Environmental Record Editing and Printing Program (EREP)
3.5

User's Guide

IBM

GC35-0151-50

Note

Before using this information and the product it supports, read the information in "Notices" on page 87.

This edition applies to Version 3 Release 5 of EREP and to all subsequent releases and modifications until otherwise indicated in new editions.

Last updated: 2021-09-30

Contents

Figures

Tables

About this document

The *EREP User's Guide* applies to EREP Version 3, Release 5.

The following operating systems can run EREP:

- DOS/VS, DOS/VSE, VSE/ESA, and VSE/Advanced Functions—known collectively in this book as ***VSE systems***.
- VS2, MVS/370, MVS/XA, MVS/ESA, and OS/390—known collectively in this book as ***MVS systems***.
- VM/370, VM/SP, VM/SP/HPO, VM/XA, and VM/ESA—known collectively in this book as ***VM systems***.

If EREP 3.5 is not installed on your system, some of the information in this book may not apply. You can find out which level of EREP your system supports by checking the release number of the EREP tape last installed; the release number is in the System Control Programming Specifications, which accompany the EREP tape.

Note: New releases of EREP are always *downward compatible.* That is, the latest version of EREP always runs on your system. They also include new functions that you can only use if you have the latest version of your operating system; but generally old functions, are not eliminated. The same is true of this book, although some very old versions of EREP (for example, IFCEREP0) are no longer supported.

Who Should Read This Publication

This publication is for people who manage and maintain data processing equipment in a system installation.

USER	DESCRIPTION
System programmers	Who set up and run EREP
IBM service representatives	Who use the EREP reports to diagnose problems in the installation's hardware devices
IBM systems engineers (SE)	Who are called when there is a problem with the running of EREP
Note: It is also for anyone who wants to find out what EREP is and how it works.	

When reading this publication, you will find a working knowledge of the operating system EREP runs under very helpful; familiarity with the system job control and entry language is also helpful, but not necessary.

Organization and Contents

The information on EREP is divided into two manuals:

MANUAL	DESCRIPTION
EREP User's Guide	Introductory and explanatory information about EREP and detailed process information for the person who may not know how to set up a job to run EREP.
EREP Reference	Reference information in quick-look-up format—for the person who is familiar with EREP and the process of setting it up, but who wants to check out syntax, message wording, or coding rules.

The information in this manual is divided into the following topics:

- Chapter 1, “Introducing EREP,” on page 1, is the introduction to EREP. This topic contains an explanation of what EREP is and what it does.
- Chapter 2, “Setting Up and Running EREP,” on page 7, describes how to define the procedures and files required to make an EREP run.
- Chapter 3, “Creating EREP Reports,” on page 15, provides instructions on how to create each type of EREP report.
- Chapter 4, “Running EREP under MVS,” on page 27, presents sample procedures for running EREP under an MVS system control program, followed by descriptions of the required system controls and usage notes.
- Chapter 5, “Running EREP under VM,” on page 49, presents sample procedures for running EREP under a VM system control program, followed by descriptions of the required system controls and usage notes.
- Chapter 6, “Running EREP under VSE,” on page 69, presents sample procedures for running EREP under a VSE system control program, followed by descriptions of the required system controls and usage notes.

z/OS information

This information explains how z/OS® references information in other documents and on the web.

When possible, this information uses cross document links that go directly to the topic in reference using shortened versions of the document title. For complete titles and order numbers of the documents for all products that are part of z/OS, see *z/OS Information Roadmap*.

To find the complete z/OS library, go to IBM Documentation (www.ibm.com/docs/en/zos).

How to send your comments to IBM

We invite you to submit comments about the z/OS product documentation. Your valuable feedback helps to ensure accurate and high-quality information.

Important: If your comment regards a technical question or problem, see instead "If you have a technical problem" on page xiii.

Submit your feedback by using the appropriate method for your type of comment or question:

Feedback on z/OS function
If your comment or question is about z/OS itself, submit a request through the IBM RFE Community (www.ibm.com/developerworks/rfe/).

Feedback on IBM® Documentation function
If your comment or question is about the IBM Documentation functionality, for example search capabilities or how to arrange the browser view, send a detailed email to IBM Documentation Support at ibmdocs@us.ibm.com.

Feedback on the z/OS product documentation and content
If your comment is about the information that is provided in the z/OS product documentation library, send a detailed email to mhvrcfs@us.ibm.com. We welcome any feedback that you have, including comments on the clarity, accuracy, or completeness of the information.

To help us better process your submission, include the following information:

- Your name, company/university/institution name, and email address
- The following deliverable title and order number: EREP V3R5 User's Guide, GC35-0151-50
- The section title of the specific information to which your comment relates
- The text of your comment.

When you send comments to IBM, you grant IBM a nonexclusive authority to use or distribute the comments in any way appropriate without incurring any obligation to you.

IBM or any other organizations use the personal information that you supply to contact you only about the issues that you submit.

If you have a technical problem

If you have a technical problem or question, do not use the feedback methods that are provided for sending documentation comments. Instead, take one or more of the following actions:

- Go to the IBM Support Portal (support.ibm.com).
- Contact your IBM service representative.
- Call IBM technical support.

Summary of Changes

Fifth Edition

This version has received editorial and terminology updates.

Fourth Edition

This version has received editorial and terminology updates.

Third Edition

This book includes new information about "Running EREP in a Multisystem Environment" on page 46.

Chapter 1. Introducing EREP

The Environmental Record Editing and Printing Program (EREP) is a diagnostic application program that runs under the MVS, VM, and VSE operating systems. The purpose of EREP is to help IBM service representatives maintain your data processing installation.

EREP edits and prints reports from the records placed in the error recording data set (ERDS) by the error recovery program (ERP) of your operating system. Some of these records are the result of device or system errors, while others are informational or statistical data. The service representative analyzes information in the EREP reports to determine if a problem exists, what the problem is, and where the problem is located.

What EREP Does

EREP processes the error records from your operating system to produce formatted reports. These EREP reports can show the status of the entire installation, an I/O subsystem, or an individual device depending upon which report you request. EREP reports vary in format depending on type:

REPORT TYPE	FORMAT
System summary	Error data in summary form
Trends	Error data by daily totals
Event history	Error data in a time sequence by occurrence

Important:

1. EREP edits and prints records that already exist; it does *not* create the error records (the operating system creates them).
2. EREP is not designed to automate media maintenance or library management. It is a service tool that shows statistical data that helps your IBM service representative determine whether a problem is media related or hardware related.

Figure 1 on page 2 shows how records are built from the device sense data and then what EREP does with those records.

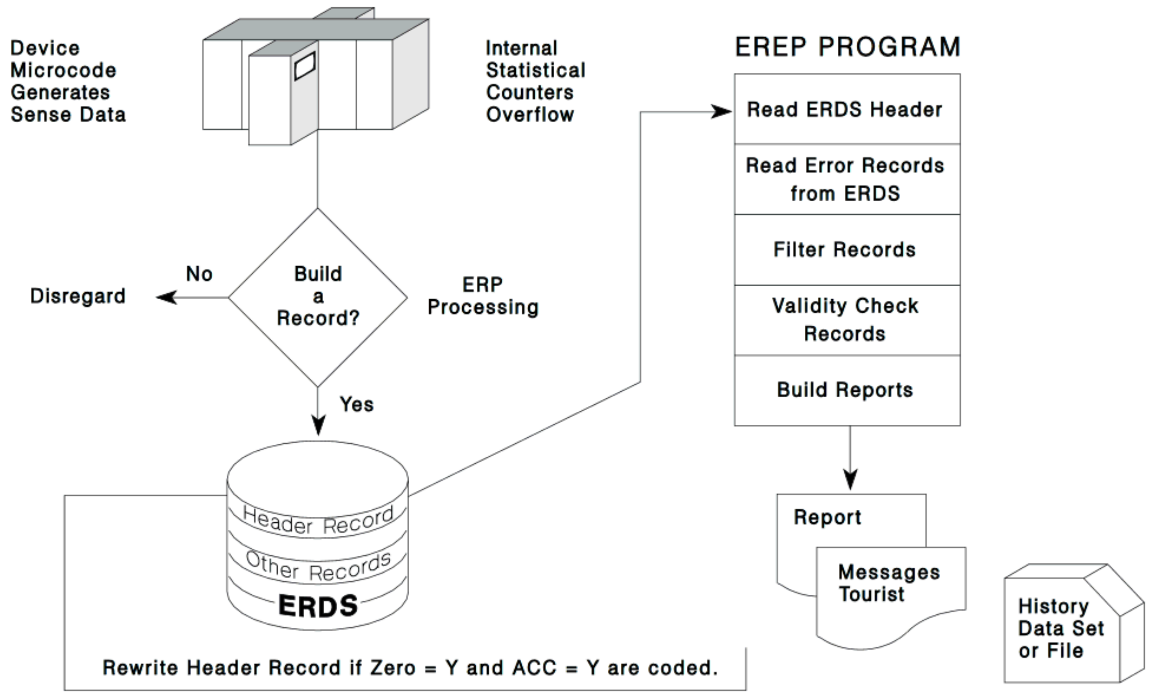

Figure 1. The Record Building and Reporting Process

Where Records Come From

Your operating system with its hardware and software captures statistical and error data, such as:

- A read error on a direct access device or tape volume
- A machine check on a processor
- An IPL of the operating system

How Data is Processed and Records Built

The system procedure executing EREP issues commands to write the buffered statistical data from the system-attached devices to the ERDS. The system ERP builds the records in the following stages:

STAGE	ACTION
1	The devices attached to the operating system generate sense data for the events encountered during the day. The sense data can be informational, error-related, or statistical.
2	The ERP of the operating system looks at the sense data. If the sense data indicates that a record should be built, the ERP takes the sense data and places it after the standard header information. The combination of the header information and the sense data becomes the error record.
3	The operating system ERP writes the records onto the system ERDS.

If any of the devices do not respond to the system commands, EREP stops and does not continue until the device that does not respond is brought back online. System diagnostics can be used to determine which device is causing the problem.

Where the Records are Stored

The records are stored in the ERDS of the operating system. The ERDS goes by a different name in each operating system:

OP. SYSTEM	ERDS NAME
MVS	ERDS (1)
VM/SP*	error recording area
VSE	SYSREC (IJSYSRC)
Note: 1. In MVS, the default name of the ERDS is SYS1.LOGREC; but for MVS release 5.1 and later, it can be installation modified.	

What EREP Does with the Records

When you run EREP, it reads records directly from the ERDS and processes them to produce the report you have requested. EREP processing includes:

- Filtering the records through the selection parameters set up for the report
- Checking records for validity
- Reviewing records to see if they belong in the reports

Note: IFCEREP1 always opens the ERDS for update mode. Users with only read authority will not be able to produce reports or copy the ERDS to a temporary history data set.

Note: With the exception of system summary report, IFCEREP1 opens the ERDS in read mode if ZERO=N is specified or defaulted. Users with only read authority can produce reports or copy the ERDS to a temporary history data set when ZERO=N is specified or defaulted. This update is available in PTF UO01597.

How EREP Filtering is Done

EREP filters through the records it read from the ERDS to determine:

- Which records satisfy the selection parameters
- The record length
- Whether the first byte of data is a valid record type
- Whether the record has been truncated by the operating system

EREP then relays the following standard message to the EREP messages file:

```
IFC120I xxx records passed filtering.
```

where xxx is the number of valid records EREP saw. If the message indicates 0 records passed filtering, the file may be missing or there may be invalid data in the file.

You may also receive the following message:

```
IFC122I xxx records ignored because truncated bit on.
```

where xxx is the number of valid records ignored.

This indicates that a number of records were truncated as they were placed in the ERDS or history file.

Note: The operating system, not EREP, truncates the records and does not indicate which records were not processed.

See "Messages Files" on page 4 for a description of messages in the EREP messages file.

How EREP Checks Records for Validity

To check all DASD OBR, MDR or A3 records for validity, EREP:

- Uses a device product specification table to check each record
- Reviews the content of certain bytes within the records

EREP accepts only records that are built to the specifications of each device.

If a record is invalid, EREP puts an IFC264I or IFC265I message and a hex dump of the record in the EREP messages file. Invalid records may be the result of:

- Invalid data, missing data, or conflicting data within the records
- Down level microcode patch to the hardware
- Program temporary fix (PTF) missing on the operating system
- PTF missing on the EREP system maintenance

See "Messages Files" on page 4 for a description of messages in the EREP messages file.

How EREP Selects Records When Building Reports

After records have been filtered and validity checked, EREP reviews each record to see whether to include it in the requested reports. Since each report has its specific criteria, not all records appear in all reports.

If you do not see a record that you feel should be in the report, check the EREP messages file to see if the record is listed. If it is not, consider the following:

- The sense data can cause a record to be excluded from a report. If sense byte 26, bit 6 is turned off in a 3390 record, the record is not included in the system exception series (SYSEXN) reports.
- The record could be a type that is not normally processed by the particular report. SIM-producing device records do not appear on a TRENDS report.
- The record may have been rejected. Records that have not passed through the previous checks are not included in the report.

How the ERDS Header Record is Reset

If the Accumulate (ACC) and the Zero ERDS (ZERO) processing parameters are set to "Y", EREP resets the ERDS header record. The header record contains pointers that locate the start and end of the error records within the ERDS. MVS and VSE use the header record to control the overwriting of previous records in the ERDS. VM/XA* and VM/ESA* erase the ERDS.

If the ERDS header record is unreadable, EREP will not be able to process the records and the operating system may not be able to write new records to the ERDS. If the header record gets overlaid or otherwise damaged, there will be no way to recover the error records within the ERDS.

Note: Using the IFCDIP00 program or rewriting the entire recorder file fixes the header record problem, but it does not allow you to read the records. Records are physically present in the file, but are still inaccessible due to the new header pointers.

Types of EREP Output

You can receive three different kinds of output in each EREP run:

- Messages files
- Report files
- History files

Messages Files

EREP puts messages and processing information about the statements it executes into the messages (TOURIST) file. The EREP messages file shows:

- Which parameters, including defaults, have been applied to the input records
- The number of records passing filtering

- The number of records processed
- How EREP has interpreted the control statements you set up
- The messages issued during processing. (For detailed descriptions of each message, refer to EREP Messages in the *EREP Reference*)

For an example of the EREP message output, refer to Using the EREP Messages File (TOURIST Output) in the *EREP Reference*.

Report Files

EREP formats information about the recorded errors into the reports that you request. You need to run EREP reports daily in large installations and weekly in smaller installations. Use your EREP reports to look for indications of system or device problems.

Important: Because there is no way to show you all the possible variations caused by different devices and different parameter combinations, your reports will not look exactly like the reports in this manual.

Your operating system, the hardware devices installed, and the version of EREP that you are running determine the reports you can print. The examples in the maintenance documentation for your specific devices show information on the types of reports available and details of the various parts of the reports.

The following table lists and categorizes the different EREP reports to help you decide which ones to include in your EREP runs:

REPORT TYPE	CATEGORY	TELLS YOU	REFER TO
System summary	Overview	If there are problems	“System Summary Report” on page 15
Trends report	Overview	If there are problems	“Trends Report” on page 18
Event history	Overview	If there are problems	“Event History Report” on page 20
System exception series	Analysis	Where the problems are	“System Exception Report Series” on page 22
Threshold reports	Detail	What the problems are	“Threshold Summary Report” on page 23
Detail print reports	Detail	What the problems are	“Detail Edit and Summary Reports” on page 25

Note: Some of the reports, such as the system exception series, have several parts; be aware that a lot of output can be generated.

For examples of statements used to generate reports for each operating system, see one of the following topics:

- Chapter 4, “Running EREP under MVS,” on page 27
- Chapter 5, “Running EREP under VM,” on page 49
- Chapter 6, “Running EREP under VSE,” on page 69

Important: Use the maintenance documentation for your device to find detailed directions about interpreting EREP reports for your specific hardware, the significance of particular fields within reports that can be device specific, and variations within reports due to hardware specifics.

History Files

You should always create a history file before running EREP reports. By creating a history file and then running all the reports against that file, you ensure that all of the reports are using the same set of records.

Examples of statements used to generate history files are in the following sections:

OP. SYSTEM	REFER TO
MVS	"Step 1: Creating a History Data Set" on page 29
VM	"Step 1: Creating a History File" on page 50
VSE	"Step 1: Creating a History Data Set" on page 69

Chapter 2. Setting Up and Running EREP

This section gives you the general guidelines you will need for invoking and running EREP. It includes the following topics:

Topic
“Invoking EREP” on page 7
“Defining Files and Storage Requirements for EREP” on page 8
“Parameters and Control Statements” on page 8
“Maintaining ERDS Data Integrity” on page 13
“Clearing the ERDS” on page 13
“Multisystem Installations” on page 13
“A Planning Checklist” on page 14

Invoking EREP

You run EREP by executing a procedure containing the operating system EREP command and its associated parameter and control statements. You can only request one type of report each time you execute the EREP command for your system. You may produce any number of different type reports by including additional EREP commands with the associated parameters and control statements.

The following table contains descriptions of how to execute EREP under each operating system and the topics where you can find detailed instructions.

OP. SYSTEM	PROCESS	REFER TO
MVS	Define the input and output data sets using JCL DD statements. The JCL submits the job as a batch job or interactively via TSO. Put the *IFCEREP1* program in the JCL EXEC statement. Include the EREP parameters on the EXEC statement or as part of SYSIN in-stream data with the EREP control statements.	Chapter 4, “Running EREP under MVS,” on page 27
VM/SP	Define the input and output files using *FILEDEFs* and then issue the *CPEREPXA* EXEC. Enter the *CPEREPXA* and EREP operands using one of the methods shown in “Entering CPEREPXA Operands” on page 60. The EREP parameters and control statements are included in the EREP operands.	Chapter 5, “Running EREP under VM,” on page 49
VSE	Define the input and output data sets using JCS TLBL or DLBL statements, and the necessary logical units using ASSGN statements. The JCS submits the job as a batch job or interactively via ICCF. Put the *IFCEREP1* program in the JCS EXEC statement. List your EREP parameter and control statements as in-stream data to be read from the SYSIPT logical unit.	Chapter 6, “Running EREP under VSE,” on page 69

Defining Files and Storage Requirements for EREP

You must provide system controls that create the interface between EREP and the operating system's data management functions. The following table shows where to find instructions for setting up system controls and defining files for each operating system:

OP. SYSTEM	REFER TO
MVS	"MVS System Controls" on page 40
VM	"Defining Files for CPEREPXA" on page 56
VSE	"VSE System Controls" on page 77

Parameters and Control Statements

All of the operating systems use the same parameters and control statements to tell EREP what specific information to print in the reports.

The parameters and control statements can be grouped according to the kinds of information they convey to the EREP program as shown in the following table:

TYPE	INFORMATION CONVEYED	REFER TO
Report parameters	Which report to produce	"EREP Report Parameters" on page 8
Selection parameters	Which records to select for the requested report	"EREP Selection Parameters" on page 9
Processing parameters	How to control the processing of error records and report output	"EREP Processing Parameters" on page 10
Control statements	How to: • Direct EREP processing • Supply more information about the system's configuration • Provide organization to the requested reports	"EREP Control Statements" on page 10

Note:

1. "EREP Report Parameters" on page 8through "EREP Control Statements" on page 10 describe all the EREP parameters and control statements.
2. Table 1 on page 11 shows parameters that *cannot* be used together.
3. For the syntax of each parameter and control statement, refer to Introduction to EREP Controls in the *EREP Reference*.

EREP Report Parameters

Use the report parameters in the following table to request reports from EREP.

REPORT PARAMETERS	WHAT THEY DO
EVENT	Produces a three-part event history report that lists errors in chronological order. Shows how often errors occur and in what order. Used to establish a pattern and identify problems.

REPORT PARAMETERS	WHAT THEY DO
PRINT	Produces a series of detail edit or summary reports, or both, for the selected record types. The number of reports depends on the input and selection parameters. **Note:** `PRINT=SD` is the default report parameter. The other options are shown in the syntax for the print parameter: `PRINT={AL | DR | NO | PS | PT | SD | SU}` The only way to run EREP without producing any report output is to code `PRINT=NO`.
SYSEXN	Produces a system exception report series covering processors, channels, DASD, optical, and tape subsystems.
SYSUM	Produces a condensed two-part system summary report of all errors for the principal system elements: CPU, channels, storage, SCP, and the I/O subsystem
THRESHOLD	Produces a summary of a 3410, 3420, and 8809 tape subsystem, including media statistics and permanent errors that exceed the limits set on the parameter.
TRENDS	Produces a two-part trends report that presents error records logged for the various system elements during a maximum of 30 days. This report presents the errors in chronological order by Julian date.

Note:

1. Table 1 on page 11 shows parameters that *cannot* be used together.
2. For the syntax of each parameter, refer to Parameter Descriptions in the *EREP Reference*.

EREP Selection Parameters

Use the selection parameters in the following table to select the records for EREP to use.

SELECTION PARAMETERS	TELLS EREP TO:
CPU (Processor serial and machine type numbers)	Use only the records associated with this particular processor.
CPUCUA (Processor serial number and channel unit address)	Use only the records associated with this channel unit attached to this processor.
CUA (channel unit address or number)	Use only the records associated with this particular channel unit address or channel unit number.
DATE	Use only the records created during this date range.
DEV (Device type)	Use only the records associated with this particular device type; or, conversely, do not use the records associated with this device type.
DEVSER (Device serial number)	Use only the OBR records associated with this tape device serial number. (Use only for the THRESHOLD report and only with the 3410, 3420, and 8809 tape OBR records.)
ERRORID (Error identifier)	Use only the MCH and MVS software records containing this particular error identifier.
LIA/LIBADR (Line interface [base] address)	Use only the 3705, 3720, 3725 or 3745 communication controller records containing this line interface address.

SELECTION PARAMETERS	TELLS EREP TO:
MOD (Processor model)	Use only the records containing this processor machine type (number).
MODE (370 or 370XA)	Use only the records created in this operating mode.
SYMCDE (Fault symptom code)	Use only the 33XX DASD records containing this particular fault symptom code.
TERMN (Terminal name)	Use only the VTAM OBR records containing this terminal name.
TIME	Use only the records created during this time range.
TYPE (Record type)	Use only the records of the specified types.
VOLID (Volume serial number)	Use only the 33XX DASD or 34XX tape records containing this volume serial number.

Note:

1. Table 1 on page 11 shows parameters that *cannot* be used together.
2. For the syntax of each parameter, refer to Parameter Descriptions in the *EREP Reference*.

EREP Processing Parameters

Use the processing parameters in the following table to control the way EREP processes the records you selected.

PROCESSING PARAMETERS	WHAT THEY DO
ACC (Accumulate)	Tells EREP to copy the records used for the report into an output history file.
HIST (History)	Tells EREP that its input consists of records in history files.
LINECT (Line count)	Tells EREP that each page of the report output must contain this number of lines.
LINELEN (Line length)	Tells EREP that each line of the system summary report output may contain up to this number of characters.
MERGE (Merge)	Tells EREP that its input consists of records from both the ERDS and history files.
SHORT (Short OBR)	Tells EREP to print out short-form OBR records in detail edit report output.
TABSIZE (Table size)	Tells EREP that the sort table it uses for internal processing must be this size.
ZERO (Zero ERDS)	Tells EREP that when this report is complete, to change the header pointer to allow the ERDS to be overwritten with newly collected errors.

Note:

1. Table 1 on page 11 shows parameters that *cannot* be used together.
2. For the syntax of each parameter, refer to Parameter Descriptions in the *EREP Reference*.

EREP Control Statements

Use the control statements in the following table to give EREP information about your configuration and set up the overall criteria for the way EREP creates the report.

CONTROL STATEMENTS	WHAT THEY DO
CONTROLLER	Tells EREP to combine the error records associated with this particular control unit and its attached devices.
DASDID	Tells EREP that this is the configuration of the 33XX DASDs within each subsystem; identifies those that do not provide physical IDs for the system exception report series. This control statement applies only to the system exception report series.
ENDPARM	Tells EREP that this is the end of the in-stream EREP parameters; the in-stream data that follows consists of EREP control statements.
LIMIT	Tells EREP to produce output for the system exception reports only when the number of megabytes processed per error is less than the megabytes specified by the error frequency value and the number of times the error occurs is greater than or equal to the number specified by the count value. This control statement applies only to the system exception report series.
SHARE	Tells EREP to combine the records for these devices that are shared between systems. This control statement applies to all the reports that generate I/O device summaries.
SYSIMG	Tells EREP to modify the CPU serial numbers for *n*-way processors so that those processors operating in the same system image are reported under the same CPU serial number.

Note: For more information on using control statements, refer to EREP Control Statements in the *EREP Reference*.

EREP Parameter Combinations

To help you to avoid using invalid parameter combinations, Table 1 on page 11 shows the parameters that *cannot* be used together. An *X* in a column indicates which two parameters cannot be used together; for example the ACC and the threshold parameters cannot be used together. Numbers in the column are identified in the notes following the table.

Table 1. EREP Selection, Processing, and Report Parameter Combinations

	Processing Parameters								**Selection Parameters**														
	ACC	**HIST**	**LINECNT**	**LINELEN**	**MERGE**	**SHORT**	**TABSIZE**	**ZERO**	**CPU**	**CPUCUA**	**14 CUA**	**DATE**	**DEV**	**DEVSER**	**ERRORID**	**LIA/LIBADR**	**MOD**	**MODE**	**SYMCDE**	**TERMN**	**TIME**	**TYPE**	**15 VOLID**
REPORT																							
EVENT						X								X									
PRINT			1			2								X									
SYSEXN						X								X									
SYSUM						X								X									
THRESHOLD	X					X		X	X	X			3		X	X	X		X	X		X	
TRENDS						X								X									
PROCESSING																							

Table 1. EREP Selection, Processing, and Report Parameter Combinations (continued)

	Processing Parameters								**Selection Parameters**														
	ACC	**HIST**	**LINECNT**	**LINELEN**	**MERGE**	**SHORT**	**TABSIZE**	**ZERO**	**CPU**	**CPUCUA**	**14 CUA**	**DATE**	**DEV**	**DEVSER**	**ERRORID**	**LIA/LIBADR**	**MOD**	**MODE**	**SYMCDE**	**TERMN**	**TIME**	**TYPE**	**15 VOLID**
ACC	X							4						X									
HIST		X			X			X															
LINECT			X																				
LINELEN				X																			
MERGE		X			X									X									
SHORT						X																	
TABSIZE							X							X									
ZERO	4	X						X	X	X	X	X	X		X	X	X	5	X	X	X	X	X
SELECTION																							
CPU								X	X	X							X						
CPUCUA								X	X	X	X			X			X					6	
CUA 14								X		X	X											6	
DATE								X				X											
DEV								X					X	7		8						9	10
DEVSER	X				X		X			X			7	X	X	X	X		X	X		11	12
ERRORID								X						X	X							12	
LIA/LIBADR								X					8	X		X			X	X			X
MOD								X	X	X				X			X						
MODE								5										X					
SYMCDE								X						X		X			X	X		11	X
TERMN								X						X		X			X	X		11	X
TIME								X													X		
TYPE								X		6	6		9	11	12				11	11		X	13
VOLID 15								X					10	12		X			X	X		13	X

Note:

1. Invalid when PRINT=NO.
2. Invalid when PRINT=DR, NO, SD, or SU.
3. Invalid except for DEV=(34XX, 3410, 3420, or 8809).
4. Invalid for ZERO=Y if ACC=N.
5. Invalid except when you code or default MODE=ALL, which indicates no record selection.
6. Only affects the selection of record types that contain a CUA: CCH(C), DDR(D), MDR(T), MIH(H), and OBR(O).
7. DEVSER is only used for the threshold report summary, so the following are the only devices allowed: 3410, 3420, 8809, and 34XX.
8. LIA/LIBADR applies only to TP communication controllers, so the following are the only valid devices: 3705, 3720, 3725, and 3745.
9. DEV is valid with only the following record types: DDR(D) MIH(H), OBR(O), MDR(T), and A3(A).
10. VOLID applies only to 33XX DASD and 34XX tape devices.

11. Only affects the selection of record types that contain a symptom code: OBR(O).
12. Only affects the selection of record types that contain an error ID: MCH(M) and SFT(S).
13. Only affects the selection of record types that contain a volume ID: OBR(O) and MDR(T).
14. The CUA parameter is not supported for A2 and A3 records.
15. The VOLID parameter is not supported for A3 records, even if they contain a volume ID.

Maintaining ERDS Data Integrity

Whenever you offload data from a file, you run the risk of losing, duplicating, or otherwise ruining the data. Some of the potential problems are:

IF...	THEN...
The ERDS fills too quickly, it can overflow.	Transfer the contents of the ERDS to a history file. See "Clearing the ERDS" on page 13 for instructions.
One or more I/O devices are not able to complete the transfer of data from buffered-log when requested by EREP. EREP can go into a wait mode, the job will not complete, and reports will not be produced.	Transfer the contents of the ERDS to a history file, and then use the history file to produce EREP reports. See "Clearing the ERDS" on page 13 for instructions.

If the step or job fails and you subsequently rerun it, be aware that you may duplicate some of the records already on the output.

If the step does not fail, but no records are copied to the output, subsequent steps can be using an empty input file for the EREP reports.

Clearing the ERDS

When ERDS is almost full, the recording routines sense it and issue a message.

Run EREP with the parameters required to clear the ERDS (only MVS and VSE use *IFCOFFLD*).

Important: This EREP run may request a system summary; but it should always include ACC=Y and ZERO=Y, to write the contents of the ERDS to another file and to clear the ERDS so the recording routines can write more records to it.

For information on your operating system's procedures, see the topics shown in the following table:

OP. SYSTEM	REFER TO
MVS	"Clearing the ERDS When Near Full on MVS" on page 45
VM	"Clearing the ERDS when Near Full on VM" on page 64
VSE	"Clearing the ERDS when Near Full on VSE" on page 80

Multisystem Installations

If your MVS, VM, or VSE installation has multiple processors running under the same or different operating systems, it may be possible to combine all of your error records into one history file.

Use the steps shown in Table 2 on page 14 to combine the error records.

Table 2. Combining Multisystem Error Records

STEP	ACTION
1	Unload the ERDS on each machine according to the appropriate method for that operating system. See "History Files" on page 6 for information on creating a history file.
2	Choose one machine on which to run the EREP reports.
3	Transfer the history files from each of the other machines to the one chosen in step 2. Use the method of transfer provided by the operating system.
4	Combine the history files from all of the machines. Use the method of combining files provided by the operating system.
5	Run the EREP reports appropriate for your installation. Use the combined file as the history file.
Note: If you merge VSE with VM or MVS records into a single file, that input file must be defined as a standard label file.	

For details on setting up job streams within your operating system, see one of the following sections:

OP. SYSTEM	SECTION
MVS	"Running EREP in a Multisystem Environment" on page 46
VM/SP	"VM SP Error Recording with Guest Systems" on page 67
VSE	"Running EREP in a Multisystem Environment" on page 81

A Planning Checklist

The following checklist contains questions to ask as you set up EREP for your installation:

1. Does the IBM service representative agree with the report content and sequence?
2. Has the ERDS been copied to a history file? See "History Files" on page 6.
3. Are the DASDID statements ready for System Exception reports? Refer to DASDID Control Statements in the *EREP Reference*.
4. Are the LIMIT statements ready for System Exception reports? Refer to LIMIT Control Statement in the *EREP Reference*.
5. Are the SHARE and CONTROLLER statements ready? Refer to SHARE Control Statements and CONTROLLER Control Statements in the *EREP Reference*.
6. Is the storage available for EREP adequate?
7. Is EREP set up to run automatically? For information on automating your EREP run in each operating system see the following topics:

OP. SYSTEM	HEADING
MVS	"Automating the Running of EREP" on page 47
VM	"Automating the Running of EREP" on page 67
VSE	"Automating the Running of EREP" on page 82

8. Is the report distribution set up to include:

   a. the IBM service representative

   b. the system programmer or administrator

Chapter 3. Creating EREP Reports

This section provides instructions on how to create each type of EREP report. Descriptions of each report are provided to help you select the reports you need to adequately monitor your installation.

EREP reports are designed to give you a variety of views of the data being processed. EREP produces:

- Overview reports, from which you can determine *if* there are problems
- Analysis reports, from which you can determine *where* there are problems
- Detail reports, from which you can determine *what* the problems are

In order to decide which report to run at which time, you need to understand what each one is telling you. The following reports are described in this section:

Topic
"System Summary Report" on page 15
"Trends Report" on page 18
"Event History Report" on page 20
"System Exception Report Series" on page 22
"Threshold Summary Report" on page 23
"Detail Edit and Summary Reports" on page 25

Note:

The reports are listed from most general to most specific, because the most effective way to use EREP reports is to start with the most general and work toward the most specific.

The *EREP Reference* contains examples of each report described in this section.

System Summary Report

The system summary report provides an overview of errors for each of your installation's principal parts, or subsystems:

- Processors (CPU)
- Channels
- Subchannels
- Storage
- Operating system control programs (SCPs)
- I/O subsystems.

Important: The system summary report does not go into detail; it shows how many errors and exceptions are recorded overall. It is a good place to start when evaluating the performance of your system.

Refer to System Summary Report in the *EREP Reference* for a set of sample reports.

Description of the System Summary Report

The system summary report has two parts:

PART	DESCRIPTION
1	Summarizes errors by CPUs from all but the I/O subsystem.

PART	DESCRIPTION
2	Summarizes errors recorded in the I/O subsystem.

Note:

1. The record counts are listed by CPU. Refer to How EREP Assigns Numbers to CPUs in the *EREP Reference* for an explanation of the way the number identifiers are assigned.

   EREP can report information from a variable number of CPUs depending upon your operating system, type of printer and what parameters you specify. Information from the remaining CPUs are grouped together under serial number X'FFFFFF'.

   It is also possible to have multiple internal CPUs reported under one serial number. Refer to SYSIMG Control Statement in the *EREP Reference* for more information.
2. DASD and tape are listed by strings in the system summary.
3. A field with all 9’s means that the number was larger than the print position allowed.
4. A dash (–) in part 2 of the system summary means there are no records for this DEVNO/CUA on this processor (CPU).
5. It is most useful to address the permanent errors first.

System Summary Part 1

The first part of the system summary report varies according to the mode of the records it summarizes as shown in the following table:

RECORD MODE	CONTAINS
370	• Counts of machine checks (MCH records) • Channel checks (CCH records) by channel
370XA	• Machine-check totals • Counts of subchannel logouts (SLH records) by channel path ID • Channel report words (CRW records) created by both hardware and software

Note:

1. For MVS only, actual software error records are included in the report.
2. Counts of software events that may or may not be associated with errors (IPLs and system termination) are shown in the first part of the system summary.

System Summary Part 2

The second part of the system summary is a condensed report of every permanent and temporary error recorded for the I/O devices in your installation, listed under the CPU associated with the error.

When your CPUs share I/O devices, you must use SHARE control statements for the system summary if you want to see I/O errors combined for all the possible paths to a device that is common to different systems. Refer to SHARE Control Statements in the *EREP Reference* for details.

The temporary errors appearing in part 2 of this report are totals of temporary read/write errors and statistical data.

The temporary and permanent I/O errors are listed by product or device groups. Table 3 on page 17 shows the product groups in the order they appear in part 2 of the system summary and the trends reports.

Table 3. The Order of Product Groups in the Reports

ORDER	PRODUCT GROUP
1	Console and unit record devices: 1. Operator’s console 2. Card reader 3. Card punch 4. Printer 5. OCR/MICR
2	Direct-access storage devices: 1. Disk 2. Drum/fixed-head file 3. Mass storage system 4. Optical
3	Tape devices
4	Displays (channel-attached)
5	Teleprocessing (TP) communications controllers
6	Terminals
7	Other devices: 1. Channel-to-channel adapter 2. Cryptographic unit 3. Dynamic pathing availability (DPA)
8	Unknown/unrecognized devices

Errors are presented by control unit or device address for each device type as shown in the following table:

Records type	Control unit or device address
For 370 records	The device address is the CUA.
For 370XA records	The device address is the device number.
For both 370 and 370XA records	The errors are combined.

DASD is presented as follows:

- DASD with serial numbers or DASDIDs show only total counts since hardware error conditions are not caused by CPU.
- DASD with serial numbers in the sense records (for example, 3990 and 9343) indicate subsystems by type and SSID value (as set in the control unit).
- DASD with DASDID indicate the subsystem by the control unit ID (first byte of the DASDID).

The I/O error data is summarized by the control unit/device address or number of the device reporting each error.

Physical ID identifiers appear in the combination format of SCUID-CTLID-DEVID.

Generating System Summary Reports

Turn to the headings shown in the following table for examples of statements used to generate the system summary reports in each operating system:

OP. SYSTEM	REFER TO
MVS	"Step 2: Generating a System Summary Report" on page 30
VM	"Step 2: Generating a System Summary Report" on page 51
VSE	"Step 2: Generating a System Summary Report" on page 70

Note:

1. Refer to SYSUM — System Summary (Report Parameter) in the *EREP Reference* for details about the system summary report parameter.
2. When you code ACC=Y with SYSUM, EREP always clears the ERDS, even if you code ZERO=N.

Use the following selection parameters to customize your system summary report:

DATE
LINELEN
MODE
TIME

Note: Specifying parameters other than these may result in misleading reports. See Table 1 on page 11 for restrictions on using parameter combinations.

The following table shows the type of error records and their source in the system summary report.

TYPE	SOURCE
CCH	CPUs, channels
CRW	CPUs, channels
EOD	operating systems
IPL	Operating systems
MCH	CPUs
MDR	I/O devices; including SCUs, controllers
OBR	I/O devices; including SCUs, controllers
SFT	MVS operating system
SLH	CPUs, channels

Trends Report

The trends report is a two-part report that presents error records logged for the various system elements during a maximum of 30 days. Trends reports present the pattern and frequency of errors on a daily basis. You can use these reports to see when the errors began, their pattern, and when they end.

Refer to Trends Report in the *EREP Reference* for a set of sample reports.

Description of the Trends Report

The trends report presents error data in chronological order, by the Julian day (1 through 365) and consists of the two parts shown in the following table:

PART	DESCRIPTION
1	Presents errors by type of failure: CPU, channel, storage, and SCP. It contains IPL, MCH, CCH/SLH/CRW, and program error (software) records for each processor (CPU).
2	Presents permanent and temporary I/O errors for the product groups in the order shown in Table 3 on page 17.

Note:

1. The trends report does not report on SIM producing devices such as 3990/3390 DASD.
2. The 9340 direct access storage subsystems are not shown in the trends report.
3. Within product groups, the errors are presented by device address or number or physical ID within generic device or product types.
4. The CPU associated with the record appears on the line with the device address or number. Devices that provide physical IDs are associated with the control unit and not with a CPU.
5. DASD and tape devices are listed by DEVNO or CUA.

Generating Trends Reports

Turn to the headings shown in the following table for examples of statements used to generate the trends report in each operating system:

OP. SYSTEM	REFER TO
MVS	“Step 11: Generating a Trends Report” on page 39
VM	“Step 11: Generating a Trends Report” on page 56
VSE	“Step 11: Generating Trends Reports” on page 76

Note:

1. Refer to TRENDS — Trends Report (Report Parameter) in the *EREP Reference* for details about the trends report parameter.
2. The *last* 30 days of error data is displayed unless you use the DATE selection parameter to specify another set of dates.
3. 30 consecutive days is the maximum number of days that can be requested.
4. Use SHARE control statements to combine all the errors reported by an I/O device that is connected to more than one system. Refer to SHARE Control Statements in the *EREP Reference* for more information.
5. If your installation has more than 16 CPUs, EREP produces the report using records from the first 15 processors (CPU)s it encounters.

   Information from the remaining CPU is grouped together under serial number X'FFFFFF'.
6. It is possible to have multiple internal processors (CPUs) reported under one serial number and thus increase EREP’s capabilities. Refer to SYSIMG Control Statement in the *EREP Reference* for more information.

Use the following parameters to customize your system summary report:

Parameters

CUA	MODE
DATE	TIME
DEV	TYPE

Note:

1. Specifying parameters other than these may result in misleading reports.
2. See Table 1 on page 11 for restrictions on using parameter combinations.
3. Refer to DEV — Device Type (Selection Parameter) in the *EREP Reference* for some restrictions on the record types you can select.

The following table shows the type of error records and their source in the trends report.

TYPE	SOURCE
CCH	CPUs, channels
CRW	CPUs, channels
IPL	Operating systems
MCH	CPUs
MDR	I/O devices
OBR	I/O devices
SFT	MVS operating system
SLH	CPUs, channels

Event History Report

The event history report consists of one-line abstracts of selected information from each record. The event history report shows errors in a time sequence that allows you to see how often and in what order errors occur. It also allows you to establish a pattern and diagnose problems.

Refer to Event History Report in the *EREP Reference* for a set of sample reports.

Description of the Event History Report

The event history is divided into the three parts shown in the following table:

PART	DESCRIPTION
1	Is a template showing the headings used for the record-dependent data from each type of record. It does the following: • Guides in the interpretation of information in the other sections of the report • Explains terms • Provides one set of heading templates for 370 and another for 370XA reports
2	Is the event history. It provides information for up to 256 processors (CPUs).
3	Is a summary, by CPU identifier, of all the records presented in the report, with totals for each record type. It provides information for up to 16 CPUs. If your installation has more than 16 CPUs, EREP produces the report using records from the first 15 CPUs it encounters. Information from the remaining CPUs is grouped together under column heading CPUS>E. Refer to How EREP Assigns Numbers to CPUs in the *EREP Reference* for an explanation of the identifiers.

It is possible to have multiple internal CPUs reported under one serial number and thus increase EREP's capabilities. Refer to SYSIMG Control Statement in the *EREP Reference* for details.

Generating Event History Reports

Turn to the headings shown in the following table for examples of statements used to generate the event history reports in each operating system:

OP. SYSTEM	REFER TO
MVS	“Step 4: Generating an Event History Report” on page 32
VM	“Step 4: Generating Event History Reports” on page 52
VSE	“Step 4: Generating Event History Reports” on page 71

Refer to EVENT — Event History (Report Parameter) in the *EREP Reference* for details about the event history report parameter.

Use the following selection parameters to customize your event history report:

CPU
CUA
DATE
DEV
MODE
TERMN
TIME
TYPE
VOLID

Note: Specifying parameters other than these may result in misleading reports. See Table 1 on page 11 for restrictions on using parameter combinations.

Important: The CONTROLLER control statement is not appropriate for the event history because the report is a *chronological* presentation of errors.

Table 4 on page 21 shows the type of error records and their source in the event history report.

Table 4. Error Record Types and Sources for Reports

TYPE	SOURCE
A1	External time reference
A2	Serial link
A3	I/O devices
CCH	CPUs, channels
CRW	CPUs, channels
DDR	I/O devices
EOD	operating systems
IPL	Operating systems
MCH	CPUs, non-TP devices
MDR	I/O devices
MIH	Missing interrupt handler
OBR	I/O devices
SFT	MVS operating system
SLH	CPUs, channels

Table 4. Error Record Types and Sources for Reports (continued)

TYPE	SOURCE

System Exception Report Series

The system exception series is a series of reports that list software and hardware error data in a variety of ways to help you identify problems within your subsystems.

Refer to System Exception Report Series in the *EREP Reference* for a set of sample reports.

Description of the System Exception Series

The system exception report series contains a two-part system error summary and a series of subsystem exception reports. EREP accumulates error data and usage statistics on subsystem components then summarizes the information by component for the subsystem exception reports. These reports are produced for some hardware subsystems, but not all of them. To find which subsystems generate system exception reports refer to Product-Dependent Information in the *EREP Reference*.

System Error Summary

The system error summary presents data in chronological order. The report has two parts as shown in the following table:

PART	DESCRIPTION
1	• Presents CPU errors and channel checks • Prints a summary of IPL, EOD, and restart records • Prints one page of output for each supported CPU in the installation
2	• Combines the I/O errors for all supported subsystems, DASD, optical, and tape • Includes physical IDs, error descriptions, and probable failing units

The probable failing unit (PFU) is the component on which the error most likely occurred and is shown for:

- CPU errors
- Channel errors
- DASD errors
- Tape errors

The following table shows the type of error records and their source in parts 1 and 2 of the system error summary.

TYPE	SOURCE
CCH	CPUs, channels
DDR	I/O devices; including channels, SCUs, controllers, volumes EOD operating systems
IPL	Operating systems
MCH	CPUs
OBR	I/O devices; including channels, SCUs, controllers, volumes

Subsystem Exception Report Series

EREP formats each of the reports in the subsystem exception report series according to the requirements of the hardware involved.

EREP produces a different series of subsystem exception reports for each type of hardware.

The following table shows the type of error records and their source in the subsystem exception report series.

TYPE	SOURCE
A3	33XX DASD, 34XX Tape
CCH	CPUs, channels
MCH	CPUs
MDR	33XX DASD, 34XX Tape, 3995 Optical
OBR	33XX DASD, 34XX Tape, 3995 Optical, 9246 Optical, 9247 Optical

Generating System Exception Reports

Turn to the headings shown in the following table for examples of statements used to generate the system exception reports in each operating system:

OP. SYSTEM	REFER TO
MVS	"Step 3: Generating a System Exception Report" on page 31
VM	"Step 3: Generating System Exception Reports" on page 51
VSE	"Step 3: Generating System Exception Reports" on page 71

Note:

1. Refer to SYSEXN — System Exception Reports (Report Parameter) in the *EREP Reference* for details about the SYSEXN report parameter.
2. Set up additional controls for the system exception reports, using the DASDID, LIMIT and SHARE control statements, before you request the report series. Refer to EREP Parameters and EREP Control Statements in the *EREP Reference* for detailed descriptions.
3. Specifying report parameters other than DATE and TIME may result in misleading reports.
4. Run the system exception report series every day to avoid reworking the same errors and to make sure that the probable failing unit analysis is accurate.

Threshold Summary Report

The threshold summary report shows all the permanent read/write errors, temporary read/write errors, and media statistics for each volume mounted, using the OBR and MDR records, for 3410, 3420, and 8809 tape devices.

Note: The system exception series is a replacement for the threshold summary. Consider switching to the system exception series.

Refer to Threshold Summary Report in the *EREP Reference* for a set of sample reports.

Description of the Threshold Summary Report

The data in the threshold summary report is grouped by tape subsystem. The report has four sections as shown in the following table:

SECTION	DESCRIPTION
DEV(ice) STATISTICS	Shows one line of statistical and error data for every demount record whose error count exceeds the read or write threshold you coded on the report parameter.
PERMANENT ERROR SUMMARY	Shows a one-line entry for *every* permanent error. A permanent error can be a read error, a write error, or an equipment check. This section ignores threshold settings so there are no limits.
TEMPORARY ERROR SUMMARY	Shows a summary of all temporary errors recorded for each device number or CUA, whether they exceeded your threshold or not.
VOLUME STATISTICS	Shows the errors and usage statistics by volume serial number using *each* MDR and OBR record from the first three sections of the report. This section also ignores threshold settings so there are no limits.
Note: • The first three sections appear once for each processor in your installation. • The columns in the fourth section of the report are titled differently depending on the device type involved.	

Note:

1. Refer to Threshold Summary Report Information in the *EREP Reference* for how the columns differ and for the device types supported by the threshold summary reports.
2. Information for up to 256 CPUs can be provided in the threshold summary.
3. It is possible to have multiple internal processors reported under one serial number and thus increase EREP's capabilities. Refer to SYSIMG Control Statement in the *EREP Reference* for details.

Generating Threshold Summary Reports

Turn to the headings shown in the following table for examples of statements used to generate the system summary reports in each operating system:

OP. SYSTEM	REFER TO
MVS	"Step 5: Generating a Threshold Summary Report" on page 33
VM	"Step 5: Generating Threshold Summary Reports" on page 52
VSE	"Step 5: Generating Threshold Summary Reports" on page 72

Note: Refer to THRESHOLD — Threshold Summary (Report Parameter) in the *EREP Reference* for details about the threshold summary report parameter.

Use the parameters shown in the following table to customize your threshold summary report:

Parameters			
CUA	DEV	MODE	VOLID
DATE	DEVSER	TIME	

Note: Specifying parameters other than these may result in misleading reports. See Table 1 on page 11 for restrictions on using parameter combinations.

The following table shows the type of error records and their source in the threshold summary.

TYPE	SOURCE
MDR	3410, 8809 tape devices
OBR	3410, 3420, 8809 tape devices

Detail Edit and Summary Reports

The detail edit and summary reports provide environmental information, hexadecimal dumps and summaries of errors to determine their nature and causes.

Refer to Detail Edit and Summary Reports in the *EREP Reference* for a set of sample reports.

Description of the Detail Edit and Summary Reports

The detail edit and summary reports allow you to look at the error records on the two levels shown in the following table:

REPORT TYPE	DESCRIPTION
Detail edits	Format every record you have selected on a separate page, including a hexadecimal dump of the record
Detail summaries	Summarize selected data from the record and total the number of records that meet your selection criteria; some detail summaries show only the total number of selected records. EREP produces one detail summary per processor (CPU) for each record type selected.

Note:

1. The format and content of the detail edits and summaries vary according to the type of record and the device or product involved.
2. These reports cover all products and devices and all record types except DASD CCH.
3. DASD does not use the combined outboard record/miscellaneous data record (OBR/MDR) detail summary (PRINT=PS|SD|SU,TYPE=OT) or the MDR detail edit and summary reports, because the DASD subsystem exception report summarizes the DASD devices.
4. VTAM OBRs do not appear on the print summary reports.

Generating Detail Edit and Summary Reports

Turn to the headings shown in the following table for examples of statements used to generate the detail edit and summary reports in each operating system:

OP. SYSTEM	REFER TO
MVS	“Step 6: Generating a CCH and MCH Detail Edit Report” on page 34
	“Step 7: Generating an MDR and OBR Detail Report for Controllers” on page 35
	“Step 8: Generating a Detail Summary for I/O Errors” on page 36
	“Step 9: Generating a Detail Edit Report for Software Records” on page 37
VM	“Step 6: Generating CCH and MCH Detail Reports” on page 53
	“Step 7: Generating MDR and OBR Detail Reports for Controllers” on page 53
	“Step 8: Generating Detail Summaries for I/O Errors” on page 54
	“Step 9: Generating Detail Reports for Software Records” on page 54

OP. SYSTEM	REFER TO
VSE	“Step 6: Generating MCH, CCH and CRW Detail Reports” on page 72
	“Step 7: Generating MDR and OBR Detail Reports for Controllers” on page 73
	“Step 8: Generating Detail Summary reports for I/O Errors” on page 74
	“Step 9: Generating Detail Reports for Software Records” on page 75

Note:

1. Refer to PRINT — Print reports (report parameter) in the *EREP Reference* for details about the PRINT report parameter.
2. Every selection parameter except DEVSER is valid with the PRINT report parameter.
3. The PRINT parameter can produce *data reduction* reports for some devices that format and summarize environmental data gathered by the device. See Data Reduction Report for an example of this report.
4. Use selection parameters to limit the PRINT reports, if you do not want to see detailed reports for every error record on your ERDS or history file.

   An EREP run in which you request PRINT reports can generate a large quantity of printed output. When you code PRINT=PT without using the date, time and type selection parameters, EREP produces a detail edit of every available record. Coding PRINT=PS produces those same detail edits, plus detail summaries of every type of record EREP found in the input file.
5. The PRINT report can provide information for up to 256 CPUs, except for PRINT=PT, which can report on an unlimited number of CPUs.
6. It is possible to have multiple internal processors reported under one serial number. Refer to SYSIMG Control Statement in the *EREP Reference* for details.
7. Table 4 on page 21 shows the types of error records and their sources for the detail edit and summary reports.

Chapter 4. Running EREP under MVS

MVS systems require system controls that create the interface between EREP and the operating system. This section contains information that is needed to run EREP on the MVS operating system. The following table shows where to find the subjects and examples in this section:

HEADING
“Example Descriptions” on page 27
“Step 1: Creating a History Data Set” on page 29
“Step 2: Generating a System Summary Report” on page 30
“Step 3: Generating a System Exception Report” on page 31
“Step 4: Generating an Event History Report” on page 32
“Step 5: Generating a Threshold Summary Report” on page 33
“Step 6: Generating a CCH and MCH Detail Edit Report” on page 34
“Step 7: Generating an MDR and OBR Detail Report for Controllers” on page 35
“Step 8: Generating a Detail Summary for I/O Errors” on page 36
“Step 9: Generating a Detail Edit Report for Software Records” on page 37
“Step 10: Updating a History Tape” on page 38
“Step 11: Generating a Trends Report” on page 39
“MVS System Controls” on page 40
“Coding the JCL” on page 42
“MVS Storage Requirements” on page 43
“Information about the MVS System Control Program (SCP)” on page 44
“Information about the ERDS” on page 45
“Running EREP in a Multisystem Environment” on page 46
“Automating the Running of EREP” on page 47

Example Descriptions

Use the examples in this topic as models for your EREP runs. The detailed explanations in later sections are provided to help you understand the examples.

The following is an example of job control language (JCL) to execute a series of EREP reports as it would appear in a file without the annotation of the more detailed example provided in “Step 1: Creating a History Data Set” on page 29 through “Step 10: Updating a History Tape” on page 38:

```
//EREPPRNT JOB ,ESTER,
//  MSGCLASS=T,NOTIFY=C961231,USER=C961231
//*-------------------------------------------------------------------*/
//*  STEP0:    COPIES SYS1.LOGREC TO TEMPORARY DATA SET              */
//*-------------------------------------------------------------------*/
//S0      EXEC PGM=IFCEREP1,REGION=1024K,
//             PARM='ACC,ZERO=N'
//SERLOG   DD  DISP=SHR,DSN=SYS1.LOGREC
//ACCDEV   DD  DISP=(NEW,PASS),DSN=&ERRDATA,
//             UNIT=SYSDA,SPACE=(CYL,(2,2)),
//             DCB=(RECFM=VB,BLKSIZE=6144)
//DIRECTWK DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//EREPPT   DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST  DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN    DD  DUMMY
//*
//*-------------------------------------------------------------------*/
//*  STEP1:    PRINTS SYSTEM SUMMARY REPORT                          */
//*-------------------------------------------------------------------*/
//S1      EXEC PGM=IFCEREP1,REGION=1024K,
//             PARM='HIST,ACC=N,SYSUM'
//ACCIN    DD  DISP=(OLD,PASS),DSN=&ERRDATA
//DIRECTWK DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//EREPPT   DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST  DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN    DD  DUMMY
//*
//*-------------------------------------------------------------------*/
//*  STEP2:    PRINTS SYSTEM EXCEPTION REPORTS                       */
//*-------------------------------------------------------------------*/
//S2      EXEC PGM=IFCEREP1,REGION=1024K,
//             PARM='HIST,ACC=N,SYSEXN,TABSIZE=128K'
//ACCIN    DD  DISP=(OLD,PASS),DSN=&ERRDATA
//DIRECTWK DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//EREPPT   DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST  DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN    DD  DUMMY
//*
//*-------------------------------------------------------------------*/
//*  STEP3:    PRINTS SIM DETAIL AND EDIT SUMMARIES                  */
//*-------------------------------------------------------------------*/
//S3      EXEC PGM=IFCEREP1,REGION=1024K,
//             PARM='HIST,ACC=N,ZERO=N,PRINT=AL'
//ACCIN    DD  DISP=(OLD,PASS),DSN=&ERRDATA
//DIRECTWK DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//EREPPT   DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST  DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN    DD  DUMMY
//*
//*-------------------------------------------------------------------*/
//*  STEP4:    PRINTS EVENT HISTORY                                  */
//*-------------------------------------------------------------------*/
//S4      EXEC PGM=IFCEREP1,REGION=1024K,
//             PARM='HIST,ACC=N,EVENT'
//ACCIN    DD  DISP=(OLD,PASS),DSN=&ERRDATA
//DIRECTWK DD  DISP=(NEW,DELETE),UNIT=SYSDA,SPACE=(CYL,(10,10))
//EREPPT   DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST  DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN    DD  DUMMY
```

The JCL in “Step 1: Creating a History Data Set” on page 29 through “Step 10: Updating a History Tape” on page 38 is an example of one way to set up an EREP run with several steps.

This is only an example: You must decide which reports are relevant to your installation, in what order they should be generated, and how often they should be run.

Important: Jobs coded to invoke EREP upon receipt of messages indicating that LOGREC is near-full must be coded with the correct message number. An IFB060E is issued when LOGREC is near full on an MVS 4.2 system. An IFB080E is issued when LOGREC is near full on an MVS 5.2 or OS/390 R1 and above system.

“Step 1: Creating a History Data Set” on page 29 creates a history data set that is used in the other reports. By creating a history file and then running all the reports against that file, you ensure that all of the reports are using the same set of records.

The parameters and control statements for this example are shown in the data sets listed on the SYSIN (DD) statements. The data sets containing the EREP parameters (EREP.PARMS) are included with each step of the example.

Table 5 on page 29 shows some of the EREP control statements that could be in the EREP.CONTROLS data set used in each step of the example.

Table 5. Contents of EREP.CONTROLS

EREP CONTROL STATEMENTS	EXPLANATION
`LIMIT 3830,EQUCHK=5,OVRN=10`	To limit the number of records appearing on the reports. The DASD subsystem exception report shows temporary equipment checks and overrun errors for a 3830 control unit only if there are 5 or more equipment checks or 10 or more overruns recorded against the device.
`SHARE=(011111.01BX,022222.02BX)`	Causes the records from DASD drive 0 (device addresses 01B0 and 02B0) to be combined and presented as data for 01B0 on CPU 011111.

Note: In Table 5 on page 29:

- The control statements are specific to the installation and the reports requested.
- "EREP Control Statements" on page 10 describes what each of the control statements does.
- EREP Control Statements in the *EREP Reference* provides the information to code the control statements.

"Parameters and Control Statements" on page 8 describes how to use the EREP parameters and EREP control statements. The EREP parameters and control statements are described in greater detail in Introduction to EREP Controls in the *EREP Reference*.

Step 1: Creating a History Data Set

Use the following example to:

- Initiate an EREP job with several steps.
- Create a history data set to use in later report generation.
- Copy the records from the ERDS to another disk data set.
- Zero the ERDS.

EXAMPLE JCL	EXPLANATION
`//EREPDAY JOB [accounting information]… MSGLEVEL=1`	Initiate EREPDAY, an EREP job with several steps.
`//STEP1 EXEC PGM=IFCEREP1,PARM='CARD'`	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
`//SERLOG DD DSN=SYS1.LOGREC,DISP=SHR`	Define the system error recording data set (ERDS) as the input data set. (The default name of the ERDS is SYS1.LOGREC; but for MVS release 5.1 and later, it can be installation modified.)

`//ACCDEV    DD  UNIT=SYSDA,DSN=EHISTORY,`
`//             DISP=(NEW,PASS,CATLG),`
`//             SPACE=(CYL,(10,5)),`
`//             DCB=(RECFM=VB,BLKSIZE=4000)`
	Define and allocate space for the output history data set. The ACCDEV data set receives the records EREP copies from the ERDS. This data set is used as input for the rest of reports in EREPDAY.

EXAMPLE JCL	EXPLANATION
`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP1),`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP1), are listed below.

`//          DD  DSN=EREP.CONTROLS,`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP control statements needed for all the reports, see Table 5 on page 29. EREP uses only those that apply to the report requested in this step.

/*	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP1) contains the following EREP parameters:

EREP PARAMETERS	EXPLANATION
PRINT=NO	Do not format any reports yet
ACC=Y	Direct EREP to copy the records from the ERDS to EHISTORY.
ZERO=Y	Direct EREP to clear the ERDS.
ENDPARM	End of EREP parameters. ENDPARM must begin on column 1. Parameters may be indented.

Step 2: Generating a System Summary Report

Use the following example to generate a system summary report from the records on the working history data set:

EXAMPLE JCL	EXPLANATION
//STEP2 EXEC PGM=IFCEREP1,PARM='CARD'	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)	Define the history input data set. EHISTORY is the data set created in "Step 1: Creating a History Data Set" on page 29. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//              SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP2),`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP2), are listed below.

EXAMPLE JCL	EXPLANATION

`//          DD   DSN=EREP.CONTROLS,`
`//               DISP=(OLD,PASS)`
	Define the data set containing the EREP control statements needed for all reports, see Table 5 on page 29. EREP uses only those that apply to the report requested in this step.

`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP2) contains the following EREP parameters:

EREP PARAMETERS	EXPLANATION
SYSUM	Request the system summary.
TYPE=MC	Select the records by type: **C** CCH/CRW/SLH: Channel check/channel report word/subchannel logout records **M** MCH: Machine check records
HIST	Input records are in the history data set.
ACC=N	No history data set generated. When you code ACC=Y with SYSUM, EREP always clears the ERDS, even if you code ZERO=N.
ENDPARM	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

Step 3: Generating a System Exception Report

Use the following example to produce a series of system exception reports:

EXAMPLE JCL	EXPLANATION

`//STEP3   EXEC  PGM=IFCEREP1,REGION=4M,`
`//             PARM='CARD'`
	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42. and increase the amount of virtual storage (region size). The system exception series requires a large sort table (TABSIZE), and needs more virtual storage (REGION). “MVS Storage Requirements” on page 43 describes the virtual storage requirements, and “Increasing Region Size” on page 44 describes how to increase the region size.

`//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)`	Define the history input data set. EHISTORY is the data set created in “Step 1: Creating a History Data Set” on page 29. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//             SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.

EXAMPLE JCL	EXPLANATION
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP3),`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP3), are listed below.

`//          DD  DSN=EREP.CONTROLS,`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP control statements needed for all the reports, see Table 5 on page 29. EREP uses only those that apply to the report requested in this step.

/*	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP3) contains the following EREP parameters:

EREP PARAMETERS	EXPLANATION
SYSEXN	Request the system exception series.
HIST	Input records are in the history data set.
ACC=N	No history file generated.
TABSIZE=512K	System exception processing requires a large sort table.
ENDPARM	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

Step 4: Generating an Event History Report

Use the following example to generate an event history report:

EXAMPLE JCL	EXPLANATION
//STEP4 EXEC PGM=IFCEREP1,PARM='CARD'	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)	Define the history input data set. EHISTORY is the data set created in "Step 1: Creating a History Data Set" on page 29. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//              SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP4),`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP4), are listed below.

EXAMPLE JCL	EXPLANATION
	EREP control statements do not apply to the event history report, so the EREP.CONTROLS data set is not needed.
`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP4) contains the following EREP parameters:

EREP PARAMETERS	EXPLANATION
`EVENT`	Request an event history report. Note the absence of selection parameters; the report is to include every record.
`HIST`	Input records are in the history data set.
`ACC=N`	No history data set generated.

Step 5: Generating a Threshold Summary Report

Use the following example to produce threshold summary reports:

EXAMPLE JCL	EXPLANATION
`//STEP5 EXEC PGM=IFCEREP1,PARM='CARD'`	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
`//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)`	Define the history input data set. EHISTORY is the data set created in "Step 1: Creating a History Data Set" on page 29. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//             SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP5),`
`//             DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP5), are listed below.

`//          DD  DSN=EREP.CONTROLS,`
`//             DISP=(OLD,PASS)`
	Define the data set containing the EREP control statements needed for all reports, see Table 5 on page 29. EREP uses only those that apply to the report requested in this step.

`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP5) contains the following EREP parameters:

EREP PARAMETERS	**EXPLANATION**
`THRESHOLD=(001,015)`	Request a threshold summary.
`HIST`	Input records are in the history data set.
`ACC=N`	No history data set generated. The default value for ACC with THRESHOLD is N—an exception to the rule—but it is wise to code the parameter anyway.
`ENDPARM`	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

Important: The system exception series is a replacement for the threshold summary. Consider switching to the system exception series.

Step 6: Generating a CCH and MCH Detail Edit Report

Use the following example to generate a set of detail edit and summary reports of all machine and channel checks:

EXAMPLE JCL	**EXPLANATION**
`//STEP6 EXEC PGM=IFCEREP1,PARM='CARD'`	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
`//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)`	Define the history input data set. EHISTORY is the data set created in "Step 1: Creating a History Data Set" on page 29. Multiple history data sets can be concatenated on this DD statement.
`//DIRECTWK  DD  UNIT=SYSDA,` `//              SPACE=(CYL,(10,10))`	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.
`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.
`//SYSIN     DD  DSN=EREP.PARMS(STEP6),` `//              DISP=(OLD,PASS)`	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP6), are listed below.
`//          DD  DSN=EREP.CONTROLS,` `//              DISP=(OLD,PASS)`	Define the data set containing the EREP control statements needed for all the reports. EREP uses only those that apply to the report requested in this step.
`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP6) contains the following EREP parameters:

EREP PARAMETERS	**EXPLANATION**
`PRINT=PS`	Request detail edits and summaries of the input records.

EREP PARAMETERS	EXPLANATION
TYPE=MC	Select the records by type: **C** CCH/CRW/SLH: Channel check/channel report word/subchannel logout records **M** MCH: Machine check records
HIST	Input records are in the history data set.
ACC=N	No history data set generated.
ENDPARM	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

Step 7: Generating an MDR and OBR Detail Report for Controllers

Use the following example to generate a set of detail summary reports of all errors for the following communications controllers:

- 3704
- 3705
- 3720
- 3725
- 3745

EXAMPLE JCL	EXPLANATION
`//STEP7 EXEC PGM=IFCEREP1,PARM='CARD'`	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
`//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)`	Define the history input data set. EHISTORY is the data set created in “Step 1: Creating a History Data Set” on page 29. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//              SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP7),`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents for the parameter data set, EREP.PARMS(STEP7), are below.

`//          DD  DSN=EREP.CONTROLS,`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP control statements needed for all the reports, see Table 5 on page 29. EREP uses only those that apply to the report requested in this step.

`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP7) contains the following EREP parameters:

EREP PARAMETERS	EXPLANATION
PRINT=SU	Request only detail summaries.
TYPE=OT	Select the records by type. **O** OBR: Outboard records; unit checks **T** MDR (formerly TPR): Miscellaneous data records
DEV=(3704,3705,3720,3725,3745)	Select by device type.
HIST	Input records are in the history data set.
ACC=N	No history data set generated.
ENDPARM	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

Step 8: Generating a Detail Summary for I/O Errors

Use the following example to generate a set of detail summary reports of all I/O errors not already covered in the preceding reports:

EXAMPLE JCL	EXPLANATION
`//STEP8 EXEC PGM=IFCEREP1,PARM='CARD'`	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
`//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)`	Define the history input data set. EHISTORY is the data set created in "Step 1: Creating a History Data Set" on page 29. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//             SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP8),`
`//             DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP8), are listed below.

`//          DD  DSN=EREP.CONTROLS,`
`//             DISP=(OLD,PASS)`
	Define the data set containing the EREP control statements needed for all the reports, see Table 5 on page 29. EREP uses only those that apply to the report requested in this step.

/*	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP8) contains the following EREP parameters:

EREP PARAMETERS	EXPLANATION
PRINT=SU	Request only detail summaries.

EREP PARAMETERS	EXPLANATION
`TYPE=DOTH`	Select the records by type: **D** DDR: Dynamic device reconfiguration records **O** OBR: Outboard records; unit checks **T** MDR (formerly TPR): Miscellaneous data records **H** MIH: Missing interrupt records
`DEV=(N34XX,N3704,N3705,N3720,N3725,N3745 )`	Select by device type; excluding those already covered.
`HIST`	Input records are in the history data set.
`ACC=N`	No history data set generated.
`ENDPARM`	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

Step 9: Generating a Detail Edit Report for Software Records

Use the following example to generate a set of detail edit and summary reports of all software and operational records:

EXAMPLE JCL	EXPLANATION
`//STEP9 EXEC PGM=IFCEREP1,PARM='CARD'`	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
`//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)`	Define the history input data set. EHISTORY is the data set created in "Step 1: Creating a History Data Set" on page 29. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//              SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP9)`
`//              DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP9), are listed below.

	EREP control statements do not apply to software records, so the EREP.CONTROLS data set is not needed.
`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP9) contains the following EREP parameters:

EREP PARAMETERS	**EXPLANATION**
PRINT=PS	Request both detail edits and summaries.
TYPE=SIE	Select the records by type: **S** Software (SFT): System abends and other software events **I** System initialization (IPL): Initial program load **E** System termination (EOD): End of day and other terminating events
HIST	Input records are in the history data set.
ACC=N	No history data set generated.
ENDPARM	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

Step 10: Updating a History Tape

Use the following example to:

- Copy the records on the input data set (EHISTORY) to the history tape (EREP.HIST.TAPE).
- Delete the old history data set (EHISTORY).
- Use the updated history tape as the input for the final step, “Step 11: Generating a Trends Report” on page 39.

EXAMPLE JCL	**EXPLANATION**
//STEP10 EXEC PGM=IFCEREP1,PARM='CARD'	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.
//ACCIN DD DSN=EHISTORY,DISP=(OLD,PASS)	Define the history input data set. EHISTORY is the data set created in “Step 1: Creating a History Data Set” on page 29. Multiple history data sets can be concatenated on this DD statement.
//ACCDEV DD DSN=EREP.HIST.TAPE, // DISP=(MOD,PASS), // VOL=(,RETAIN), // DCB=(RECFM=VB, // BLKSIZE=12000)	EREP.HIST.TAPE receives the records. If EREP.HIST.TAPE does not exist, use DISP=(NEW,PASS) to create it.
//DIRECTWK DD UNIT=SYSDA, // SPACE=(CYL,(10,10))	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.
//EREPPT DD DUMMY	Dummy statement since no report is to be formatted.
//TOURIST DD SYSOUT=A,DCB=BLKSIZE=133	Define and allocate space for the output data set that holds EREP messages and processing information.
//SYSIN DD DSN=EREP.PARMS(STEP10), // DISP=(OLD,PASS)	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP10), are listed below.

EXAMPLE JCL	EXPLANATION
	The EREP.CONTROLS data set is not needed.
`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP10) contains the following EREP parameters:

EREP PARAMETERS	EXPLANATION
PRINT=NO	Request no report output.
HIST	Input records are in the history data set.
ACC=Y	The records are to be copied from the working data set to the output data set named on the ACCDEV DD statement.

Step 11: Generating a Trends Report

Use the following example to generate a trends report covering a maximum of 30 days of records from the newly updated history tape:

EXAMPLE JCL	EXPLANATION
`//STEP11 EXEC PGM=IFCEREP1,PARM='CARD'`	Execute IFCEREP1 with the parameters in the SYSIN DD statement, see Table 7 on page 42.

`//ACCIN     DD  DSN=EREP.HIST.TAPE,`
`//             DISP=(OLD,KEEP)`
	Define the history input data set. EREP.HIST.TAPE is the "new" input data set, containing the records EREP is to use for this trends report. EREP.HIST.TAPE is the data set created in "Step 10: Updating a History Tape" on page 38. Multiple history data sets can be concatenated on this DD statement.

`//DIRECTWK  DD  UNIT=SYSDA,`
`//             SPACE=(CYL,(10,10))`
	Define and allocate DASD space for the temporary work data set needed to process history (ACCIN) input records.

`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds the EREP report.
`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`	Define and allocate space for the output data set that holds EREP messages and processing information.

`//SYSIN     DD  DSN=EREP.PARMS(STEP11),`
`//             DISP=(OLD,PASS)`
	Define the data set containing the EREP parameters needed for this step. The contents of the parameter data set, EREP.PARMS(STEP11), are listed below.

`//          DD  DSN=EREP.CONTROLS,`
`//             DISP=(OLD,PASS)`
	Define the data set containing the EREP control statements needed for all the reports, see Table 5 on page 29. EREP uses only those that apply to the report requested in this step.

`/*`	This delimiter is optional; refer to the JCL manual for your MVS system.

EREP.PARMS(STEP11) contains the following parameters:

EREP PARAMETERS	EXPLANATION
TRENDS	Request the trends report. Without the DATE parameter, trends uses the last 30 days of records.
HIST	Input records are still in a history data set rather than the ERDS. The history data set referred to here is the one updated or created in Step 10.
ACC=N	No history data set generated.

EREP PARAMETERS	EXPLANATION
ENDPARM	End of parameters. ENDPARM must begin on column 1. Parameters may be indented.

MVS System Controls

MVS systems require system controls that create the interface between EREP and the operating system's data management functions. You provide these system controls as part of the EREP run, as follows:

//JOB statement
Required; initiates the job.

//EXEC statement
Required; executes the EREP program.

The following table shows where to find information about the coding options for the EREP parameters on the EXEC statement:

TO CODE THE PARAMETERS	SEE
As in-stream data	Table 7 on page 42 and the SYSIN DD statement in "MVS System Controls" on page 40
Or on the JCL EXEC statement	Table 7 on page 42

Important: The EXEC statement is one place to request more storage to accommodate EREP using the REGION parameter. See "Increasing Region Size" on page 44 for more information.

//ACCIN DD statement
Optional; defines the history input data set.

Important: ACCIN file *MUST* have been created from an EREP ACC=Y statement. Files built with system utilities may cause unpredictable results.

The history input can be in more than one data set. You can concatenate the DD statements, making sure the record formats (RECFM) are either blocked or unblocked but not both.

The data set with the largest blocksize must be first in the concatenation, so that the system allocates a device suitable for all the data sets.

You may use the ERDS or a history data set for input. Table 6 on page 41 shows some ways to combine input from ERDS and

See "Data Control Block (DCB) Requirements" on page 43 for information about the DCB requirements.

//DIRECTWK DD statement
Optional; defines and allocates DASD space for the temporary work data set needed to process history (ACCIN) input records.

//SERLOG DD statement
Optional; defines the system error recording data set (ERDS) as the input data set. (The default name of the ERDS is SYS1.LOGREC; but for MVS release 5.1 and later, it can be installation modified.)

Note: The SERLOG statement only defines the ERDS. You cannot use the SERLOG statement to define a data set copied from the ERDS.

When you include history data sets as input for your report you must define them on the ACCIN statement. This dataset is not EAS eligible on EAV devices. OPEN fails with 'IEC142I 113-44' if EATTR=OPT is coded in the JCL for the dataset. Table 6 on page 41 shows some ways to combine input from ERDS and history data sets.

Table 6. How to Define Inputs to EREP

WHEN INPUT IS	PARAMETER REQUIRED	DD REQUIRED
Only ERDS	Neither HIST nor MERGE	SERLOG
Only History files	HIST, not MERGE	ACCIN
Both History and ERDS	MERGE, not HIST	ACCIN and SERLOG

See “Information about the ERDS” on page 45 for information about the ERDS processing.

//ACCDEV DD statement

Optional; defines and allocates space for the output history data set. You need this statement if you want EREP to accumulate the records to an output data set after completing the report.

The following is an example of the ACCDEV statement:

```
//ACCDEV   DD  DSN=C961231.EREP.OUTPUT.DATA,DCB=(BLKSIZE=6000,
//         RECFM=VB),UNIT=SYSDA,DISP=(NEW,CATLG),
//         SPACE=(CYL,(50,10),RLSE)
```

See “Data Control Block (DCB) Requirements” on page 43 for information about the DCB requirements.

//EREPPT DD statement

Optional; defines and allocates space for the output data set that holds the EREP report.

You must code this DD statement whenever you request a report. The blksize should be a multiple of 133 for a 132 character line length. See SYSOUT in the examples in Table 7 on page 42. You may view the report at a terminal by specifying the SYSOUT class for online display.

The following example shows how to define this data set:

```
//EREPPT   DD  DSN=C961231.EREP.EREPPT30,DCB=(BLKSIZE=13300,
//         RECFM=FBM,LRECL=133),UNIT=SYSDA,
//         SPACE=(CYL,10,,),DISP=(NEW,CATLG)
```

See examples in Table 7 on page 42.

//TOURIST DD statement

Required; defines and allocates space for the output data set that holds EREP messages and processing information.

The blksize should be a multiple of 133 for a 132 character line length. See SYSOUT in the examples in Table 7 on page 42. You can send the TOURIST output to the SYSOUT class, let it default to the message class for the job, or spool it to a JES device.

//SYSIN DD statement

Required; defines the data set you use to enter EREP controls as in-stream data.

You must supply a SYSIN DD statement as follows:

- You can include EREP parameters, if you code PARM='CARD' on the EXEC statement. The parameters must precede the control statements with ENDPARM separating them. See Table 7 on page 42 for more detailed information.
- You must code EREP control statements as SYSIN data.
- You must use a dummy statement when you have no control statements or parameters to enter. Code it as:

```
//SYSIN DD DUMMY
```

“Parameters and Control Statements” on page 8 describes how to use the EREP parameters and EREP control statements. The EREP parameters and control statements are described in greater detail in Introduction to EREP Controls in the *EREP Reference*.

Coding the JCL

Table 7 on page 42 contains examples that illustrate several ways to code the JCL statements for your EREP run. Consult the JCL manual for your MVS system for further information.

Table 7. JCL Examples for Running EREP on MVS

METHOD | **EXAMPLE**

To code PARM='CARD' and enter the parameters and the control statements on the SYSIN statement:

```
//STEP    EXEC  PGM=IFCEREP1,PARM='CARD'
//ACCIN     DD  DSN=EREP.HISTORY,DISP=(OLD,PASS)
//DIRECTWK  DD  UNIT=SYSDA,SPACE=(CYL,(10,10))
//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN     DD  *
PRINT=PS
HIST
ACC=N
TYPE=OT
ENDPARM
SHARE …
LIMIT …
DASDID …
CONTROLLER …
/*
```

Note: ENDPARM must begin on column 1. Parameters may be indented.

To code the parameters on the EXEC statement when the control statements are in the data set specified on the SYSIN statement:

```
//STEP    EXEC  PGM=IFCEREP1,PARM=('PRINT=PS,HIST,ACC=N')
//ACCIN     DD  DSN=EREP.HISTORY,DISP=(OLD,PASS)
//DIRECTWK  DD  UNIT=SYSDA,SPACE=(CYL,(10,10))
//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133
//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133
//SYSIN     DD  DSN=EREP.CNTRL,DISP=(OLD,PASS)
```

The EXEC statement may be coded with or without the parentheses and with a single set of quotes *only* if all of the parameters fit on one line.

```
//STEP    EXEC  PGM=IFCEREP1,PARM='PRINT=PS,HIST,ACC=N'
⋮
```

Parentheses and the individual quotes is the preferred method.

```
//STEP    EXEC  PGM=IFCEREP1,PARM=('PRINT=PS',HIST,'ACC=N')
⋮
```

If the parameters do not fit on one line, then parentheses and individual quotes are required.

```
//STEP    EXEC  PGM=IFCEREP1,PARM=('PRINT=PS',HIST,
//                                 'ACC=N','TYPE=OT')
⋮
```

Table 7. JCL Examples for Running EREP on MVS (continued)

METHOD	EXAMPLE

To code the parameters and the control statements in data sets specified on the SYSIN statement:	`//STEP    EXEC  PGM=IFCEREP1,PARM='CARD'`
	`//ACCIN     DD  DSN=EREP.HISTORY,DISP=(OLD,PASS)`
	`//DIRECTWK  DD  UNIT=SYSDA,SPACE=(CYL,(10,10))`
	`//EREPPT    DD  SYSOUT=A,DCB=BLKSIZE=133`
	`//TOURIST   DD  SYSOUT=A,DCB=BLKSIZE=133`
	`//SYSIN     DD  DSN=EREP.PARMS,DISP=(OLD,PASS)`
	`//          DD  DSN=EREP.CNTRL,DISP=(OLD,PASS)`
	
	Note: This is the method used in the examples shown in "Step 1: Creating a History Data Set" on page 29 through "Step 10: Updating a History Tape" on page 38.

"Parameters and Control Statements" on page 8 describes how to use the EREP parameters and EREP control statements. The EREP parameters and control statements are described in greater detail in Introduction to EREP Controls in the *EREP Reference*.

Data Control Block (DCB) Requirements

The DD statements you code in the JCL for an EREP run define and allocate storage for the data sets EREP uses.

The input (ACCIN) and output (ACCDEV) data sets have special DCB requirements as shown in the following table:

Data set	DCB requirements
ACCIN	If the data set resides on an unlabeled tape volume or is not included in a data set control block (DSCB) you must supply the RECFM and BLKSIZE values.
ACCDEV	If your ACCDEV data set is not specified via your System Managed Storage ACS routines, refer to the JCL Reference Manual, values are provided here for example.

Note:

1. The blocksize for a tape data set must be at least 2004.
2. A blocksize of 6144 for a DASD data set allows for the various blocking factors among DASD and improves performance.
3. Refer to the JCL manual for your system for more information.

MVS Storage Requirements

EREP requires at least 100KB of virtual storage (region size) for its internal sort table. The recommended region size is 4MB.

The following table shows the relationship between virtual storage size and the number of records that can be processed.

FOR	ENTRIES AND RECORDS
All reports except the system exception series	Each 1KB (1024 bytes) of table size holds approximately 100 entries, so that EREP can process approximately 2400 records in a 24KB sort table. The MVS TABSIZE default, 100KB, provides for a 24KB sort table.
The system exception series	Each 1KB of table size holds approximately 20 entries, so the recommended value for TABSIZE when requesting SYSEXN is 512KB.

Increasing Region Size

EREP can use two different sorting algorithms for its reports; the faster one requires additional storage equal to TABSIZE.

EREP always tries to obtain the additional storage, and uses the faster sort routine if the storage is available.

You can significantly improve EREP's performance, if you increase your region size by the value of TABSIZE over the requirements outlined in Table 8 on page 44.

Several conditions can require you to increase the region size when running EREP. Table 8 on page 44 shows these conditions and recommended amounts of region increase for each.

Table 8. MVS Region Size Increases for EREP

INFLUENCING FACTOR	INCREASE REGION SIZE BY
You are using the TABSIZE parameter	The specified value of TABSIZE minus 4KB
You are using EREP control statements	The specified or default value of TABSIZE
You are requesting the system exception report series	6 times the specified or default value of TABSIZE
You are using any of the following selection parameters: CPU CPUCUA CUA DEV DEVSER LIA/LIBADR MOD SYMCDE VOLID	4KB for any or all
You are requesting detail edit reports (PRINT=PT, PS or AL)	4KB for each processor

Use the REGION parameter on either the JOB or EXEC statement to increase the virtual storage (region) size. Refer to the JCL manual for your MVS system for information on how to change the region size.

DASD Storage for DIRECTWK

MVS requires DASD space for EREP's temporary work data set whenever your input includes records on a history data set. You request this storage using the SPACE parameter on the DIRECTWK DD statement. Table 7 on page 42 shows examples of coding for the SPACE parameter.

The amount of storage depends on the device type and the number of records to be processed. For the capacities of different types of DASD, refer to your DASD publications and your system's hardware manuals.

Information about the MVS System Control Program (SCP)

The following information can help you avoid potential problems as you create the interface between EREP and your system control program (SCP).

Access Methods

EREP retrieves error records from the ERDS both:

- Sequentially, through the QSAM access method
- Randomly, through the MVS system macro EXCP (execute channel program)

It writes records to an output data set or buffer sequentially, through QSAM. If you request specific devices for EREP's output data, they must be supported by QSAM.

Creation and Processing of Software (SFT) Records

IBM system components' recovery routines create SFT records whenever IBM code is known or suspected to be the cause of a failure. The records contain data about:

- SCP failures
- Operator-initiated restarts
- Program damage caused by machine checks

The software records contain the system diagnostic work area (SDWA) control block and its extensions for the failing task or request block. VS1 VTAM and MVS software records reflect software abends of both application and system programs.

Information about the ERDS

The following section contains information about the ERDS on MVS. (The default name of the ERDS is SYS1.LOGREC; but for MVS release 5.1 and later, it can be installation modified.)

Important: EREP edits records that already exist; it does *not* create the error records.

Initialization of the Error Recording Data Set (ERDS) in MVS

The ERDS is created and initialized at system generation by the disk initialization program, IFCDIP00. In MVS/370* systems, the ERDS must reside on the system residence volume.

The ERDS consists of a header record followed by a time-stamp record for use in IPL records and by space for error and environmental records. Refer to *EREP Reference* for an example of the ERDS header record.

You can run the IFCDIP00 service aid to reinitialize the ERDS. You can use IFCDIP00, with the IEHPROGM utility, to reallocate the ERDS data set.

Moving or Altering the ERDS

If you move or change the size of the ERDS data set, you *must* re-IPL your system.

Clearing the ERDS When Near Full on MVS

If the ERDS is filling up too quickly or EREP goes into a wait mode because some of the I/O devices are not able to complete the transfer of data from buffered-log, you can use the EREP procedure IFCOFFLD to offload the records to another data set. IFCOFFLD preserves the data on the ERDS and gives you a summary report to help you find the problem.

The following table shows what IFCOFFLD does and gives a JCL example.

IFCOFFLD	EXAMPLE

1. Generates a system summary without dumping the buffered and in-storage statistical data to the ERDS
2. Copies the records from the ERDS to the ACCDEV data set
3. Clears the ERDS
	`//OFFLD   JOB   accounting/programmer information`
	`//STEP1   EXEC   PGM=IFCOFFLD`
	`//SERLOG  DD  DSN=SYS1.LOGREC,DISP=OLD`
	`//ACCDEV  DD  UNIT=unit,DSN=name,DISP=(status,disposition),`
	`//            DCB=(RECFM=VB,BLKSIZE=size)`
	`//TOURIST DD  SYSOUT=A,DCB=BLKSIZE=133`
	`//EREPPT  DD  SYSOUT=A,DCB=BLKSIZE=133`
	`//SYSIN   DD  DUMMY`
	`/*`

Note: You must replace the words in lower case letters on the ACCDEV statement to define the data set to store the off-loaded records. You can set up the emergency off-load job so it writes the ERDS records to either a new data set or to an existing data set, depending on how you want to handle the input to your regular EREP run.	

Statistical Data on the ERDS

EREP forces the MVS system to write statistical and usage data about your I/O devices to the ERDS when you request the following reports:

- System summary
- System exception series
- Threshold summary
- Trends report
- Any report for which you have specified a device type (DEV) or ZERO=Y

The data comes from counters that are associated with the devices. The operating system dumps this data to the ERDS in the form of MDR and OBR records.

You can see the usage statistics for I/O devices by running a detail edit report and specifying the device type (DEV parameter in "EREP Selection Parameters" on page 9).

Running EREP in a Multisystem Environment

EREP will invoke formatters for MVS constructs. The formatters invoked are downward compatible. In an environment where different releases of MVS are running, the newest release of those formatters should be invoked by either of the following methods:

- Formatting reports under the highest release system.
- Using steplib to access the highest levels of formatters in the corresponding SYS1.MIGLIB.

You can combine history data sets as input to EREP by concatenating DD statements for them on the ACCIN statement. You will need to make sure the space allocated for the DIRECTWK data set is large enough to hold all the input records, since EREP copies the records to DIRECTWK before starting its selection processing.

In a multisystem environment, where I/O devices are shared between processor systems, take special care to make sure you get complete and accurate reports about the shared devices.

The following table presents suggested procedures for running EREP in a multisystem environment.

ACTIONS	STEPS AND EXAMPLES
Put all DASDID, LIMIT and SHARE statements into a separate data set.	• Specify the data set name on the SYSIN DD statement for: – System summary – System exception series – Threshold summary – Trends report – PRINT reports for shared I/O devices • Code the EREP parameters on the EXEC statement; do not use PARM='CARD' in these steps. See Table 7 on page 42 for an example.
Reorder the EREP job steps shown in the sample EREP job in "Step 1: Creating a History Data Set" on page 29 through "Step 10: Updating a History Tape" on page 38 to run the processor and SCP detail reports *before* the system-level reports.	1. Create a history data set without requesting a report. 2. Run MCH and CCH detail edit reports. 3. Run software detail edit reports. 4. Run detail edit reports for dedicated I/O devices (for example, 2305). 5. Run event history against the working data set. 6. Concatenate the history data sets from each system on the ACCIN DD statement, then run the following reports: a. System summary b. System exception c. Detail edit reports for shared I/O devices 7. Add the records from the concatenated data sets to an existing permanent history data set. **Note:** You may not want to run *all* of these reports every time.

Recommendations:

1. Develop a technique to make sure that each system's ERDS has been copied before the first step that uses concatenated input runs. For example, include a step that creates a named data set, then test for that data set before requesting the first system-level report.
2. Install this procedure on each system in the complex, so reports can be run from any one of them at any time.

Automating the Running of EREP

EREP should be run regularly and frequently. You can set up a series of jobs in cataloged procedures, that can be started by the operator or by a timer at set intervals. You can create several procedures to cover various situations.

The sample EREP run in "Step 1: Creating a History Data Set" on page 29 through "Step 10: Updating a History Tape" on page 38, with the steps concatenated in a cataloged procedure, is an example with all the kinds of reports you may want to include.

Chapter 5. Running EREP under VM

This section contains information that is needed to run EREP on the VM SP operating system. The following table shows where to find the topics and examples in this section:

Topic
"Example Descriptions" on page 49
"Step 1: Creating a History File" on page 50
"Step 2: Generating a System Summary Report" on page 51
"Step 3: Generating System Exception Reports" on page 51
"Step 5: Generating Threshold Summary Reports" on page 52
"Step 6: Generating CCH and MCH Detail Reports" on page 53
"Step 7: Generating MDR and OBR Detail Reports for Controllers" on page 53
"Step 8: Generating Detail Summaries for I/O Errors" on page 54
"Step 9: Generating Detail Reports for Software Records" on page 54
"Step 11: Generating a Trends Report" on page 56
"Step 4: Generating Event History Reports" on page 52
"Step 10: Updating the History Tape" on page 55
"VM SP System Controls" on page 56
"Using the CPEREPXA EXEC" on page 58
"Entering CPEREPXA Operands" on page 60
"Information about the Error Recording Area (ERDS)" on page 64
"VM SP Error Recording with Guest Systems" on page 67
"Automating the Running of EREP" on page 67

Important: Error record handling is different in VM/XA and VM/ESA than for VM SP. See the XA and ESA sections in "Information about the Error Recording Area (ERDS)" on page 64. Also, the EREP userid on a VM machine is *NOT* the same as the EREP product that generates reports. Problems with the EREP userid should be dealt with by VM support, not EREP product support.

Example Descriptions

Use the examples in this topic as models for your EREP runs. The detailed explanations in later sections are provided to help you understand the examples.

The following is an example of an EXEC to run a system summary report, a system exception report, and a detail edit report as it would appear in a file without the annotation of the more detailed example provided in "Step 1: Creating a History File" on page 50 through "Step 10: Updating the History Tape" on page 55:

```
GLOBAL TXTLIB ERPTFLIB EREPLIB
************************************************************************
*  DEFINE INPUT/OUTPUT FILES AND RUN SYSTEM SUMMARY REPORT.
************************************************************************
FILEDEF ACCIN DISK IR22987 DUMREC01 A
FILEDEF ACCDEV DISK EHISTORY DAILY A
FILEDEF TOURIST DISK TOURIST SYSUM A
FILEDEF EREPPT DISK REPORT SYSUM A
CPEREPXA PARMFILE SYSUM A
************************************************************************
*  DEFINE OUTPUT FILES AND RUN SYSTEM EXCEPTION REPORT.
************************************************************************
FILEDEF TOURIST DISK TOURIST SYSEXN A
FILEDEF EREPPT DISK REPORT SYSEXN A
CPEREPXA PARMFILE SYSEXN A
************************************************************************
*  DEFINE OUTPUT FILES AND RUN DETAIL EDIT REPORT.
************************************************************************
FILEDEF TOURIST DISK TOURIST PRINTPS A
FILEDEF EREPPT DISK REPORT PRINTPS A
CPEREPXA PARMFILE PRINTPS A
```

Important: CPEREPXA is the current version of CPEREP.

The EREP run shown in "Step 1: Creating a History File" on page 50 through "Step 10: Updating the History Tape" on page 55 is presented as steps in an EXEC with multiple executions of the CPEREPXA EXEC using different sets of operands. The operands in the examples have been put in files named on each CPEREPXA EXEC. The contents of the files are included in each step of the example.

This is only an example: You must decide which reports are relevant to your installation, in what order they should be generated, and how often they should be run.

The first step of the EXEC defines the input and output files needed for EREP and creates a history file to use in the steps that follow. EREP automatically sends its output to the devices named on the EREPPT and TOURIST file definition statements. The execution of the CPEREPXA EXEC includes the writing and routing of the output.

"Parameters and Control Statements" on page 8 describes how to use the EREP parameters and EREP control statements. The EREP parameters and control statements are described in greater detail in Introduction to EREP Controls in the *EREP Reference*.

Step 1: Creating a History File

Use the following example to:

- Create a history file to use in later report generation.
- Copy the records from the error recording area to the working history file.
- Clear the error recording area.

EXAMPLE	EXPLANATION
`FILEDEF DIRECTWK DISK DIRECTWK EREPWORK *`	Workspace for EREP; required when there is history input. This file is for later steps. In this first step, the input is in the file defined by CPEREPXA as SERLOG; see "Defining Files for CPEREPXA" on page 56.
`DET 181`	You want to keep the records on disk, so you must detach (or redefine) TAPE 181. CPEREPXA expects the ACCDEV file to be on TAPE 181, already defined by the system. See "Overriding Input and Output FILEDEFs for CPEREPXA" on page 58.
`FILEDEF ACCDEV DISK EHISTORY DAILY * (RECFM VB BLKSIZE 12000`	Output history file.

EXAMPLE	EXPLANATION
`FILEDEF EREPPT PRINTER (NOCHANGE BLKSIZE 133`	Send the reports to the printer. Note that EREP requires a 132-position printer.
`FILEDEF TOURIST TERMINAL (BLKSIZE 133`	EREP informational messages will appear on the screen, instead of being printed with the report.
EXEC CPEREPXA FILE1 INPUT A	Execute CPEREPXA with FILE1 INPUT A, the file containing the parameters.

FILE1 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
PRINT=NO	Do not format any reports yet. Want no printed reports yet.
ACC=Y	Direct EREP to copy the records from the ERDS to EHISTORY DAILY.
ZERO=Y	Direct EREP to clear the ERDS.
ENDPARM	End of EREP parameters; EREP control statements follow.

Important: The syntax of the FILEDEF command may differ from the examples, depending upon the version of VM you are running. Refer to the Command Reference manual for your operating system to be sure you are using this command correctly.

Step 2: Generating a System Summary Report

Use the following example to produce system summary reports:

EXAMPLE	EXPLANATION
FILEDEF ACCIN DISK EHISTORY DAILY *	Redefine EHISTORY DAILY as ACCIN; this the file created in "Step 1: Creating a History File" on page 50. This file is used as input for the remaining reports.
EXEC CPEREPXA FILE2 INPUT A	Execute CPEREPXA with FILE2 INPUT A, the file containing the parameters.

FILE2 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
SYSUM	Request the system summary report.
HIST	Input records are in the history file.
ACC=N	No history file generated. When you code ACC=Y with SYSUM, EREP always clears the ERDS, even if you code ZERO=N.
ENDPARM	End of parameters; control statements follow.

Step 3: Generating System Exception Reports

Use the following example to produce system exception reports:

EXAMPLE	EXPLANATION
`EXEC CPEREPXA FILE3 INPUT A`	Execute CPEREPXA with FILE3 INPUT A, the file containing the parameters.

FILE3 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`SYSEXN`	Request the system exception series.
`HIST`	Input records are in the history file.
`ACC=N`	No history file generated.
`TABSIZE=512K`	System exception processing requires a large sort table.
`ENDPARM`	End of parameters; control statements follow.

Step 4: Generating Event History Reports

Use the following example to generate an event history report:

EXAMPLE	EXPLANATION
`EXEC CPEREPXA FILE10 INPUT A`	Execute CPEREPXA with FILE10 INPUT A, the file containing the parameters.

FILE10 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`EVENT`	Request an event history report. Note the absence of selection parameters; the report is to include every record.
`HIST`	Input records are in the history file.
`ACC=N`	No history file generated.

Step 5: Generating Threshold Summary Reports

Use the following example to produce a threshold summary report:

EXAMPLE	EXPLANATION
`EXEC CPEREPXA FILE4 INPUT A`	Execute CPEREPXA with FILE4 INPUT A, the file containing the parameters.

FILE4 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`THRESHOLD=(001,015)`	Request a threshold summary.
`HIST`	Input records are in the history file.
`ACC=N`	No history file generated.
`ENDPARM`	End of parameters; control statements follow.

Important: The system exception series is a replacement for the threshold summary. Consider switching to the system exception series.

Step 6: Generating CCH and MCH Detail Reports

Use the following example to generate detail edit and summary reports of all machine and channel checks:

EXAMPLE	EXPLANATION
`EXEC CPEREPXA FILE5 INPUT A`	Execute CPEREPXA with FILE5 INPUT A, the file containing the parameters.

FILE5 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`PRINT=PS`	Request detail edits and summaries of the input records.
`TYPE=MC`	Select the records by type: **C** CCH/CRW/SLH: Channel check/channel report word/subchannel logout records **M** MCH: Machine check records
`HIST`	Input records are in the history file.
`ACC=N`	No history file generated.
`ENDPARM`	End of parameters; control statements follow.

Step 7: Generating MDR and OBR Detail Reports for Controllers

Use the following example to generate detail summary reports of all errors for the following communications controllers:

3704
3705
3720
3725
3745

EXAMPLE	EXPLANATION
`EXEC CPEREPXA FILE6 INPUT A`	Execute CPEREPXA with FILE6 INPUT A, the file containing the parameters.

FILE6 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`PRINT=SU`	Request only detail summaries.

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`TYPE=OT`	Select the records by type: **O** OBR: Outboard records; unit checks **T** MDR (formerly TPR): Miscellaneous data records
`DEV=(3704,3705,3720,3725,3745)`	Select by device type:
`HIST`	Input records are in the history file.
`ACC=N`	No history file generated.
`ENDPARM`	End of parameters; control statements follow.

Step 8: Generating Detail Summaries for I/O Errors

Use the following example to generate detail summary reports of I/O errors not already covered in the preceding reports:

EXAMPLE	EXPLANATION
`EXEC CPEREPXA FILE7 INPUT A`	Execute CPEREPXA with FILE7 INPUT A, the file containing the parameters.

FILE7 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`PRINT=SU`	Request only detail summaries.
`TYPE=DOTH`	Select the records by type: **D** DDR: Dynamic device reconfiguration records **O** OBR: Outboard records; unit checks **T** MDR (formerly TPR): Miscellaneous data records **H** MIH: Missing interrupt records
`DEV=(N34XX,N3704,N3705,N3720,N3725,N3745 )`	Select by device type; excluding those already covered.
`HIST`	Input records are in the history file.
`ACC=N`	No history file generated.
`ENDPARM`	End of parameters; control statements follow.

Step 9: Generating Detail Reports for Software Records

Use the following example to generate detail edit and summary reports of all software and operational records:

EXAMPLE	EXPLANATION
`EXEC CPEREPXA FILE8 INPUT A`	Execute CPEREPXA with FILE8 INPUT A, the file containing the parameters.

FILE8 INPUT A contains the following CPEREPXA operands.

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`PRINT=PS`	Request both detail edits and summaries.
`TYPE=SIE`	Select the records by type: **S** Software (SFT): System abends and other software events **I** System initialization (IPL): Initial program load **E** System termination (EOD): End of day and other terminating events
`HIST`	Input records are in the history file.
`ACC=N`	No history file generated.

Step 10: Updating the History Tape

To update the history tape:

1. Copy the records from the input history file to the permanent history tape.
2. Delete the input history file.

Use the following example to update a history tape:

EXAMPLE	EXPLANATION

`FILEDEF ACCDEV TAP1`
`          (RECFM VB BLKSIZE 12000`
	This is the archive history tape, on which the working file is to be copied.
	Important: `DISP MOD` as an option on the FILEDEF statement makes the drive forward space to the end of the file before writing, rather than overwriting the file. This is only valid with a standard label tape. SL would then also be required on the FILEDEF statement.

`EXEC CPEREPXA FILE11 INPUT A`	Execute CPEREPXA with FILE11 INPUT A, the file containing the parameters.

FILE11 INPUT A contains the following CPEREPXA operands:

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`PRINT=NO`	Request no report output.
`HIST`	Input records are in the history file.
`ACC=Y`	Create a history file.

Step 11: Generating a Trends Report

Use the following example to generate a trends report covering a maximum of 30 days of records from the newly updated history tape:

EXAMPLE	EXPLANATION
`FILEDEF ACCIN TAP2 (RECFM VB BLKSIZE 12000`	Redefine the former ACCDEV file as ACCIN; the updated history tape is now being used as input.
`EXEC CPEREPXA FILE9 INPUT A`	Execute CPEREPXA with FILE9 INPUT A, the file containing the parameters.

FILE9 INPUT A contains the following CPEREPXA operands.

PARAMETERS AND CONTROL STATEMENTS	EXPLANATION
`TRENDS`	Request the trends report. Without the DATE parameter, trends uses the last 30 days of records.
`HIST`	Input records are still in a history file rather than the ERDS.
`ACC=N`	No history file generated.
`ENDPARM`	End of parameters; control statements follow.

VM SP System Controls

Enter the CPEREPXA EXEC to execute the EREP program from the CMS environment, using the following procedure:

STEP	ACTION
1	Log on to a virtual machine that has been established for the CE (normally, a Class F user ID).
2	IPL CMS.
3	Issue FILEDEF statements for the files required by EREP. See "Defining Files for CPEREPXA" on page 56.
4	Have the system operator mount any required tape volumes for use as input and output files.
5	Issue the CPEREPXA EXEC. See "CPEREPXA Operands Syntax and Coding" on page 59 and "Using the CPEREPXA EXEC" on page 58.
6	Enter CPEREPXA operands via one of the methods detailed in "Using the CPEREPXA EXEC" on page 58 and "Entering CPEREPXA Operands" on page 60.

Defining Files for CPEREPXA

The CPEREPXA EXEC processor invokes IFCEREP1. The command processor defines the files necessary for running EREP. You can change the FILEDEFs before executing the CPEREPXA EXEC and override the definitions the command processor would use. The following default file definitions are set up by CPEREPXA:

```
FILEDEF EREPPT PRINTER (NOCHANGE BLKSIZE 133
FILEDEF SYSIN DISK SYSIN EREPWORK X3
FILEDEF SERLOG DISK SERLOG EREPWORK (BLOCK 4096
FILEDEF TOURIST TERMINAL (BLKSIZE 133
FILEDEF DIRECTWK DISK DIRECTWK EREPWORK X4
FILEDEF ACCDEV TAP1 (NOCHANGE RECFM VB BLKSIZE 12000
FILEDEF ACCIN TAP2 (NOCHANGE RECFM VB BLKSIZE 12000
```

EREPPT

Is EREP's printer file, to which it sends the report output. You can override this FILEDEF with one of your own before issuing the CPEREPXA EXEC. See the following example to change the destination to TERMINAL:

```
⋮
FILEDEF EREPPT TERMINAL (NOCHANGE BLKSIZE 133
⋮
```

CPEREPXA leaves the FILEDEF for EREPPT intact at the end of the run, in case you supplied it.

SYSIN

Is the file where CPEREPXA puts your parameters and control statements. It is put on the read/write disk having the most available space, and is automatically erased at the end of the run. If there is no data for SYSIN, CPEREPXA issues:

```
⋮
FILEDEF SYSIN DUMMY
⋮
```

When you are entering operands by the file entry method, make sure the name of the file on this FILEDEF statement is the same as the file you name on the CPEREPXA EXEC line:

```
⋮
FILEDEF SYSIN DISK EREP PARMS A (RECFM F
EXEC CPEREPXA EREP PARMS A
⋮
```

SERLOG

Is a simulation of the ERDS data set, required by the OPEN and CLOSE macros that EREP issues during its processing. When SERLOG is the input file, the records are read from the VM SP error recording area; no SERLOG file exists on any disk. See "Information about the Error Recording Area (ERDS)" on page 64 for more information about VM's error recording.

TOURIST

Is the message data, which is directed to the file you defined or your terminal screen. The messages and diagnostic information EREP writes to this file include printer control characters, which might appear on the display screen as unknown characters.

DIRECTWK

Is a work file EREP uses when there is history input. This file can be quite large, because it contains all the input error records selected from the history tape. DIRECTWK is put on the read/write disk having the most available space, and is erased at the end of the run.

ACCDEV

Is the output history file, used if you specify ACC=Y. CPEREPXA puts this file on tape drive 181, but you can override that definition with your own FILEDEF prior to issuing the CPEREPXA EXEC. If you define ACCDEV to any device other than tape 181 you must detach tape 181 and issue the alternate tape or disk commands and options; see "Overriding Input and Output FILEDEFs for CPEREPXA" on page 58 for details.

CPEREPXA leaves the FILEDEF for ACCDEV intact at the end of the run.

ACCIN

Is the input history file, used if you specify HIST=Y or MERGE=Y.

Important: ACCIN file MUST have been created from an EREP ACC=Y statement. Files built with system utilities may cause unpredictable results. ACCIN must NEVER point to the XAEREPIO RECORD or XAEREPMC RECORD files. When accessing those files, the SERLOG statement as it appears in the manual is used. DO NOT address the XAEREPIO RECORD or XAEREPMC RECORD files directly in any FILEDEF. CPEREPXA puts this file on tape drive 182, but you can override that definition with your own FILEDEF prior to issuing the CPEREPXA EXEC. If you define ACCIN to any device other than tape 182 you must detach tape 182 and issue the alternate tape or disk commands and options; see "Overriding Input and Output FILEDEFs for CPEREPXA" on page 58 for details.

CPEREPXA leaves the FILEDEF for ACCIN intact at the end of the run.

Overriding Input and Output FILEDEFs for CPEREPXA

CPEREPXA puts the input history file (ACCIN) on tape drive 182 and the output history file (ACCDEV) on tape drive 181. Use the FILEDEF command to redefine the ACCDEV and ACCIN files and designate other devices when you want to do the following:

- Use a history tape from another system as input
- Accumulate the data to another tape drive
- Accumulate the data to a disk file

CPEREPXA may not use the devices you defined in your FILEDEFs if you specify the one or more of the following parameters:

- ACC=Y
- HIST=Y
- MERGE=Y

To override the CPEREPXA defaults and use the devices you defined take the following actions:

FILE	ACTION
For ACCDEV	Detach tape 181 before running CPEREPXA and do one of the following: • Issue CMS TAPE commands to position the tape you defined • Define the file to a disk
For ACCIN	Detach tape 182 before running CPEREPXA and do one of the following: • Issue CMS TAPE commands to rewind the ACCIN tape you defined • Define the file to a disk

Using the CPEREPXA EXEC

You have the following options for entering and executing the CPEREPXA command:

METHOD	DESCRIPTION	REFER TO
Prompting	Enter CPEREPXA by itself and the system prompts you for the individual operands.	"Prompting Method" on page 60
File Entry	Create a file that contains the operands for this execution of CPEREPXA. Include the name, type and mode of the file on the CPEREPXA EXEC. This is the method used in the examples in "Step 1: Creating a History File" on page 50 through "Step 10: Updating the History Tape" on page 55.	"File Entry Method" on page 62

METHOD	DESCRIPTION	REFER TO
Stacked Entry	Create a CMS EXEC with &STACK control statements to enter the operands as in-stream data before the EXEC statement.	"Stacked Entry Method" on page 62
Mixed Entry	Combine the above methods in various ways.	"Mixed Entry Method" on page 63

Note: Refer to the user's guide for your version of CMS for more information on coding CMS EXECs and executing CMS commands.

CPEREPXA Operands Syntax and Coding

To run the CMS CPEREPXA EXEC use the following syntax:

```
CPEREPXA [filename filetype {filemode | *}]
```

In order to use the CPEREPXA EXEC, you must be in the CMS environment and have a user privilege class that allows access to the records in the error recording area.

Important: With the latest releases of VM, it is possible for an installation to redefine the privilege classes overriding those set by IBM. The privilege class for EREP is usually class F.

You control EREP under VM SP using the same parameters and control statements that you use for running EREP under MVS. Those parameters and controls are *operands* for the CPEREPXA EXEC.

Coding Rules

You must follow these rules when entering operands:

- Separate a keyword and its associated values from a following keyword operand by one or more blanks, or by a comma.

```
⋮
PRINT=PS,TYPE=MC,HIST,ACC=N
ENDPARM
⋮
```

- You can put each parameter or control statement on a separate line. The sample input operand files in "Step 1: Creating a History File" on page 50 through "Step 10: Updating the History Tape" on page 55 use this method.
- You may enter operands in any sequence.
- When the allowed values of an operand where are Y and N,

```
⋮
HIST=Y,ACC=N
⋮
you may omit =Y and code only the keyword.
```

```
⋮
HIST,ACC=N
⋮
CPEREPXA always interprets this form of the operand as specifying YES,
regardless of the default value.
```

For more details see *EREP Reference*.

Unique CPEREPXA Operands

In addition to the regular EREP parameters and control statements, you need two other operands for running EREP via CPEREPXA. These are the CLEAR operand and the TERMINAL operand, both of which act as processing controls.

CLEAR operand

Clears (zeros) and reinitializes the error recording area (ERDS).

The syntax is CLEAR.

```
⋮
CLEAR
```

Important: You must have the proper user privilege class to erase and reinitialize the error recording area. In most VM SP installations, this is privilege class F.

TERMINAL operand

Instructs CPEREPXA to stop reading EREP parameters and control statements from a separate file and to prompt the user for them instead. You use this CPEREPXA operand to change your EREP controls dynamically. Having set up an input file containing CPEREPXA operands, you have the choice of using those operands or overriding them with others entered from the terminal.

The syntax is TERMINAL[=Y] | =N.

```
⋮
TERMINAL
```

See "Mixed Entry Method" on page 63 for more information.

Using EREP Controls as CPEREPXA Operands

The same rules and restrictions apply to EREP controls regardless of the system they are being used for. However, when you enter them as CPEREPXA operands, you must also follow the rules imposed by the VM SP facility. "Entering CPEREPXA Operands" on page 60 lists those rules along with several possible ways to present the operands to CPEREPXA.

Entering CPEREPXA Operands

This section covers the following methods of entering operands for CPEREPXA:

METHOD OF ENTERING OPERANDS
"Prompting Method" on page 60
"File Entry Method" on page 62
"Stacked Entry Method" on page 62
"Mixed Entry Method" on page 63

Prompting Method

Type CPEREPXA on the command line and press the Enter key, the system prompts you with:

```
ENTER PARAMETER STATEMENTS OR NULL TO PROCESS
You then type in your CPEREPXA operands.
If you fill up the entire command line before all the operands are
entered, press the Enter key again for another prompting message and a clear
command line.
You can continue in this way until all your operands for that report are
entered.
When finished, press the Enter key to signal with a null command line the end of
the string of operands.
```

To invoke CPEREPXA using only EREP's default values, respond to the first prompting message by pressing the Enter key, to enter a null line. The following table shows a complete CPEREPXA operation as initiated from the virtual machine console by entering CPEREPXA operands.

CONSOLE ACTION BY SYSTEM AND USER	EXPLANATION
`logon …`	User logs on to the CE userID.
`ipl cms`	User IPLs CMS.
⋮	
`R;`	The system indicates successful initialization of CMS.
`mount accum 181 please put ring in.`	User requests a tape for CPEREPXA use. 181 is the virtual address of the default ACCDEV tape.
`TAPE 181 ATTACHED`	The records used for the report will be copied onto the file at virtual address TAPE 181 as part of this EREP run. If the HIST function is to be used, you must also request that the history tape be attached at address 182.
fi tourist terminal fi directwk disk erep cmsut1 a fi ereppt print fi accdev tape sl (recfm vb blksize 12000 fi accin disk erep hist a (recfm vb	User defines the files needed to run EREP from VM. You can allow these to default to VM SP system FILEDEFs. See "Defining Files for CPEREPXA" on page 56.
`cperepxa`	Invokes CPEREPXA, without naming a control file. The operation defaults to the prompting method of operand entry.
`ENTER PARAMETER STATEMENTS OR NULL TO PROCESS`	The system prompts the user for EREP parameters and control statements as operands for the CPEREPXA EXEC.
`print=ps hist acc=y dev=(3480)`	User enters CPEREPXA operands. The requested report consists of detail edits and summaries of all records containing the device type code for the 3480 tape device.
`ENTER PARAMETER STATEMENTS OR NULL TO PROCESS`	The system prompts again for operand input.
endparm share=XA.cpuser.ccuX,share=XA.cpuser.ccuX	User ends parameter operand input and begins entering control statement operands.
`ENTER PARAMETER STATEMENTS OR NULL TO PROCESS`	The system prompts again for operand input.
	User enters a null line to start processing.
⋮	
`PRT FILE 2546 FROM user ID SENT TO printaddr NOHOLD`	The end of EREP processing is signalled by the CMS PRINT message. Any TOURIST messages generated by the EREP program or CMS will appear on the terminal, unless you request another output class in FILEDEF TOURIST.

CONSOLE ACTION BY SYSTEM AND USER	EXPLANATION
Note: • Lowercase letters indicate entries by the terminal user; uppercase letters indicate system responses. • Any IFCxxxI messages in the TOURIST output are from EREP; all the EREP messages are documented in EREP Messages in the *EREP Reference*. • CMS may also issue messages in the course of EREP processing, prefixed with DMSIFC or DMSREA. These can be found in the VM messages manual.	

File Entry Method

Create a file that contains the operands you want in effect for this execution of CPEREPXA. Execute the CPEREPXA EXEC with the name, type and mode of the file as operands. There must be a FILEDEF in effect for SYSIN with the same file name as your operand file. The operands are arranged in the file according to the rules listed under "Coding Rules" on page 59. Note that input records are truncated at column 71.

To invoke CPEREPXA using only EREP's default values, issue the SYSIN FILEDEF for an empty file.

In practice, you will probably want to have several different files containing the various operand combinations needed to run CPEREPXA for your installation. See the user's guide for your system editor for information on how to create a file for the operands.

The following EXEC illustrates the use of a separate file to enter CPEREPXA operands:

```
&TRACE ERR
FILEDEF EREPPT PRINTER (NOCHANGE BLOCK 133 PERM
FILEDEF TOURIST PRINTER (NOCHANGE BLOCK 133 PERM
FILEDEF DIRECTWK DISK EREP CMSUT1 &DISK?
FILEDEF ACCIN DISK HIST RECORDS A (RECFM VB
FILEDEF SYSIN DISK EREP PARMS A (RECFM F
EXEC CPEREPXA EREP PARMS A
&EXIT
```

The file named EREP PARMS A contains:

```
HIST
ACC=N
TABSIZE=500K
SYSEXN
ENDPARM
SHARE=(020402.0736,220402.0736)
SHARE=(020402.0735,220402.0735)
LIMIT 33XX,ALL=15
LIMIT 3420,HR1600=025(1),HW1600=010(15)
LIMIT 3420,VR1600=025(1),VW1600=010(15)
DASDID CPU=020402,CH=07,SCU=14,STR=0238
```

In this example:

- The system exception report output (EREPPT) and the message output (TOURIST) both go to the printer.
- The records for EREP to process are in a history file (ACCIN) on disk.

The series of sample EXECs in "Step 1: Creating a History File" on page 50 through "Step 10: Updating the History Tape" on page 55 use this method of entering operands.

Stacked Entry Method

The CMS EXEC &STACK control statement allows you to enter commands or operands as in-stream data before coding the CPEREPXA EXEC. It is another way to avoid having to recode parameters and control statements each time you run EREP.

Precede each operand by &STACK, one to each input record. CPEREPXA reads the operands in the order in which you have stacked them.

To invoke CPEREPXA from an EXEC using only EREP's default values, code only a null &STACK statement in the EXEC.

The following example illustrates the use of the &STACK control statement within a CMS EXEC to enter operands for the CPEREPXA EXEC:

```
&TRACE ERR
FILEDEF EREPPT PRINTER (NOCHANGE BLOCK 133 PERM
FILEDEF TOURIST TERMINAL (NOCHANGE BLOCK 133
FILEDEF DIRECTWK DISK EREP CMSUT1 &DISK?
FILEDEF ACCIN DISK HIST RECORDS A (RECFM VB
&STACK HIST
&STACK ACC=N
&STACK TABSIZE=500K
&STACK SYSEXN
&STACK ENDPARM
&STACK SHARE=(020402.0736,220402.0736)
&STACK SHARE=(020402.0735,220402.0735)
&STACK LIMIT 33XX,ALL=15
&STACK LIMIT 3420,HR1600=025(1),HW1600=010(15)
&STACK LIMIT 3420,VR1600=025(1),VW1600=010(15)
&STACK DASDID CPU=020402,CH=07,SCU=14,STR=0238
&STACK
EXEC CPEREPXA
&EXIT
```

In this example, the EREP controls are in line, to be read by CPEREPXA in the order they are listed. Note the null &STACK statement following the DASDID statement. Without it, CPEREPXA prompts the terminal user for more EREP control statements.

The report output produced by this example is sent to the printer, but the TOURIST output appears on the terminal screen.

See the user's guide for your version of CMS for information about coding and using EXECs.

Mixed Entry Method

You can combine the previous methods of entering CPEREPXA operands to make the process more efficient and flexible. See the procedure in the following table:

STEP	ACTION
1	Enter the CPEREPXA EXEC followed by the name of a file containing operands, one of which is the TERMINAL operand.
2	CPEREPXA reads the operands from the named file until it reaches TERMINAL.
3	Then it prompts the terminal user for operands.

This allows you to enter operands at the time of EREP's execution to dynamically tailor a report to your immediate requirements.

The following examples show two ways to enter operands using the mixed entry method.

1. With the TERMINAL operand in an input file:

```
&TRACE ERR
FILEDEF EREPPT PRINTER (NOCHANGE BLOCK 133 PERM
FILEDEF TOURIST PRINTER (NOCHANGE BLOCK 133 PERM
FILEDEF DIRECTWK DISK EREP CMSUT1 &DISK?
FILEDEF ACCIN DISK HIST RECORDS A (RECFM VB
FILEDEF SYSIN DISK EREP PARMS A (RECFM F
EXEC CPEREPXA EREP PARMS A
&EXIT
```

   This EXEC invokes CPEREPXA to produce detail edit reports of all the MCH records in the history file (HIST RECORDS A). The reports (EREPPT) and the messages (TOURIST) output are being sent to the printer.

The file named EREP PARMS A contains:

```
PRINT=PT
HIST
ACC=N
TYPE=M
TERMINAL
```

The TERMINAL operand causes CPEREPXA to prompt the user for more input. For example you could:

- Enter a DATE parameter
- Enter the CPU parameter, to narrow the selection of records from the history file
- Enter ENDPARM, followed by SHARE or CONTROLLER control statements

2. With no null &STACK statement in a CMS EXEC:

```
&TRACE ERR
FILEDEF EREPPT PRINTER (NOCHANGE BLOCK 133 PERM
FILEDEF TOURIST PRINTER (NOCHANGE BLOCK 133 PERM
FILEDEF DIRECTWK DISK EREP CMSUT1 &DISK?
FILEDEF ACCIN DISK HIST RECORDS A (RECFM VB
&STACK PRINT=PT HIST ACC=N
EXEC CPEREPXA
&EXIT
```

This EXEC produces detail edit reports of all the records on input file HIST RECORDS A. In the absence of a null &STACK statement following the &STACK statement with parameters, CPEREPXA prompts for more input. Then, you may specify the following:

- TYPE, DATE, TIME, or DEV parameter, to limit the records EREP processes
- ENDPARM Followed by SHARE or CONTROLLER statements

In this example, the report and tourist output are being sent to the printer. To change the destination of output, you must change the FILEDEFs for the output files. See “Defining Files for CPEREPXA” on page 56 for more information.

Information about the Error Recording Area (ERDS)

This section covers the following topics:

REFER TO
“Clearing the ERDS when Near Full on VM” on page 64
“Initialization of the Error Recording Area (ERDS) in VM” on page 65
“Error Record Recording and Retrieval on XA and ESA” on page 65
“ERDS Form on XA and ESA” on page 65
“ERDS Handling on XA and ESA” on page 66
“General Procedure Flow on XA and ESA” on page 66

Clearing the ERDS when Near Full on VM

Set up an EXEC for the operator to run when the page-full message appears on the console. See “Step 1: Creating a History File” on page 50 for an example. See “Entering CPEREPXA Operands” on page 60 for more about running EREP under VM.

Important: The records in the error recording area can only be off-loaded by a normal EREP run that includes ACC=Y and ZERO=Y. IFCOFFLD is not used with VM systems.

Initialization of the Error Recording Area (ERDS) in VM

The following table describes the ERDS initialization:

TO	DESCRIPTION
Initialize the ERDS	Use the CP routines that format each of the recording cylinders and set up the logical pages to receive error records. Each recording cylinder has a header followed by the space for error records.
Reinitialize the error recording area	Issue the CPEREPXA EXEC specifying CLEAR as the only operand. CLEAR creates a new header and re-formats each cylinder. For details about CLEAR see "Unique CPEREPXA Operands" on page 59.

The following table decribes the VM SP ERDS when it is on a count-key-data device or a fixed-block-architecture device:

DEVICE ARCHITECTURE	DESCRIPTION
Count-key-data device	It consists of at least two adjacent cylinders allocated on the system residence pack. The VM ESP RDS are referred to as the "error recording cylinders."
Fixed-block-architecture device	It consists of any number of adjacent pages assigned on the system residence volume. The VM SP error recording routines see the ERDS as a series of logical pages.

Error Record Recording and Retrieval on XA and ESA

In the XA and ESA environments you can control whether or not the operating system:

- Builds error records
- Builds the ERDS

The following table describes error recording and retrieval on XA and ESA.

FUNCTION	DESCRIPTION
When the recording function in VM is set off	No error records are built. Problems with the data processing environment will not be reflected in any records being passed to the ERDS.
When the recording function in VM is set on	Records are built by the operating system error recording program.
When the retrieval function in VM is set off	The records are placed in a storage buffer. The size of the storage buffer is controlled by the user. A message is issued when the storage buffer fills.
When the retrieval function in VM is set on	The records are pulled from the storage buffer and appended to the existing ERDS or used to build a new ERDS.

ERDS Form on XA and ESA

The following table describes the form of ERDS on the VM SP, XA and ESA.

VM SP	XA AND ESA
In the VM/SP environment records are built and placed in the ERDS by the operating system. There is no method for you to locate or access the system-owned ERDS.	In XA and ESA ERDS files are built on the EREP user ID A-disk. The ERDS consists of two files: • XAEREPIO RECORD • XAEREPMC RECORD These two physical files are treated by EREP the same way the ERDS on VM/SP is treated.

ERDS Handling on XA and ESA

Be very cautious when doing any file manipulations to XAEREPIO RECORD or XAEREPMC RECORD, the two files that XA and ESA build onto the EREP user ID's A-disk. You may prevent EREP from properly accessing these files. To EREP, there is no difference between these two files and the system-owned ERDS of VM SP. But these are files on a disk and the normal file handling operations can be performed upon them.

The following are some of the commonly attempted file manipulations:

- Changing the FILEDEF statements for ACCIN to point to the XAEREPIO or XAEREPMC RECORD files
- Changing the name of the XAEREPIO or XAEREPMC files to another name and accessing the files as ACCIN
- Copying the records from XAEREPIO or XAEREPMC directly to tape without running EREP against the data first
- Doing a SENDFILE of the XAEREPIO or XAEREPMC files to another system and trying to concatenate them with the other system's history data

The VM system will allow these manipulations, but the resulting problems will only occur when you try to run EREP reports using the files.

There is a two byte length indicator in the first two bytes of each of the records in ERDS. EREP processing strips the length indicator bytes off each record, and places the remaining bytes in the ACCDEV file. EREP expects the records to be in this modified format when the file is designated as the ACCIN. If you bypass the EREP file processing and do not make the required modifications EREP issues a IFC1201 message indicating that no records passed filtering in the TOURIST output.

General Procedure Flow on XA and ESA

The recommended procedural flow for an EREP run is:

STEP	ACTION
1	Log on to the EREP user ID.
2	Issue the `#cp external` command.
3	When you receive the message HCPRET592A ENTER END OR SUMMARY, enter END.
4	Run CPEREPXA to create the reports and any output files.
5	Restart record retrieval by entering RETRIEVE EREP.
6	Disconnect from the EREP machine.

Note:

1. The EREP user ID is a VM system ID that is known to the operating system. The EREP user ID is not part of the EREP product. If any service issues arise, be sure to speak with the correct service organization for the specific problem you have.

2. For more specific information on the recording and retrieval functions, refer to *VM/XA SP Real System Operation or VM/ESA System Operation*. These functions are a part of the operating system, not of EREP.

Stopping retrieval keeps new records from being written to the ERDS while EREP is reading. The records built during this time go to the storage buffer until the retrieval function is restarted.

The example job step shown in "Step 1: Creating a History File" on page 50 creates a working history file with the records from ERDS.

To minimize the time your operating system has to store records, restart the retrieval function after the working copy of ERDS is made.

VM SP Error Recording with Guest Systems

The input to EREP through VM SP can be quite different from the input used by the other systems because VM SP creates records differently.

Both the VSE and MVS systems write records to their ERDS via SVC 76. When a VSE or MVS system is running as a guest in a virtual machine, VM SP can tell when the guest system issues an SVC 76 and can divert the record to its own error recording area. In the process, VM SP translates the virtual address of the device originating the record to a real address, so the records are meaningful to a user.

VM SP does not divert every record created by a guest system to the error recording area. Records from devices dedicated to the virtual machine, of certain types, or containing an error are reflected back to the virtual machine and recorded on the guest system's ERDS.

When VM SP reflects a record back to the virtual machine, the addresses in the record remain virtual. This means that sense data logged for I/O error conditions is associated with a logical device rather than the actual device. Such sense data is of little use in identifying problem devices.

Capturing All the Data for EREP

EREP uses the records in the VM SP error recording area as the only input (unless you use the HIST or MERGE operand). The SERLOG FILEDEF, which implies VSE or MVS guest input (ERDS), is only a simulation of that file, required because of format differences between the error recording area and the system ERDS.

The result of this can be misleading reports, because the VM SP error recording area did not contain the records that would have been on a guest system ERDS. This can be a problem especially with OBR records for your TP devices. If you only run EREP under VM, you might be missing some errors.

One way to make sure you get reports about all the possible errors in your system is to run EREP under VM SP and then run it again under each guest operating system.

Another way to make sure you are seeing all your error records in the EREP reports is to combine the data from your system's ERDS and the error recording area before requesting any reports. Then run EREP under either system using the combined records as history input.

Automating the Running of EREP

EREP should be run regularly and frequently. You can set up a procedure or series of EXECS that can be started by the operator or automatically by a timer.

The sample EREP runs in "Step 1: Creating a History File" on page 50 through "Step 10: Updating the History Tape" on page 55 cover all the kinds of reports you would want to see from an EREP run. Add the system controls to make the series of EXECs into a single EXEC to run the whole series of reports; or you can name each of the EXECs and run each report separately.

Chapter 6. Running EREP under VSE

This section contains information that is needed to run EREP on the VSE operating system. The following table shows where to find the topics and examples in this section.

Topic
"Step 1: Creating a History Data Set" on page 69
"Step 2: Generating a System Summary Report" on page 70
"Step 3: Generating System Exception Reports" on page 71
"Step 5: Generating Threshold Summary Reports" on page 72
"Step 6: Generating MCH, CCH and CRW Detail Reports" on page 72
"Step 7: Generating MDR and OBR Detail Reports for Controllers" on page 73
"Step 8: Generating Detail Summary reports for I/O Errors" on page 74
"Step 9: Generating Detail Reports for Software Records" on page 75
"Step 11: Generating Trends Reports" on page 76
"Step 4: Generating Event History Reports" on page 71
"Step 10: Updating the History Tape" on page 75
"VSE System Controls" on page 77
"Special Considerations for EREP Parameters and Controls" on page 78
"VSE Storage Requirements" on page 78
"Information about the VSE System Control Program (SCP)" on page 79
"Initialization of the Error Recording Data Set (ERDS) in VSE" on page 80
"Clearing the ERDS when Near Full on VSE" on page 80
"VSE History File (IJSYSHF)" on page 81
"Running EREP in a Multisystem Environment" on page 81
"Automating the Running of EREP" on page 82

The JCS in "Step 1: Creating a History Data Set" on page 69 through "Step 11: Generating Trends Reports" on page 76 shows a VSE job with several steps. The first step creates a history data set which is used in the remainder of the steps.

This is only an example: You must decide which reports are relevant to your installation, in what order they should be generated, and how often they should be run.

All of the EREP parameters and control statements are described in Introduction to EREP Controls in the *EREP Reference*.

Step 1: Creating a History Data Set

Use the following example to:

- Create a history set
- Copy SYSREC (error recording data set) to a tape data set:

EXAMPLE	EXPLANATION
`// JOB EREPJOB`	
`// TLBL HISTOT`	Create the tape label for the output (ACCDEV) data set.

`// PAUSE   ASSIGN SYS009 TO A SCRATCH TAPE`
`// PAUSE   ISSUE ROD COMMAND`
	Instructions to the operator. Before EREP begins copying the records, the operator must mount a scratch tape for use in creating the working history tape and must issue the ROD command to force the recording of statistical data on SYSREC.

`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
	EREP parameters and control statements are entered as in-stream data.
`PRINT=NO`	Do not format any reports yet.
`ACC=Y`	Copy the records from SYSREC to SYS009 (output history data set).
`ZERO=N`	Want to preserve SYSREC so it can be merged with the monthly history tape in the job in "Step 11: Generating Trends Reports" on page 76.
`/*`	End of in-stream data.
`// MTC REW,SYS009`	Rewind the ACCDEV (now history) tape.

Step 2: Generating a System Summary Report

Use the following example to generate a system summary report from the records on the working history data set:

EXAMPLE	EXPLANATION
`// TLBL HISTINT`	Create a tape label for the input history data set (ACCIN).
`// ASSGN SYS008,SYS009`	Assign the ACCDEV data set to the input (ACCIN) logical unit.
`// ASSGN SYS009,UA`	Release the logical unit used for ACCDEV.
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`SYSUM`	Request the system summary.
`HIST`	The input records are on the working history data set, rather than SYSREC.

EXAMPLE	EXPLANATION
`ACC=N`	No history file generated. When you code ACC=Y with SYSUM, EREP always clears the ERDS, even if you code ZERO=N.
`TABSIZE=50K`	The default value for TABSIZE is 4K; judge the need for a larger sort table by the number of records on SYSREC and the kind of report being requested.
`ENDPARM`	End of parameters; control statements follow.
`SHARE …`	For I/O devices shared between processors.
`/*`	End of in-stream data.
`// MTC REW,SYS008`	Rewind the tape holding the working history data set.

Step 3: Generating System Exception Reports

Use the following example to produce system exception reports:

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`SYSEXN`	Request the system exception report series.
`HIST`	The input records are on the working history data set, rather than SYSREC.
`ACC=N`	No history file generated.
`TABSIZE=100K`	The system exception series requires a larger sort table than other reports; see "VSE Storage Requirements" on page 78.
`ENDPARM`	End of parameters; control statements follow.
`DASDID …`	For DASD without physical IDs.
`LIMIT …`	Limiting the number of records in the report.
`SHARE …`	For tape drives shared between processors.
`/*`	End of in-stream data.
`// MTC REW,SYS008`	Rewind the tape holding the history data set.

Step 4: Generating Event History Reports

The event history report contains one-line abstracts of all records, in chronological order. Use the following example to generate this report:

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`EVENT`	Request an event history. Note the absence of selection parameters; the report is to include every record.
`HIST`	Input records are in the history data set.
`ACC=N`	No history file generated.
`TABSIZE=50K`	Default is 4K.
`/*`	End of in-stream data. Event history uses no control statements.
`// MTC RUN,SYS008`	Rewind and unload the tape holding the history data set.

Step 5: Generating Threshold Summary Reports

Use the following example to produce a threshold summary report:

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`THRESHOLD=(001,015)`	Request a threshold summary.
`HIST`	Input records are in the history data set.
`ACC=N`	No history file generated.
`TABSIZE=50K`	Default is 4K.
`ENDPARM`	End of parameters; control statements follow.
`SHARE …`	For devices shared between processors.
`/*`	End of in-stream data.
`// MTC REW,SYS008`	Rewind the tape holding the history data set.

Important: The system exception series is a replacement for the threshold summary. Consider switching to the system exception series.

Step 6: Generating MCH, CCH and CRW Detail Reports

Use the following example to generate detail edit and summary reports of all machine and channel checks, and all channel report words:

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`PRINT=PS`	Requesting detail edits and summaries of the input records.
`TYPE=MC`	Selecting the records by type: **C** CCH/CRW/SLH: Channel check/channel report word/subchannel logout records **M** MCH: Machine check records
`HIST`	Input records are in the history data set.
`ACC=N`	No history file generated.
`TABSIZE=50K`	Default is 4K; not enough for detail processing of specific record types.
`/*`	End of in-stream data.
`// MTC REW,SYS008`	Rewind the tape holding the history data set.

Step 7: Generating MDR and OBR Detail Reports for Controllers

Use the following example to generate MDR and OBR detail summary reports of all errors for the following communications controllers:

```
3704
3705
3720
3725
3745
```

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`PRINT=SU`	Request only detail summaries.
`TYPE=OT`	Select the records by type: **O** OBR: Outboard records; unit checks **T** MDR (formerly TPR): Miscellaneous data records
`DEV=(3704,3705,3720,3725,3745)`	Select by device type.
`HIST`	Input records are in the history data set.

EXAMPLE	EXPLANATION
`ACC=N`	No history file generated.
`TABSIZE=50K`	Default is 4K.
`ENDPARM`	End of parameters; control statements follow.
`SHARE …`	For devices shared between processors.
`/*`	End of in-stream data.
`// MTC REW,SYS008`	Rewind the tape holding the history data set.

Step 8: Generating Detail Summary reports for I/O Errors

Use the following example to generate detail summary reports of all I/O errors not already covered in the preceding reports.

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`PRINT=SU`	Request only Detail Summaries.
`TYPE=DOTH`	Select the records by type: **D** DDR: Dynamic device reconfiguration records **O** OBR: Outboard records; unit checks **T** MDR (formerly TPR): Miscellaneous data records **H** MIH: Missing interrupt records
`DEV=(N34XX,N3704,N3705,N3720,N3725,N3745 )`	Select by device type; excluding those already covered. Note the absence of N33XX; EREP does not produce detail summaries for 33XX DASD anyway, so they need not be excluded.
`HIST`	Input records are in the history data set.
`ACC=N`	No history file generated.
`TABSIZE=50K`	Default is 4K.
`ENDPARM`	End of parameters; control statements follow.
`SHARE …`	For devices shared between processors.
`/*`	End of in-stream data.
`// MTC REW,SYS008`	Rewind the tape holding the history data set.

Step 9: Generating Detail Reports for Software Records

Use the following example to generate detail edit and summary reports of all software and operational records:

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`PRINT=PS`	Request both detail edits and summaries.
`TYPE=SIE`	Select the records by type: **S** Software (SFT): System abends and other software events **I** System initialization (IPL): Initial program load **E** System termination (EOD): End of day and other terminating events
`HIST`	Input records are in the history data set.
`ACC=N`	No history file generated.
`TABSIZE=50K`	Default is 4K.
`/*`	End of in-stream data.
`// MTC REW,SYS008`	Rewind the tape holding the history data set.

Step 10: Updating the History Tape

To update the history tape, copy the records on the input history tape (SYS008) to the permanent history tape (SYS009, as EREP.HIST.TAPE) either creating or updating it.

Use the following example to update a history tape:

EXAMPLE	EXPLANATION
`// TLBL HISTOT,'EREP.HIST.TAPE',nnn`	The output (ACCDEV) data set.
`// TLBL HISTINT,'EREP.HIST.TAPE',nnn`	The input (ACCIN) data set.

`// PAUSE  MOUNT SCRATCH TAPE AND`
`           ASSGN SYS009`
	Instructions for the operator; assigning the ACCDEV logical unit.

`// PAUSE  MOUNT EREP.HISTORY.TAPE`
`           AND ASSGN SYS008`
	Instructions for the operator; assigning the ACCIN logical unit.

EXAMPLE	EXPLANATION
`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`PRINT=NO`	Request no report output.
`MERGE`	Merge the records from the old history data set and SYSREC, which contains the latest data because it was updated while the first 8 steps were being run.
`ACC=Y`	Write the combined records to the ACCDEV tape.
`ZERO=Y`	Clear SYSREC.
`/*`	End of in-stream data.
`// MTC REW,SYS009`	Rewind the ACCDEV tape; it is the input for the final step.
`// MTC RUN,SYS008`	Rewind and unload the old history (ACCIN) tape.

Step 11: Generating Trends Reports

Use the following example to generate a trends report covering a maximum of 30 days of records from the newly updated history tape:

EXAMPLE	EXPLANATION

`// TLBL HISTINT,'EREP.HIST.TAPE',nnn`
`// ASSGN SYS008,SYS009`
	Define the former ACCDEV data set as ACCIN; the updated history tape is now being used as input.

`// EXEC IFCEREP1`	Statement to execute EREP. **Important:** Partition size must be large enough to accommodate EREP and a sort table as indicated by TABSIZE. See "VSE Storage Requirements" on page 78.
`TRENDS`	Request the trends report. Without the DATE parameter, trends uses the last 30 days of records.
`HIST`	Input records are still in a history data set rather than SYSREC.
`ACC=N`	No history file generated.
`TABSIZE=50K`	Default is 4K.
`ENDPARM`	End of parameters; control statements follow.
`SHARE …`	For devices shared between processors.
`/*`	End of in-stream data.
`// MTC RUN,SYS008`	Rewind and unload the monthly history tape.
`/&`	

VSE System Controls

VSE requires system controls to create the interface between EREP and the operating system's data management functions.

Provide with Each EREP Run

Provide the following as part of the EREP run:

// JOB JOBNAME
This statement notifies the operating system of the EREP job.

// TLBL HISTINT or // DLBL HISTIND
(// EXTENT SYS008,xxxx,1,,xxxx,x, for DASD)
Defines the tape (TLBL) or DASD (DLBL) input history data set.

The EXTENT statement is only required if the history data set resides on DASD. Refer to the appropriate VSE publications for information on coding EXTENT statements.

Important: The input history tape must have a standard label, and the blocksize for the input history data set cannot exceed 4000.

You must use either the history data set or SYSREC, or both as input to EREP.

// ASSGN SYS008,cuu
Assign the history data set to a logical unit, which is at address **cuu** (one-digit channel, two-digit unit address). You must code this statement if you use the history data set.

The history input is always assigned to SYS008.

// TLBL HISTOT or DLBL HISTOD
(// EXTENT SYS009,xxxx,1,,xxxx,xx, for DASD)
Defines the tape (TLBL) or DASD (DLBL) output history data set: the ACCDEV data set. If you code ACC=Y, you must code this label statement and the following ASSGN statement, so EREP knows where to put the records it accumulates.

The output history tape must have a standard label.

// ASSGN SYS009,cuu
Assign the output (ACCDEV) data set to logical unit SYS009.

Assignments at Initialization

The following assignments should have been made when the partition was first initialized. If they were not, you must re-IPL in order to make the assignments.

// ASSGN SYSIPT,cuu
Code EREP parameters and control statements, following the EXEC statement, as in-stream data. The system reads the data from the SYSIPT logical unit, so this ASSGN statement is always required.

// ASSGN SYSLST,cuu
Assign the data set for EREP (TOURIST) messages and EREP reports output to system logical unit SYSLST. Refer to Using the EREP Messages File (TOURIST Output) in the *EREP Reference* for information about TOURIST output.

// ASSGN SYSLOG,cuu
Assign the system log data set, required in case SYSLST is not available, to system logical unit SYSLOG.

To Execute EREP

Use the following system controls to execute the program:

// EXEC PGM=IFCEREP1,SIZE=xxxK or SIZE=AUTO
Executes the EREP program. You may need to use the SIZE parameter to make sure there is enough storage to hold the EREP program and its sort tables. See "VSE Storage Requirements" on page 78.

Important: The input and output files should be assigned to different EXTENTs.

Special Considerations for EREP Parameters and Controls

Consider the following as you prepare the JCS for your EREP run:

- You may only specify EREP parameters and control statements on input statements (as in-stream data which is read from SYSIPT).
- If you want the latest statistical and usage data included in the reports, the operator must issue the record on demand (ROD) command before running EREP against SYSREC, to force the system to dump the in-core and buffer counters to SYSREC before EREP begins its processing.
- If VSE message OP77I appears after an EREP job is submitted, increase the SIZE parameter value on the EXEC card. It might also be necessary to increase the partition size. See "VSE Storage Requirements" on page 78.

VSE Storage Requirements

EREP requires at least 100KB of virtual storage. This provides for a sort table of 4KB, the VSE TABSIZE default.

The 4KB sort table permits the processing of approximately 400 records for a report.

EREP issues the GETVIS macro to obtain storage. GETVIS requires 1KB for each 100 records over 400.

Increasing Partition Size

If you have to increase the size of the sort table using the TABSIZE parameter, you also must increase the size of your virtual partition by the amount you specify for the TABSIZE value minus 4KB.

EREP can use two different sorting algorithms for its reports; the faster one requires additional storage equal to TABSIZE.

If you increase your partition size by the value of TABSIZE over the requirements outlined in Table 9 on page 78, you will significantly enhance EREP's performance.

EREP always tries to obtain the extra storage, and can use the faster sort routine if the storage is available.

Several cases require you to increase the partition size when running EREP. Table 9 on page 78 shows these cases and recommended amounts of partition increase for each.

Table 9. VSE Partition Size Increases for EREP

INFLUENCING FACTOR	AMOUNT OF PARTITION INCREASE
You are using the TABSIZE parameter	The specified value of TABSIZE minus 4KB
You are including EREP control statements	The specified or default value of TABSIZE

Table 9. VSE Partition Size Increases for EREP (continued)

INFLUENCING FACTOR	AMOUNT OF PARTITION INCREASE
You are using any of the following selection parameters: CPU CPUCUA CUA DEV DEVSER LIA/LIBADR MOD SYMCDE VOLID	4KB for any or all
You are requesting detail edit reports (PRINT=PT, PS or AL)	4KB for each processor
You are requesting a detail summary of 33XX records (not available in EREP 2.3 and later releases).	7KB
A processor requires frames for detail edit output (3L3X only)	150KB
You are requesting the system exception report series	6 times the specified or default value of TABSIZE

Because you might not know how many input records to expect and these partition size increases are generous, you may want to code SIZE=AUTO on the EXEC statement for the cases that require a lot of virtual storage.

In rare instances, this may create a storage problem. The following are ways to correct this problem:

- Increase the partition size to 1.7M or larger
- Code SIZE=xxxK instead of SIZE=AUTO, where xxx is 100 plus 1 for each 100 records over 400. For example, for 900 records you would code SIZE=105K.
- Do not code the SIZE parameter at all.

Information about the VSE System Control Program (SCP)

The VSE and VSE/Advanced Functions systems are currently packaged with EREP Version 3 Release 5.0.

Access Methods

EREP retrieves error records from SYSREC both:

- Sequentially, using the Sequential Access Method (SAM)
- Randomly, using the execute channel program (EXCP) system macro

It writes records to an output data set or buffer sequentially, through SAM. If you request specific devices for EREP’s output data, they must be supported by SAM.

Creation and Processing of Software Records

The VSE systems record events associated with system operation in the following record types:

RECORD TYPE	DESCRIPTION
System initialization (IPL) records	The IPL record includes a reason code and a subsystem code, supplied by the operator as part of the interactive IPL process. These two codes help identify the reason for the IPL and the device or program (if any) that failed.
Termination (EOD) records	The EOD record is written in response to a ROD (record on demand) command issued before shutting down the system for the day. The ROD command forces the dumping of statistical data counters and buffered logs to SYSREC, thus preserving the latest environmental data about your hardware systems. It also causes RDE to write the end-of-day record to SYSREC.

The VSE systems do not create type X'4X' software (SFT) records in response to abnormal termination.

EREP processes these records for:

- The system summary and trends reports, grouping them among PROGRAM ERRORs.
- The system error summary report of the system exception series, under IPL/RESTART/TERMINATION.

Initialization of the Error Recording Data Set (ERDS) in VSE

The VSE error-recording data set is called the system recorder file, or SYSREC(IJSYSRC), and is assigned the logical name IJSYSRC. SYSREC is created on the SYSRES volume during the first IPL following system generation. The SYSRES volume also contains:

- The hard copy file (IJSYSHC)
- The system history file (IFSYSHF)

In this book, SYSREC refers only to the system recorder file, or ERDS.

The operator creates and initializes SYSREC by issuing the IPL command *SET RF=CREATE* right after IPL and before the first //JOB statement.

Important: SYSREC is permanently assigned during IPL. The JCS ASSGN statement ignores any existing SYSREC assignments.

SYSREC consists of a header record after it is initialized. *EREP Reference* contain examples of the SYSREC header record.

You cannot reinitialize SYSREC without re-IPLing the system and reissuing the *SET RF=CREATE* command.

The EREP control parameter ZERO and the special EREP program IFCOFFLD merely zero out the data set; they do not remove the header.

Clearing the ERDS when Near Full on VSE

If the ERDS is filling up too quickly you can use the special EREP procedure named IFCOFFLD to off-load the records to another data set. IFCOFFLD preserves the data on the ERDS and gives you a summary report that can help you find the problem that caused the ERDS to fill up.

IFCOFFLD does essentially the same thing a normal ERDS off-load procedure does:

- It produces a system summary using the records on the ERDS.
- It copies the records to an output data set and clears the ERDS.

The only difference between the two procedures is that IFCOFFLD does not cause the dumping of statistical data to the ERDS prior to reading records for the system summary. This difference is significant because it:

- Prevents the loss of statistical data

- Saves time

The following table shows what IFCOFFLD does and gives a JCS example.

IFCOFFLD	EXAMPLE

1. Generates a system summary report without dumping the buffered and in-storage statistical data to SYSREC
2. Copies the records from SYSREC to the output data set (SYS009)
3. Zeros out SYSREC
	`// TLBL   HISTOT`
	`// ASSGN  SYS009,TAPE`
	`// EXEC   IFCOFFLD`
	`/*`
	`/&`

Note: It would be advantageous to assign your regular permanent EREP history data set to SYS009 for IFCOFFLD. You should also set up an IFCOFFLD procedure to be started from the console as needed. See "Automating the Running of EREP" on page 82.	

Statistical and Usage Data Written to SYSREC

Statistical and usage data are written to SYSREC when the operator issues the record on demand (ROD) command before running an EREP report. The data comes from counters that are associated with the devices and that are located either in buffers or in storage, depending on the device. The operating system dumps this data to the ERDS in the form of MDR and OBR records.

VSE History File (IJSYSHF)

Do not confuse the EREP history data set with the VSE history file named IJSYSHF.

The VSE history file contains information about the components of the system and the program fixes applied to those components. It is updated by MSHP (maintain system history program) and reflects the change level of your system.

The EREP history data set contains error records, either copied directly from SYSREC or accumulated after a report is run. It can be a cumulative data set, updated daily or weekly. It can be used as input to the EREP program, either by itself or in combination with SYSREC.

The EREP history data set is not created by the system; it is created when you specifically request it during an EREP run, see "Step 1: Creating a History Data Set" on page 69.

Running EREP in a Multisystem Environment

If your MVS, VM, or VSE installation has multiple processors running under the same or different operating systems, it may be possible to combine all of your error records into one history file. Use the steps shown in Table 2 on page 14 to combine the error records.

For details on setting up job streams with your other operating systems see:

HEADING
"Multisystem Installations" on page 13
"Running EREP in a Multisystem Environment" on page 46

HEADING
“VM SP Error Recording with Guest Systems” on page 67

Automating the Running of EREP

EREP should be run regularly and frequently. You can set up a procedure or series of EXECS that can be started by the operator or automatically by a timer.

The sample EREP runs in “Step 1: Creating a History Data Set” on page 69 through “Step 10: Updating the History Tape” on page 75 cover all the kinds of reports you would want to see from an EREP run. Add the system controls needed to make a run into a cataloged procedure and you will have your basic EREP setup.

You can set up two procedures that will create and update a monthly history tape:

1. One for the first run of the month
2. The other for subsequent weeks

The operator chooses and runs the appropriate procedure once a week.

Appendix A. Accessibility

Accessible publications for this product are offered through IBM Documentation (www.ibm.com/docs/en/zos).

If you experience difficulty with the accessibility of any z/OS information, send a detailed message to the Contact the z/OS team web page (www.ibm.com/systems/campaignmail/z/zos/contact_z) or use the following mailing address.

IBM Corporation
Attention: MHVRCFS Reader Comments
Department H6MA, Building 707
2455 South Road
Poughkeepsie, NY 12601-5400
United States

Accessibility features

Accessibility features help users who have physical disabilities such as restricted mobility or limited vision use software products successfully. The accessibility features in z/OS can help users do the following tasks:

- Run assistive technology such as screen readers and screen magnifier software.
- Operate specific or equivalent features by using the keyboard.
- Customize display attributes such as color, contrast, and font size.

Consult assistive technologies

Assistive technology products such as screen readers function with the user interfaces found in z/OS. Consult the product information for the specific assistive technology product that is used to access z/OS interfaces.

Keyboard navigation of the user interface

You can access z/OS user interfaces with TSO/E or ISPF. The following information describes how to use TSO/E and ISPF, including the use of keyboard shortcuts and function keys (PF keys). Each guide includes the default settings for the PF keys.

- *z/OS TSO/E Primer*
- *z/OS TSO/E User's Guide*
- *z/OS ISPF User's Guide Vol I*

Dotted decimal syntax diagrams

Syntax diagrams are provided in dotted decimal format for users who access IBM Documentation with a screen reader. In dotted decimal format, each syntax element is written on a separate line. If two or more syntax elements are always present together (or always absent together), they can appear on the same line because they are considered a single compound syntax element.

Each line starts with a dotted decimal number; for example, 3 or `3.1` or `3.1.1`. To hear these numbers correctly, make sure that the screen reader is set to read out punctuation. All the syntax elements that have the same dotted decimal number (for example, all the syntax elements that have the number `3.1`) are mutually exclusive alternatives. If you hear the lines `3.1 USERID` and `3.1 SYSTEMID`, your syntax can include either `USERID` or `SYSTEMID`, but not both.

The dotted decimal numbering level denotes the level of nesting. For example, if a syntax element with dotted decimal number 3 is followed by a series of syntax elements with dotted decimal number 3.1, all the syntax elements numbered 3.1 are subordinate to the syntax element numbered 3.

Certain words and symbols are used next to the dotted decimal numbers to add information about the syntax elements. Occasionally, these words and symbols might occur at the beginning of the element itself. For ease of identification, if the word or symbol is a part of the syntax element, it is preceded by the backslash (\) character. The `*` symbol is placed next to a dotted decimal number to indicate that the syntax element repeats. For example, syntax element `*FILE` with dotted decimal number 3 is given the format `3 \* FILE`. Format `3* FILE` indicates that syntax element `FILE` repeats. Format `3* \* FILE` indicates that syntax element `* FILE` repeats.

Characters such as commas, which are used to separate a string of syntax elements, are shown in the syntax just before the items they separate. These characters can appear on the same line as each item, or on a separate line with the same dotted decimal number as the relevant items. The line can also show another symbol to provide information about the syntax elements. For example, the lines `5.1*, 5.1 LASTRUN, and 5.1 DELETE` mean that if you use more than one of the `LASTRUN` and `DELETE` syntax elements, the elements must be separated by a comma. If no separator is given, assume that you use a blank to separate each syntax element.

If a syntax element is preceded by the `%` symbol, it indicates a reference that is defined elsewhere. The string that follows the `%` symbol is the name of a syntax fragment rather than a literal. For example, the line `2.1 %OP1` means that you must refer to separate syntax fragment OP1.

The following symbols are used next to the dotted decimal numbers.

? indicates an optional syntax element
: The question mark (?) symbol indicates an optional syntax element. A dotted decimal number followed by the question mark symbol (?) indicates that all the syntax elements with a corresponding dotted decimal number, and any subordinate syntax elements, are optional. If there is only one syntax element with a dotted decimal number, the `?` symbol is displayed on the same line as the syntax element, (for example `5? NOTIFY`). If there is more than one syntax element with a dotted decimal number, the `?` symbol is displayed on a line by itself, followed by the syntax elements that are optional. For example, if you hear the lines `5 ?, 5 NOTIFY`, and `5 UPDATE`, you know that the syntax elements `NOTIFY` and `UPDATE` are optional. That is, you can choose one or none of them. The `?` symbol is equivalent to a bypass line in a railroad diagram.

! indicates a default syntax element
: The exclamation mark (`!`) symbol indicates a default syntax element. A dotted decimal number followed by the `!` symbol and a syntax element indicate that the syntax element is the default option for all syntax elements that share the same dotted decimal number. Only one of the syntax elements that share the dotted decimal number can specify the ! symbol. For example, if you hear the lines `2? FILE, 2.1! (KEEP), and 2.1 (DELETE)`, you know that `(KEEP)` is the default option for the `FILE` keyword. In the example, if you include the `FILE` keyword, but do not specify an option, the default option KEEP is applied. A default option also applies to the next higher dotted decimal number. In this example, if the `FILE` keyword is omitted, the default `FILE(KEEP)` is used. However, if you hear the lines `2? FILE, 2.1, 2.1.1! (KEEP)`, and `2.1.1 (DELETE)`, the default option KEEP applies only to the next higher dotted decimal number, `2.1` (which does not have an associated keyword), and does not apply to `2? FILE`. Nothing is used if the keyword `FILE` is omitted.

*** indicates an optional syntax element that is repeatable**
: The asterisk or glyph (*) symbol indicates a syntax element that can be repeated zero or more times. A dotted decimal number followed by the * symbol indicates that this syntax element can be used zero or more times; that is, it is optional and can be repeated. For example, if you hear the line `5.1* data area`, you know that you can include one data area, more than one data area, or no data area. If you hear the lines `3* , 3 HOST, 3 STATE`, you know that you can include `HOST, STATE`, both together, or nothing.

Notes:

1. If a dotted decimal number has an asterisk (*) next to it and there is only one item with that dotted decimal number, you can repeat that same item more than once.
2. If a dotted decimal number has an asterisk next to it and several items have that dotted decimal number, you can use more than one item from the list, but you cannot use the items more than once each. In the previous example, you can write `HOST STATE`, but you cannot write `HOST HOST`.
3. The * symbol is equivalent to a loopback line in a railroad syntax diagram.

+ indicates a syntax element that must be included

The plus (+) symbol indicates a syntax element that must be included at least once. A dotted decimal number followed by the + symbol indicates that the syntax element must be included one or more times. That is, it must be included at least once and can be repeated. For example, if you hear the line `6.1+ data area`, you must include at least one data area. If you hear the lines `2+, 2 HOST, and 2 STATE`, you know that you must include `HOST, STATE,` or both. Similar to the * symbol, the + symbol can repeat a particular item if it is the only item with that dotted decimal number. The + symbol, like the * symbol, is equivalent to a loopback line in a railroad syntax diagram.

Notices

This information was developed for products and services that are offered in the USA or elsewhere.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
United States of America

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

This information could include missing, incorrect, or broken hyperlinks. Hyperlinks are maintained in only the HTML plug-in output for IBM Documentation. Use of hyperlinks in other output formats of this information is at your own risk.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Corporation
Site Counsel
2455 South Road

Terms and conditions for product documentation

Applicability

Personal use

Commercial use

reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user, or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session cookies that collect each user's name, email address, phone number, or other personally identifiable information for purposes of enhanced user usability and single sign-on configuration. These cookies can be disabled, but disabling them will also eliminate the functionality they enable.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, see IBM's Privacy Policy at ibm.com®/privacy and IBM's Online Privacy Statement at ibm.com/privacy/details in the section entitled "Cookies, Web Beacons and Other Technologies," and the "IBM Software Products and Software-as-a-Service Privacy Statement" at ibm.com/software/info/product-privacy.

Policy for unsupported hardware

Various z/OS elements, such as DFSMSdfp, JES2, JES3, and MVS™, contain code that supports specific hardware servers or devices. In some cases, this device-related element support remains in the product even after the hardware devices pass their announced End of Service date. z/OS may continue to service element code; however, it will not provide service related to unsupported hardware devices. Software problems related to these devices will not be accepted for service, and current service activity will cease if a problem is determined to be associated with out-of-support devices. In such cases, fixes will not be issued.

Minimum supported hardware

The minimum supported hardware for z/OS releases identified in z/OS announcements can subsequently change when service for particular servers or devices is withdrawn. Likewise, the levels of other software products supported on a particular release of z/OS are subject to the service support lifecycle of those products. Therefore, z/OS and its product publications (for example, panels, samples, messages, and product documentation) can include references to hardware and software that is no longer supported.

- For information about software support lifecycle, see: IBM Lifecycle Support for z/OS (www.ibm.com/software/support/systemsz/lifecycle)
- For information about currently-supported IBM hardware, contact your IBM representative.

Trademarks

Glossary

This glossary contains a list of terms used within the Environmental Record Editing and Printing Program library.

A

AFP
: Advanced Function Printing.

B

BPI
: Bits per inch.

BTAM
: Basic telecommunications access method.

BUFE
: Buffer error.

BYTES RD/SRCHD
: Megabytes read/searched.

C

CAT
: Channel availability table.

CCF
: Channel-check frame.

CCH
: Channel-check handler.

CCHCRH
: CCH channel reconfiguration hardware.

CCHINC
: CCH incomplete record.

CCU
: Channel control unit.

CCW
: Channel control word.

CDDA
: Command data.

CE
: IBM customer engineer (changed to IBM service representative).

central processor (CP)
: One of the internal processors that is part of a central processing complex.

channel
: The physical connector between a processor and an input/output device, usually via a control unit of some kind. In the case of the extended architecture (System 370/XA), the hardware channels are replaced by subchannels, which are capable of dynamic variation controlled by microcode in the processor complex.

  While this book refers to "subchannels" when discussing fields in 370XA report output, it uses "channel" in the general sense to mean the connection between controller and device.

channel-check frame (CCF)
: The record on the ERDS that EREP uses to format channel-check records from the 303X group of processors.

channel-check handler (CCH)
A S/370 hardware feature that, when a channel error occurs, records information about the error and issues a message to the operator. In VSE, machine check analysis and recording performs a similar function. The records created in both cases are called CCH records.

channel-report word (CRW)
In S/370XA, a part of the channel-subchannel recovery mechanism. It contains information about channel incidents reported through machine checks, specifying the error environment and the severity of the error. MVS/XA builds a CRW record that, in combination with the subchannel logout handler record, replaces the CCH record.

CHK
Check.

CHNL
Channel.

CHP
Channel path ID.

CHPID
Channel path ID.

CHR
Channel reporting (error).

CK
Check.

CKD
Count key data.

CLNACT
Cleaner action.

CMD
Command.

CMND
Command.

CMS
Conversational monitor system.

CNT
Count.

CNTRL
Control.

CNTRLR
Controller.

code
The programming-language instructions that make up a computer program. As a verb, "to code" is the same as "to write code".

COMP
Component.

CONS+UR
Console plus unit record.

controller
A single unit that provides an interface between one or more storage control units and a group of devices. Controllers usually reside within the same unit as the lowest drive addresses.

CORR
Correctable.

COR
Corrected.

CP
Central processor.

CPC
Central processing complex.

CPU serial number
A 6-digit hexadecimal number. The first digit identifies the central processor within the central processing complex. The second digit identifies the plant where the CPU was manufactured. The remaining digits identify the sequence number. For example, 120003 is CP 1 of the third CPC manufactured at plant two.

CRH
Channel reconfiguration hardware.

CRW
Channel-report word.

CSCH
Clear subchannel.

CSECTID
Control section (CSECT) identification.

CSID
Channel set ID.

CSW
Channel status word.

CT
Controller; count.

CTCA
Channel-to-channel adapter.

CTLID
Controller ID.

CTLR
Controller.

CU
Control unit.

CUA
Channel-control unit-device address.

CUD
Control unit detecting (error).

CUR
Control unit reporting (error).

D

DATAXFR
Data transfer.

DATA CKS CORR/RTRY
Data checks correctable/retry.

DCB
Data control block.

DCI
Dedicated connection interface.

DDR
Dynamic device reconfiguration.

DDROPR
DDR operator requested.

DDRSYS
DDR system requested.

DEV
Device number.

DEVNO
Device number.

DEVNUM
Device number.

DEVT
Device type.

DLBL
DASD label.

DNO
Device number.

DOS (VS)
Disk Operating System. An obsolete name, replaced by VSE, Virtual Storage Extended. In this book, "VSE" includes and implies all releases of this operating system, from DOS to VSE/ESA.

DPA
Dynamic pathing availability.

DRCT
Storage director.

DTE
Date.

dynamic device reconfiguration
A facility that allows a demountable volume to be moved, and repositioned if necessary, without abnormally terminating the job or repeating the IPL procedure. The MVS operating systems create DDR records to provide information about operator-assisted recovery involving the relocation of tape and movable DASD volumes.

E

EBCDIC
Extended binary code decimal interchange code.

ECC
Error correction code.

ECW
Extended control word.

EOD
End of day.

EQUCHK
Equipment check.

EQUIP
Equipment.

ERDS
Error-recording dataset.

EREP
Environmental record editing and printing program.

ERP
Error-recovery program/processing.

ERROPS
Error operations.

error-recovery dataset
Input to the IFCEREP1 program. In MVS systems, the ERDS is SYS1.LOGREC; in VSE systems, it is SYSREC; in VM, it is the error-recording area or cylinders.

error-recovery program/processing
System routines that detect and process errors, writing records to the ERDS.

ERSGAP
Erase gap.

ESIO
I/O devices on ESCON link.

ESW
Extended status word.

EXCP
Execute channel program.

EXTD
External damage.

FBA
Fixed block access.

FCF
Function control flag.

FCG
Floating channel group.

FLG
Flag.

FMT
Format.

FRF
Function request flag.

FRR
Function recovery routines.

FTA
File tape adapter.

H

hard machine check or error
A hardware error that disables the processor or other unit.

HDR SER
Header (tape)/serial number of drive that created tape.

HIRS
Hardware instruction retry (successful).

HSCH
Halt subchannel.

I

IC
Incident code.

ICHPT
Installation channel path table.

ID
Identification.

initial program load (IPL)
The process by which an operating system is initialized at the beginning of the day or session. At IPL, the system operator enters the installation-specific information the operating system must have

in order to manage the installation’s computing system and handle the installation’s application programs. This information includes system parameters, system dataset definitions, and other information needed so the operating system can begin operating.

installation
A data processing system location; for example, a computer center housing processors, I/O devices, other hardware devices, the software that controls the machines, and the people who control the computer center.

INV
Invalid.

INVK
Invoked.

IOB
Input output block.

IPL
Initial program load.

IRB
Interrupt response block.

J

JCL
Job control language.

JCS
Job control statement.

K

KB
Kilobyte.

L

LEN
Length.

LMAT
Load-module-address table.

LSQA
Local system queue area.

M

machine-check frame (MCF)
The record, on the ERDS, that EREP uses to format machine-check records from the 303X group of processors.

machine-check handler (MCH)
A S/370 hardware feature that analyzes errors and attempts recovery by retrying the failing instruction. If unsuccessful, it causes an interrupt that triggers the creation of an error record. In VSE systems, machine check analysis and recording performs similar functions. The records created in either case are called MCH records.

MB
Megabyte.

MCF
Machine-check frame.

MCH
Machine-check handler.

MCHTRM
MCH System terminated.

MCIC
Machine check interrupt code.

MCK
Machine check.

MDC
Maintenance device code.

MDR
Miscellaneous data record.

MDRDAS
DASD MDR record.

MI
Maintenance information.

MICR
Magnetic ink character recognition.

MIH
Missing-interrupt handler.

miscellaneous data record (MDR)
A record type that records error and usage information from buffered control units or communications controllers, and device failures on TP devices connected to 3705/3725 communications controllers. The record is created when there is an overflow of statistical counters; its purpose is to provide more information about the accompanying failure.

missing-interrupt handler (MIH)
An MVS and MVS/XA facility that keeps track of I/O interrupts, informing the operator and creating a record whenever an expected interrupt fails to occur in a preset time interval.

MIX
The XA version of the missing-interrupt handler.

MOD
Module.

MSHP
Maintain system history program.

MVS, MVS/ESA, MVS/XA
Multiple Virtual Storage, Multiple Virtual Storage/Enterprise Systems Architecture, and Multiple Virtual Storage/Extended Architecture, two versions of the System/370 operating system that are extensions of OS/VS2.

This manual uses "MVS" to refer to a family of operating systems that controls System/370 computing systems. "MVS" includes MVS/370, MVS/XA and MVS/ESA.

N

NCP
Network control program.

network management vector transport (NMVT)
An SNA management services request unit that flows over an active session between a device implementing an SNA physical unit and a device implementing an SNA control point.

NMVT
Network management vector transport.

O

OBR
Outboard recorder.

OBRDMT
OBR demount record.

OBRDPA
OBR dynamic pathing availability.

OBRDPS
OBR dynamic pathing validation analysis.

OBREOD
OBR End-of-day.

OBRPRM
OBR Permanent error record.

OBRPTH
OBR Permanent path error record.

OBRSHT
OBR Short record.

OBRTMP
OBR Temporary error.

OCR
Optical character recognition.

Operating System/Virtual Storage (OS/VS)
A family of operating systems that control IBM System/370 computing systems. OS/VS includes VS2, MVS/370, MVS/XA and MVS/ESA. This book refers to these operating systems by the general term "MVS".

OS/VS
Operating System/Virtual Storage.

OS/VS2
Virtual Storage 2 (MVS, Version 1). MVS/370; one of the MVS operating systems.

outboard recorder (OBR)
In VSE systems, the outboard recorder is a feature that records pertinent data about an unrecoverable I/O error. MVS systems create a similar record from information recorded when an I/O device is in *unit-check* status. The resulting record in both cases is called an OBR record.

OVERRN
Overrun.

OVERRUN CDDA
Overrun command data.

OVRN
Overrun.

P

PCCA
Physical configuration communications area.

PCT
Product control table.

PCUA
Primary channel-control unit-device address.

PDAR
Program damage assessment and repair.

PERM
Permanent.

PFU
Probable failing unit.

PR/SM
Program resource/system manager.

PRGM INT
Program-initiated.

PRI
Primary.

PRM
Permanent.

product control table (PCT)
The internal table that contains data EREP needs in order to identify and process records from a particular IBM device or product.

PROG-EC
Program-extended control mode.

PSF
Print Services Facility.

PSW
Program status word.

PUB
Physical unit block.

Q

QSAM
Queued sequential access method.

R

RCT
Record control table.

RCVRYXIT
Recovery exit module.

RD
Read error.

RDE
Reliability data extractor.

REC-TYP
Record type.

ROD
Record on demand.

RPA
Return point address.

RSM
Real storage manager.

RTM
Recovery termination manager.

RTN
Routine.

RTRY
Retry.

R/W
Read/write.

S

S/370 and S/370XA
Computing systems built around large IBM processors. XA stands for Extended Architecture, the architecture basis for the 3081 and later processors, characterized by 31-bit addresses. S/370

implies not only the processor but also the many other data processing devices that can be connected to it to make a 370 (or 370XA) data processing *system*.

SCD
System control data.

SCP
System control program.

SCSW
Subchannel status word.

SCU
Storage control unit.

SCUA
Secondary channel-control unit-device address.

SCUID
Storage control unit ID.

SD
Storage director.

SDR
Statistical data recorder.

SDWA
System diagnostic work area.

SE
Systems Engineer.

SEC
Secondary.

SEEKS CNTR/HH
Seek errors cylinder track/head

SFT
Software record. A record that is produced as part of the system error recovery process. It includes such software-specific information as the ERRORID and the system diagnostic work area control block and its extensions for the failing task or request block. MVS and AIX/ESA ® build software records.

SFTABN
SFT ABEND record.

SFTLST
SFT lost record.

SFTMCH
SFT machine error, recoverable.

SFTPI
SFT program interrupt.

SFTRST
SFT restart.

SIM
Service information messages.

SIO
Start I/O.

SKS
Seeks; data access errors.

SLH
Subchannel-logout handler.

SNA
Systems network architecture.

SNID
Sense path group ID (DPA).

Soft machine check or error
A hardware error that is not disabling.

SPID
Set path group ID (DPA).

SQA
System queue area.

SRC
System reference code.

SRCHD
Searched.

SRF
Service record file.

SSYS ID
Subsystem identifier.

STOR
Storage error.

storage control unit
A functional unit which resides between channels and controllers.

STSCH
Store subchannel.

SSCH
Start subchannel.

subchannel
The extended architecture version of "channel". See also *channel.*

subchannel-logout handler
A S/370XA feature that provides detailed model-independent information relating to a subchannel; the subchannel logout describes equipment errors detected by the channel subsystem. MVS/XA and MVS/ESA build an SLH record that, in combination with the CRW record, replaces the CCH record.

subsystem
In hardware terms, a group of devices that function together to perform I/O operations. An I/O subsystem can consist of a control unit (controller) and its associated drives—either disk or tape; or it can consist of *all* the DASD or tape storage—including drives and controllers—in an installation. In the case of newer DASD, the I/O subsystem also includes storage control units and storage directors, within the controller.

SVC
Supervisor call.

syntax
The relationships among the elements and characters in a parameter or language statement. For our purposes, the way you have to code something in order for the program to understand and accept it.

SYSGEN
System generation.

system control program
The minimum software package that will make your operating system work.

system generation
The process of selecting optional parts of an operating system and of creating a particular operating system tailored to the requirements of a data processing installation. Can also include I/OGEN, which is the time when the system programmer defines the installation's computing system configuration to the operating system.

Systems Engineer
The person responsible for helping you maintain the IBM software in your installation.

T

TCO
Triple capacity option.

TEMP
Temporary.

TERM
Terminal.

TLBL
Tape label.

TMP
Temporary.

TP
Teleprocessing.

TPF
Transaction processing facility.

transaction processing facility (TPF)
A high performance, real-time operating system designed for message-driven applications that require high availability and rapid response time at high message volumes.

TSCH
Test subchannel.

U

UCB
Unit control block.

V

virtual machine (VM)
A time-sharing system control program that manages the resources of an IBM System/370 computing system so that multiple remote terminal users have a functional simulation of the computing system (a virtual machine) at their disposal. This book uses "VM" to mean all versions of the Virtual Machine system control program, including VM/370, VM/System Pro duct, VM/SP/High Performance Option, VM/ESA, and VM/XA.

Virtual Storage Extended (VSE)
A family of disk operating systems that controls IBM System/360 and System/370 computing systems and includes VSE and VSE/Advanced Functions.

VM
Virtual machine.

VOLID
Volume serial number.

VS2
Virtual Storage 2 (MVS, Version 1). MVS/370; one of the OS/VS operating systems.

VSAT
Virtual storage address table.

VSE
Virtual Storage Extended.

VSE/AF
Virtual Storage Extended/Advanced Functions.

W

WRT
Write error.

Index

GC35-0151-50