]
```

Operand	Description
`YES`	Produce all sections of the diagnostic report unless a later PRINT initialization parameter is issued to disable specific sections.
`NO`	Do not produce the diagnostic report unless a later PRINT initialization parameter is issued to produce selected sections of the report. NO is the default.

Usage notes

The ENCPARM DIAG command prints all the diagnostics sections of the ESP Encore job run report. To print selected sections of the report, issue the PRINT command. For details, see "ESP Encore job run report" in the *ESP Encore User's Guide*.

Example

The following command produces all the diagnostics sections of the report, except the MULTIVOLUME section:

```
ENCPARM DIAG  YES
ENCPARM PRINT  MULTIVOL(NO)
```

ENCPARM FORCE: Run a Job Even If ESP Encore Predicts Errors

Purpose

ENCPARM FORCE causes ESP Encore to continue with a job run, even when ESP Encore predicts errors.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM FORCE [YES|NO]
```

Operand	Description
YES	The job runs despite errors that ESP Encore predicted. The job runs until it fails on the first predicted error.
NO	The job does not run if errors ESP Encore predicted errors. NO is the default.

Example

The following ENCPARM FORCE initialization parameter causes ESP Encore to allow the job to run even though it predicted errors.

```
ENCPARM FORCE YES
```

ENCPARM GDGADJ: Specify How ESP Encore Processes Relative Generations For a Job Restart

Purpose

Sometimes a relative generation reference does not refer to the correct generation when a job restarts. You can use ENCPARM GDGADJ to control whether ESP Encore uses absolute generation numbers or relative generation numbers for the restart.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM GDGADJ [YES|NO] MOVED([YES|NO])
```

Operand	Description
YES	During a restart, ESP Encore replaces a relative generation number in the JCL with the absolute generation number stored in the EXH record for the job being restarted. YES is the default. **Note:** If other jobs create new generations before the restart, the absolute generation replacements might not be valid for the restart job. ESP Encore does not modify absolute generation numbers in the JCL.
NO	ESP Encore uses the relative and absolute GDG generations currently in the JCL.
MOVED(YES)	ESP Encore replaces a relative generation number in the JCL with its absolute equivalent, even if the absolute generation was moved to another volume (which implies that another job might have deleted and then recreated the absolute generation). YES is the default. Use this operand only with ENCPARM GDGADJ YES.
MOVED(NO)	ESP Encore replaces a relative generation number in the JCL with its absolute equivalent only if the absolute generation was not moved to another volume (which might happen if another job deleted and then recreated the absolute generation). Use this operand only with ENCPARM GDGADJ YES.

Example 1

The following ENCPARM GDGADJ command causes ESP Encore to replace relative generation numbers with their absolute equivalent, even if the generation data set has been moved to a different volume.

```
ENCPARM GDGADJ YES MOVED(YES)
```

Example 2

The following JCL runs:

```
//TESTJOB1 JOB (TEST1000),'TEST',MSGCLASS=X,CLASS=A
//*
//STEPA    EXEC PGM=IEFBR14
//*
//STEPB    EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DSN=TCY.TEST.GDG(0),DISP=SHR
//SYSUT2   DD DSN=TCY.TEST.NEW,
//            DISP=(NEW,CATLG),
//            SPACE=(TRK,1),UNIT=SYSDA,
//            DCB=(LRECL=80,BLKSIZE=80,RECFM=FB)
//*
//STEPC    EXEC PGM=DJCDEC
```

1. STEPA issues RC=0.
2. STEPB reads in TCY.TEST.GDG.G0002V00, catalogs TCY.TEST.NEW, and issues RC=0.
3. STEPC abends with completion code SB37.
4. TESTJOB2 runs and creates TCY.TEST.GDG.G0003V00.
5. TESTJOB1 restarts from STEPB.

Here are the results with various ENCPARM GDGADJ settings:

GDGADJ	MOVED	Generation of TCY.TEST.GDG read by ESP Encore in STEPB
YES	NO	G0002V00
YES	YES	G0002V00
NO	NO/YES	G0003V00

Example 3

The following JCL runs:

```
//JOB1     JOB (TEST1000),'TEST',MSGCLASS=X,CLASS=A
//*
//STEPA    EXEC PGM=IEFBR14
//*
//STEPB    EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT1   DD DUMMY,
//         DCB=(LRECL=80,BLKSIZE=80,RECFM=FB)
//SYSUT2   DD DSN=TCY.TEST.GDG(+1),
//         DISP=(NEW,CATLG),
//         SPACE=(TRK,1),UNIT=3480,
//         DCB=PCY.GDGMODEL.DATA
//*
//STEPC    EXEC PGM=DJCDEC
```

1. STEPA issues RC=0.
2. STEPB catalogs TCY.TEST.GDG.G0002V00, and issues RC=0.
3. STEPC abends with completion code SB37.
4. JOB2 runs and creates TCY.TEST.GDG.G0003V00.
5. JOB3 runs, uncatalogs, and deletes TCY.TEST.GDG.G0002V00, then creates and catalogs the same generation on another volume.
6. JOB1 restarts from STEPB.

Here are the results with various ENCPARM GDGADJ settings:

GDGADJ	MOVED	Generation of TCY.TEST.GDG created by ESP Encore in STEPB
YES	NO	G0004V00
YES	YES	G0002V00
NO	NO/YES	G0004V00

ENCPARM HONORCC: Control ESP Encore Condition Code Checks of Previous Job Runs

Purpose

The ENCPARM HONORCC initialization parameter specifies whether ESP Encore considers condition codes from the previous run when restarting a job.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM HONORCC  [ALL|NONE]
```

Operand	Description
`ALL`	During a job restart, ESP Encore considers the previous run's condition codes. These condition codes are used in any condition code test in the restart. ALL is the default.
`NONE`	During a job restart, ESP Encore does not consider the previous run's condition codes.

Example

In the following JCL, if STEP1 ends with condition code 0, STEP3 is bypassed; otherwise, STEP3 runs.

```
//STEP1   EXEC  PGM=P1
...
//STEP2   EXEC  PGM=P2
...
//STEP3   EXEC  PGM=P3,COND=(0,EQ,STEP1)
...
```

However, if STEP1 ends with condition code 0 and STEP2 abends, the restart from STEP2 proceeds as follows:

- If you issue ENCPARM HONORCC NONE, STEP3 runs.

  In this case, ESP Encore does not consider the STEP1 condition code.

- If you issue ENCPARM HONORCC ALL, STEP3 is bypassed.

  In this case, ESP Encore considers the previous run's condition codes, so the condition code test in STEP3 is true.

ENCPARM IGNOREDS: Specify Data Sets ESP Encore Ignores on Job Restart

Purpose

The ENCPARM IGNOREDS command specifies the data sets that ESP Encore ignores when deciding which data sets to delete or recreate.

For these data sets

- No errors are predicted.
- No cleanup is done.
- No auto-restoring is performed.
- No security checking is performed.
- No adjustment of GDGs is done.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM IGNOREDS  [DSNAME(-|dsname)]
                  [JOBNAME(-|jobname)]
                  [DDNAME(-|ddname)]
                  [PGMNAME(-|pgmname)]
```

Operand	Description
DSNAME(- |*dsname*)	The data sets to be ignored. The default is a hyphen (-), which matches any data set name.
JOBNAME(- |*jobname*)	The jobs referring to the data sets you want ESP Encore to ignore. The default is a hyphen (-), which matches data sets that any job name refers to.
DDNAME(- |*ddname*)	The DD names for the data sets that you want ESP Encore to ignore. The default is a hyphen (-), which matches data sets that any DD name refers to.
PGMNAME(- |*pgmname*)	The executing programs referring to the data sets you want ESP Encore to ignore. The default is a hyphen (-), which matches data sets that any program refers to.

Usage notes

You might include wildcard characters in the value of the DSNAME, JOBNAME, DDNAME, and PGMNAME operands:

- An asterisk matches a single character.
- A hyphen matches a string of characters.

You can only use hyphens in the last character of a value. If you do not specify a value, the default is a hyphen (-), which matches any value.

You can issue as many ENCPARM IGNOREDS commands as you want. ESP Encore compares the value of each operand with the corresponding value in the job. If all values match, ESP Encore ignores the corresponding data set.

When a data set name matches an ENCPARM IGNOREDS command, ESP Encore ignores that data set name everywhere it occurs in the job.

Example 1

The following ENCPARM IGNOREDS command causes ESP Encore to ignore any data set specified in an IMSLOG DD statement:

```
ENCPARM IGNOREDS DDNAME(IMSLOG)
```

Example 2

The following ENCPARM IGNOREDS initialization parameter causes ESP Encore to ignore any data set if the following conditions apply:

- A step-executing program, PAY287, refers to the data set.
- DD statement SYSUT1 specifies the data set.

```
ENCPARM IGNOREDS PGMNAME(PAY287) DDNAME(SYSUT1)
```

ENCPARM MODIFY: Modify the Internal Processing of ESP Encore

Important: The settings for ENCPARM MODIFY must be the same for the master and all proxies.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM MODIFY COMPRESS(YES|NO)
               DSEXIT(module)
               GDGENQ(YES|NO)
               PACING(15|nnn)
               USECSI(YES|NO)
```

Operand	Description
COMPRESS(YES|NO)	Determines whether each EXH record is compressed when a job completes its execution. Compressing the records saves space, but it requires extra CPU cycles to expand the records for a potential restart. NO is the default.
DSEXIT(*module*)	Specifies an optional exit routine that ESP Encore invokes when processing each data set in a job. This user-defined exit routine can make decisions regarding how ESP Encore handles each data set in a job. For example, you can use DSEXIT to support complex naming conventions that ESP Workload Manager's wildcard capability does not support.
GDGENQ(YES|NO)	Determines whether ESP Encore enqueues on the GDG version of the data set that will be used during the restart. If you specify NO, the restart job can fail due to interference from other jobs in the system at the same time. YES is the default.

Operand	Description
PACING(15|*nnn*)	The maximum number of records per second collected and passed to the EXH data set. If you specify PACING(0), PACING(15) is used. Specifying a higher value than the default can improve EXH data set performance. However, higher values increase the number of records that can be lost in case of system problems. The default value of 15 is sufficient for most installations because it gives you a maximum throughput of 54,000 EXH data set slots per hour (15 records/second times 3600 seconds/hour).
USECSI(YES|NO)	Determines whether ESP Encore uses CSI (Catalog Search Interface) to search the system catalog. If you specify NO, ESP Encore uses the IDCAMS catalog access. YES is the default. Your system must use the integrated catalog facility (ICF) for ESP Encore to use CSI. If your system does not use ICF, ESP Encore uses the IDCAMS catalog access.

Examples

The following example indicates that compression should be performed for records written to the EXH data set:

```
ENCPARM MODIFY COMPRESS(YES)
```

The following example selects the CSI catalog access:

```
ENCPARM MODIFY USECSI(YES)
```

ENCPARM PREDICT: Specify the Error Types That ESP Encore Predicts

Purpose

The ENCPARM PREDICT initialization parameter controls what type of errors ESP Encore predicts. By default, ESP Encore predicts all errors.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM PREDICT error-type([YES|NO]) [error-type([YES|NO])...]
                [BYPASS(nnn)]
                [PROG(program)]
                [NOPROG(program)]
```

Operand	Description
error-type(YES)	ESP Encore predicts this type of error. YES is the default. See "Error types" on page 317 for the list of valid *error-type* values.
error-type(NO)	Specifies that ESP Encore does not predict this type of error.
BYPASS(*nnn*)	ESP Encore treats *nnn* as a special condition code and does not predict DSNOTFOUND or NOTCATLG2 errors for these steps. **Note:** If ESP Encore knows that a job step never executes, you do not need to code the BYPASS operand to avoid DSNOTFOUND or NOTCATLG2 errors.
PROG(*program*)	ESP Encore does not predict DSNOTFOUND errors for all job steps after the first step that runs *program*.
NOPROG(*program*)	Reverses the previous setting of the PROG(*program*) operand. This operand is useful when PROG(*program*) is the global default.

Error types

The *error type* must be one of the following:

Error Type	Description
DCB	The DCB attributes specified in the DCB parameter are inconsistent.
DSNOTFOUND	The data set name cannot be found in the catalog.
DUPNAME	The data set name already exists on the specified volume.
GDGDEF	The GDG base is not defined.
GDGGEN	The specified GDG generation does not exist.
NOTCAT2	The data set cannot be cataloged because it is already cataloged.
NOTDEL8	The data set cannot be deleted because it does not exist.
NOTONVOLUME	The requested data set is not on the volume specified in the DD statement or in the catalog.
NOTUNCAT2	The data set cannot be uncataloged because it is not cataloged.
PGMNOTFOUND	The specified program cannot be found.
RDNR	The job cannot restart from the specified FROMSTEP since that step contains the RD=NR parameter.
SECURITY	A security violation has been detected.
SPACE	The SPACE parameter is missing for a new data set on DASD.
UNIT	A UNIT parameter is missing for a new data set with VOLUME specified.

Example — NOTCAT2 and SPACE

When you issue the following ENCPARM command, ESP Encore does not predict NOTCAT2 and SPACE parameter errors.

```
ENCPARM PREDICT NOTCAT2(NO) SPACE(NO)
```

Note: ESP Encore still predicts all other errors.

Example — BYPASS

Sometimes you may code a COND parameter as follows so that a step is bypassed:

```
...
//STEP5  EXEC PGM=ABC,COND=(999,NE)
//SYSUT1 DD   DSN=PCY.DATA,DISP=SHR
...
```

If data set PCY.DATA in the preceding example does not exist, ESP Encore predicts a "data set not found" error, even though the step will not run. To prevent the error prediction

- Code the following ESP Encore initialization parameter:

```
PREDICT BYPASS(999)
```

Note: You can also change the COND parameter to avoid the error prediction problem. In the following example, `COND=(999,NE)` is changed to `COND=(0,LE)`:

```
...
//STEP5    EXEC PGM=ABC,COND=(0,LE)
//SYSUT1   DD DSN=PCY.DATA,DISP=SHR
...
```

Because a step cannot have a completion code of less than 0, ESP Encore knows that STEP1 in the preceding example never executes. In this case, ESP Encore does not predict a “data set not found” error.

Example — PROG

In the following JCL, STEP01 executes IDCAMS to define a data set that STEP02 refers to. As a result, ESP Encore predicts a DSNOTFOUND error because ESP Encore does not see the IDCAMS control commands:

```
//STEP01 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *
  DEFINE CLUSTER -
   (NAME(A.B.C) -
    VOL(volume))
    ...
//*
//STEP02 EXEC PGM=MYPGM
//DD1    DD DSN=A.B.C,DISP=SHR
```

To stop ESP Encore from predicting DATA SET NOT FOUND errors for all steps after STEP02 (the IDCAMS step), code the following ESP Encore initialization parameter:

```
PREDICT PROG(IDCAMS)
```

Example — NOPROG

PROG(ABC) is issued at the beginning of the following Application definition, so it applies to all jobs in the Application. NOPROG(ABC) is issued in the job definition for job P2, so it applies to that job only. Job P1 and P2 run program ABC.

```
JCLLIB ‘PA.JCL.CNTL’
APPL  PAYROLL
OPTIONS RESTARTSTEP
ENCPARM PREDICT PROG(ABC)

JOB P1
  RUN DAILY
ENDJOB

JOB P2
  ENCPARM PREDICT NOPROG(ABC)
  RUN DAILY
ENDJOB

JOB P3
  RUN DAILY
ENDJOB
```

In this case

- ESP Encore does not predict any DSNOTFOUND errors in job P1 for all steps after the one that runs program ABC.
- ESP Encore predicts all DSNOTFOUND errors in job P2, even though job P2 runs program ABC.
- ESP Encore predicts all DSNOTFOUND errors in job P3 because job P3 does not run program ABC.

ENCPARM PRINT: Select Sections of the ESP Encore Job Run Report to Print

Purpose

ESP Encore usually only prints the job run report sections that are relevant to the current run. However, you can print other report sections by issuing the ENCPARM PRINT command. For details, see "ESP Encore job run report" in the *ESP Encore User's Guide*.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM PRINT section-name([YES|NO])
              section-name([YES|NO])
              ...
```

Operand	Description
section-name(YES|NO)	Specifies whether a section of the ESP Encore job run report is produced. The default, YES, applies only to specified section names. For example, to print errors you can issue ENCPARM PRINT ERRORS(YES) or just ENCPARM PRINT ERRORS. The possible values for *section-name* are

Value	Report section printed
ACTION2	Dump of Action Table section
ACTIONS	Action Summary section
CATALOG	Catalog section
ERRORS	Errors section
EXHFILE	EXH data set Statistics section
HISTORY	History of Previous Job section
MULTIVOL	Multivolume data sets section
RESTART	Job Restart Summary section
RETRIEVAL	EXH Record-Retrieval Details section
SIOTJFCB	SIOT/JFCB section
SMF	SMF Records section
SUBSYS	Subsystem Information section
SUMMARY	Job Summary section
UNIT	Unit Names section
VTOC	VTOC section

Usage notes

Enabling all report sections

The ENCPARM DIAG YES command enables all the options listed under the ENCPARM PRINT command.

Report sections not controlled by ENCPARM PRINT

The following report sections are always produced:

- Title
- Parameter Summary
- Initialization Parameters

ESP Encore produces the report’s "Severe Error" section only when a severe error occurs. For example, an ESP Encore condition code 44 (a job's EXH record is no longer in the EXH data set) causes ESP Encore to produce the report’s "Severe Error" section.

Example

The following command adds the "History of Previous Job" section and the "Subsystem Information" section to the ESP Encore report.

```
ENCPARM PRINT HISTORY(YES) SUBSYS
```

ESP Encore assumes the default, YES, for the operand SUBSYS.

ENCPARM PURGE: Configure Automatic Purge of ESP Encore Job History Records

Purpose

The ENCPARM PURGE command sets conditions under which ESP Encore automatically purges records from the EXH data set before creating a new record.

Regardless of whether you use ENCPARM PURGE, run the CYBRMPRG utility regularly to maintain your EXH data set.

Important: You can only enter ENCPARM PURGE in the initialization parameters or in page mode. If you enter it at the Application level or job level, it is ignored.

Important: You can enter more than one ENCPARM PURGE command. They are kept in the same sequence as they are entered. The first ENCPARM PURGE command that matches the job name is executed. Once the match is found, the remaining ENCPARM PURGE commands are skipped. To achieve the desired results, enter ENCPARM PURGE commands from the most specific to the most generic.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM PURGE JOB(job-mask) [AGE(days)] [INCOMPLETEAGE(days)]
                    [KEEP(number)] [INCOMPLETEKEEP(number)]
```

Operand	Description
JOB(*job-mask*)	A job name mask string that might contain wildcard characters
AGE(days)	The number of days that complete job groups matching the job name mask are kept on the EXH data set. When a complete job group reaches or exceeds the specified age, it is subject to purging.

Operand	Description
INCOMPLETEAGE(*days*)	The number of days that incomplete job groups matching the job name mask are kept on the EXH data set. When a complete job group reaches or exceeds the specified age, it is subject to purging.
KEEP(*number*)	The number of complete job groups matching the job name mask that are kept on the EXH data set
INCOMPLETEKEEP(*number*)	The number of incomplete job groups matching the job name mask that are kept on the EXH data set

Usage notes

Batch jobs

The ENCPARM PURGE command is ignored when it is issued in the ESP Encore step of a batch job.

Job groups

A job group is composed of an initial job and all restarted jobs stemming from that initial job. A complete job group is one in which the most recent job has executed successfully; otherwise, it is an incomplete job group. The age of the most recent member of a job group determines the age of the whole job group.

Listing jobs and default purge values

- You can list the jobs in the index of an EXH data set by running the CYBRMLST utility or the CYBRMPRG utility with the TEST operand set to YES.
- You can list the default purge values that are stored on the EXH data set by running the CYBRMQRY utility.

The utilities are in the sample library (SSCPSAME).

Setting the default automatic job purge values

If you do not specify values for the optional operands, ESP Encore uses the default values taken from the EXH data set.

- You can set the default purge values for the automatic job purge when you allocate the EXH data set with the CYBRMALC utility.
- You can change the default automatic job purge values with the CYBRMKEP utility.
- You can override the default automatic job purge values with the ENCPARM PURGE command.

Automatic job purging

Important: Automatic job purging occurs gradually as new jobs that match the purge initialization parameters run.

A completed job group is purged when one of the following conditions is met:

- The most recent job in the group is older than AGE.
- The completed job group is the oldest job group in the EXH data set and a new job group is created that exceeds the KEEP limit.

An incomplete job group is purged when one of the following conditions is met:

- The most recent job in the group is older than INCOMPLETEAGE.
- The incomplete job group is the oldest incomplete job group in the EXH data set and a new job group is created that exceeds the INCOMPLETEKEEP limit.

Purging jobs manually

You can immediately purge jobs from the EXH data set with the CYBRMPRG utility. Use the CYBRMPRG utility regularly to maintain your EXH data set.

Example

In this example, the default purge criteria stored on the EXH data set are

```
AGE=3
INCOMPLETEAGE=10
KEEP=4
INCOMPLETEKEEP=6
```

ENCPARM PURGE is used to modify the default purge values:

```
ENCPARM PURGE JOB(A-) AGE(14)
ENCPARM PURGE JOB(B-) INCOMPLETEAGE(12)
ENCPARM PURGE JOB(-) AGE(4) INCOMPLETAGE(6) KEEP(3)
INCOMPLETEKEEP(5)
```

The default purge values are modified as follows:

Jobs	Purge values

Name starting with A	AGE=14
	INCOMPLETEAGE=10
	KEEP=4
	INCOMPLETEKEEP=6

Name starting with B	AGE=3
	INCOMPLETEAGE=12
	KEEP=4
	INCOMPLETEKEEP=6

All other job names	AGE=4
	INCOMPLETEAGE=6
	KEEP=3
	INCOMPLETEKEEP=5

ENCPARM TAPESCR: Request a Tape Scratch From a TMS

Purpose

The ENCPARM TAPESCR initialization parameter specifies how ESP Encore scratches tape data sets.

Note: This initialization parameter applies only if you use a tape-management system.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM TAPESCR PROG(pgmname)|
                COMMAND('command_text') CONSOLE(consname)
                                        CART(token)
```

Operand	Description
PROG(*pgmname*)	The name of the tape management program that ESP Encore calls to scratch the tape data set
COMMAND('*command_text*')	The text of an operator command that ESP Encore issues to scratch a tape data set. The maximum length of the command is 126 characters. If your command text is longer than 126 characters, use a tape-scratch program and invoke it with TAPESCR PROG(*progname*)
CONSOLE(*consname*)	The name of a console from which the command is issued. Specify CONSOLE only when you specify the COMMAND operand. ESP Encore does not issue an error if you specify CONSOLE without COMMAND.
CART(*token*)	The response token that is associated with the command. Specify CART only when you specify the COMMAND operand. ESP Encore does not issue an error if you specify CART without COMMAND.

Usage notes

PROG and COMMAND are mutually exclusive. CONSOLE and CART should only be specified if COMMAND is specified.

If no tape-management system is installed, ESP Encore only uncatalogs the tape data set using the IDCAMS NOSCR command. However, if a tape-management system is present, ESP Encore also executes the interface program named in *pgmname* or issues the command in *command_text*.

Using symbolic variables in a tape-scratch command

You can include the following symbolic variables in a tape-scratch command. Symbolic variables are replaced by their assigned values before the command is issued.

- &DSN — name of a data set to be scratched
- &VOL — list of VOLSERs that store the data set
- &EXPDT — data set expiry date
- &RETPD — data set retention period
- &TYPE — type of ESP Encore run (I-initial run, B-backout run, R-restart run)

Note: IBM restricts the length of the longest operator command to 126 characters. If your installation uses very large multi-volume data sets, the VOLSER list substitution might exceed the legal operator command length. In this situation, use a tape-scratch program and invoke it with TAPESCR PROG(*progname*).

You can list the tape-scratch command before and after the symbolic variable substitution processing by issuing the TRACE CYBRM577 command. Because the TRACE CYBRM577 degrades performance, remove it before production.

Using a tape-scratch program

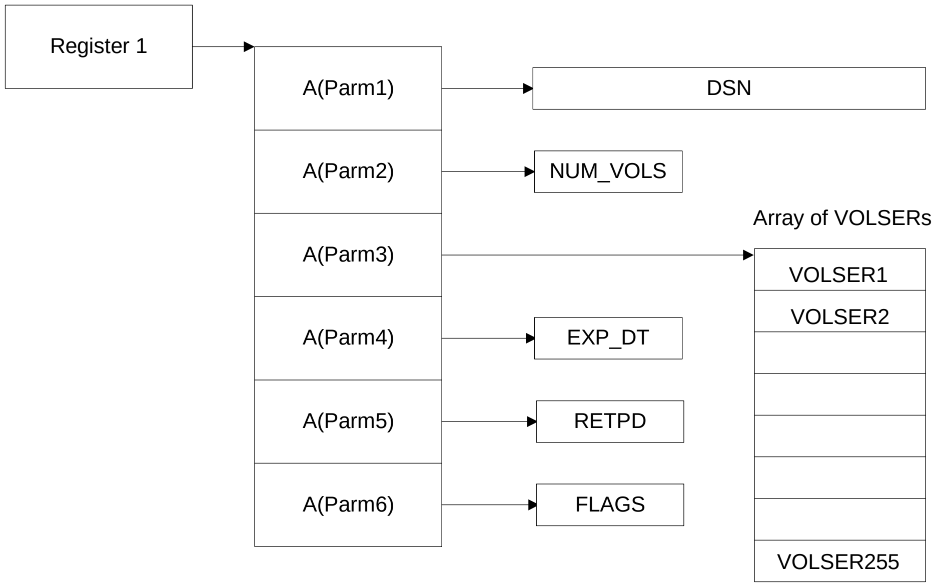
A tape-scratch program is attached; that is, it runs as an operating system subtask and its parameter list is in the standard z/OS subroutine parameter list format. The program is invoked in the AMODE (addressing mode) that was specified during its link-edit run.

The following table describes the parameters that ESP Encore passes to any program specified in ENCPARM TAPESCR.

Parameter	Description
DSN	Data set name. It is up to 44 characters long.
NUM_VOLS	Number of volumes this data set is stored on. It is a binary full-word counter. The maximum is 255 volumes (the operating system limit).
VOL_SER	Array of VOLSERs. Each array entry is six bytes long and contains a volume serial number in character format.
EXP_DT	Data set expiry date. The date is in the Julian date format YYDDD. It is a three-byte field. The first byte contains a binary representation of YY, the next two bytes contain the binary representation of DDD. For example YYDDD=99365 is stored as X'63016D'.

Parameter	Description
RETPD	Data set retention period. The date is in the Julian date format YYDDD. It is a three-byte field. The first byte contains a binary representation of YY, the next two bytes contain the binary representation of DDD.
FLAGS	A two-byte field that encodes ESP Encore run type as follows: • X'8000' – initial run • X'4000' – restart run • X'2000' – backout run • X'1000' – cleanup run

The following is the z/OS standard parameter list format:



Example — Tape-scratch program

In this example, ESP Encore invokes a program called MYTAPSCR whenever a tape data set should be scratched:

```
ENCPARM TAPESCR PROG(MYTAPSCR)
```

In this example, TAPESCR PROG attaches the program MYTAPSCR, creating a system subtask. The parameter list described previously is passed to MYTAPSCR. The program is invoked in the AMODE that was specified during its link-edit run.

Example — Symbolic parameters in TAPESCR COMMAND

In the following example, a ENCPARM TAPESCR initialization parameter issues a START command for a started task called SCRTAP. Parameters P1 and P2 pass to the started task:

```
TAPESCR COMMAND('S SCRTAP,P1=&DSN,P2=&VOL')
```

Assuming symbolic parameter &DSN=TAP1 and &VOL=SER=(TAPE01,TAPE02), the command appears as follows after the values are substituted:

```
S SCRTAP,P1=TAP1,P2='TAPE01,TAPE02'
```

The started task that is invoked must have a matching definition of parameters:

```
//SCRTAP PROC P1='NODSN',P2='NOVOL'
//STEP1  EXEC SCRTAP1,PARM='&P1,&P2'
```

If the parameters do not match, you might see the following JCL card that will cause a JCL error:

```
//IEFPROC.IEFRDER DD …a list of your mismatched parameters…
```

The started task will fail.

Note: In this example, the parameters passed to the program SCRTAP1 come from the invocation line. The parameters from the invocation line (and also from the JCL PARM field) are in the varying string format as described in the IBM documentation:

LL	DATA

In the example LL = 18 (binary halfword) followed by
DATA = TAP1,TAPE01,TAPE02 in the character format. Note that the operating system removes the single quotation marks surrounding the list of parameter P2 so that program SCRTAP1 does not see them.

ENCPARM TRACE: Specify Modules to Be Traced

Purpose

The ENCPARM TRACE initialization parameter specifies the name or prefix of one or more modules that ESP Encore should trace.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM TRACE (modulename1, modulename2 ...)[VERBOSE|NVERBOSE]
```

Operand	Description
module_name	A module or a set of modules that ESP Encore should trace. If the name is less than eight characters long, ESP Encore traces all modules beginning with that prefix. If ESP Encore traces only one module, parentheses are optional.
VERBOSE	Produces any additional trace messages for ESP Encore programs that have tracing enabled. Verbose tracing requires additional processing, so use it only if CA Technical Support requests you to do so.
NVERBOSE	Disable verbose tracing. See the VERBOSE operand.

Usage notes

The ENCPARM TRACE command causes trace information to be printed among the normal reports the ESP Encore step produces.

Since tracing degrades performance, use this command for diagnostics purposes only. Usually CA Technical Support asks you to run a trace.

Example

In the following example, ESP Encore produces trace information for module CYBRM574 and CYBRM402:

```
TRACE (CYBRM574,CYBRM402)
```

In the following example, verbose trace information for all ESP Encore modules is generated:

```
TRACE CYBRM VERBOSE
```

ENCPARM VOLUME: Specify Data Set Processing Based on Volume Serial

Purpose

Use the ENCPARM VOLUME initialization parameter to

- Specify special processing options for selected volumes.

  Selected volumes are either excluded from ESP Encore processing or included in ESP Encore processing.

- Specify the string the installation uses to identify a migrated data set.

  The operand MIGRATE applies installation-wide and it can only have a single value.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM VOLUME EXCLUDE(excl-vol-list)
               INCLUDE(incl-vol-list)
               MIGRATE(MIGRAT|mig-vol)
```

Operand	Description
EXCLUDE(*excl-vol-list*)	A list of volume serial numbers that represent volumes on which no maintenance should be done. ESP Encore does not delete any data set that resides on a volume in the list. You can use the wildcard characters asterisk and hyphen.
INCLUDE(*incl-vol-list*)	A list of volume serial numbers that represent volumes on which maintenance should be done. INCLUDE overrides previous EXCLUDE operands.

Operand	Description
MIGRATE(MIGRAT|*mig-vol*)	The volume serial numbers that indicate that a data set has been migrated or archived to tape. MIGRAT is the default value. **Note:** MIGRAT is the value the IBM data-management software uses. Other data-management software vendors might use different values. Consult your system programmer to determine the correct value for your site. Migrated or archived data sets are indicated by MIGRATED in the ESP Encore job run report's "Catalog" section (see "Diagnostic sections of the ESP Encore job run report" in the *ESP Encore User's Guide*). Also, for these data sets, ESP Encore does not predict the following errors: • DATA SET ALREADY DELETED • DATA SET NOT YET EXPIRED, NOT DELETED • PERMANENT DATA SET NOT ON VOLUME

Example

In the following example, ESP Encore does not delete data sets from SYSRES or from any volume whose volume serial begins with PAY, except for PAY002. ESP Encore considers any data set cataloged on ARCIVE as migrated or archived.

```
ENCPARM VOLUME EXCLUDE(SYSRES,PAY-) MIGRATE(ARCIVE)
ENCPARM VOLUME INCLUDE(PAY002)
```

ENCPARM WARNING: Display Extra Warning Messages to Support Manual Restart Tasks

Purpose

In some restart situations, you might need to manually adjust data sets. For example, you might have to recreate a missing data set or adjust a data set that has DISP=MOD specified in the DD statement. To remind you about the required adjustments, you can have ESP Encore issue extra warning messages in the Restart Action Summary screen. You can still restart a job when you see the extra warning messages.

Availability

ESP Encore

Where defined

Master and proxy

Syntax

```
ENCPARM WARNING GDGALL(YES|NO)
                MISSING(YES|NO)
                DISPMOD(YES|NO)
```

Operand	Description
GDGALL(YES|NO)	Determines whether ESP Encore issues a warning message when a job restarts and a generation is missing from a GDG concatenation that existed when the job first ran. NO is the default. **Note:** The ENCPARM WARNING GDGALL(YES) command must be issued for the job's initial run and restart. If it is not, the job's initial run does not track the generations included in the original GDG concatenation. **Note:** Jobs can still run if warning messages are issued for a missing GDG generation.
MISSING(YES|NO)	Determines whether ESP Encore issues a warning message if a data set no longer exists and ESP Encore intends to clean up the data set. Another job or user might have already deleted the data set. NO is the default.
DISPMOD(YES|NO)	Determines whether ESP Encore issues a warning message if a data set with DISP=MOD is specified in a job. These data sets might require special handling by the user before a restart. ESP Encore usually tries to auto-restore these data sets. If it cannot, it issues the warning. NO is the default.

Example — Warning message for WARNING GDGALL(YES)

```
WARNING: A CONCATENATED GDG GENERATION IS MISSING.
THE MISSING GENERATION IS PA.TESTGDGB.G0017V00
THIS GENERATION WAS PART OF A GDG CONCATENATION.
```

Example — Warning message for WARNING DISPMOD(YES)

```
WARNING: A DISP=MOD DATA SET WAS MODIFIED BY THE JOB.
THE DATA SET IS PA.TEST.DD4001 ON VOL003
THIS DATA SET MIGHT NEED TO BE RECREATED BEFORE RESTARTING THIS
JOB.
```

Example — Warning message for WARNING MISSING(YES)

```
WARNING: DATA SET PA.TEST.DD4001 IS MISSING.
(THIS DATA SET HAS ALREADY BEEN CLEANED UP BY SOME OTHER JOB OR
USER.)
```

ENCRYPT: Specify encryption key for communications

Purpose

The ENCRYPT initialization parameter enables encrypted communications between the ESP Agent or Workstation Server and the ESP Manager using a globally defined encryption key.

You can code more than one ENCRYPT initialization parameter in the AGENTDEF Agent definition data set.

An ENCRYPT initialization parameter applies to Agents subsequently defined until the next ENCRYPT initialization parameter is encountered.

Where defined

Master

AGENTDEF data set

WSSPARM data set

Syntax

```
ENCRYPT KEY(key)|KEYNAME(keyname) [ALL|NOALL]
```

Operand	Description
KEY(*key*)	The encryption key for communications. The form is `X'nnnnnnnnnnnnnnnn'` where X signifies hexadecimal format and *n* is a hexadecimal number. The encryption key can have up to 16 characters (8 bytes). •
KEYNAME(*keyname*)	The name of the encryption key for communications. The key name is defined with the CRYPTKEY command. •
ALL	Activates encryption using the defined encryption key for all subsequent Agents with the following exceptions: • If the AGENT initialization parameter uses the NOENCRYPT operand, no encryption is performed. • If the AGENT initialization parameter uses the ENCRYPT operand with a key defined, encryption is performed with the key defined in the AGENT initialization parameter.
NOALL	Activates encryption using the defined encryption key on the subsequent Agents whose initialization parameter includes the ENCRYPT operand with no key defined. NOALL is the default.

Examples

The following is an example of a single ENCRYPT initialization parameter:

```
ENCRYPT KEY(X'010203040506ABCD')
```

The following is a sample AGENTDEF using multiple ENCRYPT initialization parameters:

```
MANAGER NAME(CM_CENTRAL) TCPIP
ENCRYPT KEY(X'010203030501ADDD') ALL
MAPUSER JDOE TO(PROD) AGENT(TORSUN1)
AGENT TORSUN1 ADDRESS(xxx.xx.xx.xx) PORT(9999) UNIX ASCII +
   TCPIP PREFIXING NOENCRYPT
AGENT TORNT1 ADDRESS(xxx.xx.xx.xx) PORT(9998) NT ASCII +
   TCPIP PREFIXING
AGENT TORAS4001 ADDRESS(xxx.xx.xx.xx) PORT(9997) AS400 EBCDIC +
   TCPIP PREFIXING ENCRYPT(X'010203040506AFFC')
ENCRYPT KEY(X'1010101010102CBBD')
AGENT CHI_AS ADDRESS(CHIAS01) PORT(3001) AS400 EBCDIC TCPIP +
   PREFIXING
AGENT NYC_AS ADDRESS(CHIAS02) PORT(3002) AS400 EBCDIC TCPIP +
   PREFIXING ENCRYPT
AGENT SFO_AS ADDRESS(CHIAS03) PORT(3003) AS400 EBCDIC TCPIP +
   PREFIXING ENCRYPT(X'010203040506DEEC')
```

The following table explains the encryption used for communication:

AGENT	Key Used	Explanation
TORSUN1	No encryption	Agent does not use encryption because NOENCRYPT is coded.
TORNT1	010203030501ADDD	Agent uses the global key defined in the first ENCRYPT initialization parameter with ALL operand because neither ENCRYPT nor NOENCRYPT is coded.
TORAS4001	010203040506AFFC	Agent uses the key defined with its ENCRYPT operand.
CHI_AS	No encryption	Agent does not use encryption because neither ENCRYPT nor NOENCRYPT are coded and the second ENCRYPT initialization parameter specifies NOALL by default.
NYC_AS	1010101010102CBBD	Agent uses the global key defined in the second ENCRYPT initialization parameter because ENCRYPT is coded without a key.
SFO_AS	010203040506DEEC	Agent uses the key defined with its ENCRYPT operand.

ENQSELF: Set enqueue on every job

Note: You can also use the ENQSELF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The ENQSELF initialization parameter indicates that every job in all ESP Workload Manager Applications will enqueue on itself.

Note: The ENQSELF initialization parameter use resources implicitly. Therefore, a resource file (RESFILE) and the network topology must be defined before you can use it.

Note: We recommend that you use resources rather than ENQSELF for this purpose.

Where defined

Master and proxy

Syntax

```
ENQSELF [ALWAYS|NOTALWAYS]
```

Operand	Description
ALWAYS	All jobs will enqueue on themselves. This is useful if you are using the NOTWITH statement with numerous, small groups of jobs that are mutually exclusive.
NOTALWAYS	Only jobs with enqueues originating from the NOTWITH statement will automatically enqueue on themselves.

Usage notes

Specifying the NOTWITH statement in the scope of a job requires that the targeted job enqueue on itself. This occurs if either ENQSELF is set to ALWAYS or the targeted job has a NOTWITH statement. Thus setting ENQSELF to ALWAYS allows the use of unmatched NOTWITH statements.

Example

In the following example, job NWN003 has no NOTWITH statement, so it will execute at the same time as job NWN001 unless ENQSELF is set to ALWAYS. Otherwise, job NWN003 must have a NOTWITH NWN001 statement to be mutually exclusive with job NWN001.

```
APPL NWN00A
JCLLIB 'CYBER.JCL.CNTL'
JOB NWN001
  RUN WEEKDAYS
  NOTWITH NWN003
ENDJOB
JOB NWN002
  RUN WEEKDAYS
  ENDJOB
JOB NWN003
  RUN WEEKDAYS
  ENDJOB
```

ERMSTEP: Add restart step

Note: You can also issue the ERMSTEP initialization parameter as a command.

Purpose

The ERMSTEP initialization parameter identifies the JCL to be added to a job that requires ESP Encore.

Where defined

Master

Syntax

```
ERMSTEP {'string'}
        {('string'[,'string']...)}
```

Operand	Description
string	The exact JCL to be added to a job

Usage notes

Each string that the ERMSTEP initialization parameter identifies is added to jobs ESP Workload Manager submits that have OPTIONS RESTARTSTEP specified.

Enclose multiple strings in parentheses. Do not use comments on the last string specified.

Examples

Running ESP Encore

```
ERMSTEP '//ENCORE EXEC CYBENCOR'
```

Replacing initial job step insertion

Initial step AUTOVARs insert JCL before the first job step specified. However, the ESP Encore restart step is automatically inserted before any initial job steps, causing a conflict. This example describes a work-around to this problem.

If you want to use an AUTOVAR to insert initial job steps but cannot because you are using ESP Encore, specify the JCL in the ERMSTEP statement instead. For example, to add an output with a JESDS parameter, specify the following:

```
ERMSTEP('//*OUTPUT CARDS ADDED',-
  '//PRT1 OUTPUT JESDS=ALL,CLASS=A,DEST=TORONTO',-
  '//PRT2 OUTPUT JESDS=ALL,CLASS=R,DEST=CHICAGO',-
  '//ENCORE EXEC E510ENC')
```

The above example routes output for Class A to Toronto and Class R to Chicago.

ESPGROUP: Specify ESP complex ID

Purpose

The ESPGROUP initialization parameter is required for CCCHK and for ESP Encore if you have multiple subsystems.

The ESPGROUP initialization parameter specifies the ID of the ESP Workload Manager master and proxy complex for use with CCCHK or ESP Encore. If you use CCCHK or ESP Encore, an ESP Workload Manager master and all the related ESP Workload Manager proxies must have the same ESPGROUP name.

The ESPGROUP initialization parameter must also be specified to group together all ESP Workload Manager subsystems that are part of a tracking group.

Where defined

Master and proxy

Syntax

```
ESPGROUP name
```

Operand	Description
name	The name of the ESP Workload Manager group. Can be up to eight characters long, the first character alphabetical, the remaining characters alphanumeric. The default ESPGROUP is the subsystem name.

Usage notes

ESPGROUP is not related to sysplex topology.

Example

```
ESPGROUP BOSTON2
```

EVENTOPT: Set Event options

Note: You can also issue the EVENTOPT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The EVENTOPT initialization parameter allows or disallows Write to Operator (WTO) operations from being issued within an Event. EVENTOPT also allows or disallows the use of system commands from within an Event.

Restrictions

Use the EVENTOPT initialization parameter with ESP Workload Manager 4.5.1 or later. This initialization parameter supersedes the EVENTWTO initialization parameter.

Where defined

Master and proxy

Syntax

```
EVENTOPT [WTO|NOWTO]
         [VS|NOVS]
```

Operand	Description
WTO	Allows Write to Operator operations from within an Event
NOWTO	Disallows Write to Operator operations from within an Event
VS	Allows the use of z/OS commands from within an Event
NOVS	Disallows the use of z/OS commands from within an Event

Usage notes

If you specify no operands, EVENTOPT tells you the current status.

Example

The following example disallows both WTOs and system commands from being issued within an Event:

```
EVENTOPT NOWTO NOVS
```

EVENTSET: Identify Event data set

Note: You can also issue the EVENTSET initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The EVENTSET initialization parameter specifies an Event data set's name and other attributes.

Where defined

Master and proxy

ESPCOLD data set

Syntax

```
EVENTSET eventdsid [DSNAME(dsname)]
                   [SCANTIME(schedule)]
                   [DEFINE]
                   [SHR|NOSHR]
                   [JOURNAL|NOJOURNAL]
                   [BACKUPDSNAME(name)|NOBACKUPDSNAME]
```

Operand	Description
eventdsid	A unique identifier that is up to eight characters long
DSNAME(*dsname*)	The name of a VSAM KSDS defined with the correct attributes
SCANTIME(*schedule*)	Specifies when the Event data set will be scanned and, optionally, backed up. SCANTIME uses the same syntax as the command processor schedule statements. If the text string contains blanks or commas, enclose it within quotation marks.
DEFINE	Defines a new Event data set. DEFINE is the default.
SHR	The data set is shared.
NOSHR	The data set is not shared. NOSHR is the default.
JOURNAL	A record is written to SMF each time an Event is defined, updated or deleted. For more information, see "SMFREC: Specify SMF record number" on page 485.
NOJOURNAL	No records are written to SMF when the Event data set is updated. Use this keyword to cancel a previous JOURNAL request. NOJOURNAL is the default.
BACKUPDSNAME(*name*)	The name of a non-VSAM sequential data set
NOBACKUPDSNAME	Removes the BACKUPDSNAME, if it is already specified

Usage notes

Use the EVENTSET initialization parameter to define and initialize an Event data set. Use the DEFINE operand with DSNAME and SCANTIME operands. The identifier (up to eight characters long) is the permanent name to be associated with that logical entity.

When a user or group prefix is defined, an Event data set is assigned to it. The assignment is the logical identifier, rather than the actual data set name, allowing the data set name to be changed or a new data set to be assigned later.

The Event data set is scanned, usually at 24-hour intervals, to produce a schedule. Events created or modified after the scan are automatically added to the schedule. You can list the schedule with the LISTSCH command. If the SCANTIME operand is omitted, the default is 6am daily. If a backup data set is specified, the backup is performed automatically while the schedule scan is occurring.

When ESP Workload Manager encounters an I/O error on an Event data set, it issues an error message and closes the data set. Remedial action can then occur. When the problem is corrected, the EVENTSET OPEN command causes ESP Workload Manager to resume operation with the data set.

Any Events scheduled for execution while the corresponding data set is in the suspended state are queued for deferred execution. When the data set is available again, execution can resume. Activity on other data sets can proceed as normal.

Example

In the following example, a new shared Event data set called EVENT1 is defined. The actual VSAM data set name is ESP.EVENT1, and the backup data set is ESP.BACKUP.EVENT1, which will be used every morning at 5am when EVENT1 is scanned.

```
EVENTSET EVENT1 DEFINE DSNAME(ESP.EVENT1) -
  SCANTIME('5AM DAILY') BACKUP(ESP.BACKUP.EVENT1) SHR
```

EXHFILE: Identify Execution History data set

Purpose

The EXHFILE initialization parameter specifies the name of the Execution History data set that ESP Encore uses.

Where defined

Master and proxy

Syntax

```
EXHFILE dsname
```

Operand	Description
dsname	The data set name of the Execution History data set

Usage notes

You require the EXHFILE initialization parameter only if you are using ESP Workload Manager along with the rerun/restart product ESP Encore. For more information on this data set, see "Execution History (EXH) Data Set" on page 21 and the *ESP Encore User's Guide*.

The EXHFILE is always treated as shared.

Example

In the following example, the EXHFILE name is CYB.RERUN.HIST:

```
EXHFILE 'CYB.RERUN.HIST'
```

EXIT: Activate and control a user exit

Note: You can also issue the EXIT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The EXIT initialization parameter activates and controls an ESP Workload Manager user exit.

Where defined

Master

Syntax

```
EXIT FUNCTION(exittype) MODULE(modulename)
     [INIT]|[TERM]|[INIT,TERM]

EXIT [FUNCTION(exittype)] [MODULE(modulename)]
     DISABLE|ENABLE

EXIT MODULE(modulename)
     UNLOAD|RELOAD
```

Operand	Description
exittype	The logical name of the exit point. Specify one of the following: • COLDSTART • DUEOUT • EVENTDEF • EVENTEXEC • EVENTSAF • JCLSCAN • JOBENDNOTIFY • JOBPSWD • SHUTDOWN • STARTUP • USERSEND • VSCMD • WARMSTART
modulename	The name of the load module to be invoked
INIT	Execute the user exit when ESP Workload Manager is being initialized and every time the specified function occurs. The subfunction code CXDSFUNC in the common exit data area is set to CXDINIT (X'01').

Operand	Description
TERM	Execute the user exit when ESP Workload Manager is being shut down, as well as every time the specified function occurs. During the shutdown, the EXIT will execute whether the functional condition occurred or not.
DISABLE	Disable the specified user exits.
ENABLE	Enable the specified user exits.
UNLOAD	Unload the specified module from memory.
RELOAD	Reload the specified module back into memory.

Usage notes

Load module requirements

The load module specified in *modulename* should be in a library that the ESP Workload Manager subsystem can access. The load module should be re-entrant or at least re-usable.

Defining user exits with identical functions or modules

You can define multiple user exits with

- The same function and different load modules

  When the function condition occurs, the load modules are executed in the order in which they were defined.

- Different functions and the same load module

  The same physical load module is used for each function. The load macro is issued only once.

Using unloaded modules

If a load module is unloaded from memory, you must reload it before any user exit can invoke it. You unload and load modules using the UNLOAD and RELOAD operands.

Examples

Defining user exits

In this example

- Two modules, ESPUSRX1 and ESPUXDUE, are loaded.
- Upon loading module ESPUXDUE, ESP Workload Manager invokes it for initialization because of the INIT operand.
- After the parameter scan completes, the STARTUP exit invokes the module ESPUSRX1 and the module is given a chance to terminate ESP Workload Manager processing.
- If a cold start is performed, the module ESPUSRX1 is invoked again.
- Each time a job is overdue, ESP Workload Manager invokes the module ESPUXDUE.

```
EXIT FUNCTION(STARTUP) MOD(ESPUSRX1)
EXIT FUNCT(COLDSTART) MOD(ESPUSRX1)
EXIT FUNC(DUEOUT) MOD(ESPUXDUE) INIT
```

Disabling and enabling user exits

The following EXIT command disables the user exit for function JCLSCAN and module CYBJSDLT.

```
EXIT FUNCTION(JCLSCAN) MODULE(CYBJSDLT) DISABLE
```

The following EXIT command enables all JCLSCAN user exit functions.

```
EXIT FUNCTION(JCLSCAN) ENABLE
```

The following EXIT command enables all user exit functions that invoke the CYBJSDLT load module.

```
EXIT MODULE(CYBJSDLT) ENABLE
```

The following EXIT command disables all user exit functions.

```
EXIT DISABLE
```

Unloading and reloading modules

The following EXIT command unloads the CYBJSDLT load module from memory.

```
EXIT MODULE(CYBJSDLT) UNLOAD
```

The following EXIT command reloads the CYBJSDLT load module back into memory.

```
EXIT MODULE(CYBJSDLT) RELOAD
```

EXPDSTRG: Set Acceptance Conditions for Explicit Data Set Triggers

Note: You can also issue the EXPDSTRG initialization parameter as a command. For details about command syntax and operands, refer to the *ESP Workload Manager Reference Guide.*

Purpose

The EXPDSTRG initialization parameter specifies the conditions under which an explicit data set trigger notification is accepted.

Where defined

Master and proxy

ESPPARM initialization file

Syntax

```
EXPDSTRG [SECURE|VERIFY|NOVERIFY]
```

Operand	Description
SECURE	Accept an explicit data set trigger only if the following conditions are met: • The specified data set exists and is mounted. • The owner of the job executing ESPDST or the user ID issuing the DSTRIG ACTIVATE command has SAF update access to the specified data set.
VERIFY	Accept an explicit data set trigger only if the specified data set exists and is mounted.
NOVERIFY	Always accept an explicit data set trigger. NOVERIFY is the default.

Usage notes

Intended use of EXPDSTRG

Use EXPDSTRG on tracking ESP Workload Manager subsystems (subsystems with initialization parameter SMFINIT set to ON).

Overriding the EXPDSTRG initialization parameter

The EXPDSTRG command overrides the EXPDSTRG initialization parameter (if one is coded) for as long as ESP Workload Manager is active.

EXPDSTRG value at startup

The EXPDSTRG value issued in the initialization parameter or command is checkpointed. This means that it remains in effect until ESP Workload Manager is

- Cold started
- Warm started with the EXPDSTRG initialization parameter coded

The following table gives the value of EXPDSTRG for a cold start and a warm start.

Cold Start with EXPDSTRG Initialization Parameter		**Warm Start with EXPDSTRG Initialization Parameter**	
Defined	**Not defined**	**Defined**	**Not defined**
Uses the EXPDSTRG initialization parameter	Uses the default, EXPDSTRG NOVERIFY	Uses the EXPDSTRG initialization parameter	Uses the checkpointed values

Example

In this example, the VERIFY operand means that explicit data set triggers are only accepted if the data set specified in the trigger exists and is mounted:

```
EXPDSTRG VERIFY
```

EXPEDITE: Define expedite policy

Note: You can also issue the EXPEDITE initialization parameter as an OPER command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The EXPEDITE initialization parameter defines an expedite policy.

An expedite policy specifies criteria that a job must meet to be automatically expedited.

The expedite policy specifies how a job's priority should increase. The criteria are OVERDUE (the job's start or end time is overdue) and CRITICAL_PATH (the job is on the critical path). OVERDUE and CRITICAL_PATH are expedite command keywords. It also specifies what action should be performed when the job is expedited.

Applicability

Service Governor only

Where defined

Master

Syntax

```
EXPEDITE name ADD {[OVERDUE|NOOVERDUE]}
                  {[CRITICAL_PATH|NOCRITICAL_PATH]}
                  {ALL|ANY}
                  [CLASS(class)|NOCLASS]
                  [PRIORITY(priority)|NOPRIORITY]
                  [START|NOSTART]
                  [PERFORM(pgn)|NOPERFORM]
                  [SRVCLASS(srvclass)|NOSRVCLASS]
                  [ESP_PRIORITY(nn)|NOESP_PRIORITY]
                  [WOB_PRIORITY(HIGH|ABOVE_NORMAL|NORMAL|
                   BELOW_NORMAL|IDLE)]
```

Operand	Description
name	Specifies a one-to-eight-character expedite policy name. Each character must be either alphanumeric, a national character ($ # @) or an underscore (_). The name cannot be specified as OFF.
ADD	Requests that a new expedite policy be added
OVERDUE	Specifies that, if a job under ESP's control is overdue and all other possible expedite criteria are met, the job may be expedited. OVERDUE is valid for the ADD and ALTER options and ignored in the DELETE and LIST options.

Operand	Description
NOOVERDUE	Specifies that an overdue condition for a job should not be used as criteria to expedite a job. NOOVERDUE is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOOVERDUE is the default.
CRITICAL_PATH	Specifies that, if a job is on the Application's critical path and all other possible expedite criteria are met, the job may be expedited. CRITICAL_PATH is valid for the ADD and ALTER options and ignored in the DELETE and LIST options.
NOCRITICAL_PATH	Specifies that presence of a job on the Application's critical path should not be considered criteria for determining whether a job may be expedited. NOCRITICAL_PATH is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOCRITICAL_PATH is the default.
ALL	Specifies a job will be expedited if a job meets all the criteria specified in the expedite policy. If ALL or ANY are not specified in the ADD option, ALL is the default. ALL is valid for the ADD and ALTER options and ignored in the DELETE and LIST options.
ANY	Specifies a job will be expedited if a job meets any criteria specified in the expedite policy. If ALL or ANY are not specified in the ADD option, ALL is the default. ANY is valid for the ADD and ALTER options and ignored in the DELETE and LIST options.
CLASS(*class*)	Indicates the class that a z/OS job should be changed to if the job is expedited and waiting for execution. You can enter up to eight characters. CLASS is valid for the ADD and ALTER options and ignored in the DELETE and LIST options.
NOCLASS	Specifies the class of a z/OS job that is waiting for execution should not be changed if the job may be expedited. NOCLASS is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOCLASS is the default.
PRIORITY(*priority*)	Specifies the priority of a job that is waiting for execution should be changed if the job is expedited. PRIORITY is valid for the ADD and ALTER options and ignored in the DELETE and LIST options. The value of priority is within the range of 1 to 15.
NOPRIORITY	Specifies the priority of a job that is waiting for execution should not be changed if the job may be expedited. NOPRIORITY is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOPRIORITY is the default.
START	Specifies that a job that is waiting for execution should be unconditionally started if it is expedited. The START option is only supported in OS/390 V2R4 or later for JES2 systems, and OS/390 V2R8 or later for JES3 systems. START requires the system be in IBM WLM goal mode, and the job must be in a WLM-managed job class queue. You can change the job to a WLM-managed job class queue using the CLASS operand.
NOSTART	Specifies that a job that is waiting for execution should not be unconditionally started if it may be expedited. NOSTART is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOSTART is the default.

Operand	Description
PERFORM(*pgn*)	Specifies the performance group of a z/OS job executing on a system in IBM WLM compatibility mode should be changed if the job is expedited. PERFORM is valid for the ADD and ALTER options and ignored in the DELETE and LIST options. If a job is on the local JES node and the expedite policy specifies both the PERFORM and the SRVCLASS operands, the job is also assumed to be on the local sysplex. An IBM WLM mode query is issued to determine if the system the job is on is in IBM WLM compatibility mode or IBM WLM goal mode: • If the system is in compatibility mode, PERFORM is used. • If the system is in goal mode, SRVCLASS is used. If the job is on a remote JES node and the expedite policy specifies both the PERFORM and SRVCLASS operands, SRVCLASS is used.
NOPERFORM	Specifies the performance group of a z/OS job executing on a system in IBM WLM compatibility mode should not be changed if the job may be expedited. NOPERFORM is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOPERFORM is the default.
SRVCLASS(*srvclass*)	Specifies that the service class of a z/OS job executing on a system in IBM WLM goal mode should be changed if the job is expedited. SRVCLASS is valid for the ADD and ALTER options and ignored in the DELETE and LIST options. If a job is on the local JES node and the expedite policy specifies both the PERFORM and the SRVCLASS operands, the job is also assumed to be on the local sysplex. An IBM WLM mode query is issued to determine if the system the job is on is in IBM WLM compatibility mode or IBM WLM goal mode: • If the system is in compatibility mode, PERFORM is used. • If the system is in goal mode, SRVCLASS is used. If the job is on a remote JES node and the expedite policy specifies both the PERFORM and SRVCLASS operands, SRVCLASS is used.
NOSRVCLASS	Specifies the service class of a z/OS job executing on a system in IBM WLM goal mode should not be changed if the job may be expedited. NOSRVCLASS is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOSRVCLASS is the default.

Operand	Description
ESP_PRIORITY(*nn*)	Use this operand for a job that is not yet submitted. The *nn* variable is a number within the range of 1 to 99. ESP_PRIORITY allows for a dynamic change to the ESP priority of a job waiting for ESP resources. This change in ESP priority may occur any time after the ESP Application has been generated and before the job is submitted. All changes to ESP priority are recorded in the audit log. ESP_PRIORITY is valid for the ADD and ALTER options and ignored in the DELETE and LIST options.
NOESP_ PRIORITY	Use this operand for a job that is not yet submitted. Indicates ESP will not change the job ESP priority as part of the expedite policy function. NOESP_PRIORITY is valid for the ALTER option and ignored in the ADD, DELETE, and LIST options. NOESP_PRIORITY is the default.
WOB_PRIORITY (HIGH| ABOVE_NORMAL| NORMAL| BELOW_NORMAL| IDLE)	Applies to ESP System Agent Release 7 only. Use this operand to change the process priority of a UNIX or Windows job that is executing. NORMAL is the default. Specify one of the following process priorities: • HIGH — Processes that must be executed immediately. These processes can use nearly all available CPU time. • ABOVE_NORMAL — Processes that have priority above the normal level, but below the high level • NORMAL — Processes without special scheduling needs. Normal is the default. • BELOW_NORMAL — Processes that have priority above the idle level, but below the normal level • IDLE — Processes that will run only when the system is idle **Note:** • You can increase a UNIX job's process priority only if the job runs on a machine with ESP System Agent started by the root account. If ESP System Agent is not started by root and you specify a higher process priority, the job runs with the current process priority (the priority does not increase) and an error message is recorded in the job's spool file. • If a Windows job is associated with a Windows job object, you can only change the process priority for that job using the Modify Job Object option in CSF. For more information n the Modify Job Object option, see the *ESP System Agent Guide to Scheduling Workload*. • The WOB_PRIORITY operand does not set the process priority for subprocesses on Windows.

Usage notes

The expedite policy specifies how a job's priority should be increased.

The criteria are OVERDUE (the job's start or end time is overdue) and CRITICAL_PATH (the job is on the critical path). OVERDUE and CRITICAL_PATH are expedite initialization parameters operands.

Expedite Actions

The methods of accelerating a job are known as expedite actions.

Expedite actions for jobs that are not running

The following expedite action is available for a job that is not yet submitted.

Option	Action
1	Dynamically changes the ESP Workload Manager priority of a job waiting for ESP Workload Manager resources. This job priority is specified by the ESP_PRIORITY operand.

The following expedite actions are available for submitted jobs that are waiting for execution. In this case, the objective of a job expedite is to initiate the job as soon as possible or to make it run as quickly as possible when it does start.

Option	Action
1	Changes the job's JES execution class, specified by the EXPEDITE CLASS(*class*) operand
2	Changes the job's priority on the JES execution queue, specified by the EXPEDITE PRIORITY(*priority*) operand
3	Starts the job immediately, specified by the EXPEDITE START operand

Expedite actions for jobs that are running

The following expedite actions are available for z/OS jobs that are executing. In this case, the objective of a job expedite is to increase the job's dispatching priority.

Option	Action
1	Changes the job's service class, specified by the EXPEDITE SRVCLASS(*srvclass*) operand. This requires the system be in IBM WLM goal mode.
2	Changes the job's performance group, specified by the EXPEDITE PERFORM(*pgn*) operand. This requires the system be in IBM WLM compatibility mode.

The following expedite action is available for UNIX and Windows jobs that are executing. In this case, the objective of a job expedite is to change the process priority of a UNIX or Windows job.

Option	Action
1	Changes the process priority of a UNIX or Windows job as specified by the EXPEDITE WOB_PRIORITY operand.

Examples

Define an expedite policy for z/OS jobs

The following EXPEDITE command defines an expedite policy called POLICY_1. This command specifies that a job, associated with the expedite policy, should be expedited when the job is overdue. If the job is expedited while it is waiting for execution, its execution queue priority will be set to 15. If the job is expedited while the job is executing, its service class will be changed to JES_FAST.

```
EXPEDITE POLICY_1 ADD OVERDUE PRIORITY(15) SRVCLASS(JES_FAST)
```

If the following command is issued after the previous one, a job associated with expedite policy POLICY_1 will be expedited when the job is overdue and it is on the Application's critical path. If the job is expedited while it is waiting for execution, its job class will be set to F. If F is an IBM WLM-managed job class queue, the job will then start. If the job is expedited while the job is executing, its service class will be changed to JES_FAST.

```
EXPEDITE POLICY_1 ALTER CRITICAL_PATH NOPRIORITY CLASS(F) START
```

The following command has the same effect as issuing the previous two commands in succession:

```
EXPEDITE POLICY_1 ADD OVERDUE CRITICAL_PATH +
CLASS(F) START SRVCLASS(JES_FAST)
```

Define an expedite policy for UNIX or Windows jobs

The following EXPEDITE command defines an expedite policy called POLICY_2. It specifies an executing UNIX or Windows job, associated with the expedite policy, is to be expedited when the job is overdue. The job's process priority is changed to HIGH.

```
EXPEDITE POLICY_2 ADD OVERDUE WOB_PRIORITY(HIGH)
```

Note: If the job is running on a UNIX machine, ESP System Agent must be started by root for the process priority to increase. If the job is running on a Windows machine and is associated with a Windows job object, the process priority does not change. To change the process priority for a job associated with a Windows job object, use the Modify Job Object option in CSF.

Related information

For information on the expedite feature, see the *ESP Workload Manager User's Guide.*

FTP: Select SMF record

Note: You can also use the FTP initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The FTP initialization parameter specifies whether ESP Workload Manager monitors SMF record type 118 or 119 for FTP data set triggering.

Where defined

Master and proxy

Syntax

```
FTP SMF(118|119)
```

Operand	Description
`SMF(118)`	ESP Workload Manager monitors SMF record type 118 for FTP data set triggering. The default is 118.
`SMF(119)`	ESP Workload Manager monitors SMF record type 119 for FTP data set triggering.

GMTCHECK: Check Greenwich Mean Time

Purpose

The GMTCHECK initialization parameter indicates if GMT and the local time are identical on the hardware clock.

Where defined

Master and proxy

Syntax

```
GMTCHECK
```

Usage notes

GMTCHECK has no operands. It is used in installations where the hardware clock is usually set to the local time rather than to true GMT. It indicates if GMT and Local Time are the same as far as the hardware clock is concerned. If there is a discrepancy between the two clocks (for example, after a power-up), ESP Workload Manager issues a warning message and requests confirmation to continue.

HISTFILE: Define or alter a job history data set

Note: You can also issue the HISTFILE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The HISTFILE initialization parameter specifies the job history recording data set's name and other attributes.

Where defined

ESPCOLD data set

Syntax

```
HISTFILE histfid [DSNAME(dsname)]
                 [NEWDSNAME(newdsname)]
                 [BACKUPDSNAME(backupdsname)|NOBACKUPDSNAME]
                 [DEFINE|DELETE|OPEN|CLOSE|SET]
                 [SHR|NOSHR]
                 [JOURNAL|NOJOURNAL]
                 [BACKUPTIME('schedule')|NOBACKUPTIME]
```

Operand	Description
histfid	A unique ID, up to eight characters, that identifies the history data set
DSNAME(*dsname*)	The name of a VSAM KSDS defined with the correct attributes
NEWDSNAME(*newdsname*)	The name of a VSAM KSDS similar to the above
BACKUPDSNAME(*backupdsname*)	The name of a non-VSAM sequential data set
NOBACKUPDSNAME	Removes the BACKUPDSNAME, if it is already specified
DEFINE	Defines a new history data set. DEFINE is the default.
DELETE	Deletes a definition. The associated VSAM data set is not deleted. Only the logical HISTFILE ID is deleted.
OPEN	Requests that a data set is reopened
CLOSE	Closes the history data set and de-allocates it from ESP Workload Manager
SET	Requests that an existing specification is changed without the need to open or close the data set
SHR	The data set is shared.
NOSHR	The data set is not shared. NOSHR is the default.
JOURNAL	A record is written to SMF each time a history record is inserted, updated or deleted.

Operand	Description
NOJOURNAL	No records are written to SMF when the history data set is updated. Use this keyword to cancel a previous JOURNAL request. NOJOURNAL is the default.
BACKUPTIME(*schedule*)	A time at which ESP Workload Manager automatically backs up the history data set. If the text string contains blanks or commas, enclose it within quotation marks.
NOBACKUPTIME	Removes the BACKUPTIME, if it is already specified

Usage notes

Use the HISTFILE initialization parameter to define and initialize a job history data set. Use the DEFINE operand with the DSNAME operand. The identifier (up to eight characters) is the permanent name to be associated with that logical entity.

When a tracking model is defined, a job history data set ID can be specified. This identifies which data set the history data is written to. Using the ID rather than the data set name allows you to change the actual data set name without having to alter all references to it.

Example

In the following example, a new history recording data set called HISTF1 is defined. The actual data set used is GROUP2.HISTFILE and its disposition is shared.

```
HISTFILE HISTF1 DEFINE DSNAME(GROUP2.HISTFILE) SHR
```

HOME: Specify Workstation Server IP address

Purpose

The HOME initialization parameter specifies the IP address of the Workstation Server.

Where defined

WSSPARM data set

Syntax

```
HOME  HOST(ip_address_of_wss)
```

Operand	Description
HOST(*ip_address_of_wss*)	The IP address of the Workstation Server. If HOME is omitted, the Workstation Server calls the GETHOSTID TCP/IP service to determine the Workstation Server IP address.

Equivalent EXEC parameter

The following indicates the equivalent EXEC parameter for the Workstation Server started task definition. For details, see "Workstation Server started task" on page 97.

```
HOME HOST(ip_address_of_wss)
```

Example

```
HOME HOST(128.34.10.3)
```

INDEX: Identify index data set

Purpose

The INDEX initialization parameter specifies the name of the index data set.

Where defined

Master and proxy

Syntax

```
INDEX dsname [SHR|NOSHR]
             [JOURNAL|NOJOURNAL]
             [BACKUPDATASET(bkupdsn)]
             [NOREADONLY|READONLY]
```

Operand	Description
dsname	The name of the index data set
SHR	The data set is shared and RESERVE/DEQ is performed.
NOSHR	The data set is not to be shared. NOSHR is the default.
JOURNAL	A record is written to SMF each time an update, addition or deletion occurs to the index data set. For further details, see “SMFREC: Specify SMF record number” on page 485.
NOJOURNAL	No records are written to SMF when the index data set is updated. Use this keyword to cancel a previous JOURNAL request. NOJOURNAL is the default.
BACKUPDATASET(*bkupdsn*)	The name of a non-VSAM data set the index data set is backed up to
NOREADONLY	Allows ESP Workload Manager to update the index data set. NOREADONLY is the default.
READONLY	Prevents ESP Workload Manager from updating the index data set. The READONLY operand should be used only on proxy systems. If READONLY is assigned on the master system but not on the proxy systems, data set integrity might be violated. If READONLY is assigned on all systems, ESP Workload Manager cannot function properly: no system can update the data set.

Usage notes

You can set the automatic backup of the index data set. For instructions on how to set the index data set’s automatic back up, see “BKUPINDX: Back up INDEX data set” on page 257. You can also use the BKUPINDX command.

On master and proxy systems

For installations with a master and proxy environment, the proxy processors can update the index data set. If you create these data sets without the SHR keyword on the IDCAMS DEFINE statement, the proxies might try to update these data sets without proper serialization. READONLY allows full sharing of ESP Workload Manager control data sets in a master and proxy environment where the use of normal VSAM share options might pose operational difficulties, but data set integrity must be preserved.

Example

The following example specifies a shared index data set called ESP.INDEX. A backup data set, ESP.BACKUP.INDEX, is also specified.

```
INDEX ESP.INDEX SHR BACKUPDATASET(ESP.BKUP.INDEX)
```

INET: Start or set TCP/IP tracing

Note: You can also issue the INET initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The INET initialization parameter sets the sysout class for a TCP/IP trace and to start TCP/IP tracing. TCP/IP tracing is for diagnostic purposes only and is not usually started.

Where defined

Master

WSSPARM data set

Syntax

```
INET {SET} {TRACE} SYSOUT(class)
           {CLOSE {RESET|SHUTDOWN}}
           {HOST_CACHE TTL(seconds)}
     {START|S} TRACE|TR
```

Operand	Description
SET	Sets a TCP/IP object
TRACE	Sets or starts the tracing facility.
SYSOUT(*class*)	The print data set for the TRACE log is sysout. *class* is the sysout class. Specify a single character from A to Z, or 0 to 9.
CLOSE	Use this operand only under the direction of CA Technical Support.
RESET	ESP Workload Manager and Workstation Server close a TCP/IP connection forcefully.
SHUTDOWN	ESP Workload Manager and Workstation Server close a TCP/IP connection gracefully. SHUTDOWN is the default.
HOST_CACHE	Activates or de-activates TCP/IP host caching, or resets the time-to-live (TTL) interval
TTL(*seconds*)	The time-to-live value from 0 to 99999999. A value of 0 de-activates TCP/IP host caching.
START	Starts TCP/IP tracing. If specified, it must be preceded by an INET SET SYSOUT(*class*) statement.

Equivalent EXEC parameter

The following table indicates the equivalence between WSSPARM and EXEC for the definition of the Workstation Server started task. For details, see "Workstation Server started task" on page 97.

EXEC parameters	WSSPARM parameters

SET_TCPTRACE(*class*)
or
T_TCPTR(*class*)
	INET SET TRACE SYSOUT(*class*)

START_TCPTRACE(*class*)
or
S_TCPTR(*class*)
	INET SET TRACE SYSOUT(*class*)
	INET START TRACE
	Note: Two parameter lines are required.

SET_TCPCLOSE(RESET | SHUTDOWN)
or
T_TCPCL(RESET | SHUTDOWN)
	INET SET CLOSE RESET | SHUTDOWN

Usage notes

TCP/IP tracing cannot be started unless the sysout class is set. We recommend that you include an INET SET SYSOUT(*class*) initialization parameter in ESPPARM so TCP/IP tracing can be started with one command (INET START TRACE) during ESP Workload Manager execution.

For details on the use of the INET command during ESP Workload Manager execution, see the *ESP Workload Manager Reference Guide.*

Example

The following example sets the TCP/IP trace sysout class to A:

```
INET SET TRACE SYSOUT(A)
```

INFOCOMM: Control transaction server

Note: You can also issue the INFOCOMM initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The INFOCOMM initialization parameter controls transaction servers used in sending Infoserv requests.

Where defined

Master

Syntax

```
INFOCOMM {DEFINE|DISPLAY|DELETE}
         [serverid]
         [CLIENT(clientname)]
         [TPAPPL(applname)]
```

Operand	Description
DEFINE	Requests that a new transaction server be added
DISPLAY	Requests that information on all (or specific) servers for the current Application be displayed. Information displayed includes server ID, client name, and TP application name.
DELETE	Requests the deletion of a transaction server
serverid	The logical server identifier, which is up to 25 characters. This identifier will be specified on the INFOSERV command when sending a request.
CLIENT(*clientname*)	The client name for your Application. This client must be defined to Infoserv using the CLIDEF command. The name is up to eight characters.
TPALPPL(*applname*)	The TP application name of the Infoserv task this transaction server will send its requests to. *applname* is defined in the Infoserv TP parameters data set. It contains up to 44 characters.

Usage notes

For each client defined to Infoserv you must define a transaction server that will transmit requests to Infoserv through the LU6.2 Communications Facility (TP Server).

For more information, see the ESP InfoServ documentation.

Examples

In the following example, a new transaction server called INFO1 is defined. The client name for this Application is ESP and must be defined to Infoserv using the CLIDEF command. The INFO1 transaction server will send requests to the INFOSERV_MAIN TP application.

```
INFOCOMM DEFINE INFO1 CLIENT(ESP) TPAPPL(INFOSERV_MAIN)
```

To display all existing transaction servers, type

```
INFOCOMM DISPLAY
```

To delete an existing transaction server, type

```
INFOCOMM DELETE INFO1
```

ISCDEF: Identify current node name

Note: You can also issue the ISCDEF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The ISCDEF initialization parameter identifies the current node name for inter-system tracking when NJE is used as the communications method.

Where defined

Master

Syntax

```
ISCDEF NODE(nodename) [CLASS(sysoutclass)]
                      [RECEIVER(receivername)]
```

Operand	Description
NODE(*nodename*)	The current node name in one to eight characters, corresponding to the JES node name
CLASS(*sysoutclass*)	The one-character sysout class used to store tracking information
RECEIVER(*receivername*)	The receiver name that indicates which ESP Workload Manager master receives the tracking information. This name should be unique within the node and must be a one-to-eight-character alphanumeric name. For example, you could use the ESP Workload Manager subsystem name.

Usage notes

A sysout class and external writer name must be specified only if this node is to receive tracking data. The sysout class does not have to be unique to ESP Workload Manager, but the external writer name should be unique within the node. We recommend that you use the ESP Workload Manager subsystem name.

If this system is sending tracking information, this initialization parameter must precede the ISCXMTR and XMITMDL initialization parameters. For more information, see “ISCXMTR: Specify node to receive tracking data” on page 372 and “XMITMDL: Identify tracking models to transmit” on page 561.

Examples

This example identifies the JES node name as NODE3 for an ESP Workload Manager subsystem that is not to receive inter-system tracking data:

```
ISCDEF NODE(NODE3)
```

This example identifies the JES node name as NODE2. This node is to receive tracking data. A sysout class of N is specified and an external writer name of ESPX (corresponding to the ESP Workload Manager subsystem name) is used.

```
ISCDEF NODE(NODE2) CLASS(N) RECEIVER(ESPX)
```

This example illustrates the coding of the ICSDEF and ISCXMTR initialization parameters for two communicating nodes and their relationship. The example shows only the NODE and the RECEIVER operands.

Node A initialization parameters

```
ISCDEF NODE(nodeAname)   RECEIVER(nodeAreceivername)
ISCXMTR NODE(nodeBname)  RECEIVER(nodeBreceivername)
```

Node B initialization parameters

```
ISCDEF NODE(nodeBname)   RECEIVER(nodeBreceivername)
ISCXMTR NODE(nodeAname)  RECEIVER(nodeAreceivername)
```

ISCOPT: Set internodal communication option

Purpose

The ISCOPT initialization parameter specifies which communication method (for example, NJE or LU 6.2) transmits internodal tracking information.

Where defined

Master

Syntax

```
ISCOPT [NJE|LU62|NETWORK [SENDNET(ORIGNODE|ALL)]]
       [ZONECON|NOZONECON]
```

Operand	Description
NJE	JES2 NJE is used for internodal communication. NJE is the default.
LU62	TP Server (using LU 6.2) is used for internodal communication **Note:** TP Server sends tracking records to all nodes.
NETWORK	Only Network Delivery Services (NDS) is used for internodal job tracking. You can use NDS in addition to NJE or TP Server by specifying NJE or LU62. If you use ESP Infoserv, you must use the LU62 operand. **Note:** If you use NETWORK, all XMITMDL ADD commands must specify the NETNODE(*netnode*) operand.
SENDNET(ORIGNODE)	Job-tracking records are sent to the job submission JES node only. ORIGNODE is the default for NDS. **Note:** SENDNET(ORIGNODE) applies only to NDS internodal tracking models.
SENDNET(ALL)	Job-tracking records are sent to all the network nodes specified in the internodal tracking model.
ZONECON	ESP Workload Manager converts times to the destination time zone. ZONECON is the default.
NOZONECON	ESP Workload Manager does not convert times to the destination time zone. **Note:** Use this operand only on the advice of CA Technical Support.

Usage notes

Using any tracking method, the XMITMDL command or initialization parameter is used to filter which job-tracking information is sent.

For LU6.2, you also need to define a TPPARM data set to specify communication parameters.

For NJE, the ISCDEF and ISCXMTR statements might have to be specified; they are not applicable if LU6.2 is being used.

For NETWORK, we strongly recommend that you specify SENDNET(ORIGNODE) or that you let ISCOPT default to this value; otherwise internodal job-tracking records will reach network nodes that are usually not interested in or not authorized to receive this information.

INFOSERV users who use NDS for internodal job tracking must specify LU62 because INFOSERV is only supported by TP Server. They must specify NETNODE(*netnode*) in the XMITMDL ADD command or initialization parameter.

Example

In the following example, LU 6.2 is specified as the communications method for internodal tracking:

```
ISCOPT LU62
```

ISCXMTR: Specify node to receive tracking data

Note: You can also issue the ISCXMTR initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The ISCXMTR initialization parameter specifies the node name, sysout class, and external writer name for each node that is to receive tracking data from the current system when NJE is used as the communications method for inter-system tracking.

Where defined

Master

Syntax

```
ISCXMTR id {NODE(nodename)}
           {CLASS(sysoutclass)}
           {RECEIVER(receivername)}
           [TRANSACTION(numt|100)]
           [INTERVAL(nums|120)]
           [SPINM]
```

Operand	Description
id	A logical identifier. It allows tracking information to be sent to more than one ESP Workload Manager subsystem on the same node. The identifier contains up to 16 characters.
NODE(*nodename*)	The name of NJE node. It contains up to eight characters.
CLASS(*sysoutclass*)	The sysout class name. It contains one character.
RECEIVER(*receivername*)	A receiver name to indicate from which ESP Workload Manager master this master receives the tracking information. This name must be unique within the node, alphanumeric, and up to eight characters long. For example, use the ESP Workload Manager subsystem name.
TRANSACTION(*numt*)	The number of transactions to accumulate before transmitting. The default is 100.
INTERVAL(*nums*)	The number of seconds between transmissions. The default is 120.
SPINM	Sends a message to the console each time transactions are sent

Usage notes

The ISCXMTR initialization parameter is not applicable if ISCOPT LU62 is specified.

This initialization parameter is included for each system sending tracking information, and must be preceded by an ISCDEF statement. Use one ISCXMTR initialization parameter for each node to receive tracking data from this system.

A transaction is defined as an individual tracking record that can be a job start, step end or job end. ESP Workload Manager batches transactions, writes the group to a sysout file, and starts accumulating a new batch. You can define thresholds at which the batches are written. The thresholds are a combination of transaction count and time. For example, you can request that data be transmitted when 100 transactions have accumulated or when two minutes have elapsed, whichever comes first. However, if two minutes elapse with no transactions generated, no data is transmitted.

The XMITMDL initialization parameter defines the sent tracking data.

Examples

In this example, transactions are sent to NODE1 every 180 seconds or after 100 transactions, whichever comes first. The sysout class is N and the external writer name is ESPA. A message is issued whenever a batch of transactions is sent.

```
ISCXMTR N1 NODE(NODE1) CLASS(N) RECEIVER(ESPA) +
INTERVAL(180) TRANSACTION(100) SPINM
```

In this example, transactions are sent to two different master ESP Workload Manager subsystems on NODE1 using a sysout class of N. Transactions are sent to ESPP every 180 seconds or after 100 transactions, whichever comes first; transactions are sent to ESPT every 1800 seconds or after 50 transactions, whichever comes first. One or more XMITMDL initialization parameters define the transmitted data.

```
ISCXMTR N1P NODE(NODE1) CLASS(N) RECEIVER(ESPP) +
INTERVAL(180) TRANSACTION(100)
ISCXMTR N1T NODE(NODE1) CLASS(N) RECEIVER(ESPT) + INTERVAL(1800)
TRANSACTION(50)
```

This example illustrates how to code the ICSDEF and ISCXMTR initialization parameters for two communicating nodes and their relationship. The example shows only the NODE and the RECEIVER operands.

Node A initialization parameters

```
ISCDEF NODE(nodeAname)   RECEIVER(nodeAreceivername)
ISCXMTR NODE(nodeBname)  RECEIVER(nodeBreceivername)
```

Node B initialization parameters

```
ISCDEF NODE(nodeBname)   RECEIVER(nodeBreceivername)
ISCXMTR NODE(nodeAname)  RECEIVER(nodeAreceivername)
```

JESCOMCH: Specify JES command prefix

Note: You can also issue the JESCOMCH initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The JESCOMCH initialization parameter specifies the character used to prefix JES commands.

Where defined

Master and proxy

Syntax

```
JESCOMCH character
```

Operand	Description
character	The character used as a JES command prefix. If you use a punctuation character (such as a semi-colon), enclose the punctuation in single quotation marks. The dollar sign ($) is the default character.

Usage notes

If your installation uses a character other than the dollar sign to prefix JES commands, use the JESCOMCH initialization parameter. Ensure you use single quotation marks to enclose the JES command prefix if it is punctuation.

Example

The following example specifies that the number sign (#) is used to prefix JES commands:

```
JESCOMCH '#'
```

JESNAME: Specify job entry system

Note: You can also issue the JESNAME initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The JESNAME initialization parameter specifies the name of the job entry system that tracking is kept for.

Where defined

Master and proxy

Syntax

```
JESNAME name|NONE
```

Operand	Description
name	Specify the name of the JES tracking is kept for. Enter one to four characters, the first alphabetical, the remaining characters alphanumeric.
NONE	Remove the current name.

Usage notes

Use the JESNAME initialization parameter if more than one JES might exist on a single z/OS image. It prevents work running on a secondary JES from influencing work on the primary. However, its use also prevents the monitoring of started tasks (STCs) and TSO sessions (TSUs).

Example

```
JESNAME SYSA
```

JESTYPE: Specify Type of JES

Note: You can also issue the JESTYPE initialization parameter is also a command. Refer to the *ESP Workload Manager Reference Guide* for details about command syntax and operands.

Purpose

The JESTYPE initialization parameter specifies the type of JES your installation uses.

Where defined

Master and proxy

Syntax

```
JESTYPE [2|3]
        [DJC|NODJC]
```

Operand	Description
2	Signifies that JES2 is in use
3	Signifies that JES3 is in use
DJC	Specifies normal processing for DJC networks
NODJC	Specifies that existing DJC networks should be processed as Applications

Usage notes

This initialization parameter enables ESP Workload Manager to detect differences in processing requirements for the JES3 subsystems compared with JES2. You might use the NODJC operand to convert existing DJC networks to Applications without the need to modify individual ESP Workload Manager Procedures.

If JESTYPE is not coded, ESP Workload Manager uses the type given by the JES that ESP Workload Manager runs under (Release 4.4.2 and up). ESP Workload Manager 5.1 and up ignores the 2 or 3 operands, but must be coded if DJC or NODJC is coded.

Example

To signify that the JES2 subsystem is in use, and that existing DJC networks should be processed as Applications, type

```
JESTYPE 2 NODJC
```

JOBINDEX: Jobindex data set

Purpose

The JOBINDEX initialization parameter specifies the name of the jobindex data set.

Where defined

Master and proxy

Syntax

```
JOBINDEX dsname [SHR|NOSHR]
                [JOURNAL|NOJOURNAL]
                [BACKUPDATASET(bkupdsn)]
                [NOREADONLY|READONLY]
```

Operand	Description
dsname	The name of the jobindex data set
SHR	The data set is shared and RESERVE/DEQ is performed.
NOSHR	The data set is not to be shared. NOSHR is the default.
JOURNAL	A record is written to SMF each time an update, addition or deletion occurs to the jobindex data set. For details, see "SMFREC: Specify SMF record number" on page 485.
NOJOURNAL	No records are written to SMF. Use this keyword to cancel a previous JOURNAL request. NOJOURNAL is the default.
BACKUPDATASET(*bkupdsn*)	The name of a non-VSAM data set to which the jobindex data set is backed up
NOREADONLY	Allows ESP Workload Manager to update the jobindex data set. NOREADONLY is the default.
READONLY	Prevents ESP Workload Manager from updating the jobindex data set

Usage notes

To set the jobindex data set's automatic backup, follow the instructions in "BKUPJNDX: Back up JOBINDEX data set" on page 258. You can also use the BKUPJNDX command.

On master and proxy systems

For installations with an ESP Workload Manager master and proxy environment, the proxy processors can update the jobindex data set. If you create these data sets without the SHR keyword on the IDCAMS DEFINE statement, the proxies might try to update these data sets without proper serialization. READONLY allows full sharing of

ESP Workload Manager control data sets in an ESP Workload Manager master and proxy environment where the use of normal VSAM share options might pose operational difficulties, but data set integrity must be preserved.

Note: Use the READONLY initialization parameter only on proxy systems, not on the master system.

On proxy that generates Applications

On a proxy processor that generates Applications, you must specify the SHR option:

```
JOBINDEX PROXY.JOB1 SHR
```

Example

```
JOBINDEX ESP.JOBINDEX SHR BACKUPDATASET(ESP.BKUP.JOBINDEX)
```

JOBPROF: Defining a Job Profile Name

Purpose

Use the JOBPROF initialization parameter to specify a profile name for job profile data, and to indicate what data is stored in the profile.

You can also use JOBPROF as a statement to specify profile names at the Application-level, or at the job-level for job profile data. For information, see the JOBPROF statement in the *ESP Workload Manager Reference Guide*.

Where defined

Master

Syntax

```
JOBPROF profilename ON(schedule criteria)
```

Operand	Description
profilename	Specifies a name for job profile data. The profile name can be up to 104 characters long. You can use symbolic variables, or a text string within the profile name. **Note:** If the JOBPROF initialization parameter is omitted, by default ESP Workload Manager collects job statistics using the full job name.
schedule criteria	Indicates ESP Workload Manager stores data in the job profile name for the specified run. Use any valid ESP Workload Manager schedule criteria that resolves to a single date.

Usage notes

While the Application is being generated, ESP Workload Manager records the profile name of each job in the Application, and creates an entry for each job in the jobstats data set.

By default, 10 entries are kept in the jobstats data set for each profile. You can change this system setting in the JOBSTATS initialization parameter using the PINDEX operand.

During job execution, job information is saved in the jobstats data set under the profile name of each job.

The following job information is stored:

- CPU time
- Expected elapsed execution time
- Default resources
- Number of specific and non-specific tape cartridges required
- Number of print lines produced

This job information enhances ESP Workload Manager's anticipated end-time, default resource, and critical-path calculations.

Symbolic variables

The following symbolic variables are available with the JOBPROF statement:

- ESPAPPL
- ESPCUFULLNAME
- ESPCUJOB
- ESPCUQUAL

The following symbolic variables are not available with the JOBPROF statement because the profile name resolves while the Application is being generated:

- ESPAPQUAL
- ESPAPJOB
- ESPJQUAL
- ESPJOBN
- ESPAPFULLNAME
- ESPFULLNAME

Multiple JOBPROF statements

You can use multiple JOBPROF statements in a job, in an Application, or as initialization parameters. When multiple JOBPROF statements are present, ESP Workload Manager selects the last statement that matches the schedule criteria of the job, Application, or initialization parameter. You have to specify the generic statement first, followed by the more specific statements.

Examples

Using the JOBPROF initialization parameter

The following initialization parameter defines a job profile name using the full name, the Application name, and the text string ALL. If the full name is JOBA.QUAL1 and the Application name is APPL1, the job profile name for this job is JOBA.QUAL1.APPL1.ALL.

```
JOBPROF %ESPCUFULLNAME..%ESPAPPL..ALL ON('ANYDAY')
```

Using multiple JOBPROF initialization parameters

The following example shows two JOBPROF initialization parameters. ESP Workload Manager selects the last initialization parameter that matches the job's schedule criteria. You have to specify the generic statement first, followed by the more specific statements.

```
JOBPROF %ESPCUFULLNAME..%ESPAPPL..ALL ON('ANYDAY')
JOBPROF %ESPCUFULLNAME..%ESPAPPL..FRIDAY ON('FRIDAY')
```

If the full job name is JOBA.QUAL1 and the Application name is APPL1, ESP Workload Manager stores the job profile data under the profile name of JOBA.QUAL1.APPL1.ALL when the job runs on any day other than Friday.

When JOBA.QUAL1 runs on Friday, ESP Workload Manager stores the job profile data under the profile name of JOBA.QUAL1.APPL1.FRIDAY.

Related information

Job profiling is supported in the following commands. The commands are in alphabetical order in the *ESP Workload Manager Reference Guide*.

- APPLJOB(AJ) Command: Controlling Jobs and subApplications
- APPLINS Command: Insert Commands and Statements
- LJS Command: List Job or Job Profile Statistics
- MGRMSG Command: Send Messages to the Manager

JOBSTATS: Jobstats data set

Purpose

The JOBSTATS initialization parameter specifies the name of the jobstats data set. The jobstats data set contains job statistics, job-profiling information, and the number of instances of each record type to keep.

Where defined

Master

Syntax

```
JOBSTATS dsname [BACKUPDATASETNAME(dsn)]
                [JINDEX(jobStatsIndexCount)]
                [PINDEX(jobProfileIndexCount)]
                [JOURNAL|NOJOURNAL]
                [READONLY|NOREADONLY]
```

Operand	Description
dsname	Indicates the name of the jobstats data set.
BACKUPDATASETNAME(*dsn*)	Indicates the name of a non-VSAM, sequential data set to which the jobstats data set is backed up.
JINDEX(*jobStatsIndexCount*)	Indicates the number of job statistic entries that are to remain in the jobstats data set; the default is 10.
PINDEX(*jobProfileIndexCount*)	Indicates the number of job profile entries that are to remain in the jobstats data set; the default is 10.
JOURNAL	Indicates a record is written to SMF each time an update, addition or deletion occurs to the jobstats data set.
NOJOURNAL	Indicates no records are written to SMF. This is the default.
READONLY	Prevents ESP Workload Manager from updating the jobstats data set.
NOREADONLY	Allows ESP Workload Manager to update the jobstats data set. This is the default.

Usage notes

You can use the DFLTDSN initialization parameter to define the jobstats data set. For information, see “DFLTDSN: Specify Parameters for Data Sets Used by ESP Workload Manager” on page 285.

You can use the BKUPJSTS initialization parameter to set the time schedule for the jobstats data set backup. For information, see “BKUPJSTS: Back up the jobstats data set” on page 259.

Examples

In the following example, the jobstats data set is ESP.PROD.JOBSTATS and it is backed up to data set ESP.BKUP.JOBSTATS.

```
JOBSTATS ESP.PROD.JOBSTATS BACKUPDATASETNAME(ESP.BKUP.JOBSTATS)
```

The following example indicates 50 job statistic entries and 25 job profile entries will remain in the jobstats data set, and no records are written to SMF when the jobstats data set is updated.

```
JOBSTATS ESP.PROD.JOBSTATS JINDEX(50) PINDEX(25) NOJOURNAL
```

JTPEXCL: Exclude program from job tracking

Note: You can also issue the JTPEXCL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The JTPEXCL initialization parameter gives you a way to exclude a named program from affecting the job-tracking completion status.

Where defined

Master and proxy

Syntax

```
JTPEXCL [pgmname]
        (pgmname[,pgmname]...)
        [ADD|DELETE]
```

Operand	Description
pgmname	A valid program name
ADD	Adds new information to the specification for program exclusion. ADD is the default.
DELETE	Deletes excluded programs from a previous JTPEXCL definition

Usage notes

Sometimes the last step in a job executes regardless of whether other steps in the job have executed (for example, COND=EVEN). This step usually performs some cleanup activity. The JTPEXCL initialization parameter allows you to exclude the name of the program that this last step executes so it will not affect the job-tracking completion status.

Examples

To exclude the program CLEANUP from job-tracking completion status, type

```
JTPEXCL CLEANUP
```

To exclude the programs CLEANUP and DUMMY from job-tracking completion status, type

```
JTPEXCL (CLEANUP,DUMMY)
```

LOADAGDF: Load Agent definitions

Note: You can also issue the LOADAGDF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The LOADAGDF initialization parameter loads the Agent definition data set during ESP Workload Manager initialization.

Where defined

Master

Syntax

```
LOADAGDF dsname [TEST]
```

Operand	Description
dsname	The name of a sequential data set or a member of a partitioned data set. Must have LRECL=80.
TEST	All syntax and logical checks are performed, but the table is not loaded.

Example

The following example loads the Agent definition data set CYB1.ES00.PARMLIB(AGENTDEF):

```
LOADAGDF CYB1.ES00.PARMLIB(AGENTDEF)
```

LOADJTDT: Load job-tracking definition table

Note: You can also issue the LOADJTDT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The LOADJTDT initialization parameter loads a job-tracking definition table.

Where defined

Master and proxy

Syntax

```
LOADJTDT dsname
```

Operand	Description
dsname	The name of a sequential data set or PDS member containing the job-tracking definition table

Usage notes

The job-tracking definition table identifies the characteristics of jobs you want ESP Workload Manager to track.

Example

In the following example, the job-tracking definition table is called PROD.ESP.DATA(TRAKTAB):

```
LOADJTDT 'PROD.ESP.DATA(TRAKTAB)'
```

LOADNET: Specify LOADNET data set

Purpose

The LOADNET initialization parameter specifies the name of the LOADNET data set that contains all the Network Delivery Services (NDS) initialization parameters.

Where defined

Master

Syntax

```
LOADNET datasetname
```

Operand	Description
datasetname	The name of the LOADNET initialization data set

Usage notes

If you specify the LOADNET initialization parameter, the NDS program is loaded into virtual storage and the target LOADNET data set's initialization parameters are read and processed.

Example

```
LOADNET CYBER.ESPPARM(NETWORK)
```

LOADNL: Load MAILLIST data set

Note: You can also issue the LOADNL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The LOADNL initialization parameter loads the MAILLIST data set.

Where defined

Master and proxy

Syntax

```
LOADNL dsname
```

Operand	Description
dsname	The name of the MAILLIST data set

Usage note

For information on the MAILLIST data set, see to “MAILLIST data set” on page 15.

LOADSCHF: Load schedule data set into CSF

Note: You can also issue the LOADSCHF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The LOADSCHF initialization parameter copies schedule information from a sequential work data set that a SADGEN batch job initializes to the schedule data set (SCHDFILE) that you can view from CSF.

Where defined

Master

Syntax

```
LOADSCHF
```

Usage notes

The LOADSCHF initialization parameter allows you to see future, scheduled workload from CSF. To use LOADSCHF, you must allocate a SCHDFILE data set by uncommenting SCHDFILE in the CYBESS01 sample member.

The LOADSCHF initialization parameter loads the schedule work data set into memory and copies it into the schedule data set (SCHDFILE), both of which the SCHDFILE initialization parameter identifies. Information in the work data set merges with old information.

Prior to loading, execute the SADGEN command in a batch job to populate the work data set with scheduled Events.

Example

Populating the work data set

The JCL below shows a sample job for populating the work data set:

```
//JSSADG  JOB PROD1,'JANE DOE',CLASS=J,MSGCLASS=A,
//         NOTIFY=&SYSUID
//STEP1  EXEC ESPJS55,PARM='SAD'
//STEPLIB DD DSN=PROD1.JOHN550.SSCPLINK,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  SADGEN DATASET('PROD1.JOHN550.WORKFILE.TEST') -
  FROM('11.00 TODAY') -
  TO('23.59 TODAY PLUS 7 WORKDAYS') -
  EVENTSET(-) LEVEL(PROD-) THRESH(0)
```

Loading the schedule data set

The following LOADSCHF initialization parameter merges the content of the work data set with the existing content of the schedule data set. ESP Workload Manager then loads the schedule data set into the CSF scoreboard. The SCHDFILE initialization parameter identifies the work data set and the schedule data set.

```
LOADSCHF
```

Viewing the schedule data in CSF

After ESP Workload Manager loads the schedule data set, the schedule data appears in CSF as follows:

- The processing node (pnode) field is set to “SCHED”.
- The Job Status field is set to “SCHEDULED”.
- The SCHDFILE field is set to “Yes” to indicate that the workload object has a record in the schedule data set.
- The Scheduled Time field shows when the workload object will run.

The following shows a sample CSF panel:

```
NWP5 Consolidated Status: View NWP1
-------------------------------------------------------------------------
COMMAND ===>

    Job Name  Scheduled  SCHDFILE ApplName Gen# P Node    Job Status
___ NWPEVR15  12.00 FRI  Yes      APPEVR15    5 COMPLETE  COMPLETED AT  12.48 17 FEB
___ NWPEVR15  11.39 SAT  Yes      APPEVR15    - SCHED     SCHEDULED
```

Related information

For information on CSF, see the *ESP Workload Manager Operator’s Guide*.

For information on purging jobs in completed Applications from the scoreboard and the schedule data set, see the PURGSCHF command in the *ESP Workload Manager Reference Guide*.

For information on generating a scheduled activity data set, see the SADGEN command in the *ESP Workload Manager Reference Guide*.

LOADUPDT: Load user profile definition table

Note: You can also issue the LOADUPDT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The LOADUPDT initialization parameter instructs ESP Workload Manager to load a data set as the current user profile definition table for use with host security.

Where defined

Master and proxy

Syntax

```
LOADUPDT dsname
```

Operand	Description
dsname	A fully qualified data set name that can include a member name. The data set can be any fixed-length or variable-length data set with a record length of 80 or greater.

Usage notes

The user profile definition table contains one or more PROFILE statements.

Example

To load the user profile definition table from ESP.USERPROF(UPDT), type

```
LOADUPDT ESP.USERPROF(UPDT)
```

LOCAPPL: Define VTAM information

Purpose

The LOCAPPL initialization parameter defines the local application and defines the basic VTAM information necessary to initiate and receive session information.

Where defined

TPPARM data set

Syntax

```
LOCAPPL applname [LUNAME(luname)]
                 [LOGMODE(logonmode)]
                 [SESSLIM(2|n)]
```

Operand	Description
applname	The local application name. The maximum length is 44 alphanumeric characters. The first character must be alphabetic.
LUNAME(*luname*)	The name of the VTAM LU
LOGMODE(*logonmode*)	The logon mode for this application. CYBTPLOG is the default.
SESSLIM(*n*)	The number of sessions established for this application. The default is 2.

Example — Identifying local TP Server

```
LOCAPPL ESP_MONTREAL LUNAME(ESPMTL) LOGMODE(MONTLU62)- SESSLIM(2)
```

LOG: Control Message Logging for a Mailbox

Purpose

The LOG initialization parameter controls whether messages to a mailbox are written to the mail log.

Where defined

Master and proxy

MAILLIST data set

Syntax

```
LOG [ON|OFF]
```

Operand	Description
ON	Write mailbox messages to the mail log. ON is the default.
OFF	Do not write mailbox messages to the mail log

Usage notes

If the MAILLOG initialization parameter is not set to ENABLE or ON, the LOG parameter is ignored. For more information on the MAILLOG initialization parameter, see “MAILLOG: Control the Mail Log Sysout” on page 398.

If the LOG initialization parameter is not specified, messages are only written to the mail log if the MAILLOG initialization parameter is set to ON.

For information on the MAILLIST data set, see “MAILLIST Data Set” on page 49.

The LOG initialization parameter is included in the MAILLIST data set as follows:

```
SMTPPARM CLASS(A) JOBNAME(SMTP)

MAILBOX PAYROLL MAXLINES(0)
TSOUSER CYBPAY1 SYSID(SYSC)
EMAIL paymaster@company.com
LOG ON

MAILBOX CYBACCOUNTING MAXLINES(300)
TSOUSER (CYBACC1 CYBACC2) SYSID(SYSA)
LOG OFF
```

LOGTRACE: Consolidate Trace Data

Note: You can also issue the LOGTRACE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The LOGTRACE initialization parameter consolidates certain trace data into a single sysout data set. Currently, LOGTRACE consolidates trace data produced by the following commands or initialization parameters:

- XDAB
- XFRB

Where defined

Master, proxy or both

ESPPARM data set

Syntax

```
LOGTRACE SET|T SYSOUT(class)
```

```
LOGTRACE START|S
```

Operand	Description
SET|T SYSOUT(*class*)	Set the sysout class for trace data to *class*
START|S	Start consolidating trace data

Usage notes

Setting up trace data consolidation

Before you start trace data consolidation, you must set the sysout class for the trace data. For example,

```
LOGTRACE SET SYSOUT(A)
```

To include data from a particular trace in the LOGTRACE consolidation, you must specify the LOGTRACE operand in the command or initialization parameter that sets up that trace. For example, to include VSAM I/O event trace data in the consolidation, you code the following:

```
XDAB SET TRACE LOGTRACE
```

Starting trace data consolidation

Data for a trace is not generated and is not consolidated with data from other traces until you do the following:

- Start the consolidation facility with the LOGTRACE initialization parameter or command.
- Start the trace with the initialization parameter or command for that trace.

For example,

```
LOGTRACE START
XDAB START TRACE
```

Alternatives to the LOGTRACE initialization parameter

Instead of coding the LOGTRACE initialization parameter, you can:

- Code the LOGTRACE parameter in the START command.
- Issue the LOGTRACE command.

Related information

See also

- The LOGTRACE parameter in the START command in the *ESP Workload Manager Operator's Guide*.
- The LOGTRACE command in the *ESP Workload Manager Reference Guide*.

Example

Set the trace data SYSOUT class

The following initialization parameter sets the sysout class for trace data to A.

```
LOGTRACE SET SYSOUT(A)
```

Start consolidating trace data

The following initialization parameter starts consolidating trace data to the sysout data set.

```
LOGTRACE START
```

MAILBOX: Set up mailboxes

Purpose

The MAILBOX initialization parameter specifies the name of a mailbox. You can use mailboxes to set the recipients of messages that Events and NOTIFY commands send. The PROFILE initialization parameter specifies one mailbox for the user whose profile is defined. This mailbox is the default mailbox to be used for messages when an Event does not specify a mailbox.

Where defined

Master and proxy

MAILLIST data set

Syntax

```
MAILBOX box_name MAXLINES(nnn|0)
```

Operand	Description
box_name	The name of the mailbox. Mailbox names can have up to 128 characters. Spaces, commas, and wildcard characters (asterisk or hyphen) are not allowed.
MAXLINES(*nnn*)	The maximum number of lines that an Event can send to the mailbox. ESP Workload Manager ignores any lines above the maximum. The default is 0 and means no limit. **Note:** Each ESP Workload Manager message is considered a single line.

Usage notes

For information on the MAILLIST data set, see “MAILLIST data set” on page 15.

The MAILBOX initialization parameter is included in the MAILLIST data set as follows:

```
SMTPPARM CLASS(A) JOBNAME(SMTP)
MAILBOX  PAYROLL MAXLINES(0)
TSOUSER CYBPAY1 SYSID(SYSC)
EMAIL paymaster@company.com
EMAIL payservice@payservice.com
MAILBOX CYBACCOUNTING MAXLINES(300)
TSOUSER (CYBACC1 CYBACC2) SYSID(SYSA)
```

MAILLOG: Control the Mail Log Sysout

Note: You can also issue the MAILLOG initialization parameter as a command. For information about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The MAILLOG initialization parameter sets the attributes of the mail log sysout.

Where defined

Master and proxy

Syntax

```
MAILLOG [ENABLE|ON|DISABLE] [SPINCOUNT(x)] CLASS(x)
```

Operand	Description
ENABLE	If a mailbox is specified in the Event definition, that mailbox's LOG parameter controls whether messages are written to the mail log. If no mailbox is specified in the Event definition, or if the mailbox's LOG parameter is not specified, messages are not written to the mail log.
ON	If a mailbox is specified in the Event definition and that mailbox's LOG parameter is not set to OFF, messages are written to the mail log. If no mailbox is specified in the Event definition, messages are written to the mail log.
DISABLE	Do not write any messages to the mail log. DISABLE is the default.
SPINCOUNT(*x*)	The number of records written to the mail log before ESP Workload Manager automatically spins the log. If you don't specify the SPINCOUNT operand, ESP Workload Manager does not automatically spin the log.
CLASS(*x*)	The mail log's sysout class

Usage notes

Until you set the sysout class via the CLASS operand, mail logging is disabled.

You can use the TSOSEND initialization parameter to suppress all TSO user messages and the MAILLOG parameter to send them to the mail log instead. For more information on TSOSEND, see "TSOSEND: Control Sending of TSO Messages" on page 531.

Example

The following example sets mail logging to ON. The sysout class is X, and the log is spun after 99 records are written to the log.

```
MAILLOG ON SPINCOUNT(99) CLASS(X)
```

MANAGER: Specify the Subsystem Controlling the ESP Agents

Purpose

The MANAGER initialization parameter describes the ESP Workload Manager subsystem controlling the ESP Agents that follow its specification. It is the first initialization parameter in the Agent definition data set (AGENTDEF).

Where defined

Master

AGENTDEF data set

Syntax

```
MANAGER [NAME(mgrname|CENTRAL_MANAGER)]

        [TCPIP[(ipaddress)] [BIND]]

        [APPC]

        [ENCRYPT|ENCRYPT(x'nnnnnnnnnnnnnnnn')

        |ENCRYPT KEYNAME(keyname)|NOENCRYPT]
```

Operand	Description
NAME(*mgrname* | CENTRAL_MANAGER)	Optional. The name of the ESP Manager. The *mgrname* operand has a 16-character maximum.
TCPIP(*ipaddress*)	Indicates the network protocol used for communications is TCP/IP. Optionally, you can specify the DNS host name or TCP/IP address of the ESP Workload Manager master subsystem broadcast by the MGRADDR command. If a DNS host name or TCP/IP address is not specified, ESP Workload Manager uses the address returned by the GETHOSTID command. **Note:** If the DNS host name or TCP/IP address specified is invalid on the current subsystem, ESP Workload Manager does not issue the GETHOSTID command and cannot communicate with the ESP Agent.

Operand	Description
BIND	Indicates when ESP Workload Manager connects to an ESP Agent, ESP Workload Manager will bind to the *ipaddress* as specified in the TCPIP operand. Alternatively, ESP Workload Manager will bind to the default home TCP/IP address, as returned by the TCP/IP GETHOSTID command. Specifying the BIND operand guarantees that all ESP Workload Manager TCP/IP connections with ESP Agents will have the same TCP/IP address.
APPC	Indicates the network protocol used for communications is APPC. APPC is used for backward compatibility, mostly with previous-generation OS/400 Agents.
ENCRYPT	Enables encrypted communication with the Agent. The key is specified in the ENCRYPT initialization parameter.
ENCRYPT (x'*nnnnnnnnnnnnnnnn*')	Enables encrypted communication with the Agent. The key must be an even number of hexadecimal characters up to 16 characters.
ENCRYPT KEYNAME(*keyname*)	Enables encrypted communication with the Agent. The key is defined with the CRYPTKEY command.
NOENCRYPT	Disables encrypted communication with the Agent

Examples

1. The following is an example of the MANAGER initialization parameter for the AGENTDEF data set:

   ```
   MANAGER NAME(CM_CENTRAL) TCPIP
   ```

2. The following example sets the manager name CM_ESP with two different TCP/IP addresses, depending on the host subsystem.

   ```
   IF SYSNAME='SYSA' THEN DO
   MANAGER NAME(CM_ESP) TCPIP(10.1.15.1)
   ENDDO
   ELSE IF SYSNAME='SYSC' THEN DO
   MANAGER NAME(CM_ESP) TCPIP(10.1.15.3)
   ENDDO
   ```

3. The following is a sample AGENTDEF:

```
MANAGER NAME(CM_CENTRAL) TCPIP
ENCRYPT KEY(X'010203030501ADDD')
MAPUSER JDOE TO(CYBJD01) AGENT(TORSUN1)
AGENT TORSUN1 ADDRESS(xxx.xx.xx.xx) PORT(9999) UNIX ASCII +
TCPIP PREFIXING NOENCRYPT
AGENT TORNT1 ADDRESS(xxx.xx.xx.xx) PORT(9998) NT ASCII +
TCPIP PREFIXING
AGENT TORAS4001 ADDRESS(xxx.xx.xx.xx) PORT(9997) AS400 EBCDIC +
TCPIP PREFIXING ENCRYPT(X'010203040506AFFC')
```

4. In this example, the MANAGER initialization parameter specifies a DNS host name called SYSA and requests ESP Workload Manager bind to the TCP/IP address corresponding to SYSA when connecting to an ESP Agent:

```
MANAGER NAME(CM_CENTRAL) TCPIP(SYSA) BIND
```

MAPUSER: Map distributed system user IDs to authorized mainframe user IDs

Purpose

When a distributed system user ID is mapped to an authorized mainframe user ID, ESP Workload Manager can authorize the distributed system user by checking security clearance on the mainframe user ID. If there is no MAPUSER initialization parameter and the distributed system user ID is not defined to your mainframe security product, the security check fails.

Where defined

Master

AGENTDEF data set

Syntax

```
MAPUSER id TO(tso_user_id) AGENT(agentname)
```

Operand	Description
id	The identification of a user logging on to an Agent
TO(*tso_user_id*)	The TSO user ID to be used for security checks
AGENT(*agentname*)	The name of the Agent this mapping applies to

Usage notes

The MAPUSER initialization parameter also applies to a script that might issue a command. You must define or map the script's owner.

Example

The following example shows the ESP Manager will use CYBJD01 for security checks when it sees the user JDOE log on from Agent TORSUN1:

```
MAPUSER JDOE TO(CYBJD01) AGENT(TORSUN1)
```

MAXCXME: Set maximum checkpoint data set space

Note: You can also issue the MAXCXME initialization parameter as an OPER command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The MAXCXME initialization parameter sets the maximum amount of space, in bytes, that XMEs use in the checkpoint data set.

Where defined

Master and proxy

Syntax

```
MAXCXME [size|128000]
```

Operand	Description
size	Specify a number in bytes from 0 to 9999999. The default is 128000.

Usage notes

If a value of 0 is specified, no limit is applied to the amount of space that XMEs use in the checkpoint data set. XMEs are added to the data set until the data set becomes full. At that point, no more XMEs are added to the data set until sufficient space becomes available. No proactive indication of this state is given, but the use of the LISTXMEZ command immediately indicates the situation.

Note: If the checkpoint data set fills up (due to a problem), be aware that you will start losing tracking data as the TCELLS overflow, unless preventive action is taken.

MAXDORM: Set QUEUE data set's maximum dormancy

Note: You can also issue the MAXDORM initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The MAXDORM initialization parameter sets the maximum time interval, in hundredths of a second, before ESP Workload Manager reads the QUEUE data set in a shared DASD environment. This value is critical to proxy system performance.

Where defined

Master and proxy

Syntax

```
MAXDORM [interval|30000]
```

Operand	Description
interval	Specify a number between 100 and 100000 in hundredths of a second. The default is 30000.

Usage notes

In a master/proxy configuration, the master communicates with the proxies using the QUEUE data set or XCF. ESP Workload Manager also uses the QUEUE data set to pick up schedule changes, new data set trigger data, and direct trigger requests from other systems. The MAXDORM value can affect the responsiveness of these requests. Specify a value that is the maximum time you want ESP Workload Manager to take in response to a cross-system request.

For more information on the performance considerations of MINHOLD, MINDORM, and MAXDORM values, see “Performance Considerations” on page 127.

Example

The following example sets the QUEUE data set's maximum dormancy to 3500 on proxy A and proxy B and 300 on the master.

Proxy A

```
MAXDORM 3500
```

Proxy B

```
MAXDORM 3500
```

Master

```
MAXDORM 300
```

MAXLRECL: Set maximum data base record length

Important: This initialization parameter is obsolete. The largest VSAM data set record size for all ESP Workload Manager VSAM data sets is always 32760.

MAXQXME: Set maximum QUEUE data set space

Note: You can also issue the MAXQXME initialization parameter as an OPER command. For more information about the command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The MAXQXME initialization parameter sets the maximum amount of space, in bytes, that XMEs use in the QUEUE data set.

Where defined

Master and proxy

Syntax

```
MAXQXME [size|128000]
```

Operand	Description
size	Specify a number in bytes between zero and 9999999. The default is 128000.

Usage notes

A value of 0 is allowed but converted to the default of 128000 without any indication this has occurred. When an attempt to store an XME fails because the limit is reached, messages 1132W and 1133W are issued and no more XMEs are added to the data set until the condition is corrected. The messages are issued each time the overflow condition occurs. When another XME is successfully added to the data set, the next failure causes the messages to be issued again.

MCS: Control MCS extended console

Note: You can also issue the MCS initialization parameter as command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The MCS initialization parameter controls the ESP Workload Manager MCS (Multiple Console Support) extended console facility. This facility, when active, retrieves responses to system commands that ESP Workload Manager issues and writes those responses to the master job log.

Where defined

Master, proxy or shadow manager

Syntax

```
MCS [ACTIVATE [NAME(consname)] [KEY(conskey)]
    |DEACTIVATE
    |STATUS]
```

Operand	Description
ACTIVATE	Activates the ESP Workload Manager extended MCS console facility
NAME(*consname*)	The console name, up to eight characters. If omitted, the console name defaults to the value specified in the SYSID initialization parameter. If the SYSID initialization parameter is omitted, the current SMF system identifier is used.
KEY(*conskey*)	The console key name, up to eight characters. If omitted, the default is ESP.
DEACTIVATE	Deactivates the ESP Workload Manager extended MCS console facility
STATUS	Displays the status of the ESP Workload Manager extended MCS console. STATUS is the default.

Usage notes

When MCS is issued without operands, the status of the ESP Workload Manager extended MCS console is displayed.

By default, ESP Workload Manager is activated as an extended MCS console, so you don’t need to code the MCS initialization parameter if you want to use the default.

You can use the console key name in the z/OS DISPLAY CONSOLE,KEY command to identify extended MCS consoles.

We recommend that you do not specify the NAME and KEY operands.

MGRADDR: Identify manager to Agent

Note: You can also issue the MGRADDR initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The MGRADDR initialization parameter notifies one or more Agents of the ESP Workload Manager's address and, optionally, the port number so the Agent knows which Agent receiver to connect to. This IP address is always transmitted.

When the MGRADDR initialization parameter is present, ESP Workload Manager automatically notifies each Agent directly under its control, as specified, passing the IP address and, optionally, the port number of the Agent receiver that the Agent should connect to.

Where defined

Master

Syntax

```
MGRADDR {AGENT|AGENT(agentname)}
        [PORT(port)]
        [PERSISTENT(YES|NO)]
```

Operand	Description
AGENT	Sends notification to all Agents
AGENT(*agentname*)	The name of the Agent to notify, as defined in the Agent definition file. To specify multiple Agents, use the hyphen wildcard.
PORT(*port*)	The Agent receiver port number to connect to
PERSISTENT (YES|NO)	Applies to the ESP System Agent Release 7 only. Indicates a permanent change or a temporary change on the Agent. • YES — Used to permanently save the ESP Workload Manager address and receiver port number in the Agent's agentparm.txt file so the Agent knows which receiver to connect to when required. For more information on how the agentparm.txt file is updated, see the *ESP System Agent Administrator's Guide*. • NO — Indicates that the ESP Workload Manager address and receiver port number only change temporarily on the Agent. The address and port number are reset to the values defined in the ESP System Agent's agentparm.txt file the next time the Agent restarts. NO is the default.

Usage notes

Use the MGRADDR initialization parameter for any of the following reasons:

- You use ESP Agents and you move your master from one z/OS image to another. The TCP/IP address will change in this scenario.
- Your installation includes production and test systems and you move ESP Agents from the control of one system to another.

This change is temporary until the Agent restarts. If you need to make the change permanent, code the MGRADDR initialization parameter with the PORT(port) and PERSISTENT(YES) operands.

Examples

To send manager address notification to all Agents, type

```
MGRADDR AGENT
```

or

```
MGRADDR AGENT(-)
```

To send manager address notification to Agents with a name prefix of CYB, type

```
MGRADDR AGENT(CYB-)
```

To send manager address notification to Agent CYBAIX, and to inform it to connect to Agent receiver on port 5451, type

```
MGRADDR AGENT(CYBAIX) PORT(5451)
```

To send manager address notification to Agent R7AGENT, and to inform it to connect to Agent receiver on port 5451 from now on, type

```
MGRADDR AGENT(R7AGENT) PORT(5451) PERSISTENT(YES)
```

MINDORM: Set QUEUE data set's minimum dormancy

Note: You can also issue the MINDORM initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The MINDORM initialization parameter sets the minimum time interval, in hundredths of a second, before ESP Workload Manager attempts to read the QUEUE data set after ESP Workload Manager has performed a write in a shared DASD environment. This value is critical to both master and proxy system performance.

Where defined

Master and proxy

Syntax

```
MINDORM [interval|100]
```

Operand	Description
interval	Specify a number between 100 and 100000 in hundredths of a second. The default is 100.

Usage notes

The MINDORM initialization parameter can prevent an excess of I/O to the QUEUE data set in a shared DASD environment. Specifying a time interval that is too low can result in too high an I/O activity, while specifying too high a value can affect response time. Since ESP Workload Manager only attempts to read the QUEUE data set if it needs to refer to or update the information, a lower time-interval value has less impact than a higher value.

To specify MINHOLD, MINDORM, and MAXDORM values for performance in a master and proxy environment, see "Performance Considerations" on page 127.

Example

The following example sets the QUEUE data set's minimum dormancy to 250 on proxy A, 300 on proxy B, and 150 on the master.

Proxy A

```
MINDORM 250
```

Proxy B

```
MINDORM 300
```

Master

```
MINDORM 150
```

MINHOLD: Set QUEUE data set’s minimum control interval

Note: You can also issue the MINDHOLD initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The MINHOLD initialization parameter sets the time interval, in hundredths of a second, that a subsystem retains control of the QUEUE data set in a shared DASD environment. This value is critical to system performance.

Where defined

Master and proxy

Syntax

```
MINHOLD [interval|100]
```

Operand	Description
interval	Specify a number between 1 and 100000 in hundredths of a second. The default is 100.

Usage notes

The recommended value varies depending on your configuration. Set this value with MINDORM and MAXDORM.

To specify MINHOLD, MINDORM, and MAXDORM values for performance in a master and proxy environment, see “Performance Considerations” on page 127.

Example

In the following example, proxy A is twice as busy as proxy B. Proxy A retains control of the QUEUE data set for 100 hundredths of a second, proxy B for 50 hundredths of a second, and the master for 200 hundredths of a second.

Proxy A

```
MINHOLD 100
```

Proxy B

```
MINHOLD 50
```

Master

```
MINHOLD 200
```

MODE: Test the WSSPARM file

Purpose

The MODE initialization parameter allows you to verify the WSSPARM file when the file is used with the SCAN operand. If any initialization parameter in the file has incorrect syntax, ESP Workload Manager responds with appropriate messages.

Where defined

WSSPARM data set

Syntax

```
MODE  NORMAL|SCAN
```

Operand	Description
`NORMAL`	Causes the Workstation Server to proceed normally after scanning the WSSPARM file and the EXEC parameters
`SCAN`	Causes the Workstation Server to stop after scanning the WSSPARM file and the EXEC parameters

Equivalent EXEC parameter

The following indicates the equivalent EXEC parameter for the definition of the Workstation Server started task. For details, see “Workstation Server started task” on page 97.

```
NORMAL | SCAN
```

Example

To verify the WSSPARM, type

```
MODE SCAN
```

To use the WSSPARM to run a Workstation server, type

```
MODE NORMAL
```

MSG: Set message attributes

Note: You can also issue the MSG initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The MSG initialization parameter adds or deletes routing or descriptor codes to a message or range of messages. It can also change the message type.

Where defined

Master and proxy

Syntax

```
MSG {(id[,id]...)}
    {(id1:id2)}
    [ROUTE(codes)]
    [DESC(codes)]
    [DELROUTE(codes)]
    [DELDESC(codes)]
    [TYPE{(I|W|E|S)}]
```

Operand	Description
id	The message ID. It is a number between 26 and 9998. You can specify a list of numbers enclosed in parentheses and separated by blanks or commas.
id1:*id2*	A range of message IDs. Each number is between 26 and 9998, separated by a colon. You can specify a list of ranges enclosed in parentheses and separated by blanks or commas.
ROUTE(*codes*)	Requests that one or more routing codes be added when the message is used. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.
DESC(*codes*)	Requests that one or more descriptor codes be added when issuing the messages. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.

Operand	Description
DELROUTE(*codes*)	Requests that one or more routing codes be deleted from those normally used for the message. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.
DELDESC(*codes*)	Requests that one or more descriptor codes be deleted from those normally used for the message. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.
TYPE	The new message type is one of the following: • I informational message • W warning message • E error message • S severe error message

Usage notes

When a message is about to be issued, the routing and descriptor codes are modified according to the above specification. The routing and descriptor codes for the message class, as set by the MSGTYPE command, are also set.

The order of application is

1. DELROUTE and DELDESC from message type.
2. ROUTE and DESC from message type.
3. DELROUTE and DELDESC from individual message modifier.
4. ROUTE and DESC from individual message modifier.

Example

In this example

- All informational messages have routing codes 1, 2, 3, 10, and 12 added.
- Warning messages have routing codes 1, 2, 3, 8, 10, and 12 added with descriptor code 1.
- Messages 500 to 510 and 525 also have descriptor code 2 added, although they remain informational messages.
- Message 530 is converted to a warning message and, therefore, has the warning message routing and descriptor codes added. However, the descriptor code of 1 is deleted from message 530.

```
MSGTYPE INFORMATION ROUTE(1:3,10,12)
MSGTYPE WARNING ROUTE (1:3,8,10,12) DESC(1)
MSG (500:510,525) DESC(2)
MSG 530 TYPE(W) DELDESC(1)
```

MSGLIMIT: Set message limits

Note: You can also issue the MSGLIMIT initialization parameter in an ESP Workload Manager Procedure or as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The MSGLIMIT initialization parameter allows an installation to limit the number of console messages a single Event generates. Use MSGLIMIT to suppress excess messages or cancel Events that exceed a certain limit.

Where defined

Master and proxy

Syntax

```
MSGLIMIT (n1[,n2]...])
         [LIST]
         [OFF]
```

Operand	Description
n1	The number of messages after which ESP Workload Manager cancels an Event if *n1* is specified on its own.
n2	The number of messages after which ESP Workload Manager cancels an Event. If you enter two operands in MSGLIMIT, messages are suppressed after *n1* is reached and the Event is canceled after *n2* is reached.
LIST	Displays the current message limit values. The first value is the number beyond which messages are suppressed. The second value is the number is that beyond which Events are canceled. You can also use LIST when changing the message limits to echo the new limits in the response.
OFF	Turns the message limit off. All messages are issued and no Events are canceled.

Usage notes

When you enter MSGLIMIT as a command or initialization parameter, MSGLIMIT sets the default limit value. This default takes effect for any ESP Workload Manager Procedure that does not contain a MSGLIMIT statement. If a Procedure contains a MSGLIMIT statement, its value overrides the default value.

When MSGLIMIT is issued as an operator command, it overrides any previous MSGLIMIT command or initialization statement.

Examples

The following example sets the message limit to 100. When an Event exceeds 100 messages, that Event is canceled.

```
MSGLIMIT 100
```

In the following example, any message after the first 40 from an Event is suppressed. However, if the total number of messages (including the suppressed ones) exceeds 100, the Event is canceled.

```
MSGLIMIT 40,100
```

MSGPRFX: Set message prefix

Note: You can issue the MSGPRFX initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

Use the MSGPRFX initialization parameter to alter the message ID prefix for all messages generated by the subsystem.

Where defined

Master and proxy

Syntax

```
MSGPRFX prefix|ESP
```

Operand	Description
prefix	The message ID prefix, which is the character string immediately before the message number. It is a string of up to seven characters. The default is ESP.

Usage notes

When running more than one subsystem, it is a good idea to identify the source of a message. The message ID prefix can be set to a different string for each subsystem.

Code the MSGPRFX initialization parameter first in the initialization parameters because it only applies to initialization parameters specified after it, not to what is specified ahead of it. For example, if MSGPRFX is specified after your SYSPLEX initialization parameters, all the SYSPLEX messages will not contain your message prefix.

Example

To generate message identifiers in the form ESPT, type

```
MSGPRFX ESPT
```

MSGTYPE: Set message type attributes

Note: You can also issue the MSGTYPE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The MSGTYPE initialization parameter adds or deletes routing or descriptor codes to groups of messages based on their message type.

Where defined

Master and proxy

Syntax

```
MSGTYPE {INFOR|I}
        {WARNING|W}
        {ERROR|E}
        {SEVERE|S}
        [ROUTE(codes)]
        [DESC(codes)]
        [DELROUTE(codes)]
        [DELDESC(codes)]
```

Operand	Description
INFOR|I	Specifies that informational messages will be issued
WARNING|W	Specifies that warning messages will be issued
ERROR|E	Specifies that error messages will be issued
SEVERE|S	Specifies that severe messages will be issued
ROUTE(*codes*)	Requests that one or more routing codes be added when the message is issued. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.
DESC(*codes*)	Requests that one or more descriptor codes be added when issuing the messages. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.
DELROUTE(*codes*)	Requests that one or more routing codes be deleted from those normally used for the message. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.

Operand	Description
DELDESC(*codes*)	Requests that one or more descriptor codes be deleted from those normally used for the message. *codes* can be a number, range of numbers or a list of numbers and ranges of numbers. Each number must be between 0 and 16.

Usage notes

When a message is about to be issued, the message type indicator is used to locate the message-type attributes. Any message-type attributes are combined with the attributes of the message itself to form the final attributes.

The order of application of attributes when both MSG and MSGTYPE apply to a message is

1. DELROUTE and DELDESC from message type.
2. ROUTE and DESC from message type.
3. DELROUTE and DELDESC from individual message modifier.
4. ROUTE and DESC from individual message modifier.

When a message type is the target of more than one MSGTYPE subcommand the results are cumulative.

Examples

Example 1

In this example

- All informational messages have routing codes 1,2,3,10, and 12 added.
- Warning messages have routing codes 1,2,3,8,10, and 12 added with descriptor code 1.
- Messages 500 to 510 and 525 also have descriptor code 2 added, although they remain informational messages.
- Message 530 is converted to a warning message and, therefore, has the warning message routing and descriptor codes added. However, the descriptor code of 1 is deleted from message 530.

```
MSGTYPE INFORMATION ROUTE (1:3,10,12)
MSGTYPE WARNING ROUTE (1:3,8,10,12) DESC(1)
MSGTYPE (500:510,525) DESC(2)
MSG 530 TYPE(W) DELDESC(1)
```

Example 2

This example ensures that informational messages have routing codes 1,2,5, and 6 with descriptor code 2:

```
MSGTYPE I ROUTE(1,2,5)
MSGTYPE I ROUTE(6) DESC(2)
```

Example 3

If you want to reset any prior setting, specify 0 as the first routing or descriptor code.

To reset previous routing and descriptor code settings before adding the new values, type

```
MSGTYPE I ROUTE(0,6), DESC(0,2)
```

NETAUTH: Define authorization profiles

Note: You can also issue the NETAUTH initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The NETAUTH initialization parameter defines authorization profiles for remote hosts that are allowed to connect to the local Network Delivery Services (NDS) network.

Where defined

LOADNET data set only

Syntax

```
NETAUTH [ADD HOST(host)|LIST]
```

Operand	Description
ADD	Defines a new network authorization profile
HOST(*host*)	The TCP/IP address (dotted decimal format or DNS name) that is authorized to connect to the local NDS network
LIST	Lists the network authorization profiles. LIST is the default.

Usage notes

You must set the AUTHENTICATE operand of the NETWORK initialization parameter before your installation can restrict connection by remote hosts according to the authorization profiles set in the NETAUTH initialization parameter or command.

Refer to the NETAUTH command to delete authorization profiles.

Example

The following example restricts access to the local NDS network to hosts with the TCP/IP addresses 122.34.5.1 and 134.4.10.1:

```
NETWORK SET CONNECTIONS AUTHENTICATE
NETAUTH ADD HOST(122.34.5.1)
NETAUTH ADD HOST(134.4.10.1)
```

NETDEST: Define NDS destination profile

Note: You can also issue the NETDEST initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The NETDEST initialization parameter defines a Network Delivery Service (NDS) destination profile. Each network destination is uniquely identified by a network node and specifies its host attributes.

- To define the network destination as a local network node, you must code a NETNODE initialization parameter in the local ESP Workload Manager LOADNET initialization data set.
- To define the network destination as a remote network node, you must code a NETDEST ADD initialization parameter in the local ESP Workload Manager LOADNET initialization data set. You can also issue the NETDEST ADD command dynamically.

Where defined

LOADNET data set only

Syntax

```
NETDEST ADD
```

Expanded Syntax

```
ADD NETNODE(netnode)
    [HOST(host)]
    [SYSPLEX(sysplex)
       [SYSNAME|SYSTEM(sysname)]
       [XCFGROUP(xcfgroup)[XCFMEMBER(xcfmember)]]]
    [JESNODE(jesnode)[JESMEMBER(jesmember)]]
    [SYSID(sysid)]
```

Operand	Description
ADD	Creates a new network destination profile. For the applicable operands, See “ADD” on page 426.

ADD

Operand	Description
NETNODE(*netnode*)	The name of the network node. This value correspond to the value defined in the remote subsystem's NETNODE initialization parameter.
HOST(*host*)	The DNS host name or the dotted decimal IP address of the remote network node
SYSPLEX(*sysplex*)	The sysplex name of a remote ESP Workload Manager subsystem

SYSNAME(*sysname*)
SYSTEM(*sysname*)
	The name of a remote ESP Workload Manager subsystem. SYSNAME and SYSTEM are aliases.

XCFGROUP(*xcfgroup*)	The XCF group name of a remote ESP Workload Manager subsystem
XCFMEMBER(*xcfmember*)	The XCF member name of a remote ESP Workload Manager subsystem.
JESNODE(*jesmode*)	The JES or NJE node name of a remote network node
JESMEMBER(*jesmember*)	The JES or MAS member name of a remote network node
SYSID(*sysid*)	The system identifier of a remote ESP Workload Manager subsystem. *sysid* corresponds to the SYSID initialization parameter of the remote subsystem.

Usage notes

You must define a network destination profile with at least one of the following host attribute specifications:

- HOST(*host*)
- SYSPLEX(*sysplex*)
- JESNODE(*jesnode*)
- SYSID(*sysid*)

You must maintain the following dependencies in a network destination profile:

- SYSNAME(*sysname*) cannot be specified unless SYSPLEX(*sysplex*) is also specified.
- XCFGROUP(*xcfgroup*) cannot be specified unless a SYSPLEX(*sysplex*) is also specified.
- XCFMEMBER(*xcfmember*) cannot be specified unless XCFGROUP(*xcfgroup*) is also specified.
- JESMEMBER(*jesmember*) cannot be specified unless JESNODE(*jesnode*) is also specified.

If a remote ESP Workload Manager master subsystem does not always execute on the same system within its sysplex, its NETDEST specification in the local LOADNET initialization file must not specify any of the following system specific destination specification operands:

- HOST(*host*)
- SYSNAME(*sysname*)
- XCFMEMBER(*xcfmember*)
- JESMEMBER(*jesmember*)

If any host attribute specifications for a remote network node in a NETDEST command do not correspond to the actual host attributes of the network node, an NDS connection request to that network node will be rejected.

Example

The example below illustrates a group of LOADNET initialization parameters:

```
NETNODE ESPM_CHICAGO
NETDEST NETNODE(ESPM_NEWYORK) SYSPLEX(NEWYORK) JESNODE(NYC)+
SYSID(ESPM_NYC)
NETDEST NETNODE(ESPM_TORONTO) SYSPLEX(TORONTO) JESNODE(TOR)+
SYSID(ESPM_TOR)
NETWORK START CONNECTION HOST(NEWYORK) PORT(2300)
NETWORK START CONNECTION HOST(TORONTO) PORT(2300)
```

- The NETNODE initialization parameter defines ESPM_CHICAGO as the local network node name.
- The first NETDEST initialization parameter defines remote network node ESPM_NEWYORK. The sysplex name of its host system is NEWYORK, the JES node name of its host system is NYC, and its subsystem identifier is ESPM_NYC.
- The second NETDEST initialization parameter defines remote network node ESPM_TORONTO. The sysplex name of its host system is TORONTO, the JES node name of its host system is TOR, and its subsystem identifier is ESPM_TOR.
- The first NETWORK START CONNECTION initialization parameter starts a connection to remote network node ESPM_NEWYORK. Its DNS host name is NEWYORK and it is listening on TCP port 2300.
- The second NETWORK START CONNECTION initialization parameter starts a connection to remote network node ESPM_TORONTO. Its DNS host name is TORONTO and it is listening on TCP port 2300.

NETNODE: Name network node

Purpose

The NETNODE initialization parameter names a local ESP Workload Manager master subsystem in the context of a NETWORK Delivery Services (NDS) network.

Where defined

Master

LOADNET data set

Syntax

```
NETNODE netnode
```

Operand	Description
netnode	The arbitrary name of the network node for the ESP Workload Manager master subsystem that contains this NETNODE initialization parameter in its LOADNET data set. The *netnode* value has a up to 16 characters.

Usage notes

The remote subsystems' NETDEST and XMITMDL initialization parameters refer to the arbitrary name defined with the NETNODE initialization parameter in the local subsystem.

Examples

```
NETNODE ESP_CHICAGO
```

You can use an ESP Workload Manager system symbolic variable in the NETNODE initialization parameter:

```
NETNODE ESP_%JESNODE
```

NETQUEUE: Identify NETQUEUE data set

Purpose

The NETQUEUE initialization parameter specifies the name of the NETQUEUE data set. The NETQUEUE data set is a VSAM linear data set required by an ESP Workload Manager master subsystem that uses the Network Delivery Services (NDS) TRACKING services.

Where defined

Master

LOADNET data set

Syntax

```
NETQUEUE dsname
```

Operand	Description
dsname	The name of the NETQUEUE data set.

Example

```
NETQUEUE ESP.NETQUEUE
```

NETWORK: Control NDS

Note: You can also issue the NETWORK initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The NETWORK initialization parameter sets up and starts Network Delivery Services (NDS) connections.

Where defined

Master

LOADNET data set

Syntax

```
NETWORK SET|START
```

Expanded Syntax

```
SET CONNECTIONS [AUTHENTICATE|NOAUTHENTICATE]
                [BUFFERSIZE[(buffersize)]]
                [KEEPALIVE(keepalive)|NOKEEPALIVE]
                [RETRY(retry)|NORETRY]
```

Expanded Syntax

```
START {CONNECTION HOST(host) PORT(cport)
                  [RETRY(seconds)|NORETRY]
       |LISTENER PORT(lport)
       |SERVICE service
      }
```

Operand	Description
SET	Sets attributes for NDS local node. For the operands associated with SET, see “SET” on page 431.
START	Starts an NDS activity on a target remote node. For the operands associated with START, see “START” on page 432.

SET

Operand	Description
CONNECTIONS	Sets attributes for Network Delivery Services (NDS) connections
AUTHENTICATE	Specifies that a connection to or from a remote NDS network node is accepted only if the network node host name or TCP/IP address has been defined in an NDS authorization profile. For information on how to define NDS authorization profiles, see the NETAUTH command or initialization parameter.
NOAUTHENTICATE	Specifies that connections to or from a remote NDS network node are always accepted. NDS authorization profiles are ignored.
BUFFERSIZE(*buffersize*)	The buffer size in bytes for data transmitted from remote NDS network nodes. The range is from 1 to 1048576. The buffer size values are rounded as follow: • 1 to 511 — 512 • 513 to 1023 — 1024 • 1025 to 2047 — 2048 • 2049 and above — rounded up to the next 4K value **Note:** During connection, the remote NDS network node might negotiate the data buffer sizes downward.
KEEPALIVE(*keepalive*)	Specifies an interval between 30 and 14400 seconds. When an NDS connection is idle for a duration equal to the specified interval, a query is sent to the remote network node. If the remote network node does not answer the query, the connection is assumed to have been severed and the local network closes its half of the connection.
NOKEEPALIVE	Specifies that no queries are sent to the remote network node regardless of the duration of idle periods.
RETRY(*retry*)	Specifies an interval between 10 and 3600 seconds. If an NDS connection is not successfully established, the local network node will retry establishing the connection after the specified interval. These attempts will continue until the connection is successfully established or a NETWORK STOP PENDING message is issued. **Note:** The NETWORK START CONNECTION command or initialization parameter overrides this operand.
NORETRY	Specifies that an unsuccessful NDS connection will not be retried. **Note:** The NETWORK START CONNECTION command or initialization parameter overrides this operand.

START

Operand	Description
CONNECTION	Starts an NDS connection
HOST(*host*)	The TCP/IP address (dotted decimal format or DNS name) of the target NDS network host
PORT(*cport*)	The TCP/IP port number of the target NDS network host
RETRY(*seconds*)	Specifies an interval between 10 and 3600 seconds. If an NDS connection is not successfully established, the local network node will retry establishing the connection after the specified interval. These attempts will continue until the connection is successfully established or a NETWORK STOP PENDING message is issued. **Note:** The NETWORK initialization parameter overrides the NETWORK SET CONNECTIONS command or initialization parameter and the RETRY and NORETRY operands.
NORETRY	Specifies that an unsuccessful NDS connection will not be retried **Note:** The NETWORK initialization parameter overrides the NETWORK SET CONNECTIONS command or initialization parameter and the RETRY and NORETRY operands.
LISTENER	Starts an NDS listener
PORT(*lport*)	The TCP/IP port number of the listener
SERVICE *service*	Starts an NDS service. The following values are accepted: • ROUTING — for remote ESP Encore requests (ESP Encore 3.1 and higher) • TRACKING — for internodal job tracking

Usage notes

If no NETWORK SET CONNECTIONS initialization parameter is used, the following default is in effect:

```
NETWORK SET CONNECTIONS NOAUTHENTICATE BUFFERSIZE(4096)-
NOKEEPALIVE NORETRY
```

The connections are not started until the whole LOADNET initialization parameters are read. Therefore, the order of the initialization parameters in the LOADNET data set is not relevant.

Example

The example below illustrates a group of LOADNET initialization parameters:

```
NETNODE ESPM_CHICAGO
NETDEST NETNODE(ESPM_NEWYORK) SYSPLEX(NEWYORK) JESNODE(NYC)+
SYSID(ESPM_NYC)
NETDEST NETNODE(ESPM_TORONTO) SYSPLEX(TORONTO) JESNODE(TOR)+
SYSID(ESPM_TOR)
NETWORK SET CONNECTIONS AUTHENTICATE BUFFERSIZE(4096) +
KEEPALIVE(600) RETRY(300)
NETWORK START CONNECTION HOST(NEWYORK) PORT(2300)
NETWORK START CONNECTION HOST(TORONTO) PORT(2300)
NETAUTH ADD HOST(NEWYORK)
NETAUTH ADD HOST(TORONTO)
```

- The NETNODE initialization parameter defines ESPM_CHICAGO as the local network node name.
- The first NETDEST initialization parameter defines remote network node ESPM_NEWYORK. The sysplex name of its host system is NEWYORK, the JES node name of its host system is NYC, and its subsystem identifier is ESPM_NYC.
- The second NETDEST initialization parameter defines remote network node ESPM_TORONTO. The sysplex name of its host system is TORONTO, the JES node name of its host system is TOR, and its subsystem identifier is ESPM_TOR.
- The NETWORK SET CONNECTIONS initialization parameter sets the connection with a limitation to remote NDS network nodes that have an NDS network authorization profile. The buffer size is 4096 bytes. If the connections are idle for 600 seconds (10 minutes), the remote network node will be queried and the connection will be terminated if it does not respond. If the connections fail to establish, they will be retried every 300 seconds (five minutes).
- The first NETWORK START CONNECTION initialization parameter starts a connection to remote network node ESPM_NEWYORK. Its DNS host name is NEWYORK and is assumed to be listening on TCP port 2300.
- The second NETWORK START CONNECTION initialization parameter starts a connection to remote network node ESPM_TORONTO. Its DNS host name is TORONTO and is assumed to be listening on TCP port 2300.
- The two NETAUTH initialization parameters set the NDS network authorization profile for the two remote network nodes.

NJETOL: Set NJE tolerance

Note: You can also issue the NJETOL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The NJETOL initialization parameter sets or displays the tolerance for the time difference between the ESP Workload Manager submission time and the reader-on time of a remote JES3 system on an NJE connection.

Important: Jobs will stay in input submitted if held in the queue longer than the tolerance NJETOL set.

Where defined

Master

Syntax

```
NJETOL [time_before|500,time_after|500]
```

Operand	Description
time_before|500	The look-back limit in hundredths of a second. The JES3 reader-on time can be *time_before* hundredths of a second before the ESP Workload Manager submission time.The default is 500.
time_after|500	The look-ahead limit in hundredths of a second. The JES3 reader-on time can be *time_after* hundredths of a second after the ESP Workload Manager submission time. The default is 500.

Usage notes

If you issue NJETOL without operands, the current tolerances appear.

When using inter-system tracking, ESP Workload Manager must use the JES3 reader-on time. A tolerance factor might be required to allow for a time difference between the ESP Workload Manager time of submission and the JES3 assigned reader-on time at the JES3 execution node.

This initialization parameter applies regardless of the means of intersystem tracking (for example, NJE or LU6.2) but only to internodal data received from a JES3 system.

You should not increase the NJE tolerance to more than 10-minute window because this increases the chances of indiscriminate matches.

Example

In this example, the NJE tolerance is set to 30 seconds before or after the ESP Workload Manager submission time:

```
NJETOL 3000,3000
```

NODE: Define JES node

Note: You can also issue the NODE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

Use the NODE initialization parameter with the CPU initialization parameter to define your network's resource topology. The NODE initialization parameter defines each JES MAS node.

Where defined

Master only

Syntax

```
NODE name [ADD|DEL|LIST|SET]
          [ROUTEJCL('/*XEQ nodename')]
          [ACTIVE|INACTIVE]
          [SYSPLEX|NOSYSPLEX]
```

Operand	Description
name	The name you want to use. *name* might correspond to an existing JES node name or SMF ID or a group of Agents. You can use wildcard characters to mask the name with all operands except ADD. (See “Wildcards and Masking” on page 237.)
ADD	Adds a new definition
DEL	Deletes one or more existing definitions
LIST	Displays a list of one or more existing definitions. LIST is the default.
SET	Modifies attributes of existing definitions
ROUTEJCL(‘/*XEQ *nodename*’)	Specifies a JCL statement that is inserted into a job's JCL, causing the job to be routed to the appropriate node
ACTIVE	The node is considered active. ACTIVE is the default.
INACTIVE	The node is considered inactive. ESP Workload Manager does not attempt to schedule a job for the node while it is inactive.
SYSPLEX	This node belongs to the sysplex where ESP Workload Manager is running. This operand is required for resources defined with the IBM WLM operand. Use this operand when you want ESP Workload Manager to use IBM WLM workload balancing. This operand is available only if ESP Workload Manager Service Governor is installed.
NOSYSPLEX	This node does not belong to the sysplex where ESP Workload Manager is running. NOSYSPLEX is the default.

Usage note

You must code a NODE initialization parameter for every JES node (MAS) on which you want to use resources, explicitly or implicitly.

You also need to code CPU after defining NODE.

We recommend that you use the CPU and NODE names that JES knows.

Examples

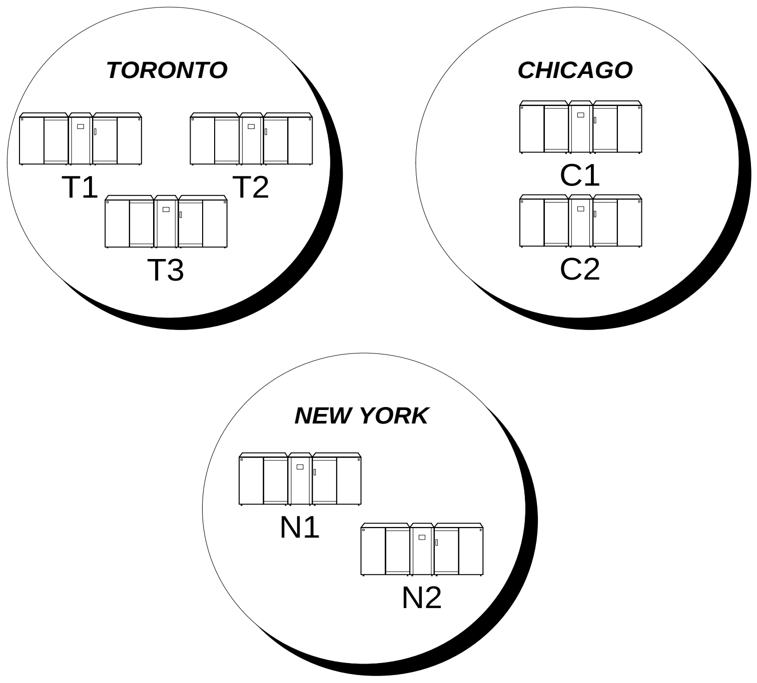
Define a single node and single CPU

This example defines a single node called TORONTO and a single CPU called MVSA:

```
NODE TORONTO ADD
CPU MVSA ADD NODE(TORONTO) CURRENT
```

Define multiple nodes

The following example defines three nodes: TORONTO, CHICAGO, and NEW YORK.



ESPPARM looks like this:

```
NODE TORONTO ADD ROUTEJCL('/*XEQ TORONTO')
CPU T1 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T1') ORDER(1)
CURRENT
CPU T2 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T2') ORDER(2)
CPU T3 ADD NODE(TORONTO) ROUTEJCL('/*JOBPARM SYSAFF=T3') ORDER(3)

NODE CHICAGO ADD ROUTEJCL('/*XEQ CHICAGO')
CPU C1 ADD NODE(CHICAGO) ROUTEJCL('/*JOBPARM SYSAFF=C1') ORDER(3)
CURRENT
CPU C2 ADD NODE(CHICAGO) ROUTEJCL('/*JOBPARM SYSAFF=C2') ORDER(4)

NODE NEWYORK ADD ROUTEJCL('/*XEQ NEWYORK')
CPU N1 ADD NODE(NEWYORK) ROUTEJCL('/*JOBPARM SYSAFF=N1') ORDER(5)
CURRENT
CPU N2 ADD NODE(NEWYORK) ROUTEJCL('/*JOBPARM SYSAFF=N2') ORDER(6)
```

Define NODE and CPU for IBM WLM workload balancing

The following example defines two CPUs that belong to the Toronto node:

```
NODE TORONTO ADD SYSPLEX
CPU T1 ADD NODE(TORONTO) CURRENT
CPU T2 ADD NODE(TORONTO)
```

All the CPUs that have the same NODE operand in the CPU initialization parameter are linked. If you have Service Governor installed, ESP Workload Manager can balance the workload among members belonging to one node if the SYSPLEX operand is coded in the NODE initialization parameter. For information on how to use WLM workload balancing, see the *ESP Workload Manager User's Guide*.

Define NODE and CPU for Agent workload balancing

The following example defines

- Three Windows NT CPUs associated with Agents called AGwin1, AGwin2, and AGwin3.
- Four UNIX CPUs associated with Agents called AGunix1, AGunix2, AGunix3, and AGunix4.

```
NODE WIN ADD
NODE UNIX ADD
CPU WIN1 ADD NODE(WIN)AGENT(AGwin1)
CPU WIN2 ADD NODE(WIN)AGENT(AGwin2)
CPU WIN3 ADD NODE(WIN)AGENT(AGwin3)
CPU UNIX1 ADD NODE(UNIX)AGENT(AGunix1)
CPU UNIX2 ADD NODE(UNIX)AGENT(AGunix2)
CPU UNIX3 ADD NODE(UNIX)AGENT(AGunix3)
CPU UNIX4 ADD NODE(UNIX)AGENT(AGunix4)
```

All the CPUs that have the same NODE operand in the CPU initialization parameter are linked. If you have Service Governor installed, ESP Workload Manager can balance the workload among the Agents belonging to one node. For information on how to use Agent workload balancing, see the *ESP Workload Manager User's Guide.*

NTRCLASS: Specify no track class

Note: You can also issue the NTRCLASS initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The NTRCLASS initialization parameter specifies a job execution class to exclude from tracking. This initialization parameter turns off the collection of type 26 and 30 SMF records for the specified class.

Restrictions

Only one NTRCLASS initialization parameter can be active. If you code more than one, the last initialization parameter coded is used.

Where defined

Tracking ESP Workload Manager subsystem

Syntax

```
NTRCLASS class
```

Operand	Description
class	The class of jobs to exclude from tracking. It is one alphanumeric character.

Usage notes

Use this option with caution. If it is turned on, you can change the class tracking, but there are no explicit operands to turn it off. To turn it off, you must issue the NTRCLASS command with a class value that does not exist. The only character NTRCLASS accepts that cannot represent a class is number sign (#).

To turn NTRCLASS off you can issue

```
NTRCLASS #
```

Note: The effect of the NTRCLASS initialization parameter remains active even when ESP Workload Manager restarts. If you remove all NTRCLASS initialization parameters from the initialization parameter data set, no class will be excluded when you IPL your system.

Example

```
NTRCLASS Z
```

NULLPSWD: Compare password with JOB card

Note: You can also issue the NULLPSWD initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The NULLPSWD initialization parameter specifies a password to be compared to the PASSWORD parameter of a JOB card.

Where defined

Master and proxy

Syntax

```
NULLPSWD password
```

Operand	Description
password	The password to be compared with a JOB card's PASSWORD parameter. The password can be up to eight characters.

Usage notes

The following table indicates whether a job will succeed or fail assuming that only a user with the password ABC can run the job.

User password	PASSWORD parameter in JOB card	NULLPSWD Initialization Parameter	Result
ABC	ABC	Ignored	succeed
ABC	Omitted	Ignored	succeed
ABC	XYZ	XYZ	succeed
ABC	XYZ	xxx (Not XYZ)	fail

PCACHE: Set Procedure caching

Note: You can also issue the PCACHE initialization parameter is also a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The PCACHE initialization parameter specifies whether ESP Workload Manager uses ESP Procedure caching and the size of the cache. ESP Procedure caching can improve CPU usage and processing speed for ESP Procedures that have over 400 jobs. For details about displaying and deleting cache entries, see the CPROC command in the *ESP Workload Manager Reference Guide*.

Where defined

Master

Syntax

```
PCACHE ENABLE|OFF|DISABLE|ON
       SIZE(32|size)
```

Operand	Description
ENABLE	Allows caching for ESP Procedures that include an INVOKE command with the CACHE operand
OFF DISABLE	Disallows the use of caching
ON	Turns on caching for all ESP Procedures unless they specify the NOCACHE operand
SIZE(*size*)	Indicates the size of the cache. The value is between 0 and 1000 MB. A value of 0 clears the cache. If you enter a value larger than 1000 MB, the size is reduced to 1000 MB. The default is 32 MB.

Usage notes

A cached copy of an ESP Procedure is also stored in the Application file as part of the Procedure Caching Record (PCR). This allows you to keep an original cache copy of an ESP Procedure after restarting ESP Workload Manager. The Application Tracking Record (ATR) refers to PCR.

The SIZE parameter applies only to the amount of main memory. If, due to memory constraints, a cached procedure has to be removed from the cache, it will remain in the PCR providing persistence of ESP Procedure caching. When a job is submitted from this procedure, ESP Workload Manager reloads it from PCR into the cache. Only the CPROC DELETE command can delete a copy of a cached procedure from

PCR. When this occurs, ESP Workload Manager re-reads the ESP Procedure and builds a new copy in the cache and the PCR.

PLEXID: ESP Workload Manager Complex ID

Note: As of PTF SEPT154, ESP Workload Manager accepts the PLEXID initialization parameter only as an alias of the ESPGROUP initialization parameter. We strongly recommend that you use ESPGROUP. For information on how to use this initialization parameter, see “ESPGROUP: Specify ESP complex ID” on page 341.

PORT: Specify Workstation Server listening port

Purpose

The PORT initialization parameter specifies the port Workstation Server will listen on.

Where defined

WSSPARM data set

Syntax

```
PORT  port
```

Operand	Description
port	The port Workstation Server will listen on.

Equivalent EXEC parameter

The following indicates the equivalent EXEC parameter for defining the Workstation Server started task. For details, see “Workstation Server started task” on page 97.

```
PORT(port)
```

Example

```
PORT 5300
```

PREALLOC: Pre-allocate data set

Note: You can also issue the PREALLOC initialization parameter as a command without using operands to display the current pre-allocation list of data sets, and with the UNALLOC operand to de-allocate a previously allocated data set. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The PREALLOC initialization parameter pre-allocates a data set.

Where defined

Master and proxy

Syntax

```
PREALLOC dsname ALLOC|UNALLOC
```

Operand	Description
dsname	The data set name
ALLOC	Requests that the data set be allocated. ALLOC is the default.
UNALLOC	Requests that the data set be unallocated

Usage notes

When you need access to the data set, ESP Workload Manager bypasses dynamic allocation for each pre-allocated data set.

Use one PREALLOC initialization parameter for each data set you wish to pre-allocate.

Example

The following is an example of commonly used pre-allocated data sets:

```
PREALLOC CYBER.PROD.JCL ALLOC
PREALLOC CYBER.COPY.JCL ALLOC
PREALLOC CYBER.DOC.LIB ALLOC
PREALLOC CYBER.PROC.LIB ALLOC
PREALLOC CYBER.TEMP.JCL ALLOC
```

PROFILE: Identify the Event prefix and data set

Purpose

Use the PROFILE initialization parameter to associate an Event prefix with an Event data set. ESP Workload Manager uses the Event data set to store Events. The initialization parameter can also associate a history data set with the Event prefix. From this history data set, a user can report the default calendars and a mailbox that the Event uses.

Where defined

Master only

user profile definition table (UPDT)

Syntax

```
PROFILE USER(uuuu) EVENTSET(eventdsid)
             [HIST(histid)]
             [CALENDAR1(cccc)]
             [CALENDAR2(dddd)]
             [MAILBOX(box_name)]
```

Operand	Description
USER(*uuuu*)	An Event prefix specification (usually a user ID). The specification may include wildcards.
EVENTSET(*eventdsid*)	The logical ID of the Event data set. Its value must be the same as the value of *eventdsid* in an EVENTSET initialization parameter.
HIST(*histid*)	The logical ID (eight-character internal name) of a history data set as defined with the HISTFILE initialization parameter. The specification may include wildcards.
CALENDAR1(*cccc*)	The name of a calendar to be used as the first default calendar
CALENDAR2(*dddd*)	The name of a calendar to be used as the second default calendar
MAILBOX(*box_name*)	The name of a default mailbox. The default mailbox contains addresses used for messages that are issued by Events matching the USER operand value when the Event definition does not specify a mailbox. You must define mailboxes in the MAILLIST data set. For information on the MAILLIST data set, see “MAILLIST Data Set” on page 49.

Usage notes

The order of PROFILE initialization parameters in the UPDT is important. Enter the PROFILE initialization parameters from the most specific to the least specific. This sequence considers the fact that ESP Workload Manager scans the table from top to bottom until it finds a PROFILE initialization parameter that matches the new Event's prefix.

The profile name can include the wildcard characters asterisk and hyphen. For more information, see "Representing characters with wildcards" on page 64.

As a minimum, ESP Workload Manager requires a UPDT loaded from a data set that contains a single PROFILE initialization parameter that sets the Eventset default for all new Events.

The HIST operand allows wildcard characters. (See "Representing characters with wildcards" on page 64.) Users can process history reports based on more than one history data set. Since only one HIST operand can be entered for a given PROFILE definition, coding an exact history data set ID restricts the user to generating reports from only that history data set. By carefully assigning history data set IDs, using appropriate wildcarding in the HIST operand, the user can access the desired set of history data sets. For example, HIST(HIST-) gives you access to all history data sets with identifiers that start with HIST.

The mailbox feature enables ESP Workload Manager to send messages to all the TSO user ID and email addresses contained in a specific mailbox.

To use the mailbox feature, you need to define the mailboxes in the MAILLIST data set and load that data set with the LOADNL command or initialization parameter. For information on how to define mailboxes, see the MAILBOX, TSOUSER, EMAIL, and SMTPPARM initialization parameters.

If you do not define mailboxes in the PROFILE initialization parameter or in the EVENT, ESP Workload Manager sends the messages to the Event owner.

Examples

Tell ESP Workload Manager where to save all new Events

In the following example, ESP Workload Manager stores all new Events in the EVENT1 Eventset:

```
PROFILE USER(-) EVENTSET(EVENT1)
```

Tell ESP Workload Manager where to save specific Events

In the following example, ESP Workload Manager stores Events with a prefix of PROD in the EVENT1 Eventset. This example also indicates that these Events use the CYBER calendar and that the user has access to the HIST1 history data set.

```
PROFILE USER(PROD) EVENTSET(EVENT1) CALENDAR1(CYBER) +
HIST(HIST1)
```

Define multiple entries in a UPDT

The following PROFILE initialization parameters show multiple entries in a UPDT:

```
PROFILE USER(PAY-) EVENTSET(PAYEVS) CALENDAR1(PAYCAL)
PROFILE USER(PROD-) EVENTSET(PRODEVS) CALENDAR1(PROD)
PROFILE USER(-) EVENTSET(EVENT1)
```

In the above example

- The first PROFILE initialization parameter tells ESP Workload Manager to store Events with a prefix of PAY- in the PAYEVS Eventset. It also indicates that these Events use the PAYCAL calendar.
- The second PROFILE initialization parameter tells ESP Workload Manager to store Events with a prefix of PROD- in the PRODEVS Eventset. It also indicates that these Events use the PROD calendar.
- The third PROFILE initialization parameter tells ESP Workload Manager to store all other Events in the EVENT1 Eventset.
- The order of these statements is important; the most generic statement should be the last one in the table.
- This sequence accounts for ESP Workload Manager scanning the table from top to bottom until it finds a PROFILE statement that matches the new Event's prefix. The table below shows where ESP Workload Manager stores some Events, based on the previous example.

EVENT NAME	EVENTSET
PAYROLL.WEEKLY	PAYEVS
PROD.DAILY_BACKUPS	PRODEVS
USER01.ADHOC_JOBS	EVENT1

Specifying a mailbox

The following example specifies that, as a default, all Events with a prefix starting with PAY send their messages to the recipients defined in the mailbox PAYROLL. You can specify another mailbox in an Event and, if you do, the mailbox specified in the Event is used.

```
PROFILE USER(PAY-) EVENTSET(PAYEVS) MAILBOX(PAYROLL)
```

Related information

For information on using host security, see the *ESP Workload Manager Security Guide*. For information on loading the user profile definition table, see the LOADUPDT command in the *ESP Workload Manager Reference Guide*.

QTRACE: Trace QUEUE data set

Purpose

The QTRACE initialization parameter causes all allocations and freeing of space to be traced.

Where defined

Master and proxy

Syntax

```
QTRACE
```

Usage notes

Use QTRACE when CA Technical Support asks you to.

Note: When activated, QTRACE cannot be disabled without

1. Bringing ESP Workload Manager down.
2. Removing the initialization parameter.
3. Bringing ESP Workload Manager up again.

QUEUE: Specify QUEUE data set

Purpose

The QUEUE initialization parameter specifies the name of the QUEUE data set.

Where defined

Master and proxy

Note: The master and proxy share the same data set.

Syntax

```
QUEUE dsname [SHR|NOSHR]
```

Operand	Description
dsname	The name of the QUEUE data set
SHR	The data set is shared. RESERVE/DEQ is performed.
NOSHR	The data set is not shared. NOSHR might be specified in a single subsystem and might result in improved performance. Coding NOSHR nullifies the effect of MINHOLD, MINDORM, and MAXDORM.

Usage notes

Each job in the QUEUE data set requires approximately 64 bytes.

The size of the QUEUE data set is limited to one extent. When ESP Workload Manager needs to perform I/O to this data set, it uses a single-channel program. A single-channel program can only read or update one extent at a time. Size is determined automatically when the data set is formatted. The maximum size is approximately 3.5 cylinders on a 3390.

Example

```
QUEUE ESP.QUEUE SHR
```

RACROUTE: Request RACINIT

Note: You can also issue the RACROUTE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The RACROUTE initialization parameter specifies how an Event is processed for security purposes. You specify whether ESP Workload Manager should issue a RACINIT prior to executing Events, and how the user ID associated with the Event is determined.

Where defined

Master and proxy

Syntax

```
RACROUTE [ON|OFF]
         [ITU|NOITU]
```

Operand	Description
ON	Specifies that ESP Workload Manager should issue a RACINIT before executing Events
OFF	Specifies that ESP Workload Manager should not issue a RACINIT before executing Events. ESP Workload Manager processes Events under its own user ID. OFF is the default.
ITU	Inherit Trigger User. All manually triggered, DSTRIG, and global variable trigger Events use the triggering user ID.
NOITU	No Inherit Trigger User. ESP Workload Manager processes Events under the user ID specified in the Event prefix. NOITU is the default.

Usage notes

RACROUTE is also available as an OPER command. You can set the values to ON or OFF, and ITU or NOITU dynamically, for example

```
OPER RACROUTE ON NOITU
```

RACROUTE values at startup

The RACROUTE values (ON or OFF and ITU or NOITU) issued in initialization parameters or commands are checkpointed so they remain in effect until explicitly countermanded.

Cold Start with RACROUTE Initialization Parameter		Warm Start with RACROUTE Initialization Parameter	
Defined	**Not Defined**	**Defined**	**Not Defined**
Uses the RACROUTE initialization parameters	Uses defaults `RACROUTE OFF NOITU`	Uses the RACROUTE initialization parameters	Uses checkpointed values

ITU and NOITU operands

For information on how to determine which user ID ESP Workload Manager processes the Event under, see "How ESP Workload Manager determines the user for batch processing" in the *ESP Workload Manager Security Guide*.

By default, ESP Workload Manager uses the Event prefix of the Event triggering the Procedure as the user ID to run production jobs.

Specifying the USER operand in the EVENT command overrides the default. In that case, ESP Workload Manager uses the user ID you specify.

Specifying ITU at the initialization parameter level or INHERIT_TRIGGER_USER at the Event level overrides the default and the USER operand, causing the triggering user ID to be used if the Event is a manually triggered, DSTRIG or global variable trigger Event.

You can override both ITU and NOITU

- With actions by security system exit routines, such as EVENTSAF (or JES exits)
- With parameters in the JCL

The following table illustrates the logic that determines which user ID ESP Workload Manager processes the Event with RACROUTE ON.

Sequence	Description	Result	Condition
Default		Uses Event prefix	
1st Override	USER operand in EVENT definition	Uses USER operand value	
2nd Override	ITU in initialization parameter or INHERIT_TRIGGER_USER in EVENT definition	Uses triggering user ID	Manually triggered, DSTRIG or global variable trigger Event
3rd Override	Exit routines	Uses user ID specified in exit routine	
4th Override	Parameters in the JCL	Uses user ID specified in the JCL	

The logic is totally independent of what the underlying security package is.

Note: We recommend that you use explicit operands and that you do not rely on the defaults. Otherwise, the logic might be activated, inadvertently resulting in behavior different from what you expect. The possible causes to investigate are

- Multiple RACROUTE initialization parameters
- Checkpointed values originating from dynamic commands and carried over during a warm start. You can only rectify this situation by issuing a RACROUTE command with the correct operands or by initiating a warm start with a RACROUTE initialization parameter with the correct operands.

Example

```
RACROUTE ON NOITU
```

REMOTE: Specify connected Workstation Server

Purpose

The REMOTE initialization parameter specifies the Workstation Server access information that the Consolidated Workstation Server (CWSS) connects to.

Where defined

WSSREMOT initialization data set for a Consolidated Workstation Server

Syntax

```
REMOTE HOST(wss_host_ip_address)
       PORT(wss_port)
       USERID(user_id) PASSWORD(password)
       [SUBSYS(subsystem_name)]
       [ENCRYPT(x'nnnnnnnnnn')]
```

Parameter	Description
HOST(*wss_host_ip_address*)	The IP address of the remote Workstation Server
PORT(*wss_port*)	The port that the remote Workstation Server listens on
USERID(*user_id*)	The user ID the CWSS uses to log on to the remote Workstation Server. The user ID must be fully authorized to process every command a user can issue on the remote Workstation Server and to access the entire ESP Workload Manager's scoreboard.
PASSWORD(*password*)	The password the CWSS uses to log on to the remote Workstation Server. The password should be defined on the target host with an indefinite expiry period. If the password is allowed to expire, an administrator must keep the WSSREMOT file up to date.
SUBSYS(*subsystem_name*)	The name of the ESP Workload Manager subsystem to connect to. If the subsystem name is omitted or incorrect, the CWSS obtains the true subsystem name from the remote Workstation Server.
ENCRYPT(x'*nnnnnnnnnn*')	Enables a client to encrypt its first USER or LOGON command

RESDEF: Define Resources

Note: You can also issue the RESDEF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The RESDEF initialization parameter allows you to define resource definitions.

Where defined

Master only

Syntax

```
RESDEF name ADD
            [ENTERPRISE|NODAL|GLOBAL|LOCAL]
            [THRESHOLD|RENEWABLE|DEPLETABLE]
            [NODE(xxx)][CPU(xxx)]
            [MAX(n)][AVAIL(0)]
            [DEVICE(xxxxx)|WLM(duration)|
              MONITOR(CPU) POLL(xx HOURS|xx MINUTES|xx SECONDS)]
            [GRAVITY|NOGRAVITY]
            [COMMENT(comment)]
```

Operand	Description
name	Indicates the resource name. Specify up to eight alphanumeric characters for default resources and up to 44 alphanumeric characters for other resources. The first character must be alphabetic or national. The resource name must not end with a period or have two consecutive periods. When using the SET operand, you can use wildcard characters.
ADD	Defines a new resource
ENTERPRISE	Indicates one resource counter controls the resource's availability across all nodes and CPUs
NODAL	Indicates one resource counter is maintained for each node
GLOBAL	Alias of NODAL
LOCAL	Indicates one resource counter is maintained for each CPU
THRESHOLD	Defines a threshold resource type
RENEWABLE	Defines a renewable resource type
DEPLETABLE	Defines a depletable resource type
NODE(*xxx*)	Refers to a node defined in the system topology. NODE limits LIST or SET to a specific node.
CPU(*xxx*)	Refers to a CPU defined in the resource topology. CPU limits LIST or SET to a specific CPU.

Operand	Description
MAX(*n*)	Sets the maximum count of the resource you are defining. The default value is 0.
AVAIL(*n*)	Sets the available count of the resource you are defining. This value should not be manipulated, but left for ESP Workload Manager to calculate.
DEVICE (*xxxxx*)	Indicates either the IBM generic or esoteric device name that ESP Workload Manager is to monitor
WLM(*duration*)	Specifies that ESP Workload Manager obtains the number of unused service units over the specified duration from IBM WLM. ESP Workload Manager obtains the number of unused service units before submitting the job. ESP Workload Manager supports the following three durations: • SUM60 — The unused service units are summed over the last 60 seconds. • SUM180 — The unused service units are summed over the last 180 seconds. • SUM600 — The unused service units are summed over the last 600 seconds. **Note:** • The WLM operand is only relevant if ESP Workload Manager Service Governor is installed. If you code the WLM operand without Service Governor installed, ESP Workload Manager ignores the WLM operand. • If you code the WLM operand, you must code the SYSPLEX operand in the NODE command or initialization parameter when you define your system nodes.
MONITOR(CPU)	Specifies that ESP Workload Manager asks all Agents to provide the percentage of CPU availability on the platforms they control at the frequency specified by the POLL operand **Note:** The MONITOR operand is only relevant if the ESP Workload Manager Service Governor is installed. If you use the MONITOR operand without Service Governor installed, ESP Workload Manager ignores the MONITOR operand.

Operand	Description
POLL(*xx* HOURS |*xx* MINUTES |*xx* SECONDS)	Specifies how often ESP Workload Manager asks all Agents to provide the percentage of CPU availability on the platforms they control. xx is a number with a maximum corresponding to 24 hours whether in hours, minutes or seconds. If POLL specifies 0, ESP Workload Manager does not ask Agents to provide the percentage of CPU availability on the platforms they control. **Note:** The POLL operand is only relevant if the ESP Workload Manager Service Governor is installed. If you use the POLL operand without Service Governor installed, the POLL operand is ignored.
GRAVITY	Indicates jobs are to be routed to other nodes the resource is available on
<u>NOGRAVITY</u>	Indicates jobs are not to be routed to other nodes a resource is available on
COMMENT (*comment*)	Used to add information about a particular resource

Usage notes

Defining a resource

Real resources

A real resource represents a hardware device such as tape drives. Since the ESP Workload Manager master is only aware of the status of devices on the image it runs on, real resources only work in a single z/OS image. Define real resources as LOCAL and RENEWABLE and, using the MAX operand, identify the number of devices available to ESP Workload Manager-submitted jobs:

```
RESDEF T3480 ADD LOCAL RENEWABLE MAX(5) DEVICE(3480)
```

The following example shows what appears when the above resource is listed:

```
RESDEF T3480 LIST
Resource T3480    Local Renewable
TORONTO  MVSA      Max=5 Avail=5 Real=5
```

ESP Workload Manager ensures there are sufficient online, unallocated devices available to satisfy the current job's resource requirements.

For example, if job PAYJOB1 requires three units of a real resource called T3480, and ESP Workload Manager sees three devices available, but only two devices are online and unallocated, ESP Workload Manager waits for real resources and PAYJOB1 goes into a RESWAIT state.

Resources with WLM operand (ESP Workload Manager Service Governor only)

You must define an IBM WLM resource as LOCAL RENEWABLE.

To determine the MAX count, see the “CPU capacity table” in the *MVS planning: Workload Management* IBM manual. From this table, choose the number of service units per second for your CPU model and multiply it by either 60, 180 or 600 for the duration SUM60, SUM180 or SUM600 respectively. You can use the result or any higher number. If your installation has CPUs of different models, use the RESDEF SET command to adjust the counts on individual CPUs.

ESP Workload Manager routes a job to one of the available systems based on the service units assigned to the job and other resources the job requests.

IBM WLM feedback is a measurement of recent but past activity; it is not a precise prediction of a CPU load by the time ESP Workload Manager submits a job. The real value of a WLM resource is the number of unused service units. The number of unused service units can differ from the number of service units available for a job if the job service class is higher than the service class of the currently executing workload.

The workload is balanced across the CPUs and nodes within the same sysplex, as long as they are all part of the same sysplex group. The node is defined with the NODE command or initialization parameter (it must include a SYSPLEX operand).

Note: Specify the same duration for all CPUs and nodes. ESP Workload Manager issues a warning message if you specify a different duration for a specific CPU or node.

Resources with MONITOR(CPU) operand (ESP Workload Manager Service Governor only)

You must define a MONITOR(CPU) resource as LOCAL RENEWABLE.

ESP Workload Manager asks all Agents to provide the percentage of CPU availability on the platforms they control. The Agents provide this percentage at the frequency specified by the POLL operand for CPUs where the corresponding CPU initialization parameter includes an AGENT operand. You can limit the scope of the resource you define by using the NODE and CPU operands.

ESP Workload Manager routes a job to one of the available systems based on the MONITOR(CPU) resources and other resources a job requests.

The workload is balanced among the CPUs that belong to the same node. The node is defined with the NODE command or initialization parameter.

Related information

For more information on setting up resources, see Chapter 8, “Setting up Resources.”

For information on specifying default resources, see “RESDFLT: Identify default resources” on page 462.

For information on how to use IBM WLM or Agent workload balancing, see the *ESP Workload Manager User’s Guide*.

Examples

Defining a local renewable resource

In the following example, IMS is a local renewable resource with a count of three. IMS resides on the TOR1 z/OS image and is used to represent access to an IMS region. The maximum number of jobs requiring one unit of the IMS resource that can run concurrently is three.

```
RESDEF IMS ADD LOCAL RENEWABLE MAX(3) CPU(TOR1)
```

Defining a global threshold resource

In the following example, NITESHFT is a global threshold resource with a count of one. NITESHFT represents a time period when specific jobs can run. The NITESHFT resource can be manipulated using the SET operand at the appropriate times to ensure night shift jobs run, for example, between 1am and 8am.

```
RESDEF NITESHFT ADD NODAL THRESHOLD AVAIL(1)
```

Defining a local threshold resource

In the following example, PAYJOB1 is a local threshold resource with a count of zero. PAYJOB1 is used to represent a job's completion. To allow successor jobs to be submitted, you can manipulate the PAYJOB1 resource using the SET operand after PAYJOB1 successfully completes.

```
RESDEF PAYJOB1 ADD LOCAL THRESHOLD AVAIL(0)
```

Defining a real resource

In the following example, T3480 is a real resource representing 3480 cartridge drives. ESP Workload Manager compares the number of tape drives currently online and not allocated with ESP Workload Manager's internal counters to determine when jobs that require the T3480 resource are eligible for submission.

```
RESDEF T3480 ADD LOCAL RENEWABLE DEVICE(3480)
```

Defining a local renewable resource

In the following example, CICS is a local renewable resource with a count of one. CICS resides on the TOR2 z/OS image and is used to represent access to a CICS region. The GRAVITY attribute indicates that the CICS resource is available to jobs on other nodes. If a job on another node requires the CICS resource, ESP Workload Manager inserts the appropriate routing information into the requesting job JCL according to the system topology statements.

```
RESDEF CICS ADD LOCAL RENEWABLE MAX(1) CPU(TOR2) GRAVITY
```

Defining a default resource

In the following example

- The RESDFLT initialization parameter or command indicates ESP Workload Manager should make default resource assignments for scratch tapes:

  ```
  RESDFLT SCRATCHCART(SCRTAPES)
  ```

- The RESDEF command defines a global depletable resource with a count of 500. ESP Workload Manager uses historical information to assign scratch tape resource requirements to each job in an Application:

  ```
  RESDEF SCRTAPES ADD NODAL DEPLETABLE AVAIL(500)
  ```

Note: When default resources are turned on, every Application ESP Workload Manager generates uses default resources. To drop a default resource, code the following:

```
RESOURCE (0,SCRTAPES)
```

Defining resources for Agent workload balancing

The following example creates the RESWIN resource and specifies that ESP Workload Manager will receive the status of every CPU in the WIN node every five minutes:

```
RESDEF RESWIN ADD NODE(WIN) LOCAL RENEWABLE MAX(100)+
    MONITOR(CPU) POLL(5 MINUTES)
```

ESP Workload Manager can balance the workload among CPUs belonging to one node.

RESDFLT: Identify default resources

Note: You can also issue the RESDFLT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The RESDFLT initialization parameter identifies the default resource assignments that ESP Workload Manager makes to all jobs.

Where defined

Master only

Syntax

```
RESDFLT [CARTRIDGE(cart)]
        [SCRATCHCART(scratch)]
        [CPUABSORPTION(cpuabs)]
        [CPUTIME(cputime)]
        [ELAPSEDTIME(elapsed)]
        [PRINTLINES(prlines)
```

Operand	Description
CARTRIDGE (*cart*)	The name of a cartridge tape resource. Default resource names are restricted to eight characters.When you specify this operand, a job ESP Workload Manager submits is automatically assigned the maximum number of tape drives the job required at a time, averaged over its previous runs. For example, a job might require a maximum of two tape drives at a time during Monday's run, a maximum of six tape drives at a time during Tuesday's run, and a maximum of four tape drives at a time during Wednesday's run. Without job profiling, ESP Workload Manager calculates the maximum number of tape drives this job required at a time, averaged over the previous three runs, as four tape drives. However, with job profiling, you can store job information on each run, providing more accurate default resource assignments. In the example above, you could create a job profile for Monday's runs to account for a lower-than-average tape drive requirement (two tape drives) and a job profile for Tuesday's runs to account for a higher-than-average tape drive requirement (six tape drives). In the RESDEF command or initialization parameter, define this resource as GLOBAL RENEWABLE and identify the device name.
SCRATCHCART (*scratch*)	The name of a scratch-tape resource. Default resource names are restricted to eight characters. If the total number of scratch tapes increases at your installation, use the RESDEF command to adjust the number of scratch tapes available to ESP Workload Manager jobs. In the RESDEF command or initialization parameter, define this resource as DEPLETABLE and specify the number of scratch tapes available to ESP Workload Manager jobs.

Operand	Description
CPUABSORPTION (*cpuabs*)	The name of a resource that reflects CPU absorption. Default resource names are restricted to eight characters. When you specify this operand, a job ESP Workload Manager submits is automatically assigned the average CPU absorption for the job's previous runs. The average CPU absorption is CPU time divided by elapsed time and expressed as a percentage. For example, if a job uses 10% of the CPU when it executes, ESP Workload Manager assigns 10 units of the CPU absorption resource to the submitted job. In the RESDEF command or initialization parameter • Define this resource as LOCAL RENEWABLE. • Set the MAX operand for this resource to the maximum CPU usage you want for ESP Workload Manager-submitted jobs. For example, if you set MAX to 80, ESP Workload Manager does not submit jobs that would cause the total CPU absorption for all submitted jobs to exceed 80%. If the total CPU absorption for all submitted jobs is 75% and a job requires 6% CPU absorption, ESP Workload Manager prevents that job from running until the total CPU absorption falls to at least 74%.
CPUTIME (*cputime*)	The name of a resource that represents the total CPU time a job uses, expressed in seconds. Default resource names are restricted to eight characters. You can use CPUTIME to prevent jobs that require more than a certain threshold of CPU time from submitting. In the RESDEF command or initialization parameter, define this resource as LOCAL THRESHOLD.

Operand	Description
ELAPSEDTIME (*elapsed*)	The name of a resource that represents the total elapsed time a job uses, expressed in minutes. Default resource names are restricted to eight characters. You can use ELAPSEDTIME to prevent jobs that require more than a certain amount of elapsed time from submitting. For example, if you are scheduling the shutdown of a certain z/OS system, you can set the maximum availability of the resource to the number of minutes before the scheduled shutdown. ESP Workload Manager does not submit a job that would not, on average, complete prior to the scheduled shutdown time. As the scheduled time draws closer, you can automatically adjust the ELAPSEDTIME value down through an automation program or through an Event. In the RESDEF command or initialization parameter, define this resource as LOCAL THRESHOLD.
PRINTLINES (*prlines*)	The name of a resource that represents the total print lines a job generates. Default resource names are restricted to eight characters. You can use PRINTLINES to prevent jobs that require more than a specified number of print lines from submitting. For example, you can prevent jobs with print line requirements that exceed the available spool space in your installation from submitting. In the RESDEF command or initialization parameter, define this resource as GLOBAL THRESHOLD or GLOBAL DEPLETABLE.

Usage notes

General notes

If you want the default resources that you specified with the RESDFLT initialization parameter to take effect immediately, restart ESP Workload Manager or code the default resources with the RESDFLT command.

You can specify one or more default resources.

Dropping default resources

After you have identified default resources, every Application ESP Workload Manager generates uses these default resources. To drop a default resource, request zero units of the resource, for example:

```
RESOURCE (0,SCRTAPES)
```

Information used to assign default resources

ESP Workload Manager uses information from the jobstats data set to calculate the default resources it assigns to jobs. ESP Workload Manager bases the default resource levels it assigns on an average of the records from the jobstats data set. If job profiles are used, the number of records included in the average depends on the number of entries for the job profile. If job profiles are not used, the number of records included in the average depends on the number of entries for the job, based on its full name, in the jobstats data set. The average includes only completed jobs; abended jobs are ignored. ESP Workload Manager updates the information each time it creates a new Application generation.

Defining resources

After you have identified the default resources your installation requires, use the RESDEF command or initialization parameter to specify the resource type (renewable, depletable or threshold) and the resource scope (local, global or enterprise).

Examples

Define a default scratch-tape resource

The following RESDFLT initialization parameter specifies SCRTAPES as a default resource for scratch tapes:

```
RESDFLT SCRATCHCART(SCRTAPES)
```

The following RESDEF initialization parameter defines SCRTAPES as a global depletable resource with a quantity of 500:

```
RESDEF SCRTAPES ADD GLOBAL DEPLETABLE AVAIL(500)
```

ESP Workload Manager uses historical information to assign scratch-tape resource requirements to each job in an Application. ESP Workload Manager automatically decrements the SCRTAPES resource as jobs run.

In the following job, the RESOURCE statement drops the SCRTAPES default resource from job A. ESP Workload Manager does not assign units of the SCRTAPES resource to job A.

```
JOB A
  RUN DAILY
  RESOURCE(0,SCRTAPES)
ENDJOB
```

The following RESDEF command makes more scratch tapes available:

```
RESDEF SCRTAPES SET AVAIL(750)
```

Related information

For information on defining resources, see “RESDEF: Define Resources” on page 456.

For information on job profiling, see “JOBPROF: Defining a Job Profile Name” on page 379.

For information on resources, see the *ESP Workload Manager User’s Guide*.

For information on requesting resources, see the RESOURCE statement in the *ESP Workload Manager Reference Guide*.

RESFILE: Specify the resource data set

Purpose

The RESFILE initialization parameter specifies the resource data set's name and characteristics.

Note: You need a resource data set only if you use resources.

Where defined

Master only

Syntax

```
RESFILE dsname [MAXCPUS(nn|7)]
               [SIZE(ss)|2097152)]
               [AFFINITY(ENFORCE|WARN|IGNORE)]
```

Operand	Description
dsname	The name of the VSAM linear resource data set
MAXCPUS(*nn*)	The maximum number of CPUs you can expect to have defined at any point in time. The default is 7.
SIZE(*ss*)	The size of the RESFILE in bytes. The default is 2,097,152 (2 million). This is the minimum amount you can specify. There is no maximum value.
AFFINITY(ENFORCE)	If the requested CPU is not defined in the resource topology, the workload is put in RESWAIT state and warning message 3895W is issued. ENFORCE is the default.
AFFINITY(WARN)	If the requested CPU is not defined in the resource topology, warning message 3895W is issued. The workload still runs when the specified resources become available.
AFFINITY(IGNORE)	The workload runs when the specified resources become available, even if the requested CPU is not defined in the resource topology.

Usage notes

You will likely need to set MAXCPUS to more than the default (7) for Agents. Consider the value of MAXCPUS carefully. If you change it later, ESP Workload Manager will dynamically reorganize the data set with the new value during initialization. Try to specify a value you do not expect to exceed. However, each CPU needs four bytes of storage space, so the higher the number, the more storage you need to allocate. Here's how to calculate the storage space you require:

Storage space required = MAXCPUs x 4 bytes per CPU x Number of Resources

For example, if MAXCPUS is set to 20 and the maximum number of resources you intend to use is 1000, the storage space you require is

20 CPUs x 4 bytes per CPU x 1000 resources = 80,000 bytes

If you want to format a new resource data set after you have used RESFILE, you need to use the RESFORM parameter on the START command. RESFORM re-initializes the resource data set. Neither a cold start nor a QFORM impact the RESFILE.

When the resource data set runs out of space, ESP Workload Manager automatically tries to expand the data set up to 4MB per session. The expansion will succeed only if the VSAM data set was allocated with sufficient amount of space; therefore, we recommend specifying secondary extent size during the data set allocation.

When increasing the value of the SIZE parameter without formatting the resource data set, ESP Workload Manager will try to expand the data set to the new requested size during initialization. Before increasing the SIZE parameter, make sure the VSAM resource data set was allocated with enough space and that secondary extent size was specified during the data set allocation.

Example

In this example, the maximum number of CPUs expected is 16 and the RESFILE data set size is eight million bytes. Also, if a CPU is not defined in the resource topology, warning message 3895W is issued, but the workload still runs.

```
RESFILE CYB1.ESP.RESFILE MAXCPUS(16) SIZE(8000000) AFFINITY(WARN)
```

ROUTING: Route ESP Workstation requests

Purpose

The ROUTING initialization parameter sets where the ESP Workstation requests are routed to. Routing also applies when you access ESP Workload Manager from the ESP Workload Manager ISPF dialogues.

For information on XCF routing services, see the *ESP Workload Manager User's Guide*.

Applicability

HAO and Service Governor only

Where defined

WSSPARM data set

Syntax

```
ROUTING  LOCAL|MASTER
```

Operand	Description
`LOCAL`	Routes ESP Workstation requests to the ESP Workload Manager subsystem that the ESP Workstation is connected to. LOCAL is the default.
`MASTER`	Routes ESP Workstation requests to the ESP Workload Manager master

Equivalent EXEC parameter

The following indicates the equivalent EXEC parameter for the definition of the Workstation Server started task. For details, see "Workstation Server started task" on page 97.

```
ROUTING(LOCAL|MASTER)
```

Example

```
ROUTING MASTER
```

RSVLOGIC: Reserve logic

Note: You can also issue the RSVLOGIC initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the RSVLOGIC initialization parameter to specify the serialization technique.

Where defined

Master and proxy

Syntax

```
RSVLOGIC {OFF|ENQ|RESERVE}
         [QNAME(CYBJSS|qname)]
         [RNAME(CYBJSS|rname)]
```

Operand	Description
OFF	No serialization is performed. OFF is the default.
ENQ	Cross-systems ENQ perform serialization.
RESERVE	Hardware RESERVE performs serialization.
QNAME(*qname*)	The QNAME to be used for the RESERVE or ENQ. It is up to eight characters long. CYBJSS is the default. **Note:** In a MIM environment, do not use CYBJSS as a QNAME.
RNAME(*rname*)	The RNAME to be used for the RESERVE or ENQ. It is up to 44 characters long. CYBJSS is the default.

Usage notes

The RSVLOGIC initialization parameter reserves ESP Workload Manager shared operational data sets, any data sets with SHR coded or defaulted, including the QUEUE data set. It does not control access to other data sets that ESP Workload Manager uses (for example, the JCL library). If you wish to turn the RSVLOGIC initialization parameter off, you can specify RSVLOGIC OFF in the initialization parameters.

With Multi-Image Manager (MIM), the QNAME used must be added to the MIM inclusion table. Do not use QNAME(CYBJSS) with MIM.

The combination of RNAME and QNAME must be the same on all ESP Workload Manager subsystems sharing the QUEUE data set.

RESERVE with GRS

If you specify RESERVE and GRS is in use, we recommend that you add the major name (the value specified for QNAME) to systems exclusion RNL.

Example

The serialization technique used is cross-system ENQ with the QNAME and RNAME of CYBERRSV:

```
RSVLOGIC ENQ QNAME(CYBERRSV) RNAME(CYBERRSV)
```

RSVLOGIC command

The RSVLOGIC command is also an OPER command that can be issued from ESP Workload Manager page mode, option G from the ESP Workload Manager main menu. This will tell you what serialization you are using. The following shows a response from the RSVLOGIC command:

```
OPER RSVLOGIC
SERIALIZATION PERFORMED BY ENQ/DEQ
QNAME(CYBERRSV), RNAME(CYBERRSV)
0 CURRENT REQUESTS OUTSTANDING
```

The following example shows you a response with no RSVLOGIC in the initialization parameters or with RSVLOGIC OFF:

```
OPER RSVLOGIC
ESP970I RESERVE LOGIC NOT INITIALIZED
```

The following example shows you a response with RNAME of CYBERRSV:

```
OPER RSVLOGIC
SERIALIZATION PERFORMED BY ENQ/DEQ
QNAME(CYBJSS), RNAME(CYBERRSV)
0 CURRENT REQUESTS OUTSTANDING
```

RUNMODE: Restrict batch usage

Purpose

Use the RUNMODE initialization parameter to restrict ESP Workload Manager batch usage.

Where defined

Master

Syntax

```
RUNMODE BATCH|NOBATCH
```

Operand	Description
BATCH	ESP Workload Manager can run as batch. BATCH is the default.
NOBATCH	ESP Workload Manager cannot run as batch and must run as a started task.

SADLINK: Link SAD data set

Note: You can also issue the SADLINK initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the SADLINK initialization parameter to identify an externally scheduled activity data set (SAD file) to ESP Workload Manager.

Where defined

Master

Syntax

```
SADLINK identifier
         DATASET(dsname)
        [GENERATION(genno)]
        [GROUP(groupname)]
```

Operand	Description
identifier	The internal identifier of a scheduled-activity data set. *identifier* is up to eight alphanumeric characters long.
DATASET(*dsname*)	The name of the sequential data set used for storing scheduled-activity data
GENERATION(*genno*)	Indicates a generation number for a generation data group (GDG). *genno* must be zero or a negative number.
GROUP(*groupname*)	The owning group

Usage notes

The SADLINK initialization parameter identifies an external SAD file with an internal identifier. Each time ESP Workload Manager initializes, or in response to a SADLOAD command, reads the contents of the SAD file, and retains necessary information in a main-storage resident look-up table. To request that the look-up table be used to resolve external linkages, specify the correct SADLINK identifier on the RESOLVE statement in an ESP Workload Manager Procedure.

If an owning group is specified, any user who has either operator authority or update access to the owning group can issue a SADLOAD command to refresh the in-core table.

Examples

To identify the scheduled-activity data set PROD.ESP.SAD to ESP Workload Manager by assigning it an internal identifier of SADDAILY, type

```
SADLINK SADDAILY DATASET(PROD.ESP.SAD)
```

To identify the scheduled-activity data set, PROD.SADDAILY, to ESP Workload Manager by assigning it an internal identifier of SAD1, type the following. An owning group of PROD is specified.

```
SADLINK SAD1 DATASET(PROD.SADDAILY) GROUP(PROD)
```

SAFCLASS: Specify host security class

Note: You can also issue the SAFCLASS initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The SAFCLASS initialization parameter specifies that an external security system is used with the host security interface. You can also assign a class name and a prefix

Where defined

Master and proxy

Syntax

```
SAFCLASS [classname|FACILITY]
         [PREFIX(resprefix|ESP)]
```

Operand	Description
classname	The name of the host security resource class ESP Workload Manager is to use. Use the FACILITY class or a class with the same characteristics as FACILITY. When using RACF, this class should allow generic characters and be GENLISTed, which improves performance.
PREFIX(*resprefix*)	The prefix to be used for all resource profiles. The prefix is useful when a host security class is used for more than one product. The resource prefix differentiates the resource profiles each product uses. PREFIX is an optional operand. If you do not use a prefix, the resource profile will not have a prefix. **Note:** We recommend that you specify a resource prefix.

Usage notes

If SAFCLASS is issued as a command with no operands, it displays the current classname and resource prefix.

Example

To specify that ESP Workload Manager is to use the resource class PROD and that all ESP Workload Manager security resources should be prefixed ESP, include the following SAFCLASS initialization parameter in ESP Workload Manager initialization stream:

```
SAFCLASS PROD PREFIX(ESP)
```

SAFOPTS: Define host security options

Note: You can also issue the SAFOPTS initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The SAFOPTS initialization parameter defines the host security options ESP Workload Manager is to use.

Where defined

Master and proxy

Syntax

```
SAFOPTS [DSALLOC|NODSALLOC]
        [MVSCMD|NOMVSCMD]
        [UTOKEN|NOUTOKEN]
        [SIM3RDPARTY|NOSIM3RDPARTY]
        [MSGSUPP|NOMSGSUPP]
        [OPERCMDS|NOOPERCMDS]
```

Operand	Description
DSALLOC	ESP Workload Manager performs a host security resource check prior to performing dynamic allocation on any data set. The allocation fails if the requester does not have read access to the resource DSALLOC.*dsname*, where *dsname* is the name of the data set allocated. If the data set is allocated on behalf of an Event, the Event's profile is used for the RACHECK. Otherwise, ESP Workload Manager will use its own user ID.
NODSALLOC	ESP Workload Manager should not perform a check prior to dynamic allocations. NODSALLOC is the default.
MVSCMD	ESP Workload Manager uses the resource MVSCMD.*cmdstring* when verifying an Event's authority to execute a z/OS command.
NOMVSCMD	ESP Workload Manager does not uses the resource MVSCMD.*cmdstring* when verifying an Event's authority to execute a z/OS command. It uses the OPER resource instead. NOMVSCMD is the default.

Operand	Description
UTOKEN	ESP Workload Manager performs the third-party RACHECK using a host security user token rather than a simple user ID. When a client makes a request, a SAF TOKEN EXTRACT is performed in the client environment, and this token passes to the service (ESP) for verification. This requires RACF release 1.9 or later. If you are using a different host security system, ensure that it supports the SAF TOKEN EXTRACT and third-party RACHECK.
NOUTOKEN	ESP Workload Manager performs a third-party RACHECK on the client's user ID. NOUTOKEN is the default.
SIM3RDPARTY	ESP Workload Manager simulates the third-party RACHECK feature. Use SIM3RDPARTY when the host security system does not support the third-party RACHECK feature. ESP Workload Manager maintains a queue of ACEEs representing clients who request ESP Workload Manager functions. The RACHECK macro identifies one of these subordinate ACEEs. RACF uses this technique to support third-party RACHECK. Do not request SIM3RDPARTY with UTOKEN.
NOSIM3RDPARTY	ESP Workload Manager uses normal third-party RACHECK processing. NOSIM3RDPARTY is the default.
MSGSUPP	ESP Workload Manager suppresses violation messages when performing a RACHECK. The violation still occurs and is tracked internally by the system security product, but the message is suppressed.
NOMSGSUPP	ESP Workload Manager does not suppress violation messages. We recommend that violation messages are not suppressed for testing but are suppressed for production installations. This allows easy tracking of violation messages during testing but prevents additional message traffic when implemented into production. NOMSGSUPP is the default.
OPERCMDS	Checks that the user ID issuing line or page mode commands the Event executes under has authority to issue the specific operator command
NOOPERCMDS	Checks that the user ID the Event executes under has authority to run OPER commands. NOOPERCMDS is the default.

Example

The following example specifies that ESP Workload Manager will

- Not use the host security interface to check authority for dynamic allocation of data sets within an Event. This is the default.
- Use the MVSCMD prefixed host security resources to control z/OS commands that are issued from within ESP Workload Manager.
- Use the RACF user ID for all security system calls. This is the default.
- Use normal RACF third-party RACHECK. This is the default.
- Suppress security violation messages that are produced from ESP Workload Manager address space, for example

```
SAFOPTS MVSCMD MSGSUPP
```

SCHDFILE: Identify schedule and work data set names

Purpose

The SCHDFILE initialization parameter identifies the schedule data set and work data set that the LOADSCHF command uses. The LOADSCHF command allows you to view future scheduled workload from CSF.

If you don't need to use the LOADSCHF command, you can omit this initialization parameter unless you want to change the following defaults for the scoreboard service. To use SCHDFILE, you must uncomment it in the CYBESS03 sample member.

Where defined

Master

Syntax

```
SCHDFILE dsname|NONE [WORKFILE(dsname2)]
                     [HASHSIZE(nnnn|907)]
                     [MAXENTRIES(nnnn|32768)]
                     [ACCOUNT(1|2|3|4)]
                    [SECURE|NOSECURE]
```

Operand	Description
dsname	The name of the VSAM data set to retain future schedule data. This operand is only required for the LOADSCHF command.
NONE	Allows ESP Workload Manager to run without a schedule data set. If NONE is specified, the WORKFILE operand is ignored. Use this operand if you don't need to use the LOADSCHF command and you want to change the defaults of the other operands.
WORKFILE(*dsname2*)	The name of the sequential data set to be used as a work data set. The work data set must be a scheduled activity data set that a SADGEN batch job produced. The WORKFILE operand is required for the LOADSCHF command.
HASHSIZE(*nnnn*)	The number of entries in the hash table. The maximum is 7993. HASHSIZE should be a prime number. The default is 907.
MAXENTRIES(*nnnn*)	MAXENTRIES is now obsolete; it remains supported for backward compatibility.
ACCOUNT	The number of the accounting field CSF is to display. The default is 1.
SECURE	Alters the CSF view, displaying only the entries that a user has access to. SECURE is the default.
NOSECURE	Displays everything on CSF, even if the user has no authority.

Usage notes

If the SCHDFILE initialization parameter is omitted, ESP Workload Manager runs without a schedule data set and uses the default values for the scoreboard service.

The hash table is a cross-reference table that ESP Workload Manager uses to index the entries in the scoreboard. Typically, the more random your job names, the larger the hash table.

Example

The following example identifies the schedule data set as CYB.ESP.SCHFILE and the work data set as CYB.ESP.WORKFILE.

```
SCHDFILE CYB.ESP.SCHFILE WORKFILE(CYB.ESP.WORKFILE)
```

Related information

For information on CSF, see the *ESP Workload Manager Operator's Guide*.

For information on purging jobs in completed Applications from the scoreboard and the schedule data set, see the PURGSCHF command in the *ESP Workload Manager Reference Guide*.

For information on generating a scheduled activity data set, see the SADGEN command in the *ESP Workload Manager Reference Guide*.

SHADGOAL: Define shadow manager options

Note: You can also issue the SHADGOAL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the SHADGOAL initialization parameter to instruct a shadow manager of the actions to take when an ESP Workload Manager master terminates.

Applicability

HAO only

Where defined

Master only

Syntax

```
SHADGOAL [MASTER(FAIL|LEAVE|QUIESCE)]
               [AFTER(seconds)]
               [WARN]
               [COMMAND('command text')
               [TAKEOVER|TRIGGER(event)]
```

Operand	Description
MASTER	Specifies that a shadow goal is defined
FAIL	The shadow goal being defined applies to the ESP Workload Manager master terminating abnormally and entering an XCF failed state.
LEAVE	The shadow goal being defined applies to the ESP Workload Manager master leaving the XCF group and entering an XCF undefined state when it terminates.
QUIESCE	The shadow goal being defined applies to the ESP Workload Manager master entering an XCF quiesced state when it terminates.
AFTER(*seconds*)	The number of seconds to wait after the ESP Workload Manager master terminates, before executing the shadow goal actions. AFTER must be a number within the range of zero to 86400. If AFTER is not specified, the default is zero.
WARN	Specifies that warning message 4397 will be issued when the shadow goal is executed

Operand	Description
COMMAND('*command text*')	The z/OS command to be issued when the shadow goal is executed
TAKEOVER	Specifies that the shadow manager should take over as the ESP Workload Manager master when the shadow goal is executed
TRIGGER(*event*)	Specifies that the shadow manager should take over as the ESP Workload Manager master and trigger the specified Event when the shadow goal is executed

Usage notes

When a shadow-enabled ESP Workload Manager becomes the active master in the XCF group, all shadow goals are deleted and the SHADGOAL command ceases to be available because it is no longer meaningful.

You can dynamically define shadow manager actions using the z/OS MODIFY command. The SHADGOAL syntax varies in the initialization parameter and the z/OS MODIFY command. For information on the SHADGOAL command, see the *ESP Workload Manager Reference Guide.*

Examples

The following example specifies that 180 seconds (3 minutes) after the ESP Workload Manager master terminates and enters an XCF quiesced state, the shadow manager issues warning message 4397 and issues z/OS command S ESPM.

```
SHADGOAL MASTER(QUIESCE) AFTER(180) WARN +
COMMAND('S ESPM')
```

The following example specifies that 300 seconds (5 minutes) after the ESP Workload Manager master abnormally terminates, the shadow manager issues warning message 4397 and takes over as the ESP Workload Manager master.

```
SHADGOAL MASTER(FAIL) AFTER(300) WARN TAKEOVER
```

SMFINT: Specify SMF interface

Purpose

The SMFINT initialization parameter specifies whether the SMF interface routine should be initialized.

Where defined

Master and proxy

Syntax

```
SMFINT {ON|OFF}
```

Operand	Description
ON	Specifies that the SMF interface routine should be initialized. ON is the default.
OFF	Specifies that the SMF interface routine should not be initialized.

Usage notes

The SMFINT initialization parameter activates or de-activates the subsystem SMF interface routine. A master system with this initialization parameter turned on can intercept and capture SMF information. This SMF information is used to update the CKPT, which writes to the TRAKFILE and HISTFILE, and passes relevant job status to the Application Manager. If the SMFINT initialization parameter is turned on in a proxy, the SMF information is intercepted, captured, and written to the QUEUE data set, where the master picks it up for processing.

The OFF operand allows the SMF interface routine to be switched off. This operand is useful when you run multiple copies of ESP Workload Manager on the same CPU and the systems share data sets. The OFF operand prevents the duplication of several functions, such as the collection of tracking data.

During a restart, SMFINT OFF overrides the previous setting. When set, SMFINT OFF persists until ESP Workload Manager restarts next.

SMFREC: Specify SMF record number

Purpose

The SMFREC initialization parameter specifies an SMF record ID ESP Workload Manager can use when journalling changes to selected database files.

Where defined

Master and proxy

Syntax

```
SMFREC nnn
```

Operand	Description
nnn	A three-digit number between 128 and 255

Usage notes

ESP Workload Manager can journal all updates, additions, and deletions to its data base files to SMF. To do this, you must give ESP Workload Manager a unique SMF record ID for use in its journal records, and specify the JOURNAL keyword on the appropriate database initialization parameter. If you are using ESP Workload Manager on more than one CPU, each system should be using the same SMF record ID.

Example

```
SMFREC 200
```

SMTPPARM: Identify SMTP server

Purpose

The SMTPPARM initialization parameter defines a Simple Mail Transfer Protocol (SMTP) server to ESP Workload Manager.

Where defined

Master and proxy

MAILLIST data set

Syntax

```
SMTPPARM CLASS(class|A)
         JOBNAME(jobname|SMTP)
         [FROMADDRESS(email_address)]
         [ORIGUSER(tso-user-id) SECURE]
```

Operand	Description
CLASS(*class*)	The JES sysout class used for batch SMTP. Do not use a held class. Class A is the default.
JOBNAME(*jobname*)	The name of your SMTP server. SMTP is the default.
FROMADDRESS(*email_address*)	The email address of the message sender, for example jdoe@mycompany.com. If you omit this operand, the sender email address is blank.
ORIGUSER(*tso-user-id*) SECURE	The TSO user ID from which email messages will be sent. If you use this operand, all email messages will be sent in TSO transmit format from the specified TSO user ID. **Note:** If SMTP is in secure mode, this operand is required.

Usage notes

Using the FROMADDRESS operand

You can use the FROMADDRESS operand to prevent messages from ESP Workload Manager from being considered spam. Message recipients can set their spam filters to allow messages from the address specified in the FROMADDRESS operand.

Using the ORIGUSER operand

The TSO user specified will automatically receive the following TSO message for each email ESP Workload Manager sends:

```
+Msg from SMTP: received spool file <number>
```

To avoid this message, use the NOINTERCOM option in the specified user's TSO profile. Alternatively, you can specify a non-existent user in the operand. However, the user you specify must be defined in the SMTP SECTABLE data set.

Coding SMTPPARM in the MAILLIST data set

In the following example, the SMTPPARM initialization parameter is coded in the MAILLIST data set:

```
SMTPPARM CLASS(A) JOBNAME(SMTP) FROMADDRESS(prod@company.com)

MAILBOX  PAYROLL MAXLINES(0)
  TSOUSER CYBPAY1 SYSID(SYSC)
  EMAIL paymaster@company.com
  EMAIL payservice@payservice.com

MAILBOX CYBACCOUNTING MAXLINES(300)
  TSOUSER (CYBACC1 CYBACC2) SYSID(SYSA)
```

The following example shows the MAILIST data set for a secure SMTP installation:

```
SMTPPARM CLASS(A) JOBNAME(SMTP) ORIGUSER(CYB001) SECURE

MAILBOX  PAYROLL MAXLINES(0)
  TSOUSER CYBPAY1 SYSID(SYSC)
  EMAIL paymaster@company.com
  EMAIL payservice@payservice.com

MAILBOX CYBACCOUNTING MAXLINES(300)
  TSOUSER (CYBACC1 CYBACC2) SYSID(SYSA)
```

The SMTPPARM settings apply to the whole MAILLIST data set.

For information on the MAILLIST data set, see “MAILLIST Data Set” on page 49.

SORTNAME: Specify Sort/Merge program name

Note: You can also issue the SORTNAME initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The SORTNAME initialization parameter specifies the name of the installation Sort/Merge utility.

Where defined

Master and proxy

Syntax

```
SORTNAME progname
```

Operand	Description
progname	The name of the Sort/Merge utility

Usage notes

Various ESP Workload Manager components, such as the REPORT processor, use the Sort/Merge utility. The SORTNAME initialization parameter identifies the name of the sort program the installation uses. Note that the sort program should support the standard sort parameters and the E15 and E35 exits.

The Sort/Merge utility default is SORT.

Example

To give the name SORT to the Sort/Merge utility the installation uses, type

```
SORTNAME SORT
```

SORTUNIT: Specify work data set unit name

Note: You can also issue the SORTUNIT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The SORTUNIT initialization parameter specifies a unit name to be used in the dynamic allocation of Sort/Merge utility work data sets.

Where defined

Master and proxy

Syntax

```
SORTUNIT unitname
```

Operand	Description
unitname	The unit name. SYSDA is the default.

Example

The following example specifies the unit name SYSDA for allocating the Sort/Merge utility work data sets:

```
SORTUNIT SYSDA
```

STRACE: Activate subroutine trace

Purpose

The STRACE initialization parameter allocates space and activates a storage management trace.

Where defined

Master and proxy

Syntax

```
STRACE 4000|nnn
```

Operand	Description
nnn	The size of TRACE table. *nnn* is an integer. The default is 4000.

Usage notes

To stop the trace, remove STRACE from the initialization parameters and restart ESP Workload Manager.

Example

```
STRACE 40000
```

SUBSYS: Set subsystem name (ESP Workload Manager)

Purpose

Use the SUBSYS initialization parameter to identify the name of the ESP Workload Manager subsystem.

In ACF2 environments, certain restrictions might exist regarding duplicate subsystem or started task names. In some instances, ACF2 might not allow the subsystem to subsequently restart if the started task name is the same on another subsystem. If this situation occurs, use the SUBSYS initialization parameter to specify a different subsystem name.

Where defined

Master and proxy

WSSPARM data set

Syntax

```
SUBSYS subname
```

Operand	Description
subname	The name of the ESP Workload Manager subsystem. *subname* is a one-to-four-character name. Ensure the name you choose is unique in the z/OS image. The default is the first four characters of the started task name.

Equivalent EXEC parameter

The following indicates the equivalent EXEC parameter for the Workstation Server started task definition. For details, see "Workstation Server started task" on page 97.

```
SUBSYS(subname)
```

Usage notes

Use the subsystem ID to construct a subsystem control table, which anchors control blocks that aid in interaddress space communication, job tracking, and so on. Each instance of ESP Workload Manager subsystem executing on a single processor requires a unique subsystem ID. Obtain the subsystem ID from the SUBSYS initialization statement.

The two most important pieces of information to remember when you need to bring up two or more ESP Workload Manager subsystems on the same z/OS processor are

- Double tracking must be prevented.
- Each ESP Workload Manager must be unique so communication between the ESP Workload Manager subsystems is possible.

The two components that you need to be aware of are the SUBSYS ID and the SYSID.

Example

If ESP Workload Manager is not defined as a subsystem when it is started, it dynamically defines itself as a subsystem.

The ESP Workload Manager subsystem name must be no more than four characters. If you are running more than one instance of ESP Workload Manager on a system, use a meaningful naming convention, giving a unique name to each ESP Workload Manager subsystem. For example, name the subsystem ESP followed by a unique character for each ESP Workload Manager subsystem, such as

```
SUBSYS ESPM
```

SUBSYS: Set subsystem name (ESP Workstation Server)

Purpose

Use the SUBSYS initialization parameter to identify the ESP Workload Manager subsystem that ESP Workstation will communicate with through the ESP Workstation Server.

Where defined

WSSPARM data set

Syntax

```
SUBSYS subname
```

Operand	Description
subname	The name of the subsystem. *subname* is a one-to-four-character name. Ensure the name you choose matches the SUBSYS initialization parameter of the ESP Workload Manager that the ESP Workstation Server communicates with.

Usage notes

The SUBSYS initialization parameter is required either as a WSSPARM command or as a parameter on the JCL EXEC statement. If SUBSYS is omitted, the WSSPARM assumes that the subsystem name is ESP.

The ESP Workstation Server uses the subsystem ID to locate the unique subsystem control table that a specific ESP Workload Manager created in system memory. The information in this table enables the ESP Workstation Server to communicate with the ESP Workload Manager via different cross-memory techniques.

Example

When the ESP Workstation Server begins executing, it will scan system memory for a subsystem control table, which the ESP Workload Manager created, whose ESPPARM member contains the following statement:

```
SUBSYS ESPM
```

SVC: Specify SVC number

Purpose

The SVC initialization parameter specifies the SVC number that the ESP TSO command uses.

ESP Workload Manager requires the use of an SVC number to allow communication between the ESP Workload Manager subsystem and the ESP TSO command. Locate an unused SVC in the range 200 to 255 and code the SVC parameter. The SVC number does not have to be SYSGENed. It does not matter if a previous SVC, of any type, used this number before. ESP Workload Manager loads the associated SVC routine (CYBJS030) into common storage and updates the SVC table. No IPL or CLPA is necessary.

Where defined

Master and proxy

Syntax

```
SVC number
```

Operand	Description
number	The number of an unused SVC of any type. *number* must be between 200 and 255.

Example

```
SVC 220
```

SYMBOL: Set symbolic variable introducer

Note: You can also issue the SYMBOL initialization parameter as command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The SYMBOL initialization parameter specifies the character to be used to introduce a symbolic variable.

Where defined

Master and proxy

Syntax

```
SYMBOL character
```

Operand	Description
character	*character* is printable. Setting the symbolic variable introducer to a character that is legal as the first character of a name can cause confusion and is not recommended. This particularly applies to the national characters, such as the dollar sign ($), number sign ($) or commercial at (@).

Usage notes

The default symbolic variable introducer character ESP Workload Manager uses is percent (%) character. If the percent character creates a potential conflict because the installation JCL contains many occurrences of the percent (%) character that can be interpreted incorrectly, you can assign any other character, such as the exclamation point (!).

A user can override a symbolic variable introducer character.

Example

To change the symbolic variable introducer character to the exclamation point (!), type

```
SYMBOL !
```

SYSID: Specify system identifier

Purpose

The SYSID initialization parameter specifies the system identifier that ESP Workload Manager is to use. Events use the SYSTEM initialization parameter in the Event definition to identify the system to process Events on. The SYSTEM initialization parameter on the Event definition must match the SYSID specified here.

Where defined

Master and proxy

Syntax

```
SYSID sysid
```

Operand	Description
sysid	The system ID. *sysid* is up to eight characters long. If this operand is omitted, the current SMF system identifier is used.

Usage notes

Use the system ID to uniquely identify an instance of ESP Workload Manager across multiple systems. When multiple instances of ESP Workload Manager share data sets, each instance of ESP Workload Manager must have a unique system ID so that the different ESP Workload Manager subsystems can send information to each other and not overlay data belonging to another ESP Workload Manager subsystem. The system ID also controls which processor can execute a particular Event.

The system ID is obtained from the SYSID initialization parameter. If the system ID is omitted, the current system SMF ID is used. Changing the system ID requires an ESP Workload Manager cold start to reset the ownership, which might happen if the SMF ID is used as a default and it changes. For this reason, we recommend that the SMF ID is not used, but that each ESP Workload Manager be given a unique system ID.

If ESP Workload Manager uses a system ID other than the current SMF ID, include the system ID in the SYSID initialization parameter. Define system IDs that are unique to ESP Workload Manager to prevent a change of the SMF ID from affecting ESP Workload Manager functions.

Note: ESP Workload Manager uses SYSID as an identifier for Events.

Example

```
SYSID SYS1
```

SYSMSGS: Intercept system messages

Note: You can also issue the SYSMSGS initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

ESP Workload Manager can intercept any system message while it is being written to the JES system message data set belonging to an individual job, started task or time-sharing user (TSU).

The JES system message data set constitutes the job's log, containing JES messages about the job's processing, allocations, job and step statistics, and data set disposition. The JES system message data set (JESYSMSG) is subject to installation parameters that can affect content and creation.

ESP Workload Manager does not intercept data written to other JES data sets, the data sets belonging to the operating system (that is, SYSLOG) or console messages.

Where defined

Master and proxy

Syntax

```
SYSMSGS ['text']
        [DISABLE|ENABLE|IGNORE]
        [CANCEL|CCFAIL|JCLERROR|WARN]
        [TSU]
        [STC]
        [ID(xxxx)]
        [COL(nn[:nn])]
        [NAME(jobname[,jobname]...)
        [EVENT(eventid)]
        [COUNT(m)]
        [ROUTE(rcode)]
        [DESC(dcode)]
        [JOBNAME]
        [WTO]
        [COMPRESS]
```

Operand	Description
text	The part of the message text that defines the message that you want to be intercepted. *text* can be a message identifier, a prefix or any part of the message text. Do not specify this operand if the DISABLE, ENABLE or IGNORE keywords are specified. However, you must specify this operand if these keywords are not specified.
DISABLE	Temporarily suspends an existing entry for the specified message ID and prevents its interception. The ID operand must be specified. No other operand may be specified, including the message text.
ENABLE	Re-enables a suspended entry for the specified message ID. The ID operand must be specified. No other operand may be specified, including the message text.
IGNORE	Invalidates and deletes an existing entry for the specified message ID and prevents its interception. The ID operand must be specified. No other operand may be specified, including the message text.
CANCEL	Requests that interception of this message text will generate a z/OS job cancellation (system abend code S222)
CCFAIL	Requests that interception of this message will cause a condition code failure
JCLERROR	Requests that interception of this message should cause a JCL error
WARN	Requests that ESP Workload Manager generate a warning message when the specified system message is intercepted. The message appears when an LTJ subcommand is issued for the job that issued the message. Note that no action will occur with this option; only a warning message will be issued.
TSU	Requests that interception of this message will only occur for time-sharing users (TSUs)
STC	Requests that interception of this message will only occur for started tasks (STCs)
ID(*xxxx*)	The identifier of a message intercept to be cleared. ID consists of four alphanumeric characters. To clear all system message interceptions, code ID(*).
COL(*nn*)	Specifies a column or range of columns where the specified message text should begin. The default is column 1.
NAME(*jobname*)	Specifies up to four job names (or job name strings), thus limiting the system message interception for a particular job or set of jobs
EVENT(*eventid*)	The Event to be triggered when the system message is intercepted. EVENT can be abbreviated to EV.

Operand	Description
COUNT(*m*)	The Event is scheduled for every *m* interception of the system message, where *m* is a number between 0 and 255. A value of 0 results in a schedule for each interception, as does a value of 1. The default is 1.
ROUTE(*rcode*)	The routing code that identifies the console this message should be sent to. If *rcode* is omitted, the default routing code is 2.
DESC(*dcode*)	The descriptor code that applies to this message. If *dcode* is omitted, the default descriptor code is 6. Use DESC(2) for a non-deletable message.
JOBNAME	The job name should be embedded within the message when it is rebroadcast on the console.
WTO	The message should be rebroadcast as a WTO message.
COMPRESS	Superfluous blanks are to be removed from the text.

Usage notes

After each IPL, you must start ESP Workload Manager before starting any task whose system messages you want to intercept.

The SYSMSGS initialization parameter can specify the failure of a job or triggering of an Event through the interception of pre-defined message IDs or message text. The message can be issued from a console or an authorized terminal and routed to any specified console. Ensure that the SYSMSGS operand is first specified in the TRACKOPT initialization parameter. If TRACKOPT SYSMSGS is not specified, system messages will not be intercepted.

The message text must not be specified for the DISABLE, ENABLE or IGNORE operands. These operands require that the ID operand also be specified. No other operands are allowed.

When the SYSMSGS initialization parameter is first used to identify message interception, it takes effect immediately. Use the ENABLE operand only after a SYSMSGS ID has been disabled.

You can display the current system message interceptions, with their message IDs, via the LSYSMSGS command.

Examples

Cancelling the job

Whenever a NOT CATLGD 2 message is generated starting anywhere between column 50 and 60, cancel the job and embed the job name when the message is rebroadcast.

```
SYSMSGS 'NOT CATLGD 2' COL(50:60) CANCEL WTO JOBNAME
```

Triggering an Event

Whenever an IEF142I system message is generated for the job PAYROLL, trigger the Event called CYBER.PAYSTEP.

```
SYSMSGS 'IEF142I' NAME(PAYROLL) EVENT(CYBER.PAYSTEP)
```

Highlighting the system message

Whenever an IEF253I system message is generated for jobs with two-character names beginning with J or any job beginning with the character K, cancel the job and trigger the Event called CYBER.CAN. Highlight the message using DESC(2). Assign an ID of 0010 to this system message interception:

```
SYSMSGS 'IEF253I' ID(1101) NAME(J*,K-) CANCEL -
EV(CYBER.CAN) DESC(2)
```

Ignoring specified system messages

The following example requests that any NOT CATLGD message starting anywhere between columns 50 and 60 will be ignored for jobs starting with the prefix UT:

```
SYSMSGS 'NOT CATLGD' COL(50:60) NAME(UT-) IGNORE
```

SYSPLEX: Define an XCF Group and Its Members

Purpose

The SYSPLEX initialization parameter supports the Cross-System Coupling Facility (XCF) component of z/OS. The SYSPLEX initialization parameter defines the name of the XCF group, what XCF member name the ESP Workload Manager subsystem will join as, and what state ESP Workload Manager will be in upon termination.

Where defined

Master and proxy

Syntax

```
SYSPLEX {GROUP(group)}
        {MEMBER(member)}
        [TERMOPT(QUIESCE|LEAVE)]
        [INTERVAL(interval)]
```

Operand	Description
GROUP(*group*)	The name the XCF group will use. The XCF group name must be up to eight alphanumeric or national characters. It must not begin with the characters SYS and must not be MISSING or UNDESIG.
MEMBER(*member*)	The member name ESP Workload Manager will use when it joins the XCF group. *member* must be unique in the XCF group. It must be up to 16 alphanumeric or national characters and must not be MISSING.

Operand	Description
TERMOPT(QUIESCE)	ESP Workload Manager enters an XCF quiesced state when it terminates normally. A quiesced XCF member is disassociated from XCF Services, but XCF retains a record of its existence. Quiesced members of an XCF group are included in the XCF DISPLAY GROUP subcommand. QUIESCE is the default.
TERMOPT(LEAVE)	ESP Workload Manager enters a LEAVE state when it terminates normally from the XCF group. LEAVE means a member goes into an undefined state and XCF does not maintain a record of the member's existence. If QUIESCE and LEAVE are omitted and the local z/OS system is a member of a sysplex (for example, uses a couple data set), QUIESCE is the default. If the local z/OS system is not a member of a sysplex, LEAVE is the default.
INTERVAL(*interval*)	Activates XCF status monitoring. XCF status monitoring helps detect loops and hangs. Each XCF member supplies an eight-byte field every interval. If the other members do not see this field, the subsystem might be looping or hanging. In this case, ESP Workload Manager issues messages ESP4331W in the console log. The value is the number of seconds between 3 and 86400. **Note:** XCF status monitoring requires HAO or Service Governor.

Example

The following example uses the full initialization parameter syntax:

```
SYSPLEX GROUP(ESP520) MEMBER(MASTER) TERMOPT(QUIESCE)+
INTERVAL(120)
```

TAPETRAK: Control tape usage count

Note: You can also issue the TAPETRAK initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The TAPETRAK initialization parameter enables ESP Workload Manager to determine which tape devices should be included in a job’s tape-device-usage count based on the largest number of tape devices used in a single step. The average tape-device-usage count for a job is recorded in the ESP Workload Manager history file and ESP Workload Manager uses this count to estimate a job's tape-device requirements when tape devices are defined as a default resource. If a job's estimated tape-device requirements are not met, ESP Workload Manager puts the job into a resource wait state, and only proceeds with job submission when the tape-device requirements are met.

Note: Tape devices are also called cartridge devices.

Where defined

Master and proxy

Syntax

```
TAPETRAK [ALL|REAL|NOVIRTUAL]
         [EXCLUDE(X'nnnn'[:X'nnnn'] …)|NOEXCLUDE]
```

Operand	Description
ALL	Specifies that all tapes (both real and virtual) are to be included in a job step’s tape-device-usage count. ALL is the default.
REAL	Specifies that real tape devices are to be included in, and virtual tape devices are to be excluded from, a job step’s tape-device-usage count

Operand	Description
NOVIRTUAL	Specifies that real tape devices are to be included in, and virtual tape devices are to be excluded from, a job step's tape-device-usage count **Note:** NOVIRTUAL is an alias of REAL.
EXCLUDE	The device numbers of the tape devices that must be excluded from a job tape-device-usage count. You can exclude • Single devices: X'*nnnn*' • Ranges of devices: X'*nnnn*':X'*nnnn*' Separate the device addresses or the range of device addresses with a space. **Note:** If you enter multiple TAPETRAK EXCLUDE initialization parameters, ESP Workload Manager will validate only the last initialization parameter. To exclude multiple devices at the same time, you must specify them in one initialization parameter.
NOEXCLUDE	No tape devices are to be explicitly excluded from a job step's tape-device-usage count. Use this operand to reverse a previous TAPETRAK initialization parameter containing the EXCLUDE operand.

Usage notes

The TAPETRAK parameter is only valid for tracking ESP Workload Manager subsystems, specifically those with initialization parameter SMFINT set to ON (the default).

You can use the TAPETRAK initialization parameter to include or exclude any tape device, real or virtual. If you use a tape-management system, like the IBM VTS (Virtual Tape System), where ESP Workload Manager can automatically detect the virtual devices, you can use the REAL or NOVIRTUAL operand to exclude all virtual devices. If you use a tape -management system like StorageTek's VSM (Virtual Storage System) where ESP Workload Manager cannot automatically detect the virtual devices, you must use the EXCLUDE operand to exclude any devices.

When you want to use only the LIST and HELP operands, you can also specify TAPETRAK as a command from TSO/ISPF page mode or from an ESP Workload Manager line mode interface. To use all other operands, you must issue TAPETRAK as a
z/OS MODIFY command.

You may use the EXCLUDE operand to exclude any device. However, the EXCLUDE operand is only meaningful when you specify tape devices.

Removing a TAPETRAK initialization parameter from the ESPPARM file does not reset the tape tracking status to its original default. You must replace the original TAPETRAK initialization parameter in the ESPPARM file by typing

```
TAPETRAK NOEXCLUDE
```

Alternatively, you can issue a TAPETRAK command similar to the following:

```
F QX10S2,TAPETRAK NOEXCLUDE
```

Examples

Your installation with IBM VTS wants to exclude virtual tape devices from a job step's tape-device-usage count. Add the following initialization parameter to the ESPPARM initialization file:

```
TAPETRAK REAL
```

Your installation with StorageTek's VSM wants to exclude virtual tape devices from a job step's tape-device-usage count. Their virtual tape device numbers are 0580-058F and 0780-078F. Add the following initialization parameter to the ESPPARM initialization file:

```
TAPETRAK EXCLUDE(X'580':X'58F' X'780':X'78F')
```

Related Information

For information about specifying tape drive resources, see the "Using real devices" section of the *ESP Workload Manager User's Guide.*

TCELL: Define tracking storage cell

Note: You can also issue the TCELL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The TCELL initialization parameter defines a global storage cell pool used for passing tracking information to the ESP Workload Manager subsystem.

Where defined

Master and proxy

Note: You must specify SMFINT ON.

Syntax

```
TCELL JOBSTART|STEPEND|JOBEND
      [COUNT(cellcount)]
      [EXP(maxexp)]
```

Operand	Description
JOBSTART	The tracking cell specification for job start tracking records
STEPEND	The tracking cell specification for step end tracking records
JOBEND	The tracking cell specification for job end tracking records
COUNT(*cellcount*)	The initial count of cells in the storage pool. The maximum value is 100. The default is 10.
EXP(*maxexp*)	The maximum expansion limit beyond which no more cells are GETMAINed. The maximum value is 5000. The default is 10.

Usage notes

TCELLS are required to pass job start, step end, and job end information to the ESP Workload Manager subsystem. Use a separate TCELL initialization parameter to define each cell pool.

When choosing the initial cell count and maximum expansion limit, consider the type of activity occurring on the system. The cell count should be large enough to accommodate approximately two to five minutes of average activity. ESP Workload Manager analyzes and returns cells to the pool as rapidly as possible.

The maximum expansion limit specifies how many additional elements are to be obtained through GETMAIN when the pool is exhausted. Any cell obtained by using GETMAIN is freed by using FREEMAIN. The expansion limit should be large enough to accommodate the average activity for 20 to 30 minutes during the periods that ESP Workload Manager might be down for maintenance.

Job purge tracking records use the same tracking cell specification as job end tracking records.

Examples

To define cell pool for job end Events, type

```
TCELL JOBEND COUNT(100) EXP(5000)
```

To define cell pool for job start Events, type

```
TCELL JOBSTART COUNT(100) EXP(5000)
```

To define cell pool for step end Events, type

```
TCELL STEPEND COUNT(100) EXP(5000)
```

TCPIP: Define TCP/IP access

Note: You can also issue the TCPIP initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The TCPIP initialization parameter specifies which vendor's TCP/IP to access (IBM HPNS or Interlink Software's TCP access) and where it is (job name for IBM or z/OS subsystem name for Interlink).

Where defined

Master and proxy

WSSPARM data set

Syntax

```
TCPIP {IBM|HPNS}[(jobname)]
      {TCPACCESS}}[(ACSS|subsys1[,subsys2])]
```

Operand	Description

IBM(*jobname*)
HPNS(*jobname*)
	Specifies IBM HPNS (High Performance Native Sockets) TCP/IP, which is the default. This option should be specified (or defaulted to) when running Version 3.2 or later of IBM TCP/IP.
	HPNS is an alias of IBM.
	jobname is the job name of the IBM TCP/IP to be accessed. If omitted, the job name specified by the TCPIPJOPBNAME keyword in the TCP/IP task is used.

TCPACCESS(*subsys1*,*subsys2*)	Specifies Interlink Software's TCP access. *subsys1* is the z/OS subsystem name of the Interlink Software TCP/IP API (Application Programming Interface) to be accessed. If omitted, ACSS is the default. *subsys2* is the z/OS subsystem name of the Interlink Software's TCP access DNR (Domain Name Resolver) to be accessed. The API and DNR components of TCP access are almost always in the same z/OS subsystem. Only specify *subsys2* is the API and DNR components are in different z/OS subsystems.

Equivalent EXEC parameter

The following table indicates the equivalence between WSSPARM and EXEC for the Workstation Server started task definition. For details, see "Workstation Server started task" on page 97.

WSSPARM parameters	EXEC parameters
TCPIP IBM[(*jobname*)]	TCPIBM[(*jobname*)]
TCPIP HPNS[(*jobname*)]	TCPHPNS[(*jobname*)]
TCPIP TCPACCESS[(*subsys1[,subsys2]*)]	TCPACCESS[(*subsys1[,subsys2]*)]

Examples

To identify the TCP/IP to be accessed as IBM HPNS running under job name IBMTCPIP, type

```
TCPIP IBM(IBMTCPIP)
```

or

```
TCPIP HPNS(IBMTCPIP)
```

To identify the TCP/IP to be accessed as Interlink Software's TCP access running under the default z/OS subsystem name ACSS, type

```
TCPIP TCPACCESS
```

To identify the TCP/IP to be accessed as Interlink Software's TCP access running under z/OS subsystem name INET, type

```
TCPIP TCPACCESS(INET)
```

TIMECHK: Check time

Purpose

The TIMECHK initialization parameter warns the operator if

- On startup, more than the specified time has elapsed since ESP Workload Manager was last up.
- During processing, the system clock changes by more than the specified time.

Where defined

Master and proxy

Syntax

```
TIMECHK [60|minutes]
```

Operand	Description
minutes	On startup, if more than the specified number of minutes have elapsed since ESP Workload Manager was last up, ESP Workload Manager issues a warning message. ESP Workload Manager requests confirmation to continue processing. During processing, if the system clock changes and the difference between the old time and the new time exceeds the specified number of minutes, ESP Workload Manager issues a warning message. ESP Workload Manager requests confirmation to continue processing. If no value is specified, the default is 60 minutes.

Usage notes

If the system clock is set to the incorrect time, the TIMECHK initialization parameter helps detect the error.

ESP Workload Manager also issues a warning message if, when ESP Workload Manager starts, the system clock is set to a time prior to the last time ESP Workload Manager was up.

Example

To cause ESP Workload Manager to issue a warning message if more than 180 minutes have elapsed since it was last up or the system clock changes by more than 180 minutes, type

```
TIMECHK 180
```

TIMEZONE: Define time zone

Purpose

The TIMEZONE initialization parameter defines the difference in time (offset) between a named time zone and the common-reference time zone. A time zone must be defined before you can use it in an expression. When ESP Workload Manager communicates with other systems (proxies, other masters or Agents), the local time zone must be set up with the reference zone common to all participants (usually, UTC).

In the TIMEZONE initialization parameter, you can specify how ESP Workload Manager obtains local time, which allows scheduling according to a time that is different from the local time the operating system reports without the need to code a time zone name on all time expressions.

Where defined

Master and proxy

Syntax

```
TIMEZONE code
         name
         offset
         SYSZONE(YES|NO)
         [TIMEREF(SYSTIME|GMT +|- hh.mm)]
```

Operand	Description
code	The time zone code. It is a number between 0 and 63. The following time codes are reserved: • 0 = LOCAL • 1 = UTC • 2 = GMT • 3 = Z
name	The name of the time zone. *name* is up to eight characters long. All characters are allowed.
offset	The time shift from the reference zone. You specify *offset* in hours and minutes separated by a period and followed by E or W for East and West. For example, 1.30E means a difference of one hour and a half East from the reference zone.
SYSZONE(YES)	Requests that the time zone shift z/OS uses be added to the offset value. SYSZONE allows ESP Workload Manager to adjust its zone shift synchronously with the operating system and eliminates the need to restart ESP Workload Manager for a seasonal time change. SYSZONE relies on the operating system's time zone being properly maintained.

Operand	Description
SYSZONE(NO)	The system zone shift from z/OS will not be included in the time zone value. NO is the default.
TIMEREF	Sets up the way to obtain local time. If TIMEREF is not specified, the local time the operating system reports will be used. TIMEREF is processed only on a TIMEZONE statement for LOCAL time zone. **Note:** The TIMEZONE statement with the TIMEREF parameter should be placed at the top of initialization statement list to set up the correct local time before other statements are processed.
SYSTIME	The system's local time is used as reference for calculating ESP Workload Manager's local time. SYSTIME is the default.
GMT	The reference for calculating ESP Workload Manager's local time is GMT, as the operating system reports
+|-*hh.mm*	The offset in hours and minutes to be applied to the reference time in order to obtain local time

Usage notes

You can use the LTZONE command to display the current time zone settings.

Note: It is critical in multiple-system configurations that the time zone accurately reflect the z/OS view of the local time. Also, UTC, GMT, and Z must be equal.

Examples

In the following example, Eastern Daylight Time is defined as zone 5 (four hours West of UTC) on a system that does not use the system zone shift:

```
TIMEZONE 5 EDT 4.0W
```

In the following example, the TIMEZONE statement defines all time zones values for Canada for a system based in the Eastern Standard Time zone and using the time zone shift:

```
TIMEZONE 0  LOCAL    0        SYSZONE(YES)
TIMEZONE 16 EASTERN  0        SYSZONE(YES)
TIMEZONE 17 CENTRAL  1.0W     SYSZONE(YES)
TIMEZONE 18 MOUNTAIN 2.0W     SYSZONE(YES)
TIMEZONE 19 PACIFIC  3.0W     SYSZONE(YES)
TIMEZONE 20 ATLANTIC 1.0E     SYSZONE(YES)
TIMEZONE 21 NFNDLAND 1.30E    SYSZONE(YES)
```

The following example simulates Pacific time for ESP Workload Manager running on a machine configured for Eastern time:

```
TIMEZONE 0  LOCAL   3.0W  SYSZONE(YES) TIMEREF(SYSTIME-3.00)
TIMEZONE 16 EASTERN 0     SYSZONE(YES)
TIMEZONE 19 PACIFIC 3.0W  SYSZONE(YES)
```

TPAPPL: Interact with remote partner application

Note: You can also issue the TPAAPPL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The TPAPPL initialization parameter manages interactions with remote partner applications.

Where defined

Master

TPPARM data set

Syntax

```
TPAPPL applname {DISPLAY}
                {START}
                {STOP}
                {ADDCONV}
                {DELCONV}
                {SHOWHELD}
                {DELHELD}
                {SUSPEND}
                {RESUME}
                {RELEASE} [LUNAME(luname)]
                          [LOGMODE(logonmode)]
                          [TRAN(trans_id)]
                          [DEFER]
```

Operand	Description
applname	The name of the remote application
DISPLAY	Displays the current status of the connection, including all conversations
START	Establishes a connection to the specified application
STOP	Terminates the connection
ADDCONV	Adds a conversation with the remote application
DELCONV	Deletes an existing conversation
SHOWHELD	Displays any held transactions
DELHELD	Deletes any held transactions
SUSPEND	Temporarily suspends interaction with the remote application
RESUME	Resumes activity on a suspended connection

Operand	Description
RELEASE	Releases any held transactions
LUNAME(*luname*)	The name of the VTAM LU. If omitted, *luname* defaults to blanks. Use LUNAME only with the START keyword.
LOGONMODE(*logonmode*)	The logon mode for this application. If omitted, *logonmode* defaults to blanks. Use LOGONMODE only with the START keyword.
TRAN(*trans_id*)	The name of the transaction ID. If omitted, *trans_id* defaults to blanks. Use TRAN only with the RELEASE, DELHELD and SHOWHELD keywords.
DEFER	Performs the specified operation in deferred mode

Examples

Initiating a TP Server connection

The following example shows that a connection is being requested with the TP application ESP_MONTREAL, whose LU name is ESPMTL. The logon mode will default to the value specified on the LOCAPPL initialization statement.

```
TPAPPL ESP_MONTREAL START LUNAME(ESPMTL)
```

Displaying held transactions

The following example requests a display of how many transactions are being held for TP application ESP_MONTREAL:

```
TPAPPL ESP_MONTREAL SHOWHELD
```

Releasing held transactions

The following example requests that all held transactions for the TP application ESP_TORONTO be released:

```
TPAPPL ESP_TORONTO RELEASE
```

Stopping a connection

The following example requests that a connection with the TP application ESP_TORONTO stop:

```
TPAPPL ESP_TORONTO STOP
```

TPCKPT: Allocate the TP Server checkpoint data set

Purpose

The TPCKPT initialization parameter specifies the name of and, optionally, clears and reformats the TP Server checkpoint data set.

Where defined

Master

TPPARM data set

Syntax

`TPCKPT DSNAME(`*`ckptdsn`*`) [CLEAR]`

Parameter	Description
DSNAME(*ckptdsn*)	The name of the checkpoint data set
CLEAR	Clears and reformats the checkpoint data set

Example

The following example shows you how to define a TP Server checkpoint data set as ESP.TP.CKPT:

```
TPCKPT DSNAME(ESP.TP.CKPT)
```

TPRETRY: Set interval to retry contacting an LU

Note: You can also issue the TPRETRY initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The TPRETRY initialization parameter sets and displays the interval at which attempts will be made to contact an LU that is down.

Where defined

Master

TPPARM data set

Syntax

```
TPRETRY applname [INTERVAL(nn)]
```

Parameter	Description
applname	The name of the remote application
INTERVAL(*nn*)	The desired interval in minutes. If not specified, the current value appears.

Example

```
TPRETRY ESP_TORONTO INTERVAL(10)
```

TRACE: Activate trace mechanism

Note: You can also issue the TRACE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The TRACE initialization parameter activates the trace facility and allows trace options to be set. TRACE is used for diagnostic purposes.

Where defined

Master and proxy

Syntax

```
TRACE [SET(id[:id][,…])]
      [RESET(id[:id][,…])]
      [SWITCH|CLOSE|OPEN|STATUS]
      [REUSE]
```

Operand	Description
SET*(id)*	Specifies one or more record identifiers you want to trace
RESET*(id)*	Specifies one or more record identifiers you no longer want to trace
SWITCH	Requests the data set currently in use for the trace facility be freed. The next trace data set will activate automatically. With this option, data sets activate in the sequence in which they were first defined. When the last data set in the sequence closes, the first trace data set to be defined is used.
CLOSE	Requests closure of the trace data set currently in use but prevents switching to the next trace data set. The buffers defined using TRACEDEF automatically continue to hold trace data in the core until a TRACE OPEN command is issued.
OPEN	Requests opening of the trace data set that was most recently active. Use OPEN after TRACE CLOSE is issued to reactivate the same trace data set.
STATUS	Requests information to indicate where the trace facility is active. This information includes how many records were written since trace was activated, the current trace data sets in use, and which data set is currently active.
REUSE	The data set currently in use for the trace facility to be checkpointed so that subsequent writes to it begin at the checkpoint rather than at the start of the data set

Usage notes

The TRACE initialization parameter is useful as a problem-solving tool. On occasion, CA Technical Support will ask you to set a specific trace record ID to access information that will help with troubleshooting.

If you wish to capture only records relating to Event processing (for example, type 601), use the ESPPARM AUDITLOG DD name in the ESP Workload Manager started task procedure. This allows the use of a pre-allocated sysout Event activity and eliminates the need to use the TRACEDEF and TRACE commands.

Examples

The following example activates the trace facility and specifies that record IDs 602 through 604, and 607 should be traced:

```
TRACE SET(602:604,607)
```

In the following example, the current data set used for the trace facility is checkpointed before switching to the next trace data set. When the current trace data set is later reused, records are written starting at the checkpoint.

```
TRACE SWITCH REUSE
```

In the following example, the trace is turned off for record IDs 602 through 604:

```
TRACE RESET(602:604)
```

TRACEDEF: Define trace data sets

Note: You can also issue the TRACEDEF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The TRACEDEF initialization parameter identifies the data sets to be used to record information collected by the TRACE facility. TRACEDEF is used for diagnostic purposes.

Where defined

Master and proxy

Syntax

```
TRACEDEF DSN(dsname[,dsname]...) [BUF(size,count)]
```

Operand	Description
DSN(*dsname*)	The name of one or more data sets to be used as trace data sets. Specifying multiple trace data sets allows you to use the SWITCH operand with TRACE to free up one trace data set and switch to another. DCB=(RECFM=V,LRECL=4096,BLKSIZE=4100)
BUF(*size*,*count*)	The buffers you want to use for the trace data sets you define. Size specifies the buffer size and count identifies how many buffers are required. *size*,*count* defaults to 4096,4, which is four buffers, each one 4096 bytes.

Usage notes

You must use TRACEDEF to define trace data sets (and optionally the buffers) before you can use the TRACE initialization parameter or command. You do not have to specify any DCB attributes when you initially allocate the data sets because ESP Workload Manager does this automatically.

The buffers you define continue to hold trace data until each one becomes full. At this point, the data is automatically written to the trace data set and another buffer is used.

Example

To define two trace data sets (TRACE1 and TRACE2) that each have three 23400-byte buffers, type

```
TRACEDEF DSN(ESPCYB.TRACE1 ESPCYB.TRACE2) BUF(23400,3)
```

TRACKDEF: Specify tracking definitions

Note: You can also issue the TRACKDEF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the TRACKDEF initialization parameter to specify tracking definitions in a job-tracking definition table.

Where defined

Master and proxy

Job-tracking definition table (JTDT) data set

Syntax

```
TRACKDEF [NAME(string)]
         [JOB]
         [STC]
         [TSU]
         [RACID(string)]
         [PGMR(string)]
         [CLASS(classid)]
         [ACCOUNT1(string)]
         [ACCOUNT2(string)]
         [ACCOUNT3(string)]
         [ACCOUNT4(string)]
         [NOTRACK|MODEL(modelname)]
```

Operand	Description
NAME(*string*)	A one-to-eight-character job name to be matched on. Wildcard characters asterisk (*) and hyphen (-) are used to perform masking. (See “Representing characters with wildcards” on page 64.) You can also use wildcards defined with the WILDCARD initialization parameter described in “WILDCARD: Define wildcards for JTDT” on page 537. NAME can also be specified as JOBNAME.
JOB	Jobs are tracked. If JOB, STC or TSU is not specified, all three entities are tracked.
STC	Started tasks are tracked. If JOB, STC or TSU is not specified, all three entities are tracked.
TSU	TSO users are tracked. If JOB, STC or TSU is not specified, all three entities are tracked.
RACID(*string*)	The security system user ID associated with the job

Operand	Description
PGMR(*string*)	The programmer name field associated with the job. All 20 character positions can be checked.
CLASS(*classid*)	The job's execution class. *classid* must be up to eight characters long in a JES3 environment.
ACCOUNT1(*string*)	The job's first account number. Only the first eight characters are checked.
ACCOUNT2(*string*)	The job's second account number. Only the first eight characters are checked.
ACCOUNT3(*string*)	The job's third account number. Only the first eight characters are checked.
ACCOUNT4(*string*)	The job's fourth account number. Only the first eight characters are checked.
NOTRACK	Jobs matching the TRACKDEF entry are not tracked. NOTRACK is the default.
MODEL(*modelname*)	The name of the tracking model to be associated with a job, started task or TSO user. If a job is tracked, you must code the MODEL operand. You can override this operand using the MODEL statement for a job in an Application.

Usage notes

A job-tracking definition table identifies the characteristics of the jobs you want ESP Workload Manager to track. ESP Workload Manager can track jobs based on job name, execution class, programmer name, account number, job type or the user ID associated with the job.

The job-tracking definition table allows the following:

- You can define your own wildcard characters in the table with the WILDCARD initialization parameter described on "WILDCARD: Define wildcards for JTDT" on page 537. These characters give you great flexibility in defining the property of the job you want ESP Workload Manager to use as the job-tracking parameter.
- You are not restricted to a job name or prefix when defining the tracking parameter. Instead, you can choose from a larger set of job properties when defining the parameter. For example, you can choose the job's execution class or the name of the programmer.

A job-tracking definition table consists of a set of WILDCARD and TRACKDEF initialization parameters, respectively, in a sequential data set or in a PDS member. The use of WILDCARD is optional.

The order of the TRACKDEF initialization parameters is important. When tracking data is received for a job, ESP Workload Manager scans the TRACKDEF entries in the sequence until it finds a match. ESP Workload Manager then takes the action that entry specifies. The entry can identify whether the job is tracked or not. If the job is tracked, the default tracking model for the job is specified. If no matching entry is found, the job is not tracked.

To track STCs or TSUs, you must use the TRACKOPT command or initialization parameter to identify the STCs or TSUs you want tracked. You can also use the TRACKDEF initialization parameters to identify which STCs or TSUs you want tracked.

In addition, you can test a job-tracking definition table using ESP Workload Manager ISPF interface—Option M.4 from the ESP Workload Manager Main Menu.

Examples

Track all jobs

In the following example, all jobs are tracked using tracking model MODEL1:

```
TRACKDEF NAME(-) MODEL(MODEL1)
```

Track jobs by name

In the following example

- The first job-tracking definition initialization parameter indicates ESP Workload Manager uses tracking model PRODJOBS to track all jobs that start with the letter J.
- The second initialization parameter indicates ESP Workload Manager uses tracking model TESTJOBS to track all jobs that start with the letter U.

```
TRACKDEF JOB NAME(J–) MODEL(PRODJOBS)
```

```
TRACKDEF JOB NAME(U–) MODEL(TESTJOBS)
```

This table shows whether ESP Workload Manager will track specific jobs, according to the above job-tracking definitions:

NAME	TYPE	TRACKING	MODEL
JTOP5200	JOB	Yes	PRODJOBS
UTOP1492	JOB	Yes	TESTJOBS
ABC	JOB	No	NA
JCICS100	STC	No	NA

Track class T jobs, regardless of job name, using tracking model MODEL1

```
TRACKDEF NAME(-) CLASS(T) MODEL(MODEL1)
```

Track all class P jobs using the tracking model PROD

```
TRACKDEF CLASS(P) MODEL(PROD)
```

Track all started tasks with the prefix CICS using the model JOBMON

```
TRACKDEF STC NAME(CICS-) MODEL(JOBMON)
```

Track all jobs

```
TRACKDEF JOB NAME(-) MODEL(DEFAULT)
```

Track all jobs using a model based on the job name

The following example tracks all jobs whose names begin with two alphabetic characters followed by five numeric characters:

```
TRACKDEF JOB NAME(-) MODEL(NAME(1:2),MODEL)
```

ESP Workload Manager derives the name of the job-tracking model it uses by concatenating the first two characters of the job name with the string MODEL. For example, job-tracking model DXMODEL tracks job DX123245.

Do not track jobs with a programmer name field starting with CYBED

```
TRACKDEF PGMR(CYBED-) NOTRACK
```

Track jobs starting with X using the model XSYS

```
TRACKDEF JOB NAME(X-) MODEL(XSYS)
```

Note: See also the example included in “Defining a job-tracking definition table” on page 65.

Related information

For information on job-tracking definition tables, see “Using Job-Tracking Definition Tables” on page 65.

For information on loading the job-tracking definition table (JTDT), see “LOADJTDT: Load job-tracking definition table” on page 386.

For information on defining wildcard characters used in a job-tracking definition table, see “WILDCARD: Define wildcards for JTDT” on page 537.

For information on defining a job-tracking model, see the DEFTM command in the *ESP Workload Manager Reference Guide*.

For information on displaying the status of tracked jobs, see the LJ command in the *ESP Workload Manager Reference Guide*.

For information on specifying job-tracking options, see the TRACKOPT command in the *ESP Workload Manager Reference Guide*.

TRACKING: Control tracking facility

Note: You can also issue the TRACKING initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The TRACKING initialization parameter enables or disables the tracking facility.

Where defined

Master and proxy

Syntax

```
TRACKING [COLLECT|NOCOLLECT|]
         [STORE|NOSTORE]
```

Operand	Description
COLLECT	Requests that SMF recording be activated. SMF tracking data is stored in the TRAKFILE and the history data sets are updated (unless NOSTORE is specified). COLLECT is the default. **Note:** COLLECT is meaningless on a master that has SMFINIT OFF.
NOCOLLECT	Requests that SMF recording be deactivated. No SMF data is collected, no tracking data is written to the TRAKFILE and the history data sets are not updated.
STORE	Requests that tracking data from SMF be stored in the TRAKFILE and that the history data sets be updated. STORE is the default.
NOSTORE	Requests that the tracking processor be quiesced. No tracking data is written to the TRAKFILE and the history data sets are not updated. The data is temporarily buffered to the checkpoint data set until STORE is requested. **Note:** NOSTORE is meaningless on a proxy.

Usage notes

If no options are specified, the current job-tracking options appear (the NOLOG field of the display is reserved for future use). If an option is specified, it is added to the current tracking options.

If you wish to perform normal job-tracking functions, the COLLECT and STORE operands should be active.

The COLLECT and NOCOLLECT operands control the collection of SMF data that the job-tracking processor uses to update TRAKFILE and the history data sets. When NOCOLLECT is specified, SMF data is not captured and the tracking is lost.

Use the NOSTORE operand to quiesce the job-tracking processor temporarily, suspending all job-tracking functions, including Job Monitoring. The NOSTORE operand is useful when you need to perform maintenance on ESP Workload Manager data sets. NOSTORE buffers tracking data to the checkpoint data set until STORE is requested.

Note: The tracking processor should not be left in the NOSTORE mode for long periods of time because the checkpoint data set could fill up causing tracking data to be lost.

When STORE is specified to bring the job-tracking processor out of quiesced state, any data that was buffered in the checkpoint data set is written to the TRAKFILE and the history data sets are updated.

Example

In the following example, all normal job-tracking options are set:

```
TRACKING COLLECT STORE
```

TRACKOPT: Set tracking options

Note: You can also issue the TRACKOPT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The TRACKOPT initialization parameter sets various tracking options.

Where defined

Master and proxy

Syntax

```
TRACKOPT [STC|NOSTC]
         [TSU|NOTSU]
         [SYSMSGS|NOSYSMSGS]
         [MASTER|PROXY]
         [JAT|NOJAT]
         [TRACK_PURGE|NOTRACK_PURGE]
         [POST_OLDEST|NOPOST_OLDEST]
```

Operand	Description
STC	Specifies that started tasks should be tracked
NOSTC	Requests that tracking of started tasks should no longer occur. NOSTC is the default.
TSU	Specifies that TSO users should be tracked
NOTSU	Requests that tracking of TSO users should no longer occur. NOTSU is the default.
SYSMSGS	Specifies that system messages should be intercepted
NOSYSMSGS	Requests that interception of system messages should no longer occur. NOSYSMSGS is the default.
MASTER	Identifies this as the master system for Application processing
PROXY	Identifies this as a proxy system
JAT	Specifies that a Job Authorization Table is used
NOJAT	Requests that a Job Authorization Table no longer be used. NOJAT is the default.
TRACK_PURGE	Jobs are tracked through the OUTPUT pnode. TRACK_PURGE is the default.

Operand	Description
NOTRACK_PURGE	Jobs are not tracked through the OUTPUT pnode.
POST_OLDEST	Jobs are posted as complete in an Application's oldest active generation.
NOPOST_OLDEST	Jobs are posted as complete in all of an Application's generations. NOPOST_OLDEST is the default.

Usage notes

TRACKOPT should be specified on each system in a multi-access spool environment

TRACKOPT can be specified as an initialization parameter as well as from a console. When issued from a console, the options in the initialization parameters will be overridden. The information is saved in the checkpoint data set, which means it is retained across IPLs. However, for a cold start, any information in the initialization parameters will be used.

When an EXTERNAL or MANUAL job completes and multiple generations of the Application exist, ESP Workload Manager must decide which generation of an Application to post the job complete in. Use the POST_OLDEST or NOPOST_OLDEST operands to control this.

Examples

The following example displays current tracking options:

```
TRACKOPT
```

The following example turns on the tracking of started tasks (`STC`):

```
TRACKOPT STC
```

The following example turns on the tracking of started tasks (`STC`) and system messages (`SYSMSGS`):

```
TRACKOPT STC SYSMSGS
```

The following example turns off the tracking of started tasks (`STC`):

```
TRACKOPT NOSTC
```

The following example specifies this system as the master:

```
TRACKOPT MASTER
```

The following example specifies this system as a proxy:

```
TRACKOPT PROXY
```

To track started tasks to completion but not through purge, and specify this system as the master, type

```
TRACKOPT STC MASTER NOTRACK_PURGE
```

TRAKFILE: Define tracking data set

Purpose

The TRAKFILE initialization parameter specifies the name of the tracking data set.

Where defined

Master and proxy

Syntax

```
TRAKFILE dsname [SHR|NOSHR]
                [NOREADONLY|READONLY]
                [CACHE(nn)|1]
```

Operand	Description
dsname	The name of the TRAKFILE data set
SHR	The data set is shared and RESERVE/DEQ is performed.
NOSHR	The data set is not shared. NOSHR is the default.
NOREADONLY	Allows ESP Workload Manager to update the TRAKFILE. NOREADONLY is the default.
READONLY	Prevents ESP Workload Manager from updating the TRAKFILE
CACHE(*nn*)	The number of megabytes of memory for caching. The default value is 1. Cache is not enabled by default. **Note:** The CACHE operand is ignored on proxy systems.

Example

In the following example, the TRAKFILE data set is ESP.TRAKFILE and it is shared:

```
TRAKFILE ESP.TRAKFILE SHR
```

TRDFLT: Specify manual trigger default

Note: You can also issue the TRDFLT initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The TRDFLT initialization parameter specifies the default to be used when an Event is triggered manually.

Where defined

Master and proxy

Syntax

```
TRDFLT [ADD|REPLACE]
```

Operand	Description
ADD	Specifies that all manual triggers for Events should be performed in addition to the next scheduled execution
REPLACE	Specifies that all manual triggers should replace an Event’s next scheduled execution. When using the TRIGGER subcommand for specific Events, you can override REPLACE with the ADD operand. REPLACE is the default.

Example

```
TRDFLT ADD
```

TSOSEND: Control Sending of TSO Messages

Note: You can also issue the TSOSEND initialization parameter as a command. For information about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The TSO initialization parameter controls the sending of TSO messages to users.

Where defined

Master and proxy

Syntax

```
TSOSEND [LOGON|IMMEDIATE|DISABLE]
```

Operand	Description
LOGON	If the user is logged on, send the message. If the user is not logged on, hold the message in SYS1.BRODCAST until the user logs on. LOGON is the default.
IMMEDIATE	If the user is logged on, send the message. Otherwise, discard the message.
DISABLE	Suppresses all TSO messages

Usage notes

The TSOSEND initialization parameter applies either when no mailbox is specified in the Event definition or when a mailbox is specified and a TSOUSER parameter is included in the mailbox definition. The TSOSEND command does not apply to messages that the NOTIFY and SEND commands initiate.

For information on setting up mailboxes, see “MAILLIST Data Set” on page 49.

You can use the TSOSEND initialization parameter to suppress all TSO user messages and the MAILLOG parameter to send them to the mail log instead. For information on the MAILLOG initialization parameter, see “MAILLOG: Control the Mail Log Sysout” on page 398.

TSOUSER: Include user ID in mailbox

Purpose

The TSOUSER initialization parameter includes a user ID in a mailbox.

Where defined

Master and proxy

MAILLIST data set

Syntax

```
TSOUSER {userid |(userid userid …)} [SYSID(system_id)]
```

Operand	Description
userid	The user ID that is to receive messages. If you specify more than one user ID, separate each with a space and enclose the list in parentheses.
SYSID(*system_id*)	The system ID where messages are to be sent. If you do not specify a system ID, messages are sent only to the system where the message is generated.

Usage notes

For information on the MAILLIST data set, see “MAILLIST data set” on page 15.

The TSOUSER initialization parameter is a component of the MAILLIST data set as follows:

```
SMTPPARM CLASS(A) JOBNAME(SMTP)
MAILBOX  PAYROLL MAXLINES(0)
TSOUSER CYBPAY1 SYSID(SYSC)
EMAIL paymaster@company.com
EMAIL payservice@payservice.com
MAILBOX CYBACCOUNTING MAXLINES(300)
TSOUSER (CYBACC1 CYBACC2) SYSID(SYSA)
```

USERDEF: Identify user definition data set

Purpose

The USERDEF initialization parameter specifies the name of the user definition data set.

Where defined

Master and proxy

Syntax

```
USERDEF dsname [JOURNAL|NOJOURNAL]
               [BACKUPDATASET(bkupdsn)]
               [SHR|NOSHR]
```

Operand	Description
dsname	The name of the user definition data set
JOURNAL	A record is written to SMF each time an update, addition or deletion occurs to the user definition data set. For details, see "SMFREC: Specify SMF record number" on page 485.
NOJOURNAL	Requests that no records be written to SMF when the user definition data set is updated. Use this keyword to cancel a previous JOURNAL request. NOJOURNAL is the default.
BACKUPDATASET(*bkupdsn*)	The name of a non-VSAM data set the user definition data set is backed up to
SHR	The data set is shared.
NOSHR	The data set is not shared. NOSHR is the default.

Usage notes

The backup data set should be allocated with about 50 percent of the space assigned to the VSAM data set and a secondary allocation of 10 percent. You do not need to assign DCB attributes on the allocation because ESP Workload Manager sets these.

You can set the automatic backup of the user definition data set. For instructions on how to do it, see "BKUPUDEF: Back up user definition data set" on page 260.

You do not need the USERDEF initialization parameter for installations using SAF.

Example

In the following example, the user definition data set is ESP.USERDEF and is shared. The backup data set name is ESP.BACKUP.USERDEF.

```
USERDEF ESP.USERDEF BACKUPDATASET(ESP.BACKUP.USERDEF)
```

USERMOD: Set user modification options

Note: You can also issue the USERMODE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide.*

Purpose

The USERMOD initialization parameter defines the user modifications to be implemented.

Where defined

Master and proxy

Syntax

```
USERMOD [SET(usermodid)]
        [RESET(usermodid)]
        [LIST]
```

Operand	Description
SET(*usermodid*)	Turns on the specified usermods. *usermodid* can be a usermod number, list of usermod numbers or a range of usermod numbers. The value for each usermod is between 1 and 255.
RESET(*usermodid*)	Turns off the usermod specified in the brackets. *usermodid* can be a usermod number, list of usermod numbers or a range of usermod numbers. The value for each usermod is between 1 and 255.
LIST	Lists the active usermods

Usage notes

User modification status is preserved across a ESP Workload Manager restart, but not an IPL.

You can specify both SET and RESET in the same USERMOD statement. In this case, SET is processed before RESET.

Examples

The following example turns on usermods 1, 5, 6, 7, 18, 22, 23, 24, 25, 26, and 27:

```
USERMOD SET(1,5:7,18,22:27)
```

The following example shows both SET and RESET in the same initialization parameter:

```
USERMOD SET(5:9) RESET(7)
```

Because SET is processed before RESET, the above example turns on usermods 5,6,8, and 9 and turns off usermod 7.

USERVFYX: Identify user verification exit

Note: You can also issue the USERVFYX initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The USERVFYX initialization parameter requests that the ESP UVFX exit be invoked to identify a user.

Where defined

Master and proxy

Syntax

```
USERVFYX exitname [DYNAMIC]
```

Operand	Description
exitname	The name of the initialization exit load module. *exitname* is the module created when you assemble the sample program in the ESPUVFX member of the distribution library.
DYNAMIC	The load module is dynamically loaded each time a cross-memory request is processed.

Usage notes

The user verification exit refers to the necessary control blocks to identify the user. The exit is invoked by the ESP Workload Manager cross-memory request processor executing in the user's address space. The USERVFYX initialization parameter is useful in installations where a non-RACF type of security package is in use, such as ACF2.

Example

To specify a user verify exit called ESPUVFX and that the load module be dynamically loaded, type

```
USERVFYX ESPUVFX DYNAMIC
```

WILDCARD: Define wildcards for JTDT

Note: You can also issue the WILDCARD initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the WILDCARD initialization parameter to define wildcard characters for use in defining tracked jobs. These characters are in addition to the asterisk and hyphen wildcard characters.

Where defined

Job-tracking definition table (JTDT) data set

Syntax

```
WILDCARD char
         string
```

Operand	Description
char	Wildcard character
string	A string containing all the valid characters that the wildcard can match

Usage notes

The wildcard character can be any printable character, with the exception of the comma, blank, left or right parentheses, semicolon or quotation mark. A hyphen between two characters indicates a range of valid characters, starting with the character on the left and extending through the EBCDIC sequence to the character on the right, inclusively.

Use of the WILDCARD initialization parameter is optional. You can use it when you have strict naming standards for jobs.

Examples

The following is an example of a job-tracking definition table:

```
WILDCARD # 0–9 /* NUMERICS */
WILDCARD $ A–Z /* ALPHABETICS */
WILDCARD + 0–9A–Z /* ALPHANUMERIC */
TRACKDEF JOB NAME(DUMMYJOB) NOTRACK
TRACKDEF JOB NAME(PAY–) MODEL(PAYMODEL)
TRACKDEF JOB NAME($$$$####) MODEL(PRODJOBS)
```

In the above example

- The wildcard character number sign (#) matches all numeric characters.
- The wildcard character dollar sign ($) matches the alphabetic characters, A through Z, inclusively.
- The wildcard character plus sign (+) matches the alphanumeric characters, A through Z, inclusively, and the digits 0 through 9, inclusively.
- The first job-tracking definition statement indicates ESP does not track a job called DUMMYJOB.
- The second job-tracking definition statement indicates ESP uses model PAYMODEL to track jobs where names begin with PAY.
- The third job-tracking definition statement indicates ESP uses job-tracking model PRODJOBSMODEL to track jobs whose names consist of four alphabetic characters followed by four numeric characters.

Related information

For information on job-tracking definition entries in a job-tracking definition table, see “TRACKDEF: Specify tracking definitions” on page 521.

For information on job-tracking definition tables, see “Using Job-Tracking Definition Tables” on page 65.

For information on loading job-tracking definition tables (JTDT), see “LOADJTDT: Load job-tracking definition table” on page 386.

WKSTART: Define start of week

Note: You can also issue the WKSTART initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The WKSTART initialization parameter specifies the day the week should start on.

Where defined

Master and proxy

Syntax

```
WKSTART dayname
```

Operand	Description
dayname	The name of the day you want the week to start on. If you do not use this operand, the start of the week defaults to Sunday.

Usage note

You can override the *dayname* value when you define a calendar.

Example

To identify that ESP Workload Manager should consider Monday as the first of the week, type

```
WKSTART MONDAY
```

WOBDEF: Define workload objects

Note: You can also issue the WOBDEF initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the WOBDEF initialization parameter to load workload object modules.

Where defined

Master and proxy

Syntax

```
WOBDEF LOAD(module[,module]...)
```

Operand	Description
module	The name of the module to support a specific type of distributed workload. When listing more than one module, separate the names with commas or spaces.

Workload object modules

From the following table, select module names from the “Module” column to specify in the WOBDEF initialization parameter.

If you set up a CSF freeform filter (see the CSF information in the *ESP Workload Manager Operator’s Guide*), you specify the short name.

Workload object type	Short name	Module	Description
AGENT_MONITOR	AM	CYBESOAM	Agent monitor
AIX_JOB	AX	CYBESOAX	AIX Agent
APPLEND	AE	CYBESOAE	APPLEND (End of Application)
AS400_JOB	A4	CYBESOA4	OS/400 Agent
BDC_JOB	BI	CYBESOBI	BDC processor
CPU_MON	CM	CYBESOCM	CPU usage monitor
DATA_OBJECT	DA	CYBESODA	Data object
DB_JOB	DM	CYBESODM	Database monitor (perform SQL queries and updates against a database)
DB_MON	MO	CYBESOMO	Database monitor (monitor a table for changes to the number of rows)
DBSP_JOB	ST	CYBESOST	Database stored procedure

Workload object type	Short name	Module	Description
DB_TRIG	TR	CYBESOTR	Database trigger job (monitor a table for rows added, deleted or updated)
DISK_MON	HM	CYBESOHM	Disk usage monitor
DSTRIG	DT	CYBESODT	Data set trigger
EJB_JOB	SB	CYBESOSB	J2EE Java Beans
EVENTLOG_MON	LM	CYBESOLM	Windows event log monitor
EXTMON	EX	CYBESOEX	External monitor (monitor a job controlled on an external scheduler, for example ESP dSeries)
FILE_TRIGGER	FM	CYBESOFM	File trigger
FTP_JOB	FT	CYBESOFT	File transfer (perform FTP file operations)
HP3K_JOB	H3	CYBESOH3	HP3000 Agent
HPUX_JOB	HP	CYBESOHP	HP-UX Agent
IP_MON	IM	CYBESOIM	IP monitor
IRIX_JOB	IR	CYBESIR	IRIX Agent
JMSP_JOB	JP	CYBESOJP	J2EE publish messages
JMSS_JOB	JS	CYBESOJS	J2EE subscribe, get messages
LINUX_JOB	LJ	CYBESOLJ	Linux Agent
MPEIX_JOB	MP	CYBESOMP	MPE/iX Agent
NCR_JOB	NC	CYBEONC	NCR Agent
NT_JOB	NT	CYBESONT	Windows NT Agent
OA_JOB	OA	CYBESOOA	Oracle applications.
OPENVMS_JOB	OV	CYBESOOV	OpenVMS Agent.
OS2_JOB	O2	CYBESOO2	OS/2 Agent
PROCESS_MON	PM	CYBESOPM	Process monitor
PS_JOB	PS	CYBESOPS	PeopleSoft Agent
PYRAMID_JOB	PJ	CYBESOPJ	PYRAMID Agent
SAP_JOB	SP	CYBESOSP	R/3 ERP Systems Agent
SAPA_JOB	SA	CYBESOSA	SAP Data Archiving Processor
SAPE_JOB	BE	CYBESOBE	SAP Event Monitor
SAPM_JOB	SL	CYBESOSL	SAP Process Monitor
SEQUENT_JOB	SQ	CYBESOSQ	IBM DYNIX/ptx (SEQUENT) Agent
SERVICE_MON	SM	CYBESOSM	Service monitor
SQL_JOB	QL	CYBESOQL	SQL Command
SUN_JOB	SJ	CYBESOSJ	SOLARIS Agent

Workload object type	Short name	Module	Description
TANDEM_JOB	TA	CYBESOTA	TANDEM NSK Agent
TEXT_MON	TM	CYBESOTM	Text file monitor
UNIX_JOB	UJ	CYBESOUJ	Generic UNIX (supports Linux, TRU64 UNIX, and UNIX System Services)

Example

```
WOBDEF LOAD(CYBESOAM CYBESODT CYBESOTM)
```

WORKDAYS: Specify workdays

Note: You can also issue the WORKDAYS initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The WORKDAYS initialization parameter identifies which days should be considered workdays.

Where defined

Master and proxy

Syntax

```
WORKDAYS (day[,day]...)
```

Operand	Description
day	The list of the days you want to be considered workdays, enclosed in parentheses. Holidays are automatically excluded.

Usage notes

If you do not use the WORKDAYS initialization parameter, the defaults are Monday through Friday, excluding holidays.

You can override the *day* value when you define a calendar.

Example

To define that every day is a workday except Saturdays and holidays, type

```
WORKDAYS (SUN,MON,TUE,WED,THU,FRI)
```

WORKMGR: Provide Enclave support

Note: You can also issue the WORKMGR initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The WORKMGR initialization parameter connects ESP Workload Manager to IBM Workload Manager.

Where defined

Master and proxy

Syntax

```
WORKMGR SUBSYSTEM_TYPE|SST(ESP|subsystemtype)
SUBSYSTEM_NAME|SSN(subsystemname) ENCLAVES|NOENCLAVES
```

Operand	Description

SUBSYSTEM_TYPE(*subsystemtype*)
SST(*subsystemtype*)
	The type of subsystem. The default is ESP. We recommend you do not override this default.
	ESP Workload Manager does not permit IBM subsystems, such as CICS and IMS.

SUBSYSTEM_NAME(*subsystemname*)
SSN(*subsystemname*)
	The name of the ESP Workload Manager subsystem. This is an eight-character field. The default is the ESP subsystem ID, determined by the SUBSYS initialization parameter in the initialization parameter data set. For more information, see "SUBSYS: Set subsystem name (ESP Workload Manager)" on page 491.

ENCLAVES	Support for enclaves is enabled. If you specify the WORKMGR initialization parameter, ENCLAVES is the default.
NOENCLAVES	Support for enclaves is disabled.

Usage notes

When you have defined the subsystem type and the subsystem name, you can disable and enable enclaves from page mode by entering the WORKMGR command.

Example

The following example uses all operands:

```
WORKMGR SST(ESP) SSN(P520) ENCLAVES
```

When ESP Workload Manager is connected to IBM z/OS Workload Manager, all enclaves run under the defined subsystem type and name.

WSSCTL: Set parameters and control message tracing

Note: You can also issue the WSSCTL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The WSSCTL initialization parameter enables you to set the values of the Workstation Server parameters and start message tracing. Message tracing records messages that a Workstation Server receives and sends, and writes them to a sysout file.

Where defined

WSSPARM data set

Syntax

```
WSSCTL {SET PARMS [INTERVAL(seconds)]
                  [PROXYEXP(seconds)]
                  [RCONNINT(seconds)]
                  [RPOLLINT(seconds)]
                  [RRSPWAIT(seconds)]
                  [TCPWAIT(seconds)]
                  [MAXCON_LOIP(connections)]
                  [MAXCON_PERIP(connections)]
                  [MAXCON_TOTAL(connections)]
       {SET TRACE SYSOUT(class)}
       {START TRACE}
```

Operand	Description
SET PARMS	Sets or changes a Workstation Server parameter
INTERVAL(*seconds*)	Sets the maximum time that the Workstation Server waits before performing a scoreboard scan. The default is 5 seconds. For a Consolidated Workstation Server, this value is the same for each separate scoreboard.
PROXYEXP(*seconds*)	Sets the time interval after which each proxy command task times out when waiting for a new command from the Consolidated Workstation Server. The default is 300 seconds.
RCONNINT(*seconds*)	Sets the time interval that each Consolidated Workstation Server waits before retrying to connect to a remote Workstation Server when a connection cannot be established. The default is 10 seconds.

Operand	Description
RPOLLINT(*seconds*)	Sets the time interval that the Consolidated Workstation Server waits before polling each remote Workstation Server in its domain to test whether they are active. The Consolidated Workstation Server skips a turn for remote Workstation Servers that send at least one message before RPOLLINT expires. The default is 300 seconds.
RRSPWAIT(*seconds*)	Sets the time interval after which a Consolidated Workstation Server client command times out when waiting for a response from a remote Workstation Server. The default is 300 seconds.
TCPWAIT(*seconds*)	Sets the maximum time interval that each remote Workstation Server waits TCP/IP operations to complete. The default is 30 seconds.
MAXCON_LOIP(*connections*)	Sets the maximum number of client connections for logons in progress from a particular TCP/IP address. The default is 2.
MAXCON_PERIP(*connections*)	Sets the maximum number of client connections per TCP/IP address. The default is 8.
MAXCON_TOTAL(*connections*)	Sets the maximum number of client connections for the Workstation Server. The default is 250.
SET TRACE	Sets Workstation Server messages tracing
SYSOUT(*class*)	The sysout class of the Workstation Server message-trace print data set. This operand is rejected if the Workstation Server started task JCL contains a TRACELOG DD statement. If you set the class in the initialization parameter, you can start tracing in the ESP Workload Manager client or as a z/OS modify command against the Workstation Server started task.
START TRACE	Starts Workstation Server messages tracing. If specified, precede START TRACE with an WSSCTL SET TRACE SYSOUT(*class*) parameter.

Equivalent EXEC parameter

The following table indicates the equivalence between WSSPARM and EXEC for the Workstation Server started task definition. For details, see “Workstation Server started task” on page 97.

WSSPARM parameters	EXEC parameters
WSSCTL SET PARMS INTERVAL(*seconds*)	INTERVAL(*seconds*)
WSSCTL SET PARMS RPOLLINT(*seconds*)	RPOLLINT(*seconds)*
WSSCTL SET PARMS RCONNINT(*seconds*)	RCONNINT(*seconds*)
WSSCTL SET PARMS RRSPWAIT(*seconds*)	PROXYEXP(*seconds*)
WSSCTL SET PARMS PROXYEXP(*seconds*)	RRSPWAIT(*seconds*)
WSSCTL SET PARMS TCPWAIT(*seconds*)	TCPWAIT(*seconds*)
WSSCTL SET PARMS MAXCON_LOIP(*connections*)	MAXCON_LOIP(*connections*)
WSSCTL SET PARMS MAXCON_PERIP(*connections*)	MAXCON_PERIP(*connections*)

WSSPARM parameters	**EXEC parameters**
WSSCTL SET PARMS MAXCON_TOTAL(*connections*)	MAXCON_TOTAL(*connections*)

WSSCTL SET TRACE SYSOUT(*class*)	SET_WSSTRACE(*class*)
	or
	T_WSSTRACE(*class*)

WSSCTL SET TRACE SYSOUT(*class*)
WSSCTL START TRACE
Note: Two parameter lines are required.
	START_WSSTRACE(*class*)
	or
	S_WSSTRACE(*class*)

Example

```
WSSCTL SET PARMS INTERVAL(60)
WSSCTL SET PARMS RPOLLINT()
WSSCTL SET PARMS RCONNINT()
WSSCTL SET PARMS RRSPWAIT()
WSSCTL SET PARMS PROXYEXP()
WSSCTL SET PARMS TCPWAIT(120)
WSSCTL SET PARMS MAXCON_LOIP(10)
WSSCTL SET PARMS MAXCON_PERIP(100)
WSSCTL SET PARMS MAXCON_TOTAL()
WSSCTL SET TRACE SYSOUT(A)
WSSCTL START TRACE
```

WSSSET: Set Workstation Server connections and timing

Purpose

The WSSSET parameter sets the number of connections allowed and the timing characteristics of those connections.

Where defined

WSSPARM data set

Syntax

```
WSSSET|SET [INTERVAL(5|seconds)]
           [PROXYEXP(300|seconds)]
           [RCONNINT(10|seconds)]
           [RPOLLINT(300|seconds)]
           [RRSPWAIT(300|seconds)]
           [TCPWAIT(30|seconds)]
           [MAXCON_LOIP(2|connections)]
           [MAXCON_PERIP(8|connections)]
           [MAXCON_TOTAL(2508|connections)]
```

Operand	Description
INTERVAL(*seconds*)	The maximum time that the Workstation Server waits before performing a scoreboard scan. The default is 5 seconds. For a Consolidated Workstation Server, this value is the same for each separate scoreboard.
PROXYEXP(*seconds*)	The time interval after which each proxy command task times out when waiting for a new command from the Consolidated Workstation Server. The default is 300 seconds.
RCONNINT(*seconds*)	The time interval that each Consolidated Workstation Server waits before retrying to connect to a remote Workstation Server when a connection cannot be established. The default is 10 seconds.
RPOLLINT(*seconds*)	The time interval that the Consolidated Workstation Server waits before polling each remote Workstation Server in its domain to test whether it is active. The Consolidated Workstation Server skips a turn for remote Workstation Servers that send at least one message before RPOLLINT expires. The default is 300 seconds.
RRSPWAIT(*seconds*)	The time interval after which a Consolidated Workstation Server client command task times out when waiting for a response from a remote Workstation Server. The default is 300 seconds.

Operand	Description
TCPWAIT(*seconds*)	The maximum time interval that each remote Workstation Server waits for TCP/IP operations to complete. The default is 30 seconds.
MAXCON_LOIP(*connections*)	The maximum number of client connections for logons in progress from a particular TCP/IP address. The default is 2.
MAXCON_PERIP(*connections*)	The maximum number of client connections per TCP/IP address. The default is 8.
MAXCON_TOTAL(*connections*)	The maximum number of client connections for the Workstation Server. The default is 2508.

Equivalent EXEC parameter

The following table indicates the equivalence between WSSPARM and EXEC for the Workstation Server started task definition. For details, see "Workstation Server started task" on page 97.

WSSPARM parameters	EXEC parameters
WSSSET|SET INTERVAL(*seconds*)	INTERVAL(*seconds*)
WSSSET|SET RPOLLINT(*seconds*)	RPOLLINT(*seconds*)
WSSSET|SET RCONNINT(*seconds*)	RCONNINT(*seconds*)
WSSSET|SET PROXYEXP(*seconds*)	PROXYEXP(*seconds*)
WSSSET|SET RRSPWAIT(*seconds*)	RRSPWAIT(*seconds*)
WSSSET|SET TCPWAIT(*seconds*)	TCPWAIT(*seconds*)
WSSSET|SET MAXCON_LOIP(*connections*)	MAXCON_LOIP(*connections*)
WSSSET|SET MAXCON_PERIP(*connections*)	MAXCON_PERIP(*connections*)
WSSSET|SET MAXCON_TOTAL(*connections*)	MAXCON_TOTAL(*connections*)

Example

You can use the following example as a template. When you do not want to use the default, fill in the parentheses with different values. Here the INTERVAL and TCPWAIT values have been changed.

```
WSSSET  INTERVAL(60)          /* Default:    5 seconds */
WSSSET  RPOLLINT()            /* Default:  300 seconds */
WSSSET  RCONNINT()            /* Default:   10 seconds */
WSSSET  RRSPWAIT()            /* Default:  300 seconds */
WSSSET  PROXYEXP              /* Default:  300 seconds */
WSSSET  TCPWAIT(120)          /* Default:   30 seconds */
WSSSET  MAXCON_LOIP()         /* Default:    2 tasks   */
WSSSET  MAXCON_PERIP()        /* Default:    8 tasks   */
WSSSET  MAXCON_TOTAL()        /* Default:  250 tasks   */
```

XCF SET SERVICE: Specify a fixed interval

Note: You can also issue the XCF SET SERVICE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The XCF SET SERVICE initialization parameter specifies a time-out interval for XCF service sends and receive acknowledgements. You can also use it to specify a wait-time after which an XCF service restarts following a suspension.

Where defined

Master and proxy

Syntax

```
XCF SET|T SERVICE TRACKING|DSTRIG|ROUTING|SCORBED
    [TIMER(timer)] [SUSPEND(time)]
```

Operand	Description
TIMER(*timer*)	Specifies a time length after which an XCF operation (XCF SEND or XCF RECEIVE ACK (acknowledgement)) times out. The time-out interval must be either 0 or a number within the range 30-1800 (seconds). A value of 0 indicates the default XCF time-out values apply. Usually the default time-out value is 60 (seconds), but in some instances it may be higher.
SUSPEND(*time*)	Specifies a time length that an XCF service waits before restarting after the XCF service has been suspended. The wait-time is within the range 0-3600 (seconds). The default suspend time for an XCF service is 60 seconds (one minute). If the value is set to 0 and an XCF service is suspended, it will be for an indefinite period of time, until it is manually started or stopped with an XCF START SERVICE or XCF STOP SERVICE command.
TRACKING	Job-tracking records are transmitted to the master from a proxy via XCF.

Operand	Description
DSTRIG	Data set triggering records are transmitted to the master from a proxy via XCF.
ROUTING	ESP Workload Manager users to connect to a proxy on any system in the sysplex and can route ESP Workload Manager commands to the master. **Note:** ROUTING requires HAO or Service Governor.
SCOREBD	ESP Workload Manager users connected to a proxy on any system in the sysplex can view the scoreboard and receive subsequent scoreboard updates. **Note:** SCOREBD requires HAO or Service Governor.

Usage notes

While an XCF service is suspended, it is possible to issue the XCF STOP and XCF START commands manually. If a manual command is issued on a suspended XCF service, it will not automatically resume.

The TIMER and SUSPEND operands are optional. If you do not specify either operand, the command response displays the TIMER and SUSPEND values for the specified XCF service.

Examples

```
XCF SET SERVICE TRACKING SUSPEND(90)

XCF SET SERVICE SCOREBD SUSPEND(120)
```

Related information

For more information about all the XCF services, see the *ESP Workload Manager User's Guide.*

For the complete set of XCF commands available, see the *ESP Workload Manager Reference Guide*.

XCF SET TRACE: Define a sysout class

Note: You can also issue the XCF SET TRACE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the XCF SET TRACE initialization parameter to define a sysout class for trace data output. Although you may code XCF SET TRACE in the initialization parameter data set or enter it as a command, we recommend that you put it in your initialization parameter data set.

Where defined

Master and proxy

Syntax

```
XCF {SET|T} {TRACE|TR} {SYSOUT|S}(class)
```

Operand	Description
SYSOUT(*class*)	The output class for sysout

Example

The following example shows you the short form of the SET TRACE command to set trace data to a sysout class of X:

```
XCF T TR S(X)
```

Related information

For more information about the XCF Trace facility, see the *ESP Workload Manager User's Guide.*

For the complete set of XCF commands available, see the *ESP Workload Manager Reference Guide*.

XCF START SERVICE: Enable XCF connection

Note: You can also issue the XCF START SERVICE initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Use the XCF START SERVICE initialization parameter to start XCF services. XCF services are registered and activated via the XCF START SERVICE initialization parameter in the initialization parameter data set. Although you may enter XCF as a command, to enable the XCF connection you must code the XCF START SERVICE initialization parameter in the initialization parameter data set.

Where defined

Master and proxy

Syntax

```
XCF {START|S} {SERVICE|SERV|S} [TRACKING]
                               [DSTRIG]
                               [ROUTING]
                               [SCOREBD]
```

Operand	Description
XCF START	Enables the XCF connection for the XCF service specified
TRACKING	Enables job-tracking records to be transmitted to the ESP Workload Manager master from an ESP Workload Manager proxy via XCF
DSTRIG	Enables data set triggering records to be transmitted to the ESP Workload Manager master from an ESP Workload Manager proxy via XCF
ROUTING	Enables ESP Workload Manager users to connect to an ESP Workload Manager proxy on any system in the sysplex and route ESP Workload Manager commands to the ESP Workload Manager master. **Note:** ROUTING requires HAO or Service Governor.
SCOREBD	Enables ESP Workload Manager users connected to an ESP Workload Manager proxy on any system in the sysplex, the ability to view the ESP Workload Manager scoreboard and receive subsequent scoreboard updates. **Note:** SCOREBD requires HAO or Service Governor.

Usage notes

You can use XCF for job tracking and data set triggering. When XCF is used, the ESP Workload Manager proxy transmits job-tracking and data set triggering records to the

ESP Workload Manager master via XCF, providing a quicker way to communicate and reducing contention on the QUEUE data set.

If the XCF TRACKING and DSTRIG services are not enabled, ESP Workload Manager will continue to write job-tracking and data set triggering records to the QUEUE data set.

Queue data set

The QUEUE data set is still required in ESP Workload Manager.

The QUEUE data set has a record of abended job information and information on jobs in the input, exec, and output pnodes.

Examples

The following examples show you the full syntax of the XCF START SERVICE command:

```
XCF START SERVICE TRACKING
XCF START SERVICE DSTRIG
XCF START SERVICE ROUTING
XCF START SERVICE SCOREBD
```

Related information

For more information about the XCF services, see the *ESP Workload Manager User's Guide.*

For the complete set of XCF commands available, see the *ESP Workload Manager Reference Guide*.

XDAB: Trace VSAM I/O Events

Note: You can also issue the XDAB initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Note: XDAB is a diagnostic tool and should not be used under normal operating conditions.

The XDAB initialization parameter traces VSAM I/O events and sends the trace data to a sysout data set.

Where defined

Master, proxy or both

ESPPARM data set

Syntax

```
XDAB SET|T TRACE|TR
     [SYSOUT(class)]|[LOGTRACE]
     [ASYNC|ALL]
     [LOCK|NOLOCK]
     [RESERVE|NORESERVE]
```

```
XDAB START|S TRACE|TR
```

Operand	Description
SYSOUT(*class*)	Send VSAM I/O trace data to a sysout data set with DDNAME DABTRACE and class specified by *class*.
LOGTRACE	Send VSAM I/O trace data to a sysout data set with DDNAME LOGTRACE. The trace data is consolidated with certain other trace data. For details, see the LOGTRACE initialization parameter.
ASYNC	Include only asynchronous VSAM I/Os in the trace data
ALL	Include asynchronous and synchronous VSAM I/Os in the trace data
LOCK	Include internal VSAM data set locking and unlocking events in the trace data
NOLOCK	Exclude internal VSAM data set locking and unlocking events from the trace data

Operand	Description
RESERVE	Include VSAM data set cross-system serialization events in the trace data
<u>NORESERVE</u>	Exclude VSAM data set cross-system serialization events from the trace data
START|S	Start VSAM I/O tracing

Usage notes

Setting up the trace

Before you can start the trace, you must set it up by coding XDAB with at least the SYSOUT or LOGTRACE operands.

If you code XDAB with the LOGTRACE operand, you must also set up trace data consolidation. For details, see the LOGTRACE initialization parameter.

Starting trace data consolidation

If you specified operand LOGTRACE, note that data for the XDAB trace is not generated and is not consolidated with data from other traces until you start the XDAB trace and start trace consolidation. For details on trace consolidation, see the LOGTRACE initialization parameter.

Asynchronous I/Os

Some VSAM data set I/Os, such as JOBINDEX file I/Os, are asynchronous for performance reasons. The ESP Workload Manager main task is not suspended when a wait is required for an asynchronous I/O completion.

Related information

See also

- The LOGTRACE initialization parameter.
- The XDAB command in the *ESP Workload Manager Reference Guide*.

Example

Set the sysout class for XDAB trace data

The following example specifies that trace data goes to DDNAME DABTRACE with the sysout class set to A.

```
XDAB SET TRACE SYSOUT(A)
```

Consolidate XDAB trace data with other trace data

The following example specifies that trace data goes to DDNAME LOGTRACE and is consolidated with certain other trace data.

```
XDAB SET TRACE LOGTRACE
```

Filter XDAB trace data

The following example limits the trace data to asynchronous VSAM I/Os and includes VSAM data set cross-system serialization events in the trace.

```
XDAB SET TRACE ASYNC RESERVE
```

Starting the XDAB trace

The following example starts the XDAB trace

```
XDAB START TRACE
```

XFRB: Trace Slot I/O Events

Note: You can also issue the XFRB initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

Note: XFRB is a diagnostic tool and should not be used under normal operating conditions.

The XFRB initialization parameter traces slot I/O events and sends the trace data to a sysout data set.

Where defined

Master, proxy or both

ESPPARM data set

Syntax

```
XFRB SET|T TRACE|TR
     [SYSOUT(class)]|[LOGTRACE]
     [ASYNC|ALL]
     [LOCK|NOLOCK]
     [RESERVE|NORESERVE]
```

```
XFRB START|S TRACE|TR
```

Operand	Description
SYSOUT(*class*)	Send slot I/O trace data to a sysout data set with DDNAME FRBTRACE and class specified by *class*.
LOGTRACE	Send slot I/O trace data to a sysout data set with DDNAME LOGTRACE. The trace data is consolidated with certain other trace data. For details, see the LOGTRACE initialization parameter.
ASYNC	Include only asynchronous slot I/Os in the trace data
ALL	Include asynchronous and synchronous slot I/Os in the trace data
LOCK	Include internal slot data set locking and unlocking events in the trace data
NOLOCK	Exclude internal slot data set locking and unlocking events from the trace data

Operand	Description
RESERVE	Include slot data set cross-system serialization events in the trace data
NORESERVE	Exclude slot data set cross-system serialization events from the trace data
START|S	Start slot I/O tracing

Usage notes

Setting up the trace

Before you can start the trace, you must set it up by coding XFRB with at least the SYSOUT or LOGTRACE operands.

If you code XFRB with the LOGTRACE operand, you must also set up trace data consolidation. For details, see the LOGTRACE initialization parameter.

Starting trace data consolidation

If you specified operand LOGTRACE, note that data for the XFRB trace is not generated and is not consolidated with data from other traces until you start the XFRB trace and start trace consolidation. For details on trace consolidation, see the LOGTRACE initialization parameter.

Asynchronous I/Os

Some slot data set I/Os, such as TRAKFILE I/Os, are asynchronous for performance reasons. The ESP Workload Manager main task is not suspended when a wait is required for an asynchronous I/O completion.

Related information

See also

- The LOGTRACE initialization parameter.
- The XFRB command in the *ESP Workload Manager Reference Guide*.

Example

Set the sysout class for XFRB trace data

The following example specifies that trace data goes to DDNAME FRBTRACE with the sysout class set to A.

```
XFRB SET TRACE SYSOUT(A)
```

Consolidate XFRB trace data with other trace data

The following example specifies that trace data goes to DDNAME LOGTRACE and is consolidated with certain other trace data.

```
XFRB SET TRACE LOGTRACE
```

Filter XFRB trace data

The following example limits the trace data to asynchronous slot I/Os and includes slot data set cross-system serialization events in the trace.

```
XFRB SET TRACE ASYNC RESERVE
```

Starting the XFRB trace

The following example starts the XFRB trace

```
XFRB START TRACE
```

XMITMDL: Identify tracking models to transmit

Note: You can also issue the XMITMDL initialization parameter as a command. For details about command syntax and operands, see the *ESP Workload Manager Reference Guide*.

Purpose

The XMITMDL initialization parameter tells ESP Workload Manager which tracking models are to be transmitted, and to which node.

Where defined

Master

Syntax

```
XMITMDL [LIST
         |ADD MODEL(modelname[,modelname]...)
             {[NETNODE(netnode[,netnode]...)]
              [NODE(nodeid[,nodeid]...)]
             }
         |DELETE {[MODEL(modelname[,modelname]...)]
                  [NETNODE(netnode[,netnode]...)]
                  [NODE(nodeid[,nodeid]...)]
                 }
        ]
```

Operand	Description
LIST	Lists specifications of transmitted tracking data. LIST is the default.
ADD	Adds specifications to the list of transmitted tracking data
DELETE	Deletes specifications to the list of transmitted tracking data
MODEL(*modelname*)	The name of the tracking model associated with the jobs for which data is to be transmitted. You can use wildcard characters to signify a group of tracking models. You can specify multiple tracking models.
NETNODE(*netnode*)	The destination network node for the internodal tracking model managed by Network Delivery Services (NDS). The specified destinations must correspond to the NETNODE operand of a NETDEST initialization parameter in the LOADNET initialization data set.

Operand	Description
NODE(*nodeid*)	• For NJE, the logical identifier of each node receiving tracking data, as identified on the ISCXMTR statement. • For LU6.2, the application name of the target TP Server. You can specify multiple nodes. **Note:** NODE is invalid if ISCOPT NETWORK is coded in your initialization parameters.

Usage notes

In a multi-access spool node, only the ESP Workload Manager master subsystem transmits the job-tracking data. The proxy processors are not involved.

If NJE is the communication method, the sysout class, external writer name, and logical identifier for the node receiving the data are defined using the ISCXMTR initialization parameter. For any job information sent, the receiving node must have one or more tracking models defined to track the jobs.

XMITMDL can be issued as an operator command, but it must be issued from an ESP Workload Manager master subsystem.

Examples

Adding tracking specifications

To specify that all tracking data for model XSYS be sent to ESP_SYSTEM_A, where this is the application name of the target TP Server, type

```
XMITMDL ADD MODEL(XSYS) NODE(ESP_SYSTEM_A)
```

To specify that tracking data for tracking models whose names begin with the characters PR or SYS be sent to the node with the logical identifier of N2 (where NJE is used for internodal tracking), type

```
XMITMDL ADD MODEL(PR-,SYS-) NODE(N2)
```

To specify that all tracking data be sent to the node with the logical identifier N1 (where NJE is used for internodal tracking), type

```
XMITMDL ADD MODEL(-) NODE(N1)
```

To specify that data for the tracking model XSYSTEMS be sent to the node ESPM_CHICAGO using NDS, type

```
XMITMDL ADD MODEL(XSYSTEMS) NETNODE(ESPM_CHICAGO)
```

To specify that data for the tracking model XSYSTEMS and tracking models with names beginning with abc be sent to the nodes master1, master2, master3, and master4 using NDS, type

```
XMITMDL ADD MODEL(XSYSTEMS,abc-) +
            NETNODE(master1,master2,master3,master4)
```

Deleting tracking specifications (using the previous example as a base)

To remove tracking model ABCDE, type

```
XMITMDL DELETE MODEL(ABCDE)
```

To remove node master4 from receiving tracking information from model XSYSTEMS, type

```
XMITMDL DELETE MODEL(XSYSTEMS) NETNODE(master4)
```

Note:

- The model XSYSTEMS is not deleted. It would be deleted if master4 was the last node associated with XSYSTEMS.
- In this case, do not use XMITMDL DELETE NETNODE(master4) because it would also prevent node master4 from receiving tracking information from models with names beginning with abc.

Glossary

AFM	Automated Framework Message. A structured message used to communicate commands and statuses between components in an ESP Workload Manager enterprise.
Agent	See ESP Agent.
AGENT_MONITOR	A workload object that monitors the status of ESP Agents.
Alert	Used to trigger an Event when a job reaches a particular stage in processing, such as job submission, job start or job failure. For example, you can set up an Alert to trigger an Event when a job in an Application becomes overdue. To monitor at the step level, use job monitoring instead.
Application	See ESP Application.
Application Monitor	An ESP Workload Manager facility for monitoring and controlling Applications in ISPF. From Application Monitor, you can issue commands, for example to insert a job into an Application, complete an Application or release a held Application.
Audit log	An audit trail of ESP Workload Manager activity. The audit log stores information on administrative activities, operator commands, and Application and Event processing. The audit log can be found in the ESP Workload Manager started task job log.

Automated Framework Message	See AFM.
Backout	The act of deleting all data sets a job previously created. For example, if a job abnormally terminates (abends), you can use ESP Encore to back out the failed job's processing as if the job had never executed.
Calendar	A collection of definitions for holidays and special days that are unique to your installation. The calendar also identifies workdays and the first day of the week.
CCCHK	A statement used to specify job processing based on condition codes. If a job produces a specified condition code, the CCCHK statement determines whether ESP Workload Manager considers the job as failed and whether the job continues to run. For example, you can immediately stop a failing job without running any subsequent steps regardless of the conditional operands in the JCL.
CCFAIL	Condition code fail. CCFAIL statements identify which condition codes cause ESP Workload Manager to consider a job as failed, preventing successor jobs from being submitted.
CLANG	Control language. CLANG is a high-level programming language developed for ESP Workload Manager.
Cleanup	The act of deleting data sets prior to a job being run or restarted. For example, before ESP Encore restarts a job, ESP Encore can delete the data sets created by that job in its previous run.
Command	A request made of ESP Workload Manager. For example, the LOADAGDF command loads the Agent Definition file.
Consolidated Status Facility (CSF)	See CSF.
COPYJCL	A user data set containing a working copy of the submitted JCL with the variables resolved. Using the COPYJCL statement, if a job fails, you can modify the working copy of the JCL and resubmit the job, without affecting the JCL source.
Critical path	The longest path, based on historical execution time, to a workload object representing a critical point in the ESP Application. The critical path is dynamic; it changes as the Application runs. An ESP Application can have multiple critical paths. A critical path can cross Applications if an Application has external jobs.
CSF	An ESP Workload Manager facility for monitoring and controlling the workload in ISPF. From CSF, you can issue commands, for example to bypass a job, restart a failed job or edit a job's JCL.
Data object	See Variable.
DJC	Dependent Job Control. A product that controls resources and job networks at the initiator level.
DJC job network	A group of related jobs where dependencies are controlled at the initiator level.

DOCLIB	A user data set containing job documentation entries. Each entry consists of control information—information ESP Workload Manager uses directly when processing a job, such as the JCL library—and user description—other information you want to store about a job, such as restart instructions.
DSTRIG	Used to trigger an Event based on data set activity at the Event level or release a job based on data set activity at the job level. For example, using DSTRIG, you can trigger an Event or release a job based on the creation or FTP transfer of a data set.
Enqueue	Allows a job to request exclusive or shared use of a resource without the need for the user to define the resource explicitly in advance. ESP Workload Manager will prevent jobs with mutually exclusive requests from executing concurrently. ESP Workload Manager enqueues behave like the z/OS enqueues.
ESP Agent	A program residing on a distributed server, allowing ESP Workload Manager or ESP dSeries to automate and manage workload running on operating systems, such as Windows, Linux, and UNIX, and ERPs, such as SAP, PeopleSoft, and Oracle E-Business Suite.
ESP Application	A group of related workload objects that run under ESP Workload Manager's control. ESP Applications allow you to group jobs that might run on different platforms and at different times into a single workflow. You define an ESP Application in an ESP Procedure.
ESP Communication Server	ESP Communication Server allows you to access ESP Workload Manager's panel interface from another z/OS image, where you might not have an ESP Workload Manager subsystem.
ESP Console Manager	ESP Console Manager is a console automation tool that allows you to reduce the message traffic to the console. You can use ESP Console Manager to monitor all console activity, review commands and messages, provide real-time statistics, automate responses and procedures, and perform simulations.
ESP dSeries	An Enterprise Job Scheduling solution that runs on distributed systems and offers Event-based automation across both mainframe and distributed systems.
ESP Infoserv	ESP Infoserv provides an automated interface between ESP Workload Manager and third-party Information Management products having a Low-Level Application Programming Interface (LLAPI). ESP Infoserv transmits data captured by ESP Workload Manager to the Information Management product using TP Server. Using ESP InfoServ, you can create and update problem records in an Information Management database, browse or edit problem records from ESP Workload Manager's CSF display, and support multiple client applications concurrently.
ESP Encore	ESP Encore is an advanced rerun and restart product that works with ESP Workload Manager to restart batch jobs. ESP Encore recommends the restart point in failed jobs, can make the necessary adjustments to batch JCL, and can perform the necessary data set cleanups for an error-free restart.

ESP Procedure

A data set containing ESP statements and commands. You define ESP Applications in ESP Procedures and invoke ESP Procedures using an Event. ESP Procedures can contain CLANG, symbolic variables, built-in functions, and REXX.

ESP Workload Manager

The core of the ESP Workload Manager enterprise, ESP Workload Manager acts as the central point of control for your workload—handling and directing all incoming messages from ESP Workstation and ESP Agents. Through ESP Agents, ESP Workload Manager manages workload on distributed systems.

ESP Workload Manager High Availability Option (HA0)

ESP Workload Manager High Availability Option takes full advantage of the IBM clustering sysplex technology to increase enterprise IT performance by improving the availability of system resources throughout the organization. HAO includes

- Shadow manager (see Shadow manager).
- IBM Automatic Restart Management (ARM) function support.
- XCF status monitoring (see XCF status monitoring).
- Universal login (see Universal login).

ESP Workload Manager Service Governor

ESP Workload Manager Service Governor optimizes workload to significantly increase the performance of the enterprise IT environment. ESP Workload Manager Service Governor includes

- Expedite (see Expedite).
- Resource balancing, which monitors all the ESP Workload Manager resources in the resource topology for availability and usage.
- Workload balancing (see Workload balancing).
- XCF status monitoring (see XCF status monitoring).
- Universal login (see Universal login).

ESP Workstation

A graphical user interface that you use to define, monitor, and control your workload. ESP Workstation runs on Microsoft Windows and communicates with ESP Workload Manager.

Event

Runs the workload defined by your Applications. When an Event is triggered, ESP Workload Manager invokes the ESP Procedure that contains your Application definition. You can use Events to schedule workload, trigger workload manually or trigger workload based on data set activity such as the arrival of a file. You can also use Events to send messages and issue operator commands.

EXH data set

Execution history data set. ESP Encore stores all its job history information in the EXH data set. Job history records are stored as job groups. A job group consists of an initial run and all the restart runs.

Expedite	Specifies how a job should be accelerated based on your criteria. Requires ESP Workload Manager Service Governor.
External job	A job defined in an ESP Application that ESP Workload Manager submits from another Application. You use external jobs to establish dependencies between jobs in different Applications. The Application that submits the job is known as the home Application. The Application where the job is defined as external is known as the distant Application.
EXTMON	A workload object used to monitor a job executed by an external scheduler, such as ESP dSeries.
File trigger	A workload object used to release a job based on file activity. For example, you can release a job based on the creation of a file or a change in file size.
Full name	A full name is a name for any workload. Full name is 1 to 64 character long and it is case-sensitive. You must specify full name in commands, functions, and so on, where a workload name is required. Full name is available in the user interfaces, reports, and variables. Valid full name characters are alphanumeric (A to Z, 0 to 9), national (@, #, $), underscores, and periods. The first character must be alphabetic or national. There cannot be consecutive periods or a period as the last character. The full name for mainframe workload must start with a valid mainframe job name.
Global variable	See Variable.
HAO	High Availability Option. See ESP Workload Manager High Availability Option (HA0).
Initial run	The first time a job is submitted for execution.
Initialization parameter	An ESP Workload Manager setting applied at initialization time. The initialization parameters define the ESP Workload Manager environment at startup.
JCLLIB	A user data set containing the JCL for z/OS jobs ESP Workload Manager submits.

Job ID An identification assigned automatically to every job by JES. The format of the Job ID varies according to the following rules:

Condition	Format
JES2 not running in z2 mode (with large job numbers disabled)	JOB*nnnnn*
JES2 running in z2 mode (with large job numbers enabled)	J*nnnnnnn*
JES3	For job numbers less than 100000: JOB*nnnnn* For job numbers greater than 99999: J*nnnnnnn*

Note: The same format used for the JES job ID is used by JOBONQ.

Job monitoring Used to trigger an Event when a job reaches a particular stage in processing, such as at the end of a step, at the end of a job or when a job is purged. Unlike Alert processing, job monitoring supports step-level checking.

Job name Job name is a field in the user interfaces and reports and is available in a number of variables. It is the shorter of the first eight characters of a full name or the characters before the first period in a full name. Valid job name characters are alphanumeric (A to Z, 0 to 9), national (@, #, $), and underscores (for distributed workload only). The first character must be alphabetic or national.

Link A workload object in an Application that is automatically marked complete when its predecessor and time dependencies are met. You can use a link, for example, to simplify job relationships, keep an Application active, and notify someone when something happens or does not happen.

Long name A long name is a 1-to-64-character description for distributed workload only. A long name is case sensitive and can include any character. You specify the long name in the LONGNAME operand of the job definition statement. Long name can be displayed in CSF, included in reports, and coded in CLANG through the %ESPLONGNAME variable. By default, if

- The long name specified meets the standards for a full name
- The full name specified is a job name or job name and qualifier

the long name becomes the full name of the workload. Your administrator can override this default by applying usermod 59.

Manual job Used to create dependencies on jobs that are submitted outside of ESP Workload Manager, for example a job that a user manually submits. When the manually submitted job completes, ESP Workload Manager releases the Manual job's successors.

Master An ESP Workload Manager subsystem that centrally controls the workload across multiple z/OS images.

Operand A parameter of a statement, command or initialization parameter. For example, the LOADAGDF command has an optional operand that identifies the AGENTDEF data set you want to load.

Page mode A command line interface for entering ESP Workload Manager commands in ISPF. Page mode produces scrollable output.

Pnode Processing node. ESP Workload Manager uses the INPUT, EXEC, and OUTPUT pnodes to track a job. In CSF, pnodes represent the processing states of a job such as PREDWAIT, RESWAIT, WAITING, READY, and COMPLETE. You can define additional manual pnodes, for example to represent the distribution phase of output.

Procedure See ESP Procedure.

PROCLIB A user data set that contains ESP Procedures. You define your Applications in ESP Procedures.

Proxy An ESP Workload Manager subsystem that captures SMF data for the workload running on that z/OS image and transmits that data to the master.

Qualifier A qualifier is the one to eight characters after a job name and period in a full name. If there are more than eight characters after the period or if there is another period in the full name, qualifier is a system-generated number, for example, ~0000001. Qualifiers are shown in the user interface and in reports and is available in variables. Valid qualifier characters are alphanumeric (A to Z, 0 to 9), national (@, #, $), and underscores. A qualifier can be used to distinguish a job from other jobs with the same job name, give a more descriptive name to a job, and pass symbolic variables.

Rerun To re-execute a job as an initial run. Before running the job, ESP Workload Manager resets the job to the state before it ran.

Resource Any type of real or abstract object that affects a job's ability to run successfully and can be quantified. Resources are classified into the following three types:

Renewable resource — A borrowed resource. When ESP Workload Manager submits a workload object, it removes the renewable resource from the resource pool. The resource, however, is not permanently used up. Instead, the resource returns to the resource pool when the workload object finishes running, whether it was successful or not. You can use a renewable resource, for example, to represent tape drives, a database or sort workspace.

Depletable resource — A consumed resource. When ESP Workload Manager submits a workload object, it permanently removes the depletable resource from the resource pool. You can replenish the resource using an ESP Workload Manager command, so other jobs can use the resource. You can use a depletable resource, for example, to represent the number of scratch tapes available or a permanent DASD space.

Threshold resource — A sizing resource. ESP Workload Manager sizes the workload object against the threshold resource's current level. For example, if the threshold resource is set to two, ESP Workload Manager only submits workload objects that require two or fewer units of the resource. You can use a threshold resource, for example, to prevent certain workload objects from executing until a NITESHIFT period begins.

Note: Enqueues are implicit resources.

Restart	To re-initiate a run of a job using information from the previous execution. For example, you can use ESP Encore to restart a job from the point of failure to the end of the job.
SADGEN	A user data set containing scheduled activity information on jobs. You use this data set for schedule-activity reports.
Schedule criteria	Specification of scheduling requirements for Events and jobs. You use free-format, everyday English as scheduling terms to specify schedule criteria. ESP Workload Manager has a built-in understanding of general scheduling terms. You can add your own unique scheduling terms, special processing periods, holidays, and other special days using calendars.
Service Governor	See ESP Workload Manager Service Governor.
Shadow manager	Allows your environment to shift the workload to another ESP Workload Manager master so processing can continue should your system fail. Requires ESP Workload Manager High Availability Option (HA0).
Statement	Gives information to ESP Workload Manager about an object. For example, the RUN statement specifies a job's run criteria in a job definition. You always code statements within an ESP Procedure.
SubApplication	Groups of workload objects that belong to an ESP Application. You can use subApplications to break up a large Application into smaller, easier-to-manage job groups.
Symbolic variable	See Variable.
SYMLIB	A user data set that contains symbolic variables.
Sysmsg	Used to intercept system messages written to the JES system message data set. The JES system message data set consists of the job's log. After ESP Workload Manager intercepts a system message, you can cancel a job, fail a job, trigger an Event or issue a WTO (write-to-operator) message.
Tag	A user-defined field used to filter jobs with a common characteristic in CSF or to pass information to JCL. For example, you can tag high-priority jobs and filter these jobs in CSF.
Task	A workload object in an Application that can require manual completion or can complete automatically when its dependencies are met (self-completing). You can use a task to represent a process that requires manual intervention, such as the checking of a report during Application processing, or a process that can be automated.

TEMPLIB	A user data set that contains temporary or override JCL. In an Application definition, use the TEMPLIB statement to specify the temporary JCL library you want to use as the default for all jobs following this statement.
Tracking model	A centralized definition of the attributes and processing phases of a group of jobs. ESP Workload Manager uses job-tracking models to track jobs and store job history for reporting.
Universal login	Allows ESP Workload Manager clients connected to an ESP Workload Manager proxy to route subsystem requests to the ESP Workload Manager master (XCF routing service) and to view the CSF and Application Monitor scoreboards (XCF scoreboard service). Requires ESP Workload Manager High Availability Option (HA0) or ESP Workload Manager Service Governor.
User data sets	See COPYJCL, DOCLIB, JCLLIB, PROCLIB, SADGEN, SYMLIB, and TEMPLIB.
Variable	**Data object** — Serves as a data repository. A data object is coded in an Application as any other workload object. You can store variables in a data object and retrieve those variables from a data object in another Application. You can also store, update, and delete data in a data object at run time using AFMs. **Global variable** — A variable available across Applications. You store global variables in global variable tables. A global variable trigger triggers an Event when a designated global variable changes. **Symbolic variable** — An integer or a character string whose value can change during ESP Workload Manager processing. When ESP Workload Manager encounters a symbolic variable, it substitutes the current value of that variable. You can use symbolic variables to define date operands, job names, job dependencies, and so on. You can use ESP Workload Manager's built-in symbolic variables or define your own.
Workday	A calendar definition defines a site's workdays. The default workdays are Monday through Friday, excluding any holiday that has been defined for the site.
Workload balancing	Selects where to run workload using IBM WLM workload balancing for z/OS workload and Agent workload balancing for Agent workload. Requires ESP Workload Manager Service Governor.
Workload object	A representation of work to be scheduled. An Application consists of workload objects. Workload objects include jobs that execute programs, such as mainframe JCL, Windows command files, UNIX scripts, and AS/400 programs, objects that monitor for conditions, such as data set and file activity, and objects that run on ESP Workload Manager, such as links and tasks.
XCF status monitoring	Monitors ESP Workload Manager XCF members' activity for any possible problems. Requires ESP Workload Manager High Availability Option (HA0) or ESP Workload Manager Service Governor.

Index