



IBM Enterprise Developer Server Guide for z/OS

Version 5.0.0



IBM Enterprise Developer Server Guide for z/OS

Version 5.0.0

Note

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

Fourth Edition (October 2005)

This edition applies to Version 5.0.0 of IBM Enterprise Developer Server (product number 5655-I57) and to all subsequent releases and modifications until otherwise indicated in new editions.

You can order publications through your IBM representative or the IBM branch office serving your locality.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

© Copyright International Business Machines Corporation 1994, 2005. All rights reserved.

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

Contents

Trademarks.	vii
Terminology Used in This Document.	vii

About This Document	ix
Who Should Use This Document	ix

Part 1. Preparing to Install. 1

Chapter 1. Preparing for the Installation of Enterprise Developer Server.	3
--	----------

Chapter 2. Storage Requirements for Enterprise Developer Server 5

Virtual Storage Requirements	5
Enterprise Developer Server Load Module Storage	5
Application Load Module Storage	5
COBOL Dynamic Storage	6
Enterprise Developer Server Dynamic Storage	6
Storage Requirements for CICS	7
Disk Storage Requirements for Enterprise Developer Server	7
Work Database Space For Segmented Applications	7

Chapter 3. Installation Considerations. . . 9

z/OS Batch Considerations	9
DL/I Considerations.	9
DB2 Considerations	9
CICS Installation Considerations	10
DL/I Considerations	10
DB2 Considerations	10
Security Considerations	10
Monitoring and Tuning	10
CICS Utilities	11
Using the data Build Descriptor Option	11
Modifying CICS Resource Tables	11
IMS Installation Considerations.	11
IMS/ESA Exploitation	12
DB2 Considerations	12
Security Considerations	12
Monitoring and Tuning	12
IMS System Definition.	13
IMS Control Region	13
Work Database	13

Chapter 4. Customizing Enterprise Developer Server 15

General Customization Considerations for z/OS	15
Customizing Enterprise Developer Server	15
Security Considerations	15
Performance Considerations	15
Customizing Build Scripts	16
Modifying the Language Environment Run-time Option	16

Using Generated Programs with PL/I Programs	16
Installation and Language-Dependent Options for z/OS	16

Part 2. Administering on z/OS Systems 21

Chapter 5. General System Considerations for z/OS Systems . . . 23

Considerations that Affect Performance	23
Build Descriptor and Compiler Options	23
Modules in Memory	23
Files and Databases.	24
Defining and Loading VSAM Program Data Files.	24
Defining VSAM Data Sets	24
Loading Data in the Files.	26
Support for DBCS terminals	27
Extended Addressing Considerations for Enterprise Developer Server	27
Database Considerations	27
Preparing Programs	28
Checking Access Authorization.	28
Backing Up Data	28

Chapter 6. System Considerations for CICS. 29

Required File Descriptions	29
Segmented and Nonsegmented Processing	30
Using Transient Data Queues for Printing in z/OS CICS	30
z/OS CICS terminal printing	31
Special Parameter Group for the FZETPRT Program	31
PRTBUF Parameter.	32
PRTMPP Parameter.	33
PRITYP Parameter	33
FORMFD Parameter	33
CICS Entries for FZETPRT (DBCS only)	34
Using the New Copy Function	34
Specifying Recovery Options in the CICS Tables	34
Considerations that Affect Performance	35
Residency (Modules in Memory) Considerations	35
Using and Allocating Data Files in CICS.	36
Considerations for Using DB2 in CICS	40
Considerations for Using DL/I in CICS	40
Setting up the National Language	41

Chapter 7. System Considerations for z/OS Batch 43

Required File Descriptions	43
Using VSAM Program Data Files in z/OS Batch	44
Considerations for Using DB2 in z/OS Batch	44
Recovery and Database Integrity Considerations	44
Considerations for Using DL/I in z/OS Batch	44

Defining the Program Specification Block (PSB)	44
Recovery and Database Integrity Considerations	45
Performance Considerations for z/OS Batch	45
Runtime JCL	45

Chapter 8. System Considerations for IMS 47

Required File Descriptions	47
Defining the Program Specification Block (PSB)	48
Processing Modes	49
Printing Considerations for IMS	49
Recovery and Database Integrity Considerations	49
Considerations that Affect Performance	50
Residency Considerations and the IMS Preload Function	50
Database Performance	52
Limiting MFS Control Blocks	53
Monitoring and Tuning the IMS System	53
Considerations for Using DB2 in IMS	53
Recovery and Database Integrity Considerations	54
Checking Authorization	54
Considerations for Using DL/I in IMS	54
Recovery and Database Integrity Considerations	55
Maintaining the Work Database in IMS	55
Deleting Old Records from the Work Database	55
Expanding the Work Database	57
Supporting Multiple Work Databases	60
Considerations for Message Format Services in IMS	61

Part 3. Preparing and Running Generated Applications 63

Chapter 9. Output of Program Generation on z/OS Systems 65

Allocating Preparation Data Sets	65
List of Program Preparation Steps after Program Generation	67
Deploying generated code to USS	67
Outputs of Generation	67
Objects Generated for Programs	70
Objects Generated for Tables	71
Objects Generated for Form Groups	71

Chapter 10. z/OS Builds 73

z/OS Build Server	74
Starting a z/OS Build Server	76
Starting a USS Build Server	79
Stopping servers	79
Configuring a build server	79
Working with Build Scripts	79
Working with z/OS Build Scripts	79
Converting JCL to Pseudo-JCL	81

Chapter 11. Preparing and Running a Generated Program in CICS 85

Modifying CICS Resource Tables	85
Program Entries (PPT)	85
Transaction Entries (PCT)	86
Destination Control Table Entries (DCT)	86

File Control Table Entries (FCT)	87
Resource Control Table Entry (RCT)	87
Using Remote Programs, Transactions, or Files	87
Modifying CICS Startup JCL	87
Making New Modules Available in the CICS Environment	87
Making Programs Resident	88
Running Programs under CICS	88
Controlling Diagnostic Information in the CICS Environment	88
Printing Diagnostic Messages in the CICS Environment	88

Chapter 12. Preparing and Running Generated Programs in z/OS Batch . . . 89

Running Main Programs under z/OS Batch	89
Examples of Runtime JCL for z/OS Batch Programs	89
Running a Main Batch Program with No Database Access	90
Running a Main Batch Program with DB2 Access	90
Running Main Batch Program with DL/I Access	90
Running a Main Batch Program with DB2 and DL/I Access	91
Recovery and Restart for Batch Programs	92

Chapter 13. Creating or Modifying Run-time JCL on z/OS Systems 93

Tailoring JCL before Generation	93
Modifying Run-time JCL	94

Chapter 14. Preparing and Running Generated Programs in IMS/VS and IMS BMP 97

Modifying the IMS System Definition Parameters	97
Defining an Interactive Program	97
Defining Parameters for a Batch Program as an MPP	98
Defining Parameters for a Batch-Oriented BMP Program	99
Defining Parameters for a Transaction-Oriented BMP Program	99
Creating MFS Control Blocks	99
Making New Modules Available in the IMS Environment	100
Preloading Program, Print Services, and Table Modules	100
Running Programs under IMS	101
Starting a Main Program Directly	101
Starting a Main Transaction Program Using the /FORMAT Command	101
Running Transaction Programs as IMS MPPs	101
Running Batch Programs as MPPs	103
Running a Main Program under IMS BMP	103
Examples of Runtime JCL for IMS BMP Programs	104
Running a Main Batch Program as an IMS BMP Program	104
Running a Main Batch Program as an IMS BMP Program with DB2 Access	105
Recovery and Restart for IMS BMP Programs	106

Chapter 15. Moving Prepared Programs to Other Systems from z/OS Systems 107

Moving Prepared Programs To Another z/OS System	107
Maintaining Backup Copies of Production Libraries	107

Part 4. Utilities 109

Chapter 16. Using Enterprise Developer Server Utilities on z/OS Systems. 111

Using the CICS Utilities Menu.	111
New Copy	112
Diagnostic Message Printing Utility	114

Chapter 17. Diagnostic Control Options 117

Change or View Diagnostic Control Options for a Transaction	118
Change or View Default Diagnostic Control Options	119

Chapter 18. Using the Parameter Group Utility. 121

Chapter 19. IMS Diagnostic Message Print Utility 125

Part 5. Diagnosing Problems . . . 127

Chapter 20. Diagnosing Problems for Enterprise Developer Server on z/OS Systems 129

Detecting Errors	129
File and Database Errors—Category 1	129
File and Database Errors—Category 2	130
File and Database Errors—Category 3	130
Reporting Errors	131
Controlling Error Reporting in CICS.	131
Controlling Error Reporting in IMS Environments	131
Controlling Error Reporting in z/OS Batch	132
Error Reporting Summary	132
Transaction Error	132
Run Unit Error	132
Catastrophic error	133
Enterprise Developer Server Error	133
Using the Enterprise Developer Server Error Panel	133
Printing Diagnostic Information for IMS	134
ERRDEST Message Queue	134
IMS Log Format	135
Running the Diagnostic Print Utility.	136
Printing Diagnostic Information for CICS	136
CICS Diagnostic Message Layout.	136
Running the Diagnostic Print Utility.	137

Analyzing Errors Detected while Running a Program	138
---	-----

Chapter 21. Finding Information in Dumps 139

Enterprise Developer Server ABEND Dumps.	139
COBOL or Subsystem ABEND Dumps	139
Information in the Enterprise Developer Server Control Block	140
Information in an Application.	140
How to Find the Current Position in a Program at Time of Error	141

Chapter 22. Enterprise Developer Server Trace Facility 143

Enabling Enterprise Developer Program Source-Level Tracing with Build Descriptor Options	143
Activating a Trace	144
Activating a Trace Session for CICS	144
Activating a Trace Session for z/OS Batch.	147
Deactivating a Trace Session	149
Printing Trace Output	149
Printing the Trace Output in CICS	149
Printing the Trace Output in z/OS Batch	149
Reporting Problems for Enterprise Developer Server.	149

Chapter 23. Common Messages during Preparation for z/OS Systems . 151

Common Abend Codes during Preparation	151
DB2 Precompiler and Bind Messages	151
COBOL Compilation Messages	151

Chapter 24. Common System Return Codes for z/OS Systems 153

Common SQL Return Codes	153
Common DL/I Status Codes	155
Common VSAM Status Codes.	155
OPEN request type	156
CLOSE request type	156
GET/PUT/POINT/ERASE/CHECK/ENDREQ request types	156
COBOL Status Key Values	157

Chapter 25. Enterprise Developer Server Return Codes and Abend Codes for z/OS Systems 159

Return Codes	159
ABEND Codes	159
CICS Environments	159
IMS, IMS BMP, and z/OS Batch Environments	161
z/OS Batch	162

Chapter 26. Codes from Other Products for z/OS Systems 163

Common System Abend Codes for All Environments	163
--	-----

LE Run-time Messages	164
COBOL Run-time Messages	164
Common COBOL Abend Codes	165
Common IMS Runtime Messages.	165
Common IMS Runtime Abend Codes	166
Common CICS Run-time Messages	167
Common CICS Abend Codes	167
COBOL Abends under CICS	168

Part 6. Appendixes 169

Appendix. Enterprise Developer Server Run-time Messages	171
Notices	199
Index	203

Trademarks

The following terms are trademarks of the IBM[®] Corporation in the United States or other countries or both:

CICS[®]
CICS/ESA[®]
Current
DB2[®]
IBM
IMS[™]
Language Environment[®]
RACF[®]
VisualAge[®]
WebSphere[®] Studio
WebSphere
z/OS[®]

The following terms are trademarks of other companies:

Microsoft[®] Windows[®], and Windows NT[®] are trademarks or registered trademarks of Microsoft Corporation.

Windows, is a registered trademark of Sun Microsystems, Inc.

Terminology Used in This Document

Unless otherwise noted in this publication, the following references apply:

- EGL refers to Enterprise Generation Language
- CICS applies to Customer Information Control System
- “Region” or “CICS region” corresponds to CICS Transaction Server for region
- Workstation applies to a personal computer, not an AIX workstation
- The make process applies to the generic process, and not to specific make commands, such as make, nmake, pmake, polymake

About This Document

This manual provides information about customizing and administering Enterprise Developer Server in the following environments:

- z/OS UNIX System Services (USS)
- z/OS
- CICS

Note: Hereafter in this book, IBM Enterprise Developer Server for z/OS is referred to simply as “Enterprise Developer Server.”

Who Should Use This Document

This manual is intended for system administrators and system programmers responsible for installing, maintaining, and administering Enterprise Developer Server. It provides information to complete the following tasks:

- Manage system requirements
- Manage file utilization and conflicts

This manual is also intended for use by the programmers responsible for preparing and running EGL-generated programs. It provides information on the following items:

- Outputs of the generation process
- How to prepare generated programs for running
- Error codes
- How to use Enterprise Developer Server utilities
- How to diagnose and report problems

Attention IBM VisualAge Generator Users

Enterprise Developer Server provides the required components to support development and execution of programs generated by Enterprise Generation Language (EGL) or VisualAge Generator Developer.

To understand how VisualAge Generator Developer is used on z/OS, please refer to your VisualAge Generator documentation for information regarding VisualAge Generator Server for z/OS function and use.

Attention CICS Users

Please refer to the CICS documentation for the level of CICS installed on your system for detailed information regarding CICS functions and operations.

Attention: Accessing the EGL helps

To access the EGL helps, open the EGL client and select Help->Help Contents from the menu bar.

Part 1. Preparing to Install

Chapter 1. Preparing for the Installation of Enterprise Developer Server 3

Chapter 2. Storage Requirements for Enterprise Developer Server	5
Virtual Storage Requirements	5
Enterprise Developer Server Load Module Storage	5
Application Load Module Storage	5
COBOL Dynamic Storage	6
Enterprise Developer Server Dynamic Storage	6
Storage Requirements for CICS	7
Disk Storage Requirements for Enterprise Developer Server	7
Work Database Space For Segmented Applications	7

Chapter 3. Installation Considerations	9
z/OS Batch Considerations	9
DL/I Considerations	9
DB2 Considerations	9
CICS Installation Considerations	10
DL/I Considerations	10
DB2 Considerations	10
Security Considerations	10
Monitoring and Tuning	10
CICS Utilities	11
Using the data Build Descriptor Option	11
Modifying CICS Resource Tables	11
Using Spool Files	11
Temporary Storage	11
IMS Installation Considerations	11
IMS/ESA Exploitation	12
DB2 Considerations	12
Security Considerations	12
Monitoring and Tuning	12
IMS System Definition	13
IMS Control Region	13
Work Database	13
DL/I Work Database Considerations	13
DB2 Work Database Considerations	13

Chapter 4. Customizing Enterprise Developer Server	15
General Customization Considerations for z/OS	15
Customizing Enterprise Developer Server	15
Security Considerations	15
Performance Considerations	15
Customizing Build Scripts	16
Modifying the Language Environment Run-time Option	16
Using Generated Programs with PL/I Programs	16
Installation and Language-Dependent Options for z/OS	16

Chapter 1. Preparing for the Installation of Enterprise Developer Server

After selecting the production environments, do the following to prepare for the installation of the Enterprise Developer Server:

- Obtain a copy of the *Program Directory for Enterprise Developer Server for z/OS* (GI10-3241-00) (shipped with the product's installation materials).
- Determine the hardware, software, and storage requirements for the production environments selected.
- Install the hardware and software required by the Enterprise Developer Server.
- Collect information before customization.
- Understand specific environment considerations before defining applications.

Before continuing with the current document, do as follows:

1. Access the following Web site:

<http://www.ibm.com/software/awdtools/studioenterprisedev>

2. For details on product updates, click **Support** and search for information on **EGL**

3. To see what documents are available, click **Library** and review the list of publications; in particular, the entries under **Product Information**

The following document is of particular value because it includes prerequisite and planning information:

IBM WebSphere Developer for zSeries® Host Planning Guide

The document number for that guide is SC31-6599-02. Later editions of that Guide (if any) will have a higher number in the last digit.

Copies of documents are also available from the IBM Publications Center:

<http://www.elink.ibm.link.ibm.com/public/applications/publications/cgibin/pbi.cgi>

Chapter 2. Storage Requirements for Enterprise Developer Server

The following sections give approximate estimates of Enterprise Developer Server storage use by type of storage.

Virtual Storage Requirements

A program requires virtual storage for the following:

- Enterprise Developer Server load modules
- Application load modules
- COBOL dynamic area
- Enterprise Developer Server dynamic area

CICS applications also use specialized CICS storage facilities.

Enterprise Developer Server Load Module Storage

Most of the modules in the run-time function are not linked with the generated programs. Only one copy of these modules needs to be available for use by all programs generated with Enterprise Generation Language (EGL).

For z/OS, these modules can be in a library (DFHRPL), or placed in the link pack area (LPA). For CICS, you might want to make the modules resident. Refer to the Enterprise Developer Server program directory for a list of LPA eligible load modules.

Table 1. Enterprise Developer Server Reentrant Load Module Storage Estimates

Function	Size	RMODE
CICS base services	240 KB	ANY
CICS base services, 24-bit addressing mode	8 KB	24
Double-byte language ASCII/EBCDIC code conversion tables	Chinese - 50 KB	ANY

Application Load Module Storage

Application load module storage is the storage required for generated COBOL applications. The load modules are created by link-editing the generated COBOL applications produced by EGL's COBOL generation facility. The size of the load module can be determined from the linkage editor module map. The size varies depending on the functions utilized with the applications.

The application load module storage includes all generated application programs, data table programs, form group format modules, and print services programs used by a batch job step or transaction. The size of an application load module also includes the small Enterprise Developer Server programs that are statically linked with the programs. The load modules produced by link-editing the generated programs are reentrant. Each module can be linked with RMODE(ANY) so that the load module can reside in extended storage.

The size of the Enterprise Developer Server modules linked with each generated program, print services program, or data table program is shown in Table 2. These estimates should be added to the application load module size to determine the overall load module size.

Table 2. Enterprise Developer Server Statically Linked Module Storage Estimates

Environment	Application	Print service program	Data table program
CICS	2.5 KB	1 KB	1 KB
z/OS	1.3 KB	1 KB	1 KB

Note: Enterprise Developer Server modules are not statically linked with a form group format module.

COBOL Dynamic Storage

Application load modules acquire dynamic storage while they are running. The COBOL run-time library requires this storage for application data structures such as records, forms, and data tables. The storage includes both the internal and external data structures.

The COBOL data build descriptor option determines whether to acquire storage below the 16 MB line. The procedures shipped with the Enterprise Developer Server enable data (a build descriptor option) to control the value for the COBOL DATA compiler option. The default value of that build descriptor option is 31. Set data to 24 if an application calls another application or program that is linked as AMODE(24). Data table program and print services programs must also use data=24 if any application being used is linked AMODE(24).

When you generate z/OS batch or CICS programs with dynamic storage requirements greater than 64 KB, the value data=31 is required.

The amount of storage required for internal data structures is listed in the compile listing of the COBOL application when the MAP, OFFSET, or LIST compiler options are used.

Applications that run outside of CICS use COBOL external data structures to share information between applications in the same run unit. The storage required is approximately 1 KB.

Enterprise Developer Server Dynamic Storage

When applications are running, Enterprise Developer Server allocates storage as shown in Table 3 on page 7. The initial application of the run unit determines where the shared storage between Enterprise Developer Server and the generated COBOL application is allocated. If the initial application is generated with the build descriptor option data set to 24 or is link-edited with AMODE(24), this storage is allocated below the 16 MB line. Otherwise, the storage is allocated with 31-bit addresses as shown in the following table:

Table 3. Enterprise Developer Server Dynamic Storage Utilization

Function	Storage Required	24- or 31-bit Addressing mode
Persistent dynamic storage pool. The pool is extended as needed in 32 KB increments. Most transactions or jobs require only the initial allocation.	32 KB increment	31
CICS - service program dynamic storage stack	48 KB	31
z/OS batch	64 KB	24

Storage Requirements for CICS

Generated COBOL applications use the following CICS storage facilities:

Table 4. Enterprise Developer Server Use of CICS Storage Areas

Type of Storage	Function	Size
Transaction Work Area (TWA)	Enterprise Developer Server Control Block. Offset in TWA is specified in twaOffset build descriptor option.	1 KB
COMMAREA	Calls using COMMPTR	4 times the number of parameters
COMMAREA	Calls using COMMDATA	Total length of all parameters
COMMAREA	Remote calls	Total length of all parameters, plus 12

Disk Storage Requirements for Enterprise Developer Server

The auxiliary disk storage space required to install files for the Enterprise Developer Server is approximately 2 MB. Additional disk space for user programs can vary.

Work Database Space For Segmented Applications

The space required for saving application status across a terminal I/O operation in CICS is the sum of all segmented applications' data areas (maps and records) plus 6 KB per application. In CICS, disk space is used only if auxiliary temporary storage is specified as the work database during application generation.

Chapter 3. Installation Considerations

The following sections describe installation considerations for the Enterprise Developer Server.

z/OS Batch Considerations

If the installation has z/OS batch applications that gain access to relational databases, do as follows:

1. Install the correct version of DB2.
2. Create the tables in the relational database that the applications will access.
3. Follow the optional DB2-related steps for Enterprise Developer Server installation, as described in the Program Directory.
4. Define plans.

DL/I Considerations

If the installation has applications that gain access to DL/I databases, do the following:

1. Install the correct version of IMS. For more information on the correct version of IMS, see *Prerequisites for WebSphere Developer for zSeries (SC31-6352)*. This publication comes with the product or can be accessed from the IBM Publications Center at www.elink.ibm.link.ibm.com/public/applications/publications/cgibin/pbi.cgi.
2. Define databases and PSBs to IMS as described in the IMS utilities reference document.
3. Follow the optional DL/I-related steps for Enterprise Developer Server installation as described in the *Program Directory for Enterprise Developer Server for z/OS*.
4. The **data** build descriptor option defaults to **data="24"** for non-CICS environments.

DB2 Considerations

If the installation has applications that gain access to relational databases, do the following:

1. Install the correct version of DB2. For more information on the correct version of DB2, see *Prerequisites for WebSphere Developer for zSeries (SC31-6352)*. This publication comes with the product or can be accessed from the IBM Publications Center at www.elink.ibm.link.ibm.com/public/applications/publications/cgibin/pbi.cgi.
2. Create the tables in the relational database that the applications will access.
3. Follow the optional DB2-related steps for Enterprise Developer Server installation as described in the *Program Directory for Enterprise Developer Server for z/OS*.
4. Define application plans as described in the resource definition and installation and operation guides.

CICS Installation Considerations

This section discusses some general considerations when installing EGL-generated programs in the CICS environment.

DL/I Considerations

If the installation has applications that gain access to DL/I databases, you must do the following:

1. Install the correct version of IMS. For more information on the correct version of IMS, see *Prerequisites for WebSphere Developer for zSeries (SC31-6352)*. This publication comes with the product or can be accessed from the IBM Publications Center at www.elink.ibm.link.ibm.com/public/applications/publications/cgibin/pbi.cgi.
2. Define databases and PSBs to IMS as described in the IMS utilities reference document.
3. Follow the optional DL/I-related steps for Enterprise Developer Server installation as described in the *Program Directory for Enterprise Developer Server for z/OS*.
4. Add DL/I support to CICS and define databases and PSBs to CICS as described in the resource definition and installation and operation guides or in the IMS database control guide.

DB2 Considerations

If the installation has programs that gain access to relational databases, do the following:

1. Install the correct version of DB2.
2. Create the tables in the relational database to which the applications will gain access.
3. Follow the optional DB2-related steps for Enterprise Developer Server installation as described in the *Program Directory for Enterprise Developer Server for z/OS*.
4. Add DB2 support to CICS and define application plans to CICS as described in the DB2 system administration guides.

Security Considerations

CICS provides access control to resources (such as data files and programs) and transactions. This access can be controlled by the user or by the terminal.

CICS resources (such as data files, programs, destinations, journals, and temporary storage) can be assigned a security lock value. CICS users are assigned one or more key values. If a user is running an CICS transaction that is defined for resource security checking, the user's keys are checked every time a resource is requested. If the user does not have a key that matches the lock, access is denied by ending the transaction with an AEY7 ABEND code.

Monitoring and Tuning

Use CICS monitoring facilities to get information about CICS tasks.

Refer to the performance guide for your release of CICS for more information.

CICS Utilities

In the CICS environment, the Enterprise Developer Server includes a set of utilities to assist in managing the error diagnosis and control facilities of the Enterprise Developer Server environment. These utilities are EGL COBOL applications. See “Using the CICS Utilities Menu” on page 111 for more information about these utilities.

Using the data Build Descriptor Option

Set the build descriptor option data to 24 on generated COBOL programs to enable calls from the generated program to programs using 24-bit addresses, as long as the length of the COBOL dynamic storage (as defined in the COBOL working-storage section) required for the application is less than 64 KB. Programs whose dynamic storage requirements are greater than 64 KB must be compiled with the build descriptor option data set to 31. Otherwise, COBOL ends the program with a 1009 ABEND code.

Note: The build scripts and procedures shipped with the Enterprise Developer Server enables the data build descriptor option to control the value for the COBOL DATA compiler option. The build descriptor option data is set to 31 as the default for the CICS environment.

Modifying CICS Resource Tables

CICS uses tables to identify startup parameters, transactions, programs, files, databases, transient data destinations, and system locations for proper operation. The application developer must add or modify these tables to correctly identify all objects to be used in the new or changed application. The CICS tables are compiled as assembler programs and stored in a run-time library. Some tables can also be maintained through an online facility as described in the CICS resource definition online document. CICS requires that the online facility be used instead of PPT and PCT entries.

To generate model resource definition online (RDO) program and transaction definitions, specify the build descriptor option `cicsEntries=RDO`.

The CICS system initialization table needs to include `EXEC=YES`.

Add any transaction that invokes an application that uses DB2 to the resource control table (RCT) with the appropriate plan name. You can also use a resource definition.

Using Spool Files

To use the spool files, include the `SPOOL=YES` parameter in the System Initialization Table (SIT).

Temporary Storage

Temporary storage queues used by the Enterprise Developer Server must be defined as nonrecoverable. These queues start with `X'EE'`.

IMS Installation Considerations

This section discusses some general considerations for running EGL COBOL applications in the IMS environment.

IMS/ESA Exploitation

The procedures shipped with the Enterprise Developer Server cause the generated COBOL applications to be compiled with the **data="31"** build descriptor option and linked in AMODE(31) and RMODE(ANY). If the application calls another application that is linked with AMODE(24), then the **data="24"** build descriptor option is required.

You can link the generated COBOL application to run below the 24-bit line. However, if AMODE(24) is used to link the application, you must use **data="24"** build descriptor option for the following situations:

- For an application that calls an application or program that is linked as AMODE(24)
- For the first application in the run unit, if any generated application in the run unit is linked as AMODE(24) or if a non-EGL COBOL application that uses DL/I is linked as AMODE(24)
- For a table or mapping services program, if any application being used is linked as AMODE(24)

DB2 Considerations

If the installation has applications that gain access to relational databases, do the following:

1. Install the correct version of DB2. For more information on the correct version of DB2, see *Prerequisites for WebSphere Developer for zSeries (SC31-6352)*. This publication comes with the product or can be accessed from the IBM Publications Center at www.elink.ibm.link.ibm.com/public/applications/publications/cgibin/pbi.cgi.
2. Create the tables in the relational database that the applications will gain access to.
3. Follow the optional DB2-related steps for Enterprise Developer Server installation as described in the *Program Directory for Enterprise Developer Server for z/OS*.
4. Add DB2 support to IMS and define application plans to IMS as described in the DB2 system administration guide.

Security Considerations

Resource Access Control Facility (RACF) can be used to control users authority to each transaction.

Monitoring and Tuning

Potential performance problems can be tracked before they occur by checking processing statistics on a regular basis. The following are some of the statistics to monitor:

- Use the IMS DC monitor facilities to check transaction utilization. Consider preloading applications or groups of applications that are frequently used.
- Use the IMS database monitor facilities to check how effectively the databases are performing and using space.

Use the following tools to monitor IMS performance:

- The IMS Performance Analysis and Reporting System (IMSPARS, program number 5798-CQP).

This tool presents information on transaction transit times, IMS resource usage, and IMS resource availability, and detailed reports tracing individual transaction and database change activity. These reports are based on the contents of the IMS log data set.

- The Resource Measurement Facility (RMF™) II
This tool collects information about processor, channel, and I/O device utilization.
- The DB Tools product (program number 5668-856)
This tool provides information to help improve access efficiency and space utilization.

Refer to the IMS system administration document and the database administration guide for the release of IMS for additional information on monitoring the IMS online system and DL/I databases.

IMS System Definition

If you plan to use IMS, define all PSBs and transactions in the IMS system definition. In addition, define DL/I application databases.

IMS Control Region

You might need to review the values for the following:

- PSB work area pool (PSBW parameter)
- FORMAT pool (FBP parameter)
- MFS test area (MFS parameter)
- Communications input/output area (TPDP parameter)

In addition, if a DL/I work database is used, the work database would need to be added to either the control region JCL or to the dynamic allocation table.

Work Database

The work database is used to save the status of an EGL COBOL application during a CONVERSE process option, and to pass information during certain types of program-to-program message switches. The work database can be either a DL/I database or a DB2 table. The application developer specifies the **workDBType** build descriptor option when generating an application to determine which type of database is to be used. A DL/I or DB2 work database is used only for Enterprise Developer transaction applications that are generated for the IMS/VS target environment. In general, a DL/I work database performs better than a DB/2 work database.

Multiple DL/I or DB2 work databases can be installed. Use separate databases for each application system to improve performance or data availability.

DL/I Work Database Considerations

If you plan to use a DL/I implementation for the work database, you might need to tailor the database description (DBD) before running the job that creates and initializes the DL/I work database.

DB2 Work Database Considerations

If you plan to use a DB2 implementation for the work database, review the database definition before running the job that initializes the DB2 work database. A DB2 synonym needs to be created for each user and application gaining access to the DB2 work database.

The DB2 work database requires a 32KB page size. If a DB2 work database is used, you might need to increase the allocation of the 32KB buffers. To increase the allocation of buffers, modify and assemble the DB2 parameter module (default is DSNZPARM). Refer to the DB2 documents for the system for additional information.

If you select DB2, a DB2 application plan for each transaction is needed even if the EGL COBOL application itself does not require DB2.

If you select DB2 and if the Enterprise Developer Server needs maintenance applied to the module that handles the DB2 work database access, bind the application plans again for all transactions that use this database.

There are also considerations with the DB2 authorization used by the IMS program that is gaining access to the DB2 work database. For example, authorization needs to be granted to LTERM and a synonym needs to be created.

Chapter 4. Customizing Enterprise Developer Server

Before starting the customization process, determine the following:

- The target environments that application developers specify during generation
- Whether the applications use relational databases
- The IMS Work database and terminal types
- The national language support requirements

General Customization Considerations for z/OS

The following sections discuss some general considerations for running Enterprise Developer-generated applications in the supported z/OS environment.

Customizing Enterprise Developer Server

Customizing Enterprise Developer Server consists of performing some of the same procedures used to install the product on the system. These procedures are described in the *Program Directory for Enterprise Developer Server for z/OS*.

Security Considerations

The Enterprise Developer Server does not provide security services. Standard system or database manager security functions can be used with generated COBOL applications in the same way that they are used with customer-developed COBOL applications.

For example, if the applications use DB2, define DB2 application plans and give run authority to those users that are authorized to use the applications associated with the plan. The Resource Access Control Facility (RACF*) can also be used to grant users authority to read or update files.

Performance Considerations

Other chapters in this book provide detailed information on considerations that affect performance. See the following chapters for information on these performance-related topics and others:

Performance Topic	Where to Find Info
Build descriptor options	• Chapter 5, "General System Considerations for z/OS Systems," on page 23
Placing Enterprise Developer Server product and generated application modules in memory	• Chapter 5, "General System Considerations for z/OS Systems," on page 23
Residency and work-database considerations in CICS	• Chapter 6, "System Considerations for CICS," on page 29
Monitoring and tuning tools	• Chapter 6, "System Considerations for CICS," on page 29
Residency and database considerations in IMS	• Chapter 8, "System Considerations for IMS," on page 47

Customizing Build Scripts

The Enterprise Developer Server includes build scripts used for preparing generated applications for running. These build scripts can be customized to meet any data set naming conventions. Refer to the EGL helps for additional information.

Modifying the Language Environment Run-time Option

In the non-CICS environments, generated COBOL applications rely on COBOL working storage being initialized to binary zeros to determine whether Enterprise Developer Server is initialized. For LE, this is done by specifying `STORAGE=((00))` in the `CEEDOPT` CSECT.

In the non-CICS environments, generated COBOL programs that access sequential files (including print files) require the `ALL31` runtime option to be set to `OFF`. This is done by specifying `ALL31(OFF)` in the `CEEDOPT` CSECT.

The modified runtime options modules must be in a library allocated to the `STEPLIB` placed in the link pack area or in a library managed by the Virtual Lookaside Facility and Library Lookaside features of z/OS for each non-CICS z/OS environment. If those modules are in a separate library, the library must precede the library that contains the unmodified modules.

Alternatively, these options can be set for each program by creating a `CEEUOPT` load module with these options set as listed above and link-editing this module with each generated COBOL program. Refer to the Language Environment documentation for more information on creating and using a `CEEUOPT` module to set run-time options.

Using Generated Programs with PL/I Programs

If PL/I programs are used with generated COBOL applications in a non-CICS environment, you must generate the COBOL application to invoke the PL/I program using a static COBOL call. This requires the PL/I programs to be linked with the COBOL application in the same load module.

If PL/I programs are used with generated COBOL applications in the CICS environment, you must generate the COBOL application to call the PL/I program using the `CICS LINK` command. This is the default linkage for the CICS environment. The calling and called programs must not be linked together for the CICS environment.

Refer to the EGL helps for additional information.

Installation and Language-Dependent Options for z/OS

The following are the language-dependent options required for z/OS. To change the defaults use the steps outlined in the Program Directory for Enterprise Developer Server for z/OS (GI10-3241-00) to specify the settings and for instructions on customizing the Run-time Default Options and Language Dependent Options.

Table 5. Installation and Language-dependent Options for z/OS

Question	Default	Your Selection
Enterprise Developer Server Run-time Default Options		

Table 5. Installation and Language-dependent Options for z/OS (continued)

Question	Default	Your Selection
Default language code	ENU	_____
Enterprise Developer Server trace buffer size	64	_____
CICS ¹ temporary storage control interval size	16	_____
Enterprise Developer Server National Language Dependent Options (One code is needed for each national language you install. The default values vary for each language.)		
National language code (US English)	ENU	_____
Long Gregorian date format	MM/DD/YYYY	_____
Short Gregorian date format	MM/DD/YY	_____
Long Julian date format	YYYY-DDD	_____
Short Julian date format	YY-DDD	_____
Conversion table name	ELACNENU	_____
Positive response character string	YES	_____
Negative response character string	NO	_____
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Simplified Chinese)	CHS	_____
Long Gregorian date format	YYYY-MM-DD	_____
Short Gregorian date format	YY-MM-DD	_____
Long Julian date format	YYYY-DDD	_____
Short Julian date format	YY-DDD	_____
Conversion table name	ELACNCHS	_____
Positive response character string	YES	_____
Negative response character string	NO	_____
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Swiss German)	DES	_____
Long Gregorian date format	DD.MM.YYYY	_____
Short Gregorian date format	DD.MM.YY	_____
Long Julian date format	YYYY.DDD	_____
Short Julian date format	YY.DDD	_____
Conversion table name	ELACNDES	_____
Positive response character string	YES	_____
Negative response character string	NO	_____
Decimal point character	,	_____
Numeric separator character	.	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (German)	DEU	_____
Long Gregorian date format	DD.MM.YYYY	_____
Short Gregorian date format	DD.MM.YY	_____
Long Julian date format	DDD/YYYY	_____
Short Julian date format	DDD/YY	_____
Conversion table name	ELACNDEU	_____
Positive response character string	YES	_____

Table 5. Installation and Language-dependent Options for z/OS (continued)

Question	Default	Your Selection
Negative response character string	NO	_____
Decimal point character	,	_____
Numeric separator character	.	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Upper Case English)	ENP	_____
Long Gregorian date format	MM/DD/YYYY	_____
Short Gregorian date format	MM/DD/YY	_____
Long Julian date format	YYYY-DDD	_____
Short Julian date format	YY-DDD	_____
Conversion table name	ELACNENP	_____
Positive response character string	YES	_____
Negative response character string	NO	_____
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Spanish)	ESP	_____
Long Gregorian date format	DD/MM/YYYY	_____
Short Gregorian date format	DD/MM/YY	_____
Long Julian date format	DDD/YYY	_____
Short Julian date format	DDD/YY	_____
Conversion table name	ELACNESP	_____
Positive response character string	SI	_____
Negative response character string	NO	_____
Decimal point character	,	_____
Numeric separator character	.	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Japanese)	JPN	_____
Long Gregorian date format	YYY-MM-DD	_____
Short Gregorian date format	YY-MM-DD	_____
Long Julian date format	YYYY-DDD	_____
Short Julian date format	YY-DDD	_____
Conversion table name	ELACNJPN	_____
Positive response character string	YES	_____
Negative response character string	NO	_____
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Korean)	KOR	_____
Long Gregorian date format	MM/DD/YYYY	_____
Short Gregorian date format	MM/DD/YY	_____
Long Julian date format	DDD/YYYY	_____
Short Julian date format	DDD/YY	_____
Conversion table name	ELACNKOR	_____
Positive response character string	YES	_____
Negative response character string	NO	_____

Table 5. Installation and Language-dependent Options for z/OS (continued)

Question	Default	Your Selection
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Brazilian Portuguese)	PTB	_____
Long Gregorian date format	DD/MM/YYYY	_____
Short Gregorian date format	DD/MM/YY	_____
Long Julian date format	DDD/YYYY	_____
Short Julian date format	DDD/YY	_____
Conversion table name	ELACNPTB	_____
Positive response character string	SIM	_____
Negative response character string	NAO	_____
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (French)	FRA	_____
Long Gregorian date format	MM/DD/YYYY	_____
Short Gregorian date format	MM/DD/YY	_____
Long Julian date format	DDD/YYYY	_____
Short Julian date format	DDD/YY	_____
Conversion table name	ELACNFRA	_____
Positive response character string	OUI	_____
Negative response character string	NAN	_____
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Traditional Chinese)	CHT	_____
Long Gregorian date format	YYYY-MM-DD	_____
Short Gregorian date format	YY/MM/DD	_____
Long Julian date format	YYYY-DDD	_____
Short Julian date format	YY-DDD	_____
Conversion table name	ELACNCHT	_____
Positive response character string	YES	_____
Negative response character string	NO	_____
Decimal point character	.	_____
Numeric separator character	,	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____
National language code (Italian)	ITA	_____
Long Gregorian date format	MM/DD/YYYY	_____
Short Gregorian date format	MM/DD/YY	_____
Long Julian date format	DDD/YYYY	_____
Short Julian date format	DDD/YY	_____
Conversion table name	ELACNITA	_____
Positive response character string	SI	_____
Negative response character string	NO	_____

Table 5. Installation and Language-dependent Options for z/OS (continued)

Question	Default	Your Selection
Decimal point character	,	_____
Numeric separator character	.	_____
Currency symbol	\$	_____
SQL host variable indicator	:	_____
SQL host column indicator	!	_____

Part 2. Administering on z/OS Systems

Chapter 5. General System Considerations for z/OS Systems

Considerations that Affect Performance	23
Build Descriptor and Compiler Options	23
Modules in Memory	23
Files and Databases	24
Defining and Loading VSAM Program Data Files	24
Defining VSAM Data Sets	24
Defining an Alternate Index	25
Loading Data in the Files	26
Support for DBCS terminals	27
Extended Addressing Considerations for Enterprise Developer Server	27
Database Considerations	27
Preparing Programs	28
Checking Access Authorization	28
Backing Up Data	28

Chapter 6. System Considerations for CICS

Required File Descriptions	29
Segmented and Nonsegmented Processing	30
Using Transient Data Queues for Printing in z/OS CICS	30
z/OS CICS terminal printing	31
Special Parameter Group for the FZETPRT Program	31
PRTBUF Parameter	32
PRTMPP Parameter	33
PRTTYP Parameter	33
FORMFD Parameter	33
CICS Entries for FZETPRT (DBCS only)	34
Using the New Copy Function	34
Specifying Recovery Options in the CICS Tables	34
Considerations that Affect Performance	35
Residency (Modules in Memory) Considerations	35
Virtual Storage Considerations and Residency	35
Work Database Temporary Storage Queue Considerations	36
Using and Allocating Data Files in CICS	36
Defining and Loading VSAM Data Files	37
Using Remote Files	38
Defining Transient Data Queues	38
Considerations for Using DB2 in CICS	40
Associating DB2 Databases with CICS Transactions	40
Recovery and Database Integrity Considerations	40
Considerations for Using DL/I in CICS	40
Recovery and Database Integrity Considerations	40
Setting up the National Language	41

Chapter 7. System Considerations for z/OS Batch

Required File Descriptions	43
Using VSAM Program Data Files in z/OS Batch	44

Considerations for Using DB2 in z/OS Batch	44
Recovery and Database Integrity Considerations	44
Considerations for Using DL/I in z/OS Batch	44
Defining the Program Specification Block (PSB)	44
Recovery and Database Integrity Considerations	45
Performance Considerations for z/OS Batch	45
Runtime JCL	45

Chapter 8. System Considerations for IMS

Required File Descriptions	47
Defining the Program Specification Block (PSB)	48
Processing Modes	49
Printing Considerations for IMS	49
Recovery and Database Integrity Considerations	49
Considerations that Affect Performance	50
Residency Considerations and the IMS Preload Function	50
Preloading Enterprise Developer Server Modules	51
Loading Enterprise Developer Server Modules into the Link Pack Area	52
Preloading Generated Programs	52
Database Performance	52
Limiting MFS Control Blocks	53
Monitoring and Tuning the IMS System	53
Considerations for Using DB2 in IMS	53
Recovery and Database Integrity Considerations	54
Checking Authorization	54
Considerations for Using DL/I in IMS	54
Recovery and Database Integrity Considerations	55
Maintaining the Work Database in IMS	55
Deleting Old Records from the Work Database	55
DL/I Work Database	55
DB2 Work Database	56
Expanding the Work Database	57
DL/I Work Database	57
DB2 Work Database	58
Supporting Multiple Work Databases	60
DL/I Work Databases	60
DB2 Work Databases	60
Considerations for Message Format Services in IMS	61

Chapter 5. General System Considerations for z/OS Systems

This chapter describes the system requirements and considerations for administering the Enterprise Developer Server in all of the supported z/OS environments.

The following information is discussed:

- Considerations that affect performance
- Defining and loading VSAM program data files
- Support for DBCS terminals
- Extended addressing considerations for Enterprise Developer Server
- DB2 considerations
- Backing up data
- Customizing Enterprise Developer Server

Considerations that Affect Performance

Specifying certain build descriptor and compiler options and making reentrant programs resident in memory can affect the performance of EGL-generated programs.

Build Descriptor and Compiler Options

Setting build descriptor options as follows may improve run-time performance:

- checkNumericOverflow=NO
- debugTrace=NO
- fillWithNulls=NO
- initIORecords=NO
- initNonIOData=NO
- leftAlign=NO
- math=COBOL
- setFormItemFull=NO
- validateMixedItems=NO
- validateOnlyIfModified=YES

Specifying the following compiler options also may improve run-time performance:

- NOFDUMP
- NOSSRANGE
- NOTEST
- OPT

Note: Refer to the Enterprise COBOL for z/OS documentation for details on these compiler options.

Modules in Memory

Placing load modules in memory can improve performance by reducing the number of I/O operations (EXCPs). Load modules can be placed in memory by using the features of z/OS or the features of the environment in which you are running. Refer to the appropriate performance consideration sections for more detailed information about improving performance in a particular run-time environment.

General z/OS* methods to place load modules in memory are listed below:

- Place modules in the link pack area (LPA). Some of the modules that are shipped with the Enterprise Developer Server are reentrant and can be placed in the LPA. Refer to the *Program Directory for Enterprise Developer Server for z/OS* (GI10-3241-00) for information about modules that are reentrant and LPA eligible.

Generated programs, online print-service programs, form group format modules, and shared data tables are also reentrant and can be included in the LPA.

- Manage the Enterprise Developer Server data sets and the data sets containing the generated programs, online print services programs, form group format modules, and shared data tables. Use the Virtual Lookaside Facility (VLF) and the Library Lookaside (LLA) features of z/OS. Those features can place both the load modules and the partitioned data set (PDS) directories in memory.

Note: The STEPLIB and ISPLLIB libraries are searched first. For the z/OS methods, the load module (for LPA) or the data set (for VLF/LLA) cannot be contained in the STEPLIB or ISPLLIB concatenation list.

Files and Databases

Standard tuning techniques (such as buffering) can be used with files and databases used by generated COBOL programs.

Defining and Loading VSAM Program Data Files

This section describes how to define and load VSAM data sets for use as program data files in the CICS or z/OS batch environment. The section contains the following information:

- Defining VSAM data sets
- Defining an alternate index
- Loading data into the files

Defining VSAM Data Sets

VSAM data files can be serial (ESDS), relative (RRDS), or indexed (KSDS) files. Use the IDCAMS program to define a user VSAM data file. Figure 1 on page 25 shows example JCL that can be used to define the VSAM files.

```

//DEFVSAM JOB ...
//STEP1 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=A
//SYSIN DD *

/* THE FOLLOWING SAMPLE DEFINES A */
/* VSAM INDEXED FILE */

DEFINE CLUSTER (NAME(ELA1.USER.KSDS) -
              VOL(xxxxxx) -
              CYLINDERS(pp ss) -
              KEYS(1 d) -
              RECORDSIZE(aaa mmm) -
              INDEXED)

/* THE FOLLOWING SAMPLE DEFINES A VSAM */
/* NUMBERED RELATIVE RECORD FILE */

DEFINE CLUSTER (NAME(ELA1.USER.RRDS) -
              VOL(xxxxxx) -
              CYLINDERS(pp ss) -
              RECORDSIZE(aaa mmm) -
              NUMBERED)

/* THE FOLLOWING SAMPLE DEFINES A VSAM */
/* ESDS FILE */

DEFINE CLUSTER (NAME(ELA1.USER.ESDS) -
              VOL(xxxxxx) -
              CYLINDERS(pp ss) -
              RECORDSIZE(aaa mmm) -
              NONINDEXED)

```

where:

xxxxxx Specifies a valid volume serial number

pp Specifies the primary number of cylinders to be allocated

ss Specifies the secondary number of cylinders to be allocated

1 Specifies the length of the key

d Specifies the offset of the key

aaa Specifies the desired average record length

mmm Specifies the maximum record length

Figure 1. Defining VSAM Data Files

Defining an Alternate Index

An alternate index provides you with another way of gaining access to the records in a given KSDS file. Using a secondary key eliminates the need for you to keep several copies of the same information organized in different ways for different programs.

To gain access from an alternate index to the file through its prime index (base cluster), you must define a path to it. The path sets up an association between the alternate index and the base cluster, allowing the records in the data set to be available to you in different sequences. The alternate index is built after the base cluster is defined.

Figure 2 shows example IDCAMS definition commands for the base cluster and the alternate index cluster for an indexed file.

```
DEFINE CLUSTER (NAME(VSAM.KSDS.BASE.FILE) -  
  VOLUMES(xxxxxx) -  
  CYLINDERS(pp ss) -  
  KEYS(1 d) -  
  RECORDSIZE(aaa mmm) -  
  INDEXED)  
DEFINE ALTERNATEINDEX (NAME(VSAM.KSDS.ALT.INDEX) -  
  KEYS(1 d) -  
  CYLINDERS(pp ss) -  
  RELATE(VSAM.KSDS.BASE.FILE) -  
  VOLUMES(xxxxxx))  
DEFINE PATH(NAME(VSAM.KSDS.ALT.INDEX.PATH) -  
  PATHENTRY(VSAM.KSDS.ALT.INDEX))  
BLDINDEX INDATASET(VSAM.KSDS.BASE.FILE) -  
  OUTDATASET(VSAM.KSDS.ALT.INDEX)
```

where:

xxxxxx Specifies a valid volume serial number

pp Specifies the primary number of cylinders to be allocated

ss Specifies the secondary number of cylinders to be allocated

1 Specifies the key length

d Specifies the key displacement

aaa Specifies the desired average record length

mmm Specifies the maximum record length

Figure 2. Defining the Base Cluster and the Alternate Index Cluster

Loading Data in the Files

If you are using a VSAM indexed file (KSDS) and you want to open it for input only, initialize the file with at least one record. The file must have at least one record because a VSAM restriction prevents a file from being opened for input if the file is empty. While an empty file might be opened for output or both input and output, it must contain data to be opened for input.

There are several ways that you can put data into a file. One way is to create a Enterprise Developer program that uses an ADD process option to add records to an empty serial file. Once the program ends, you can use the IDCAMS REPRO command to copy the serial file into an indexed file.

Another way is to write a program that uses an ADD process option to add records to an empty indexed file. You must close the file in order to make the new records accessible.

Another way to initialize a VSAM KSDS file is to use a utility program shipped with the Enterprise Developer Server product. This utility can be used to initialize the key of a VSAM KSDS file. Figure 3 on page 27 shows how to initialize a VSAM KSDS file by setting the key to hexadecimal zeros.

```
//LOAD      JOB...
//JOB LIB   DD DSN=ELA.VxRxM0.SELALMD,DISP=SHR
//INITK     EXEC PGM=FZEZREBO,PARM='I,KSDS'
//SYS PRINT DD SYSOUT=A,DCB=(LRECL=121,BLKSIZE=121,RECFM=FB)

//KSDS      DD DSN=USER.KSDS,DISP=SHR
//SYS IN     DD DUMMY
```

Figure 3. Initializing a VSAM KSDS File

You can also use the IDCAMS utility to load initial data into an indexed file. Figure 4 shows an example of loading data into a VSAM KSDS file. The data contained in the USER.KSDS.INPUT file is loaded into the USER.KSDS data set.

```
//JOB KSDSLOAD
//LOAD EXEC PGM=IDCAMS
//SYS PRINT DD SYSOUT=*
//SYS IN    DD *
            REPRO INDDATASET('USER.KSDS.INPUT') OUTDDATASET('USER.KSDS')
/*
//
```

Figure 4. Loading a VSAM KSDS File

Support for DBCS terminals

Enterprise Developer Server provides support for the IBM Personal System/55 and the IBM 5550 family of terminals (emulating an IBM 3270 device). In addition to the basic hardware, this support uses character set F8 and four hardware attributes for double-byte character set (DBCS). The extended attributes are shift-out (SO) and shift-in (SI) enable, field outlining, color, and extended highlighting. Unpredictable results can occur if attributes are used that are not supported by the hardware.

Extended Addressing Considerations for Enterprise Developer Server

Some of the code provided with Enterprise Developer Server can run in extended addressing mode. This section describes considerations for using the extended addressing mode.

Most of the code shipped with Enterprise Developer Server runs in 31-bit addressing mode and resides above the 16MB line.

Most of the storage acquired by Enterprise Developer Server is above the 16MB line unless the first Enterprise Developer-generated program in the run unit is link-edited with AMODE(24) or generated with the build descriptor option data set to 24. The AMODE(24) program attribute specifies that the program runs in 24-bit addressing mode.

Database Considerations

This section discusses preparing programs and checking access authorization to database resources when using the following:

- DB2 on z/OS systems

Preparing Programs

Before running a program, the SQL* statements need to be analyzed and prepared.

If you use DB2, you also need to bind the DB2 program plan.

Note: Both of the above tasks are performed by the Enterprise Developer Server build process.

Checking Access Authorization

The database manager checks whether program users have the authority to access tables or run programs. The type of checking done varies depending on your system and the processing mode.

When accessing DB2 in generated COBOL programs, program users must be authorized to run the corresponding DB2 program plan and package.

DB2 requires an authorization identifier to ensure that program users have the DB2 authority to perform operations on the database and tables. The type of authorization checking done depends on whether the processing mode is static or dynamic. The authorization identifier of the program developer performing the BIND command is used for static SQL statements; the authorization identifier of the program user is used for dynamic SQL statements. Generated COBOL programs use dynamic SQL statements in either of two cases:

- The SQL statement is in an EGL prepare statement
- The EGL statement uses an SQL record, and a host variable identifies the SQL table name associated with that record

Any other SQL statements in the program are static statements. Refer to the DB2 administration manual for more information on the various ways the authorization identifier value is set.

Backing Up Data

You should regularly back up your data. This includes all files related to Enterprise Developer Server, private libraries, user-created data files, and user load libraries. System services are provided to back up and restore user libraries.

Chapter 6. System Considerations for CICS

This chapter provides additional system requirements and considerations for administering Enterprise Developer Server in the CICS environment.

The following information is discussed:

- Required file descriptions
- Segmented and nonsegmented processing
- Using transient data queues for printing
- Using the new copy function
- Recovery and database integrity considerations
- Considerations that affect performance
- Using and allocating data files
- Considerations for using DB2
- Considerations for using DL/I
- Considerations for using called remote programs

Required File Descriptions

Enterprise Developer Server requires the following files:

File Name

Description

ELAD This transient data queue is the default destination for Enterprise Developer Server error messages. Enterprise Developer Server produces error messages when it detects an error that prevents a program from continuing.

The ELAD transient data queue is defined when Enterprise Developer Server is installed. If you want to direct error messages for different transactions to different queues, define the other queues with the same characteristics as ELAD. Use the error diagnostic utility ELAC to direct error messages to the required queue. See the description of the utility in Chapter 17, "Diagnostic Control Options," on page 117 for more information.

ELACFIL

This is the error diagnostic control file. This file is created during customization.

ELAT This transient data queue is the destination for Enterprise Developer Server trace records.

If requested, Enterprise Developer Server can create trace records for selected runtime operations. The ELAT transient data queue is defined when Enterprise Developer Server is installed. For details, see Chapter 22, "Enterprise Developer Server Trace Facility," on page 143.

ELATOUT

This file is associated with the ELAT transient data queue at installation time. The output of the Enterprise Developer Server trace facility is sent to this data set. The attributes of this dataset are DSORG=PS, LRECL=133, BLKSIZE=1330, RECFM=FBA.

EZEPRINT

The file that you associate to the Enterprise Developer file name PRINTER

at resource association will be used when printing from a program that displays printer maps. This file can be defined with a file type of SPOOL or TRANSIENT. This file is normally associated with the transient data queue PRIN.

If you installed Enterprise Developer Server as described in the Enterprise Developer Server program directory, PRIN is defined as an indirect destination associated with the system printer. The maximum record length that a generated program writes to the system printer is 650 bytes for double-byte character set (DBCS) print maps and 133 bytes for single-byte character set (SBCS) print maps. The first byte is an American National Standards printer control character. The DBCS record length is longer than the physical printer line length because the print record can contain outlining and shift-out/shift-in (SO/SI) control characters that do not appear on the device.

If you are using Enterprise Developer Server to print to a file destination other than PRIN, the characteristics of that file should be the same as PRINTER.

EZEPRMG

This VSAM indexed file (KSDS) contains the parameter group records used for print control options for the Enterprise Developer Server terminal printer utility, FZETPRT. The FZETPRT program reads this file searching for the parameter group matching the transaction name that started FZETPRT.

Segmented and Nonsegmented Processing

Two different storage queues are used to support segmentation. The storage queue names have the following format:

xyyytttt

where:

- x** Specifies a byte with the hex value X'EE'
- yyy** Specifies WRK (program working storage) or MSG (current form saved across help or error display)
- tttt** Specifies the terminal ID associated with the transaction

The build descriptor option **workddb** specifies whether a main or auxiliary storage queue is used. The storage queues are deleted at the end of a run unit.

For details on segmentation, see the EGL help system.

Using Transient Data Queues for Printing in z/OS CICS

Printed output destined for a transient data queue is accumulated in temporary storage. The temporary storage queue name has the following format:

ttttnnnn

where:

- tttt** Is the transient data queue name
- nnnn** Is the EXEC Interface Block (EIB) task number

The default print destination for z/OS CICS is a transient data queue named EZEP. If you installed Enterprise Developer Server as described in the Enterprise Developer Server program directory, EZEP is an indirect destination associated

with the system printer. During program generation, this destination can be changed to any 4-character transient data queue name. The destination control table (DCT) entry for the queue determines the actual destination. The destination can be the system printer, a data set, or a terminal printer.

z/OS CICS terminal printing

The program called FZETPRT supports terminal printing. This program runs as a CICS transaction that starts automatically when records are written to the transient data queue. If Enterprise Developer Server was installed as described in the Enterprise Developer Server program directory, the transaction name is EZEZ for IBM 5550-type printers and EZEP for all other printers. To send printed output to the terminal, you must include a TYPE=INTRA for the transient data queue in the CICS destination control table (DCT). Specify EZEP or EZEZ for the transaction ID in the DCT entry. Unless you specify a terminal name in the DCT entry, the queue identifier must be the same as the terminal printer identifier. The trigger level in that entry must be set to 1 to ensure proper output. See “Printing Transient Data at a Terminal Device” on page 39 for a sample DCT entry.

When the FZETPRT program is initiated, it reads a line from the transient data queue, converts the American National Standards printer-control character to NL EOM format, and writes to the terminal printer specified in the DCT entry. The FZETPRT program buffers multiple print lines into a single CICS SEND command to improve performance.

When using terminal printing with Enterprise Developer Server, you should be aware of potential problems regarding form-feed orders and page alignment. When the FZETPRT program is triggered, a form-feed order is issued to the printer to ensure that it begins printing at the top of a page. If a second map is sent to the queue before it is emptied by the FZETPRT program, a form-feed order is not issued before the second map is printed. Page alignment can vary depending on the timing with which successive maps are sent to the queue.

Another potential problem can occur when printing successive maps. If one of the maps in the series is defined with lines equal to, or one line fewer than, the lines-per-page setting on the printer, a blank page occurs between the printed maps. To avoid this, define the map size as 2 lines fewer than the lines-per-page setting on the printer. Because the FZETPRT program inserts a new-line order to ensure that printing begins in column 1, the first line of the map to be printed is actually printed on the second line of the page. The second line must be allowed because a new-line order is added after the last line of the map, which advances the print head to the beginning of the next line. If this happens to be the first line of the following page, the next form-feed order causes the page to be skipped before printing resumes.

Another thing to consider is that although Enterprise Developer Server sometimes causes successive, stand-alone form-feed orders (“1”), the FZETPRT program suppresses all but one of these in converting them to NL EOM format.

If these form-feed considerations are too restrictive for your needs, consider using the FORMFD=NO parameter.

Special Parameter Group for the FZETPRT Program

You can provide terminal printing parameters to the FZETPRT program to vary the printed output by using a special parameter group file.

The FZETPRT program attempts to read a file named EZEPRMG for a parameter group that has the same name as the transaction used to start the FZETPRT program. For example, if the print transaction that starts the FZETPRT program is named EZEP, then FZETPRT tries to find the parameter group named EZEP. If the parameter group is not located in a file named EZEPRMG, or if EZEPRMG does not exist, then the FZETPRT program reads the DCAPRMG file to find the parameter group associated with this transaction.

When the transaction starts, the FZETPRT program reads the parameter group and varies the printer output according to the contents. If you need to use the terminal printing parameters, create a parameter group using the Enterprise Developer Server utility provided for this purpose.

For this parameter group, you can specify the following four parameters:

- PRTBUF=xxx
- PRTMPP=nnn
- PRRTYP=D
- FORMFD=NO

Note: Do not include blanks between keywords and their associated values.

PRTBUF Parameter

Use the PRTBUF parameter to set the size of the printer buffer. The number of SEND commands sent to the terminal printer depends on the size of the printer buffer. The following example shows how to specify the buffer size using the PRTBUF parameter:

```
PRTBUF=xxx
```

where:

xxx Is the size in bytes of the printer buffer

The FZETPRT program uses a default buffer size if any of the following conditions occur:

- The parameter is not specified in the parameter group.
- There is no parameter group associated with the transaction.
- The parameter keyword is misspelled.
- The value specified is not valid (values greater than 8K bytes, smaller than 480 bytes, or not numeric).
- The EZEPRMG or DCAPRMG file does not exist or is not available.

The default buffer size is 2KB (where KB equals 1024 bytes) for the standard character set printers and 480 bytes for LU type 3 printers.

For double-byte character set (DBCS) users the default buffer size and the maximum buffer size allowed is 1918 bytes. The default value is used if your specified value exceeds the maximum number of bytes.

When the buffer size is larger than the default, usage of the PRTBUF parameter is optional. However, using the PRTBUF parameter is recommended to reduce the number of SEND commands sent to the terminal. If the printer buffer size is smaller than the default, specify the real buffer size using this parameter. Not specifying the real buffer size can cause unpredictable results.

PRTMPP Parameter

Use the PRTMPP parameter to set the maximum number of print positions. The following example shows how to specify the number of print positions using the PRTMPP parameter:

```
PRTMPP=nnn
```

where:

nnn Is the physical length (maximum print position) of the printer line

The FZETPRT program assumes a default maximum print positions of 132 if any of the following occurs:

- The parameter is not specified in the parameter group.
- There is no parameter group associated with the transaction.
- The parameter keyword is misspelled.
- The value specified is not valid (not numeric).
- The EZEPRMG or DCAPRMG file does not exist or is not available.

Use caution when coding the value of this parameter. If the value entered is a valid numeric, the FZETPRT program uses the value without validating it. If the value is greater than the number of print positions available on the actual printer, possible malfunctioning can take place causing more line skips than necessary.

Note: For DBCS users, this parameter must be specified unless the printer is configured with MPP=132.

PRTTYP Parameter

Use the PRTTYP parameter if you use a DBCS printer. The following example shows how to specify the use of a DBCS printer using the PRTTYP parameter:

```
PRTTYP=D
```

Note: This parameter must be used to specify that you are a DBCS user and your output is being directed to an IBM 5550-family printer.

If you use multiple printers with different characteristics (namely different MPP, different buffer size, or DBCS versus non-DBCS printers), you need as many transaction IDs as there are printers, each one associated with the FZETPRT program. For examples of table entries for two printers, see the CICS transaction definitions provided with Enterprise Developer Server for the EZEP (non-DBCS printers) and EZEZ (DBCS printers) transactions.

FORMFD Parameter

Use the FORMFD parameter to control the form-feed orders that the FZETPRT program issues. The following example shows the format of the FORMFD parameter:

```
FORMFD=NO
```

The FZETPRT program defaults to inserting form-feed orders into the printer data stream if any of the following occurs:

- The parameter is not specified in the parameter group.
- There is no parameter group associated with the transaction.
- The parameter does not appear as FORMFD=NO.
- The EZEPRMG or DCAPRMG file does not exist or is not available.

If the parameter is specified correctly, the FZETPRT program does not insert form-feed orders for any reason.

CICS Entries for FZETPRT (DBCS only)

If you are using an SCS-type printer and you use DBCS, ensure that your system programmer has coded the destination control table (DCT) and the program control table (PCT) entries for a transaction that runs FZETPRT with the following option:

```
MSGPOPT=CCONTRL
```

The MSGPOPT option defines the optional facilities that a task can use. The CCONTRL parameter indicates that the program can control the outbound chaining of request units. Refer to the CICS manuals for more information.

Using the New Copy Function

The new copy function (either the Enterprise Developer Server new copy utility or the CICS NEWCOPY command) causes a transaction to use a new copy of a program, form group, or data table referenced in the transaction. The Enterprise Developer Server new copy utility is implemented as an EGL program in the CICS environment. Active transactions continue to use the current version of a program, form group, or data table until the transaction either completes or reaches the end of a segment. A new copy of the program, form group, or data table is then made available to the transaction by Enterprise Developer Server. Use the new copy function when programs, form groups, and data tables are modified and generated again. This enables you to install new versions of programs, form groups, and data tables onto your system without disrupting operation.

For programs and form groups you can use the CICS NEWCOPY command or the Enterprise Developer Server new copy utility to cause the new copy of the program to be used the next time a load request is issued for the program.

The Enterprise Developer Server new copy utility does a new copy for both the online print services program and the form group format module when you specify a part type of form group. If you use the CICS NEWCOPY command for a form group, you must issue the NEWCOPY for both the online print services program and the form group format module.

For data tables, you must use the Enterprise Developer Server new copy utility to cause a fresh copy of the data table to be used the next time a load request is issued for the data table. Do not use the CICS NEWCOPY command for data tables. The Enterprise Developer Server new copy utility sets a flag indicating that the new copy of the table is to be used the next time a program loads the table contents.

For more information on the Enterprise Developer Server new copy utility, see “New Copy” on page 112.

Specifying Recovery Options in the CICS Tables

EGL-generated programs can make use of all the z/OS CICS recovery and data integrity features. For a description of those features, refer to the recovery and restart information for your release of CICS.

The system initialization table (SIT) for CICS should specify DBP=XX, where XX is not equal to NO. If the DBP value is not equal to NO it prevents ASPE abends when generated programs issue CICS SYNCPOINT and CICS SYNCPOINT ROLLBACK commands.

If DTB=YES is specified on the program control table (PCT) entries for the transactions, the value specified for DBP is significant. CICS provides two dynamic backout programs, one for systems that require DL/I support and the other for systems that do not require DL/I support. These programs are provided by CICS if an entry is included in the processing program table (PPT) that specifies TYPE=GROUP and FN=BACKOUT.

Considerations that Affect Performance

This section describes factors that affect system performance and suggestions on how to improve performance. For information beyond what is stated in this section, refer to the performance guide for your release of CICS.

Residency (Modules in Memory) Considerations

The performance of a program is affected by the number of times that a running program requires access to a disk. Programs require access to disks for the following reasons:

- Locating and loading Enterprise Developer Server load modules or phases
- Retrieving and storing user data
- Locating and loading application programs, form group format modules, and online print services programs

The Enterprise Developer Server loads objects as they are needed. For example, the Enterprise Developer Server loads an application program, online print services program, form group format module, or data table when another program calls or references it. If you make an object resident, then the object remains in storage after it is loaded by the Enterprise Developer Server. You can use the RES parameter program definition to make any of these resident: an application program, online print services program, or form group format module. The data-table developer uses options in the part declaration to specify whether a data table is resident.

Virtual Storage Considerations and Residency

It is true that if an program is resident, less I/O is required for multiple loads. However, making these objects resident requires more virtual storage because the modules accumulate in storage as they are loaded and are not deleted after they are used.

When deciding what to make resident, consider the following:

- Storage constraints
- Frequency of program use
- Long running programs versus programs that are started more frequently

Because most systems have virtual storage constraints, it is not possible to make everything resident. You should establish priorities for deciding which objects you want to make resident. These residency priorities reflect a trade-off between program usage and storage constraints. Your priorities can dictate that some components of a program (such as the online print services program or form group format module) should be made resident, while other components (such as data tables) should not.

In CICS, when a program component is made resident, it remains in storage from the time it is loaded into storage until either CICS is shut down or the new copy function is used. To aid in deciding which programs should be made resident, you can use CICS shutdown statistics to determine how often a generated program or other component is loaded into the region or partition.

Generally, objects that are loaded more than once are prime candidates for residency. Examples of this a data table that is used by more than one program or a program that is called more than once.

Programs that are not frequently initiated or have long running time should not be made resident.

If you plan to run a program in pseudoconversational mode, you should consider making all components of the program resident. In pseudoconversational mode, the program and its components are deleted and are loaded again at each segment break if they are not made resident, and these actions degrade performance.

Work Database Temporary Storage Queue Considerations

When running in pseudoconversational mode (using a segmented CONVERSE process option), the data and the status associated with the program must be saved during user think time. You can control whether this information is saved into the CICS main temporary storage or auxiliary storage. Using main temporary storage can result in better performance because the data is written to memory within the CICS address space instead of writing the data to disk space.

Note: Use of main temporary storage can degrade system performance because the increased address space that is referenced can increase the paging activity. Also, CICS can experience a short-on-storage condition if the program data to be saved exceeds the available CICS storage. Therefore, if you take advantage of main temporary storage for programs requiring better performance, you should monitor your system to ensure that virtual storage problems do not occur.

The amount of data written or read on each request to CICS when saving program data and status, can also affect performance. The installation options module, ELARPIOP, specifies the largest size record Enterprise Developer Server writes to main or auxiliary temporary storage. The default size is 32KB (where KB equals 1024 bytes), which is the largest value allowed by CICS. Use a large value to ensure that the least number of write requests are required, and, if using auxiliary storage, to ensure that the least number of I/O operations are required. See the *Program Directory for Enterprise Developer Server for z/OS* for information on how to change the value in the installation options module.

Note: If you are using auxiliary storage queues, you should ensure the control interval size (CISIZE) of the VSAM data set used for auxiliary temporary storage matches the size specified in the installation options file. If the CISIZE for the data set is smaller, CICS splits the data written or read into smaller pieces and does multiple I/O operations for each Enterprise Developer Server request. Also ensure that you have an adequate number of buffers for the auxiliary temporary storage data set in order to reduce the number of physical I/O operations.

Using and Allocating Data Files in CICS

This section describes how to define data files for use in generated EGL-generated programs in the CICS.

Defining and Loading VSAM Data Files

Before CICS programs can use VSAM data files, you must define and load them. See “Defining and Loading VSAM Program Data Files” on page 24 for information on defining VSAM data sets, defining an alternate index, and loading a VSAM data set.

Adding the Job Control Statements: After the data set has been defined and loaded, add the data set name to the CICS startup JCL to allocate user files. You can also let CICS dynamically allocate the data set to the file using the information specified in the file control table (FCT). Figure 5 shows example allocation statements for an indexed, relative, and serial file, and an alternate index.

```
//KSDSFILE DD DSN=ELA1.USER.KSDS,DISP=SHR
//RRDSFILE DD DSN=ELA1.USER.RRDS,DISP=SHR
//ESDSFILE DD DSN=ELA1.USER.ESDS,DISP=SHR
//KSDSAIX DD DSN=VSAM.KSDS.ALT.INDEX.PATH,DISP=SHR
```

Figure 5. Allocating User Files

Adding the File Name to the CICS File Control Table: After the data set has been defined, loaded, and added to the CICS startup JCL, the FCT entry must be created for the file name for a CICS program to gain access to the data set. Creating an FCT entry can be accomplished using online (RDO) or macro definitions.

Figure 6 on page 38 shows sample macro definition entries that can be used to add a file name to the FCT. Enterprise Developer Server uses the name on the FILE operand. The FILE operand name must be the same as the ddname (z/OS) in the CICS startup JCL. All other operands must be the same as when you add an indexed, relative, or serial file to the FCT.

With CICS, make an entry to the FCT for every file used by a program. The CICS files can be defined as remote FCT entries.

For further information, refer to the appropriate CICS resource definition guide for your environment.

KSDS

```
DEFINE FILE(KSDSFILE) GROUP(xxxxxx)
    DSNAME(Indexed.DSName)
    DISPOSITION(SHARE) ADD(YES)
    BROWSE(YES) DELETE(YES) READ(YES)
    UPDATE(NO) RECORDFORMAT(F)
    STRINGS(8) LSRPOOLID(NONE)
    RECOVERY(NONE) NSRGROUP(GROUP1)
    INDEXBUFFERS(8) DATABUFFERS(9)
```

Alternate Index

```
DEFINE FILE(KSDSAIX) GROUP(xxxxxx)
    DSNAME(AlternateIndex.DSName)
    LSRPOOLID(NONE) DISPOSITION(SHARE)
    STRINGS(5) NSRGROUP(GROUP1)
    BROWSE(YES) DELETE(NO) READ(YES)
    ADD(NO) UPDATE(NO) RECORDFORMAT(F)
    RECOVERY(NONE) INDEXBUFFERS(5)
    DATABUFFERS(6)
```

RSDS

```
DEFINE FILE(RSDSFILE) GROUP(xxxxxx)
    DSNAME(Relative.DSName)
    DISPOSITION(SHARE) ADD(YES)
    BROWSE(YES) DELETE(YES) READ(YES)
    UPDATE(NO) RECORDFORMAT(F)
    STRINGS(8) LSRPOOLID(NONE)
    RECOVERY(NONE) NSRGROUP(GROUP1)
    INDEXBUFFERS(8) DATABUFFERS(9)
```

ESDS

```
DEFINE FILE(ESDSFILE) GROUP(xxxxxx)
    DSNAME(EntrySequenced.DSName)
    DISPOSITION(SHARE) ADD(YES)
    BROWSE(YES) DELETE(YES) READ(YES)
    UPDATE(NO) RECORDFORMAT(F)
    STRINGS(8) LSRPOOLID(NONE)
    RECOVERY(NONE) NSRGROUP(GROUP1)
    INDEXBUFFERS(8) DATABUFFERS(9)
```

***** E N D O F U S E R D A T A F I L E S *****

Figure 6. Adding a File Resource Definition

Using Remote Files

EGL-generated programs can gain access to files that do not reside on your CICS system.

Refer to the EGL helps for additional information.

Defining Transient Data Queues

Transient data queues are used in CICS for reading or writing data from tapes, disks, or other sequential files. If you associated a serial file with a transient data queue at generation, you must define a CICS destination control table (DCT) entry for the queue.

You can define the following types of transient data queues:

- Intrapartition (temporary data)
- Extrapartition (data that other non-CICS regions can use)

Intrapartition transient data files contain data that is not usable after it is read.

Defining Intrapartition Transient Data:

Passing Transient Data between CICS Transactions: This is an example of a DCT entry that can be used to pass data from one CICS transaction to another. The file destination specified at generation with the SYSNAME option should be xxxx.

```
DFHDCCT TYPE=INTRA,          C
      DESTID=xxxx,           C
      DESTFAC=FILE
```

Printing Transient Data at a Terminal Device: This is an example of a DCT entry that can be used for terminal printing in Enterprise Developer Server. At generation time, the resourceAssociation part specifies how you want to handle *printer*. The default is the first four characters, i.e., *prin*. (A DCT entry is supplied for *prin* that sends the printed output to the system printer.) The program supplied for printing, FZETPRT, reads records from the transient data queue and issues SEND commands to the terminal in order to print the records.

In this sample DCT, the PR01 terminal is to receive the printed output. PR01 is a z/OS CICS printer terminal name. You specify the *printer* destination at generation as PR01. Enterprise Developer Server writes the printed output to the transient data queue, PR01. The transaction EZEP starts and causes the program FZETPRT to run. The data is read from the transient data queue and sent to the terminal, PR01. The program control table (PCT) entries for EZEP and the processing program table (PPT) entries for FZETPRT are supplied. You must supply the destination control table and the terminal control table entries for the transient data and terminal.

```
DFHDCCT TYPE=INTRA,          C
      DESTID=PR01,           C
      DESTFAC=TERMINAL,      C
      TRANSID=EZEP,          C
      TRIGLEV=1
```

If the terminal printer is a DBCS printer, specify EZEZ as the TRANSID.

Defining Extrapartition Transient Data: Data to be read from tape or sent to a printer is contained in extrapartition transient data queues.

To provide these definitions as RDO entries, see the CICS resource definition guide.

The following two examples show how to use extrapartition transient data queues. These files can be used by non-CICS devices and by CICS.

Printing Transient Data: This is an example of a DCT entry specification that can be used to print output on a high-speed system printer. The file destination specified at generation with the systemName association element should be zzzz.

You need to add the appropriate JCL to the CICS runtime JCL to assign a printer to the file name. The following sample entry for the DCT is for printed output.

DFHDCT TYPE=EXTRA,	C
DESTID=ZZZZ,	C
DSCNAME=PRINTER	
DFHDCT TYPE=SDSCI,	C
DSCNAME=PRINTER,	C
RECFORM=VARBLKA,	C
RECSIZE=133,	C
BLKSIZE=1330,	C
TYPEFLE=OUTPUT	C

The JCL used in the extrapartition destination data queue sample requires the following JCL:

```
//PRINTER DD SYSOUT=*,DCB=(RECFM=VBA,LRECL=133,BLKSIZE=1330)
```

Considerations for Using DB2 in CICS

This section presents considerations for programs that access DB2 databases, and recovery and database integrity for DB2 programs running in the CICS environment.

Associating DB2 Databases with CICS Transactions

If the programs running under a transaction access DB2 databases, then you must define an entry in the CICS resource control table (RCT).

For information on the parameters you can specify when you define RCT entries, refer to the chapter on connecting the CICS attachment facility in the DB2 installation manual for your version of DB2.

To provide these definitions as RDO entries, see the CICS resource definition guide.

Recovery and Database Integrity Considerations

EGL-generated programs can use all the recovery and data integrity features that are provided by DB2 in the CICS environment.

Relational databases are recoverable resources. If your program makes changes to a relational database, the changes are not committed to the database until the end of a logical unit of work (LUW). If your program ends abnormally before the end of an LUW, all changes that were made since the beginning of the LUW are backed out.

Considerations for Using DL/I in CICS

This section discusses recovery and database integrity considerations for DL/I programs running in the CICS environment.

Refer to the EGL helps for additional information.

Recovery and Database Integrity Considerations

EGL-generated programs can make use of all the recovery and data integrity features that are provided by DL/I in the z/OS CICS environment.

DL/I databases are recoverable resources. If your program makes changes to a DL/I database, the changes are not committed to the database until the end of a logical unit of work (LUW). If your program ends abnormally before the end of an LUW, all changes that were made since the beginning of the LUW are backed out.

Setting up the National Language

On CICS, the national language code used for the first program in the run unit determines the language that is used for all messages for all programs in the run unit.

The next table lists the national languages that are supported for these purposes:

- To present Enterprise Developer Server messages on z/OS
- To present program-specific VisualAge Generator messages on any platform.

The code page for the language you specify must be loaded on your target platform.

Code	Languages
CHS	Simplified Chinese
CHT	Traditional Chinese
DES	Swiss German
DEU	German
ENP	Uppercase English
ENU	US English
ESP	Spanish
FRA	French
ITA	Italian
JPN	Japanese
KOR	Korean
PTB	Brazilian Portuguese

Chapter 7. System Considerations for z/OS Batch

This chapter presents system considerations for running EGL-generated programs in the z/OS batch environment.

The following information is discussed:

- Required file descriptions
- Recovery considerations
- Using VSAM program data files
- Considerations for using DB2
- Runtime JCL

Required File Descriptions

Enterprise Developer Server requires the following files:

File Name	Description
EZEPRINT	<p>This file is used when printing from a program that displays printer maps. EZEPRINT can be allocated to either a data set or to a SYSOUT class. The file must have a VBA (variable-blocked ANSI) record format.</p> <p>The maximum record length that a generated program can write to the print data set is 654 bytes for DBCS maps and 137 bytes for SBCS maps. The record length includes 4 bytes for the variable length record header, 1 byte for the American National Standards printer-control character, and the print line for the print map. The DBCS record length is longer than the printer line length because the print line can contain outlining control characters and shift-out (SO) and shift-in (SI) characters that are not displayed on the device. The logical record length defined for the data set must be greater than or equal to the length of the longest line written by the program, including the DBCS SO/SI characters.</p> <p>If you are using Enterprise Developer Server to print to a file destination other than EZEPRINT, the characteristics of that file should be the same as EZEPRINT.</p>
SYSPRINT, SYSOUT, SYSABOUT, SYSUDUMP	<p>These z/OS system files are used by EGL-generated programs. Do not specify DCB parameters for these files.</p>
ELAPRINT	<p>This system output file is used by generated programs. Specify ELAPRINT with RECFM=FBA and BLKSIZE=1330 DCB parameters.</p>
ELATRACE	<p>This file is the trace control file for the z/OS batch environment. The attributes for this data set are LRECL=80, RECFM=FB, and BLKSIZE=multiple of 80. The trace filters are specified in the ELATRACE data set.</p>
ELATOUT	<p>The output of the Enterprise Developer Server trace facility is sent to this data set in the z/OS batch environment. The attributes for this data set are DSORG=PS, LRECL=133, BLKSIZE=1330, and RECFM=FBA.</p>

Using VSAM Program Data Files in z/OS Batch

VSAM program data files must be defined before your z/OS batch program can use them. See “Defining and Loading VSAM Program Data Files” on page 24 for information on defining VSAM data sets, defining alternate indexes, and for information on loading VSAM data sets.

The DD statements for user files are generated for you and placed in the sample runtime JCL.

Considerations for Using DB2 in z/OS Batch

This section presents system considerations for database recovery and integrity for DB2 programs.

For information on running DB2 programs in z/OS batch, see Chapter 12, “Preparing and Running Generated Programs in z/OS Batch,” on page 89.

Recovery and Database Integrity Considerations

EGL-generated programs can use all the recovery and data integrity features provided by DB2.

Relational databases are recoverable resources. If your program makes changes to a relational database, the changes are not committed to the database until the end of a logical unit of work (LUW). If your program ends abnormally before the end of an LUW, all changes that were made since the beginning of the LUW are backed out. For information on when an LUW ends, see the EGL help topic called *Logical unit of work*.

Considerations for Using DL/I in z/OS Batch

This section presents the following information:

- Defining the program specification block (PSB)
- Recovery and database integrity considerations

Refer to the EGL helps for additional information.

Defining the Program Specification Block (PSB)

The following list shows considerations for defining a PSB that is used in the z/OS batch environment:

- DL/I PSBs used in the z/OS batch environment must have CMPAT=YES specified in the PSBGEN statement for the PSB. This enables you to use the CHKP and ROLB functions with the PSB.
- The PSBGEN statement must include the parameter LANG=COBOL or LANG=ASSEM.
- DL/I PSBs used in the z/OS batch environment must be defined with a minimum of two PCBs of any type in the PSB. This enables the generated COBOL program to test whether it is being started from the IMS region controller or from an OS XCTL macro in a non-Enterprise Developer program passing working storage and **dliLib.psbData** as parameters.
- z/OS batch programs can implement serial files as GSAM databases. These GSAM files are treated as a special type of database and require a PCB in the PSB. The GSAM PCBs must follow all database PCBs.

Recovery and Database Integrity Considerations

In z/OS batch DL/I programs, a commit point causes a DL/I basic CHKP (checkpoint) call. The contents of `dliLib.psbData` are used as the checkpoint identifier. After the CHKP call, `dliVar.statusCode` contains the status code returned with the CHKP call.

If the program runs under the z/OS terminal monitor program for SQL access, calling `sysLib.rollback()` results in an SQL ROLLBACK WORK.

If the program runs as a DL/I batch job, and DL/I or SQL requests have been issued, calling `sysLib.rollback()` results in a DL/I ROLB call. The IMS batch parameter BKO=Y must be specified when the batch job is started in order for the ROLB call to be honored. The BKO parameter is specified in the job step that calls the IMS control program DFSRRC00. If BKO=N is specified, DL/I returns status code AL for the ROLB call. Enterprise Developer Server treats the AL as a soft error, and no error message is issued.

Serial or print files associated with GSAM files and AUDIT service routine calls result in DL/I requests and cause the DL/I ROLB call to be issued.

Performance Considerations for z/OS Batch

See “Modules in Memory” on page 23 for information on performance considerations and the methods used to place modules in memory. These methods are particularly beneficial if the Enterprise Developer program is being called repeatedly by a non-Enterprise Developer program.

If you are running generated programs in z/OS batch, you no longer need to use the **forUpdate** option on the I/O statement prior to a **delete** or **replace** statement. Eliminating the **forUpdate** option allows for better performance, as it eliminates a COBOL read. However, make sure that you perform a **get** or **get next** before the **delete** or **replace** to ensure that the record is available.

Runtime JCL

See Chapter 12, “Preparing and Running Generated Programs in z/OS Batch,” on page 89 for examples of batch runtime JCL.

Chapter 8. System Considerations for IMS

This chapter provides additional administrative information that applies to the IMS environments.

The following information is discussed:

- Required file descriptions
- Defining the program specification block
- Processing modes
- Printing considerations for IMS
- Recovery and database integrity considerations
- Considerations that affect performance
- Considerations for using DB2
- Considerations for using DL/I
- Maintaining the work database

Required File Descriptions

Enterprise Developer Server requires the following files:

File Name	Description
ELASNAP	This is an optional file that contains the snap dump listing when a Enterprise Developer Server error occurs and the ELASNAP DD statement was included in the startup JCL. This file has a 125-byte logical record length, a 882-record block size, and a VBA (variable-blocked ANSI) record format. If this file is directed to the system logical unit SYSOUT define it with RECFM=VBA and BLKSIZE=4096.
ELAPRINT	This file is an optional output file for Enterprise Developer Server error messages. This file has a fixed block record format, a 133-byte logical record length, and a block size of 1330. If this file is directed to the system logical unit SYSOUT, define it with RECFM=FBA BLKSIZE=1330.
ELADIAG	<p>This is the default name for the optional message queue for Enterprise Developer Server error messages.</p> <p>This message queue is defined in the IMS system definition during Enterprise Developer Server installation. Refer to Chapter 19, "IMS Diagnostic Message Print Utility," on page 125 for information about printing the error messages contained in the ELADIAG message queue.</p>
ELATRACE	This is the trace control file for the IMS BMP environment. The attributes for this data set are LRECL=80, DSORG=PS, and BLKSIZE=multiple of 80. The trace filters are specified in the ELATRACE data set.
ELATOUT	The output of the Enterprise Developer Server trace facility is sent to this data set in the IMS BMP environment. The attributes for this data set are LRECL=133, BLKSIZE=1330, and RECFM=FBA.
ELAT	The output of the Enterprise Developer Server trace facility is sent to this output message queue in the IMS/VS environment. Use the ELAMQJUD job to retrieve the trace.

EZEPRINT This is the default message queue (IMS/VS) or output file (IMS BMP) for print output from generated programs. For IMS BMP programs, the print records are variable length. For single-byte languages, define EZEPRINT with LRECL=137, BLKSIZE=141, and RECFM=VBA. For double-byte languages, define EZEPRINT with LRECL=654, BLKSIZE=658, and RECFM=VBA. If the file is directed to the system logical unit SYSOUT, define it with RECFM=VBA, BLKSIZE=4096.

Defining the Program Specification Block (PSB)

You need to define both an IMS PSB and a Enterprise Developer PSB for your program. The Enterprise Developer PSB contains a subset of the information from the IMS PSB and is used to build default segment search arguments (SSAs) for the Enterprise Developer process options.

You need to generate an IMS PSB to correspond to the Enterprise Developer PSB. For IMS/VS, the IMS PSB must have the same name as the load module for the associated COBOL program. A program control block (ACB) generation is also required for the IMS/VS environment. For BMP and DL/I batch, the IMS PSB name does not have to match the program load module name.

When you define the PSBs for IMS programs, consider the following criteria:

- The I/O PCB (program control block) is automatically supplied and does not appear in the IMS or Enterprise Developer PSB source.
- Alternate PCBs are used to route output to terminals other than the originating terminal, or to other transactions. Alternate PCBs must appear before the database PCBs both in the IMS and the Enterprise Developer PSB source.
- When an Enterprise Developer program is generated for the IMS/VS or IMS BMP environment, a modifiable alternate PCB and a modifiable express alternate PCB are required, in that order, as the first two PCBs following the I/O PCB. Both of these PCBs must have the parameters ALTRESP=NO and SAMETRM=NO. To avoid having to edit your DL/I call modifications to adjust for the two required PCBs, include these PCBs whenever you plan to generate a program for the IMS/VS or IMS BMP target environments.
- The PSBGEN statement must include the parameters CMPAT=YES, and LANG=COBOL or LANG=ASSEM.
- IMS BMP programs can implement serial files as GSAM databases. These GSAM files are treated as a special type of database and require a PCB in the PSB. The GSAM PCBs must follow all database PCBs.

If a DL/I work database is used, the PCB for this database must be included in the IMS PSB. This PCB can be created using the macro ELAPCB and concatenating ELA.V1R2M0.ELASAMP as part of the SYSLIB in the PSBGEN procedure. Figure 7 on page 49 shows an example of the PCB expansion that occurs when ELAPCB is used.

WORKDBD defaults to ELAWORK. The WORKDBD parameter must be used if the DBD name is changed.

```

ELAPCB    [WORKDBD=customer-dbd-name]

--- expands into ---

PCB      TYPE=DB,DBDNAME=customer-dbd-name,PROCOPT=AP,KEYLEN=19
SENSEG   NAME=ELAWCNTL,PARENT=0
SENSEG   NAME=WORKLV01,PARENT=ELAWCNTL
SENSEG   NAME=WORKLV02,PARENT=WORKLV01
:
SENSEG   NAME=WORKLV14,PARENT=WORKLV13
SENSEG   NAME=MSGVLV01,PARENT=ELAWCNTL
SENSEG   NAME=MSGVLV02,PARENT=MSGVLV01
:
SENSEG   NAME=MSGVLV14,PARENT=MSGVLV13

```

Figure 7. Generating the DL/I Work Database PCB

Processing Modes

IMS requires segmented or single-segment mode. Refer to the EGL help for additional information on segmented and single-segment modes.

The **spaSize=xxxx** build descriptor option determines whether a program runs as IMS conversational (xxxx is greater than 0) or nonconversational (xxxx is 0). Refer to the EGL help for more information.

The work database is used for both conversational and nonconversational processing to save information during a **converse**. In conversational mode, the scratch-pad area (SPA) is used to set the transaction identifier and to save information during a program-to-program message switch. Refer to the EGL help for information on how the SPA is used for program-to-program message switching.

Printing Considerations for IMS

From Enterprise Developer Server, printing is initiated when a program processes a **print** statement for an Enterprise Developer-defined printForm. Refer to the online helps for information on defining forms for printers.

Printing is accomplished using MFS control blocks produced when the form group is generated. The default print destination in IMS is a message queue named EZEPRINT. The printer destination can be changed at generation time. You can also change the print destination at run time by changing the **converseVar.printerAssociation**. Refer to the Enterprise Developer help facility for additional information.

Recovery and Database Integrity Considerations

EGL-generated programs can make use of all the IMS recovery and data integrity features.

If your program makes changes to a recoverable resource, the changes are not committed until the end of a logical unit of work (LUW). If your program abnormally ends before the end of an LUW, all changes that were made since the beginning of the LUW are backed out.

An LUW for an IMS transaction ends whenever a commit point or a rollback occurs. A commit point occurs in IMS when one of the following occurs:

- The top-level program in a run unit ends successfully.

For IMS BMPs, a run unit consists of all EGL-generated programs and non-EGL-generated programs that transfer control among themselves using a **transfer** statement of the form *transfer to a transaction* or *transfer to a program*, or a **call** statement. For non-EGL-generated programs, this also includes transfers using an OS XCTL macro or CALL statement.

For IMS/VS, a run unit is equivalent to a single transaction and consists of all EGL-generated programs and non-EGL-generated programs that transfer control among themselves using a **transfer** statement of the form *transfer to a program* or a **call** statement. For non-EGL-generated programs, this also includes transfers using a CALL statement.

- A program uses a **converse** statement.

The best time for a commit point to occur is after terminal output and before the next terminal input. A commit point at terminal I/O synchronizes updates to the database and confirmation messages to the program user.

- A batch-oriented IMS BMP program (one that does not scan a serial file associated with the I/O PCB) calls **sysLib.commit()**.
- A batch-oriented IMS BMP program issues a **transfer** statement of the form *transfer to a transaction* and the **synchOnTrxTransfer** build descriptor option is set to "YES" for the transferred-from program.
- A program does a successful get unique to the I/O PCB.

A rollback occurs when one the following occurs:

- A program calls **sysLib.rollback()**.
- A program ends because of an error condition.

When a rollback occurs, all database changes that were made since the start of the LUW are backed out.

Considerations that Affect Performance

This section describes factors that affect system performance and suggestions on how to improve performance.

Residency Considerations and the IMS Preload Function

The performance of a program is affected by the number of times a disk is accessed while running the program. Programs require access to disks for the following reasons:

- Locating and loading Enterprise Developer Server load modules
- Retrieving and storing user data
- Locating and loading application, form group format modules, MFS print services programs, and table load modules

Enterprise Developer Server loads objects as they are needed. For example, Enterprise Developer Server loads an application, MFS print services program, form group format module, or table when another program calls or references it. The overhead of locating and loading modules can be reduced by using the IMS preload function. Preloading an object reduces the amount of I/O required for multiple loads. However, preloading generated programs requires more virtual storage for your system because preloaded modules remain in storage until the message region is shut down.

It is usually not possible for everything to be preloaded. Therefore, you should establish priorities for deciding which objects you should preload. These preloading priorities reflect a trade-off between your program usage and your storage constraints. Because of individual considerations such as storage constraints, environment, and types of programs, your priorities might dictate that some components (such as MFS print services programs) for a program be preloaded, while other components (such as tables) should not be preloaded. Make the decision on what modules to preload on an individual basis, according to how the program uses them.

When deciding what to preload, consider the following:

- Storage constraints
- Frequency of program use
- Long-running programs as compared to programs that are started more frequently

Generally, objects that are loaded more than once are prime candidates for preloading. Examples of this are a table that is used by more than one program and a program that is called more than one time. The following are some general rules for preloading:

- When deciding what to preload, consider the following objects:
 - Called programs
 - MFS print services programs
 - Form group format modules
 - Tables
 - Main programs
- Programs that are started or referenced frequently should be preloaded. In addition to programs that are loaded by IMS when a transaction is scheduled, this includes programs that are started by the Enterprise Developer **transfer** statements of the form *transfer to a program* or **call** statements.
- Programs that are not frequently initiated should not be preloaded.

See “Preloading Generated Programs” on page 52 for additional information.

Preloading Enterprise Developer Server Modules

For best performance, use the preload option for the following Enterprise Developer Server modules:

- ELARPRTR, the Enterprise Developer Server module that handles address mode switching
- ELARPRTM, the Enterprise Developer Server load module
- ELARPIOP, the installation options module
- ELARIccc (where ccc is the language code), the language-dependent options module
- ELACNccc (where ccc is the language code), the conversion table
- ELANCccc (where ccc is the language code), the module for Enterprise Developer Server constants and the fold table
- ELARSCNT, the configuration table
- ELA2SSQW, the module that supports the DB2 work database
- ELARSDCB, which is used for accessing Enterprise Developer Server sequential files
- ELA2SSQL, its alias ELA2SSQY, and ELA2SSQX
ELA2SSQL, its alias ELA2SSQY, and ELA2SSQX are used to gain access to the DB2 work database, and they support commit and rollback processing for DB2

program databases. Preload these modules only if you are using programs that were generated and bound using CSP/370RS V1R1.

The modules ELARSDCB and ELANCccc are loaded below the 16MB line. ELARSDCB is used only in reporting errors detected by Enterprise Developer Server. Both can be omitted from the preload list if storage space below the 16MB line is limited.

Note: You should also monitor the usage of the LE runtime modules. Because many are used by the generated COBOL programs, these modules might also be candidates for preloading.

Refer to the IMS documentation for your system for information on the preload option. An alternative to preloading is to place modules in the link pack area.

Loading Enterprise Developer Server Modules into the Link Pack Area

Placing modules in the link pack area causes all regions to share a single copy of the modules and saves storage space. Refer to the on-line helps for information about what modules can be put into the link pack area.

Only one version of CSP/370RS V2R1, CSP/370RS V1R1, VisualAge Generator Server V1R2, or IBM Enterprise Developer Server modules can be placed in the link pack area. If multiple releases are installed concurrently on the same system, override the link pack area by defining the correct library in the STEPLIB or JOBLIB DD statements for the region.

Preloading Generated Programs

You can reduce the overhead of searching the STEPLIB, JOBLIB, link pack area, and link list by preloading generated programs (application programs, online print services programs, map group format modules, and table modules) that are frequently used. However, in this case, virtual storage is still occupied by the modules when they are not in use.

To improve response time, you might also preload any module associated with any transaction that might require better performance, even though the module itself is not frequently used.

To preload generated programs, do the following:

1. Put the module in a LNKLIST library.
2. Include the module name in a preload member (DFSMPLEX, where xx is a two-character ID that you select) in the IMS procedure library.
3. Indicate in the JCL for the IMS message region that the preload member is to be included.

Database Performance

Database performance can be improved under IMS/ESA® by defining HIPERSPACE* buffer usage for IMS in the DFSVSMxx member. This is the same as defining many buffers for the files, but has the advantage that the HIPERSPACE buffers all come from 31-bit storage, not from within the IMS/ESA region. The tuning of database buffer pools is recommended. Refer to the IMS manuals for details on the tuning of database buffer pools.

If you have IMS/ESA installed and use a DL/I work database, make the work database nonrecoverable to reduce the amount of logging that occurs. Making the work database nonrecoverable might help improve performance.

Limiting MFS Control Blocks

Limiting the size and number of message format service (MFS) control blocks might help improve performance. MFS is used for mapping support in the IMS environment. MFS control blocks are generated using MFS utility control statements.

You can reduce the size and number of MFS control blocks that are generated by doing the following:

- In device selection of map definition, select only those Enterprise Developer devices that are used for the application system. For additional information about valid device types that can be specified, refer to the Devices list on the Map Definition Profile - Device Selection window within Enterprise Developer.
- Include in the **mfsDevice** build descriptor option only device types that your installation or application system uses. For additional information about specifying the **mfsDevice** build descriptor option, see the online helps.

Monitoring and Tuning the IMS System

You can track potential performance problems before they occur by checking processing statistics on a regular basis. The following are some of the statistics to monitor:

- Use the IMS DC monitor facilities to check transaction utilization. Consider preloading programs or groups of programs which are frequently used.
- Use the IMS database monitor facilities to check how effectively the databases are performing and using space.

You can also use the following tools to monitor IMS performance:

- The IMS Performance Analysis and Reporting System (IMSPARS, Program No. 5798-CQP). This tool presents information on transaction transit times, IMS resource usage, and IMS resource availability, as well as detailed reports tracing individual transaction and database change activity. These reports are based on the contents of the IMS log data set.
- The Resource Measurement Facility* (RMF*) II . This tool collects information about processor, channel, and I/O device utilization.
- The DB Tools product (Program No. 5685-093). This tool provides information to help improve database efficiency and space utilization.

Refer to the system administration manuals and the database administration guide for your release of IMS for detailed information about monitoring the IMS online system and DL/I databases.

Considerations for Using DB2 in IMS

This section discusses considerations for recovery, database integrity, and security issues for DB2 programs.

For information on designing and generating DB2 programs for the IMS environment, refer to the online helps.

For information on preparing DB2 programs for running in the IMS environment, see Chapter 14, “Preparing and Running Generated Programs in IMS/VS and IMS BMP,” on page 97.

Recovery and Database Integrity Considerations

EGL-generated programs can use all the recovery and data integrity features that are provided by DB2 in the IMS environment.

Relational databases are recoverable resources. If your program makes changes to a relational database, the changes are not committed to the database until the end of a logical unit of work (LUW). If your program ends abnormally before the end of an LUW, all changes that were made since the beginning of the LUW are backed out. See “Recovery and Database Integrity Considerations” on page 49 for additional information on when an LUW ends.

Checking Authorization

The database manager checks whether the program users have authority to gain access to tables or to run programs. The type of checking done varies depending on your system and the processing mode.

When using DB2 in generated COBOL programs, the program users must be authorized to run the corresponding DB2 program plan. For transaction-oriented regions, the authorization ID depends on the type of IMS security being used:

- If sign-on security is used, IMS provides the sign-on name as the authorization ID.
- If sign-on security is not used, IMS provides the name of the originating terminal as the authorization ID.

The plan used with a transaction has the same name as the program associated with the transaction.

For batch-oriented regions, the authorization ID is the contents of the ASXBUSER field, if valid, or the PSB name. The plan name is specified as one of the batch program parameters.

For more information on IMS security mechanisms, refer to the appropriate IMS manual.

Considerations for Using DL/I in IMS

This section discusses considerations for DL/I programs in the IMS environment.

See “Defining the Program Specification Block (PSB)” on page 48 for information on defining a PSB for DL/I programs.

For information on designing and generating DL/I programs for the IMS environment, refer to the EGL helps.

For information on preparing DL/I programs for running in the IMS environment, see Chapter 14, “Preparing and Running Generated Programs in IMS/VS and IMS BMP.”

Recovery and Database Integrity Considerations

EGL-generated programs can make use of all the recovery and data integrity features that are provided for DL/I databases in the IMS environment.

DL/I databases are recoverable resources. If your program makes changes to a DL/I database, the changes are not committed to the database until the end of a logical unit of work (LUW). If your program ends abnormally before the end of an LUW, all changes that were made since the beginning of the LUW are backed out. See “Recovery and Database Integrity Considerations” on page 49 for additional information on when an LUW ends.

Maintaining the Work Database in IMS

You should monitor and tune the DL/I and DB2 work databases just as you would any other DL/I database or DB2 table. You can use the normal database administration utilities to monitor these databases and to determine when they need to be reorganized to improve performance.

The activities involved in maintaining the work database are the following:

- Deleting old records from the work database
- Expanding the work database
- Supporting multiple DL/I or DB2 work databases

Deleting Old Records from the Work Database

The terminal ID is the key for the records in the work database. Each record contains a time stamp that indicates the last time the record was updated.

Deleting old records from the database reduces the amount of disk space required in the work database. You probably want to delete records in the following situations:

- Some users might run a generated program only infrequently, less than once a day, for example. In this case, you might want to delete old records on a daily or weekly basis.
- Sometimes terminal names are changed or users are moved to terminals with different names. In this case, new records are created for the new terminals, but the old records are not automatically deleted.

The utilities that delete records from the DL/I and DB2 work databases validate the date and time to ensure that your request does not result in deletion of records that are less than 24 hours old.

DL/I Work Database

Figure 8 on page 56 shows the JCL used to remove old records from a DL/I work database. The JCL is supplied as member ELAWKJCD in the ELA.V1R2M0.ELAJCL file. Specify the records you want to delete by entering the date (in Julian format) and time prior to which all records are to be deleted.

```

//*****
//** ELAWKJCD - JOBSTREAM TO CLEAN UP THE DLI WORK DATABASE
//**          FOR VISUALAGE GENERATOR SERVER.
//**
//** LICENSED MATERIALS - PROPERTY OF IBM
//** 5648-B02 (C) COPYRIGHT IBM CORP. 1994, 1998
//** SEE COPYRIGHT INSTRUCTIONS
//**
//** STATUS = VERSION 1, RELEASE 2, LEVEL 0
//**
//** TO TAILOR THIS JOBSTREAM:
//**      1. COPY A JOBCARD.
//**      2. REPLACE DATE AND TIME STAMP VALUE WITH DESIRED
//**          VALUE. ALL RECORDS WITH LESS THAN THAT DATE AND
//**          TIME WILL BE DELETED.
//**
//** RETURN CODES
//**      0 - SUCCESSFUL COMPLETION
//**      12 - FATAL ERROR. INVALID INPUT
//**      16 - FATAL ERROR. PROCESSING TERMINATED
//**
//*****
//*
//DLIWORK      EXEC IMSBATCH,MBR=ELAWKPC1,
//              PSB=ELAWKPB1,RGN=4096K
//G.STEPLIB    DD
//              DD
//              DD DSN=CEE.SCEERUN,DISP=SHR
//              DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//G.ELAPRINT   DD SYSOUT=*
//G.SYSOUT     DD SYSOUT=*
//G.SYSIN      DD *
YYDDHHMMSS

```

Figure 8. JCL to Remove Old Records from DL/I Work Database

DB2 Work Database

Figure 9 on page 57 shows the JCL used to remove old records from a DB2 work database. The JCL is supplied as member ELAWKJC2 in the ELA.V1R2M0.ELAJCL file. Specify the records you want to delete by entering the date (in Julian format) and time prior to which all records are to be deleted.

```

//*****
//** ELAWKJC2 - JOBSTREAM TO CLEAN UP THE DB2 WORK DATABASE
//**          FOR VISUALAGE GENERATOR SERVER.
//**
//** LICENSED MATERIALS - PROPERTY OF IBM
//** 5648-B02 (C) COPYRIGHT IBM CORP. 1994, 1998
//** SEE COPYRIGHT INSTRUCTIONS
//**
//* STATUS = VERSION 1, RELEASE 2, LEVEL 0
//**
//** TO TAILOR THIS JOBSTREAM:
//**      1. COPY A JOBCARD.
//**      2. REPLACE DATE AND TIME STAMP WITH THE DESIRED DATA.
//**          ALL ROWS WITH A DATE AND TIME LESS THAN THE
//**          SPECIFIED DATE/TIME WILL BE DELETED.
//**
//** RETURN CODES
//**      0 - SUCCESSFUL COMPLETION
//**      12 - FATAL ERROR. INVALID INPUT
//**      16 - FATAL ERROR. PROCESSING TERMINATED
//**
//*****
//*
//DB2WORK EXEC PGM=ELAWKPC2,REGION=4096K
//STEPLIB DD DSN=CEE.SCEERUN,DISP=SHR
//          DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSABOUT DD SYSOUT=*
//ELAPRINT DD SYSOUT=*
//ELASNAP DD SYSOUT=*
//EZESPUFI DD DSN=&&TMP1,DISP=(NEW,PASS),
//          UNIT=SYSDA,SPACE=(TRK,(1,0)),
//          DCB=(RECFM=FB,LRECL=80,BLKSIZE=80)
//SYSIN DD *
YYDDHHMMSS
//*
//DB2SPUF EXEC PGM=IKJEFT01,REGION=4096K,COND=(0,NE)
//STEPLIB DD DSN=DSN.RUNLIB.LOAD,DISP=SHR
//SYSOUT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSTSPRT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN DD DSN=&&TMP1,UNIT=SYSDA,DISP=(OLD,DELETE)

/*
//SYSTSIN DD *
        DSN SYSTEM(DSN)
        RUN PROGRAM(DSNTIAD) PLAN(DSNTIA??)
        END
/*

```

Figure 9. JCL to Remove Old Records from DB2 Work Database

Expanding the Work Database

At times, you need to expand the work database. For example, you need to expand the database when you expand the usage of an existing program system to a larger user set comprising a much larger number of terminals that gain access to EGL-generated programs.

DL/I Work Database

To expand the DL/I work database, perform the following steps:

1. Stop the DL/I database.

2. Unload the database using the old database description (DBD).
3. Change the DBD information and perform a DBD generation.
4. If you are having application control blocks (ACBs) prebuilt rather than built dynamically, build the ACBs again.
5. Delete the space allocated for the old database and allocate space for the new definition.
6. Load the database using the new DBD.
7. Make an image copy of the new database for back-up purposes as soon as it is loaded.

Refer to the database administrator's guide and the IMS utilities manual for additional information.

DB2 Work Database

You might need to expand the table spaces containing the DB2 work database because of degraded performance from too many secondary extents, or because the application users receive a DB2 message DSNP007I indicating that no more space is available.

Ideally, when the size of a DB2 table space is increased, the primary extent should be made large enough to accommodate all the data in the work database. In any case, try to minimize the number of secondary extents required to store rows in the database.

The method you use to expand the table space depends on the version of DB2 that is installed and whether the table space is user-managed.

The procedure supplied with Enterprise Developer Server that installs the work database also installs the table space as user-managed table space (no associated DB2 storage group).

Before attempting to change the size of the table space data set, you need to estimate the space requirements for the table space. One factor in your estimate is the amount of space currently used. If the space is currently DB2-managed (resulting from an earlier change in space allocation), you can get this information by first running the DB2 STOSPACE utility against the table space storage group, and then running the following query:

```
SELECT SPACE
  FROM SYSIBM.SYSTABLEPART
 WHERE TSNAME='tsname' and DBNAME='dbname';
```

The result (SPACE) gives the number of kilobytes of storage currently allocated to the table space.

If the space for the table space is user-managed, you can use the TSO LISTCAT command to obtain the space information. You need to know the data set name of the VSAM file used for table space. The data set name for the VSAM file has the following format:

```
catname.DSNDBC.dbname.tsname.I0001.Annn
```

where:

catname	Specifies the VSAM catalog name or alias
	This is the same name or alias as in the USING VCAT clause of the CREATE TABLESPACE statement.

dbname	Specifies the DB2 database name This is the same as the database name in the CREATE TABLESPACE statement.
tsname	Specifies the table space name This is the same as the table space name in the CREATE TABLESPACE statement.
nnn	Specifies the data set number For partitioned table spaces, the number is 001 for the first partition, 002 for the second, and so forth, up to the maximum of 64 partitions. For a simple or segmented table space, the number is 001 for the first data set. If the simple or segmented table space exceeds 2 gigabytes, the second data set is 002, and so forth.

To expand table space do the following:

1. Stop the DB2 database by using the command -STOP DB (dbname).
2. Make an image copy of the table space. You can use the image copy to restore the data set if the procedure is not successful.
3. Create a storage group for the table space. Do this only if the table space currently is user-managed and a storage group is not already available.
4. Change the table space definition as follows:

- If the table space data sets are user-managed, use a DB2 statement as follows:

```
ALTER TABLESPACE dbname.tsname
  USING STOGROUP stogrp
  PRIQTY pppp SECQTY ssss
```

where:

dbname.tsname	Specifies the name of the space
stogrp	Specifies the name of the storage group
pppp	Specifies new primary allocation size (in kilobytes) for the expanded table space
ssss	Specifies new secondary allocation size (in kilobytes) for the expanded table space

Note: This statement changes the table space from user-managed to DB2-managed.

- If the table space data sets are already DB2-managed, use a DB2 statement as follows:

```
ALTER TABLESPACE dbname.tsname
  PRIQTY pppp SECQTY ssss
```

where:

dbname.tsname	Specifies the name of the space
pppp	Specifies new primary allocation size (in kilobytes) for the expanded table space
ssss	Specifies new secondary allocation size (in kilobytes) for the expanded table space

5. Move the table space data. Simply changing the table space definition does not put the new size into effect. You need to move the table space to the newly allocated space. You can, for example, reorganize the table space using the DB2 REORG utility.
6. Start the DB2 database. Enter the command `-START DB (dbname)`.

Supporting Multiple Work Databases

You can use separate work databases for different application systems. For example, you might want to use separate databases for payroll and shipping to improve performance or to increase data availability. The work database is used to pass information during certain types of program-to-program message switches between applications. When this occurs, both the transferring application and the transferred-to application must use the same physical work database.

DL/I Work Databases

To create an additional DL/I work database called ELAWORK2, do the following:

1. Copy the ELAWORK DBD in the ELA.V1R2M0.ELASAMP file, and name it ELAWORK2.
2. Change the NAME parameter on the DBD statement to ELAWORK2. Also change the DD1 parameter on the DATASET statement to ELAWORK2. Make any other changes to the block size, number of blocks, and randomizing routine based on the application system requirements.
3. Make copies of the ELAWKLD and ELAWKPB1 program specification blocks (PSBs) in the ELA.V1R2M0.ELASAMP file and give them new member names. Change the NAME parameter on the program control block (PCB) statement from ELAWORK to ELAWORK2.
4. Modify job ELACJWKD in the ELA.V1R2M0.ELAJCL file to refer to the new database. This job does the DBD, PSB, and ACB generations needed for the work database, allocates the database, and then initializes it. You need to change the DD and data set names for the work database, and name the new DBD and PSB.
5. Add the new database to the JCL for your IMS control region, and to your IMS stage-1 system definition.
6. When you create IMS PSBs for applications that need to use this new database, use the ELAPCB macro to create the PCB definition for the work database. Enter the following command:

```
ELAPCB WORKDBD=ELAWORK2
```

DB2 Work Databases

To create an additional DB2 work database, do the following:

1. Create an ELAWORK table using the ELACJWK2 job in the ELA.V1R2M0.ELAJCL file. Perform the following steps before running the job:
 - a. Add an authorization ID to the CREATE TABLE command in ELAWORK2 in the ELA.V1R2M0.ELASAMP file, for example:


```
CREATE PAYROLL.ELAWORK
```
 - b. Change the table space name and index in ELAWORK2.
 - c. Change the DELETE and DEFINE CLUSTER statements to use the table space name and index you specified in ELAWORK2.
 - d. Comment out the WRKDROP step to avoid dropping the existing work database.

2. Each developer or system administrator using the payroll ELAWORK table needs to create a SYNONYM for the table. The following example shows how to use the CREATE SYNONYM command to create a synonym:

```
CREATE SYNONYM ELAWORK FOR PAYROLL.ELAWORK
```

The BIND command generated from the default BIND templates FDA2MBDB, FDA2MBDD, and FDA2MBDC bind DBRMs for Enterprise Developer Server modules to the application being generated. The CREATE SYNONYM command ensures that developers referencing the ELAWORK table use the payroll version of the table.

Considerations for Message Format Services in IMS

Enterprise Developer generates message format services (MFS) source statements used for conversing and displaying maps in IMS environments. The generated MFS source includes DEV statements, which identify the device types on which maps can be displayed and the characteristics of those devices. The device types and characteristics must be compatible with the device types and characteristics defined in the TERMINAL and TYPE macros in your IMS system definition.

The information on the generated DEV statements is controlled by the **mfsDevice**, **mfsExtendedAttr**, and **mfsIgnore** build descriptor options. Review the default specifications for the TERMINAL and TYPE and change them to be compatible with your system definition. You can delete the **mfsDevice** entries for any terminal types not supported at your installation and change the DEV statement parameters for any types for which the default parameters are not correct.

Refer to the IMS system definition reference manual for your release of IMS for additional information on the parameters for the TERMINAL and TYPE macros. Also refer to the stage-1 system definition macros for your IMS system to determine the parameters actually used for your installation. Refer to the MFS manuals for your release of IMS for additional information about the DEV statement.

You might also want to look at your non-Enterprise Developer MFS source to see the parameters that you currently specify on the DEV statement.

Once you have determined the correct values for the **mfsDevice**, **mfsExtendedAttr**, and **mfsIgnore** build descriptor options, code the default build descriptor options in all the default build descriptor files that you use when generating for the IMS or IMS BMP target environments.

Part 3. Preparing and Running Generated Applications

Chapter 9. Output of Program Generation on z/OS Systems

Allocating Preparation Data Sets	65
List of Program Preparation Steps after Program Generation	67
Deploying generated code to USS	67
Outputs of Generation	67
Objects Generated for Programs	70
Application COBOL Program	70
Sample Run-time JCL	70
Bind Commands	70
Objects Generated for Tables	71
Table COBOL Program	71
Objects Generated for Form Groups	71
Online Print Services Program	71
Batch Print Services Program	71
Form Group Format Module	71
MFS Source	71

Chapter 10. z/OS Builds

z/OS Build Server	74
Starting a z/OS Build Server	76
Starting a USS Build Server	79
Stopping servers	79
Configuring a build server	79
Working with Build Scripts	79
Working with z/OS Build Scripts	79
Writing a JCL build script	80
File Name Conversions for z/OS	81
Converting JCL to Pseudo-JCL	81

Chapter 11. Preparing and Running a Generated Program in CICS

Modifying CICS Resource Tables	85
Program Entries (PPT)	85
Transaction Entries (PCT)	86
Destination Control Table Entries (DCT)	86
File Control Table Entries (FCT)	87
Resource Control Table Entry (RCT)	87
Using Remote Programs, Transactions, or Files	87
Modifying CICS Startup JCL	87
Making New Modules Available in the CICS Environment	87
Making Programs Resident	88
Running Programs under CICS	88
Controlling Diagnostic Information in the CICS Environment	88
Printing Diagnostic Messages in the CICS Environment	88

Chapter 12. Preparing and Running Generated Programs in z/OS Batch

Running Main Programs under z/OS Batch	89
Examples of Runtime JCL for z/OS Batch Programs	89

Running a Main Batch Program with No Database Access	90
Running a Main Batch Program with DB2 Access	90
Running Main Batch Program with DL/I Access	90
Running a Main Batch Program with DB2 and DL/I Access	91
Recovery and Restart for Batch Programs	92

Chapter 13. Creating or Modifying Run-time JCL on z/OS Systems

Tailoring JCL before Generation	93
Modifying Run-time JCL	94

Chapter 14. Preparing and Running Generated Programs in IMS/VS and IMS BMP

Modifying the IMS System Definition Parameters	97
Defining an Interactive Program	97
Defining Parameters for a Batch Program as an MPP	98
Defining Parameters for a Batch-Oriented BMP Program	99
Defining Parameters for a Transaction-Oriented BMP Program	99
Creating MFS Control Blocks	99
Making New Modules Available in the IMS Environment	100
Preloading Program, Print Services, and Table Modules	100
Running Programs under IMS	101
Starting a Main Program Directly	101
Starting a Main Transaction Program Using the /FORMAT Command	101
Running Transaction Programs as IMS MPPs	101
IMS Commands	101
Keyboard Key Operation	102
DBCS Data on a Non-DBCS Terminal	102
Error Reporting	102
Responding to IMS Error Messages	102
Running Batch Programs as MPPs	103
Running a Main Program under IMS BMP	103
Examples of Runtime JCL for IMS BMP Programs	104
Running a Main Batch Program as an IMS BMP Program	104
Running a Main Batch Program as an IMS BMP Program with DB2 Access	105
Recovery and Restart for IMS BMP Programs	106

Chapter 15. Moving Prepared Programs to Other Systems from z/OS Systems

Moving Prepared Programs To Another z/OS System	107
Maintaining Backup Copies of Production Libraries	107

Chapter 9. Output of Program Generation on z/OS Systems

This chapter provides an overview of the files produced at generation time and of the steps needed to prepare code for use at run time.

Output files are transferred to z/OS, where preparation steps include running translators, precompilers, and compilers; doing link-edits; and defining control tables for the target run-time environment.

For additional information on the outputs of program generation, please refer to the EGL helps.

Allocating Preparation Data Sets

The EGL COBOL generator builds and runs a preparation command file to transfer generated objects to z/OS and to submit a preparation job (one of the generated objects) to the internal reader to complete the preparation process.

The transferred objects are stored in partitioned data sets. You allocate the required data sets using the ELACUSER CLIST shipped in the data set that has the low-level qualifier ELACLST. This CLIST was customized at product installation to set keyword default values to settings appropriate for your environment.

For you to use this CLIST, your customized data set **must** be placed before the installation data set that has the low-level qualifier SELACLST in the SYSPROC concatenation list. Make sure that every COBOL generation user has the required data sets allocated for **every** environment in which the product will be used.

The following keyword parameters within CLIST ELACUSER may either be customized within the CLIST or overridden when executing the CLIST:

Keyword	Possible Values
IMSVS	<ul style="list-style-type: none">• Y = allocate user data sets for this environment• N = do not allocate user data sets for this environment
ZOSBATCH	<ul style="list-style-type: none">• Y = allocate user data sets for this environment• N = do not allocate user data sets for this environment
ZOSCICS	<ul style="list-style-type: none">• Y = allocate user data sets for this environment• N = do not allocate user data sets for this environment
IMSBMP	<ul style="list-style-type: none">• Y = allocate user data sets for this environment• N = do not allocate user data sets for this environment
VOL	vvvvvv = serial number
UNIT	uuuuu = valid unit name
HLQ	hhhhhhh = high-level qualifier for user data sets

CLST

- FB = allocate a fixed blocked CLIST library
- VB = allocate a variable blocked CLIST library

DB2

- Y = DB2 databases will be used with this product
- N = DB2 databases will not be used with this product

LBLK

lllll = load library data set block size

An example of the command syntax to execute the CLIST is as follows:

```
ex 'myServer.v5r0m0.elac1st(elacuser) zoscics(y) zosbatch(y)
vol(at1235) unit(sysda) hlq(tsouid) db2(y)'
```

Table 6 describes the data sets that are allocated. The DD name in the table is the DD name in the build scripts that are used by the build server. The meaning of lower-case strings in the data set name is as follows:

chqlq The high-level qualifier in use at your installation.

env The generation environment. One of these:

- ZOSBATCH (for z/OS batch)
- ZOSCICS (for z/OS CICS)
- IMSVS (for IMS/VS)
- IMSBMP (for IMS BMP)

Table 6. Program Preparation User Data Set Information

DD Name	Data Set Name	Description	DCB Information	Target Environment
DBRMLIB	cghlq.env.DBRMLIB	Database request module library for DB2 programs	DSORG=PO, RECFM=FB, BLKSIZE=6160, LRECL=80	All z/OS, if DB2 used
EZEBIND	cghlq.env.EZEBIND	Bind commands	DSORG=PO, RECFM=FB, BLKSIZE=6160, LRECL=80	All z/OS, if DB2 used
EZEJCLX	cghlq.env.EZEJCLX	Batch program runtime job stream	DSORG=PO, RECFM=FB, BLKSIZE=6160, LRECL=80	ZOSBATCH
EZEOBJ	cghlq.env.OBJECT	Object library	DSORG=PO, RECFM=U, BLKSIZE=6144, LRECL=0	All z/OS
EZESRC	cghlq.env.EZESRC	COBOL source library	DSORG=PO, RECFM=FB, BLKSIZE=6160, LRECL=80	All z/OS
SYSLMOD	cghlq.env.LOAD	Load library	DSORG=PO, RECFM=U, BLKSIZE=6144, LRECL=0	All z/OS

Table 6. Program Preparation User Data Set Information (continued)

DD Name	Data Set Name	Description	DCB Information	Target Environment
EZEPCT	cghlq.env.EZEPCT	CICS PCT entries or RDO TRANSACTION entries	DSORG=PO, RECFM=FB, BLKSIZE=6160, LRECL=80	ZOSCICS
EZEPPT	cghlq.env.EZEPPT	CICS PPT entries or RDO PROGRAM entries	DSORG=PO, RECFM=FB, BLKSIZE=6160, LRECL=80	ZOSCICS

List of Program Preparation Steps after Program Generation

Enterprise Developer Server supports program preparation and installation in the z/OS environments using build scripts with Enterprise Developer Server. You must perform the steps listed in Table 7 before you can run your program in an z/OS target environment.

Table 7. Preparation Steps for z/OS Environments

Preparation Step	Environment
Transfer from workstation to the host	All
DB2 precompile	DB2 use only
CICS translation	CICS only
COBOL compile	All
Link	All
Bind	DB2 use only. an Enterprise Developer Server program.

Additionally, for CICS, you must define your program and transactions to the environment. For CICS, you do this using the program properties table (PPT) and program control table (PCT) entries or the Resource Definition Online (RDO) PROGRAM and TRANSACTION entries. For information on CICS entries, see Chapter 11, "Preparing and Running a Generated Program in CICS."

Deploying generated code to USS

The setup for deploying generated Java™ code in USS is the same as for Windows. Please see the EGL help topic *Setting up the J2EE run-time environment for EGL-generated code*.

To access the help system in the development Workbench, select Help->Help Contents from the menu bar. When the help interface appears, select Enterprise Developer Documentation, then Developing, then Enterprise Generation Language.

Outputs of Generation

After you have generated a program, you have a number of objects that need to be transferred to the host system. To place these outputs in a PDS, you must first customize the EGL build scripts. On the z/OS system, these members need to be prepared before the program can be run.

Outputs of preparation are placed in a PDS automatically. You control the high level qualifier of the PDS using the build descriptor option projectID.

By default the build scripts do not save the generated program source code.

The build scripts save the link edit file, the bind control file and the CICS entries. The CICS entries are saved because they are needed to install the program in CICS. The link edit and bind control files are saved because they are needed to reproduce a load module from the prepared object module if you want to save the prepared outputs in an SCM repository.

You can not save the load module in the repository and restore it to an environment, but you can save the object deck and relink it in a production environment. If an enterprise wants to save the source code, it is necessary to modify the fdac1, fdabcl, fdapcl, fdatcl, fdaptcl, and fdamfs build scripts. There are instructions in the build scripts on how to do this by uncommenting certain lines and commenting others.

All program and form-group objects are generated for one environment and cannot be used in another. Data tables generated and prepared in a particular environment (whether CICS or z/OS batch) can be used in another environment on the same system.

Table 8 provides information about the types of files produced by generation, including:

- Type of object produced
- Low-level qualifiers of the default PDS name to which the object is written if the build scripts are customized to save the generated files
- Whether production is controlled by a COBOL build descriptor option
- Whether the object can be modified after generation is performed

A description of each object begins on page 70.

For additional information on generation outputs, see the EGL help topics.

You can specify an alias for a program, data table, or form group, and that alias is used for generated outputs. If you do not specify an alias, the default value is the name of the part truncated to the requirements of the target environment (8 characters, for z/OS).

The name given to the outputs includes the alias or the default name, as shown by *alias* in the next table.

A bind control file is always generated and used in preparation for programs that access an SQL database. You can specify your own bind control part to be used to generate the bind control file using the bind option, or you can develop a bind control part with the same name as the program part. Otherwise, a default bind control part is generated.

Table 8. Objects Generated for Programs for a z/OS Host by the Enterprise Developer build process

File Type	PDS Low-level Qualifier	PDS Member Name	File Name on Workstation	z/OS Run-time Environment	Build Descriptor Option	Modifiable
COBOL program	EZESRC	<i>alias</i>	<i>alias</i> .CBL	All	None	No

Table 8. Objects Generated for Programs for a z/OS Host by the Enterprise Developer build process (continued)

File Type	PDS Low-level Qualifier	PDS Member Name	File Name on Workstation	z/OS Run-time Environment	Build Descriptor Option	Modifiable
Sample run-time JCL	EZEJCLX	<i>alias</i>	<i>alias</i> .JCL	z/OS Batch IMS BMP	genRunFile	Yes
Bind command	EZEBIND	<i>alias</i>	<i>alias</i> .BND	All	bind	Yes
Link Edit File	EZELINK	<i>alias</i>	<i>alias</i> .LED	All	linkEdit	Yes
Build Plan	Not applicable (see note 1)	Not applicable	<i>alias</i> BuildPlan.xml	All	prep	No
CICS Entry (See note 2)	EZEPPT	Part specified when generation was requested (<i>alias</i> .PPT)	<i>alias</i>	CICS	cicsEntries	Review and possible modification required

Table 9. Objects Generated for Tables and Transferred to a z/OS Host by the Enterprise Developer Preparation Utility

File Type	PDS Low-level Qualifier	PDS Member Name	z/OS Run-time Environment	Build Descriptor Option	Modifiable
Table COBOL program	EZESRC	<i>alias</i> .CBL	All	genDataTables	No

Table 10. Objects Generated for Form Groups and Transferred to a z/OS Host by the Enterprise Developer Preparation Utility

File Type	PDS Low-level Qualifier	PDS Member Name	File Name on Workstation	z/OS Run-time Environment	Build Descriptor Option	Modifiable
Online print services program - (See note 3)	EZESRC	<i>alias</i>	<i>alias</i> .CBL	All	genFormGroup, genHelpFormGroup	No
Batch print services program - (See note 3)	EZESRC	<i>alias</i> P1	<i>alias</i> P1.CBL	z/OS batch, IMS BMP	genFormGroup, genHelpFormGroup	No
Form group format module - (See note 4)	EZEFOBJ	<i>alias</i> FM	<i>alias</i> FM.FMT	z/OS CICS, IMS/VS	genFormGroup, genHelpFormGroup	No
MFS print services COBOL program	EZESRC	<i>alias</i>	<i>alias</i> .CBL	IMS/VS IMS BMP	genFormGroup	No
MFS control blocks	EZEMFS	<i>alias</i>	<i>alias</i> .MFS	IMS/VS IMS BMP	formServicePgmType, genFormGroup, genHelpFormGroup	No
COBOL copybook for MFS MID/MOD layout	EZECOPY	<i>alias</i>	<i>alias</i> .CPY	IMS/VS IMS BMP	formServicePgmType, genFormGroup, genHelpFormGroup	No

Notes:

1. Build plans are not transferred to the host. They define what needs to be sent to the host. Specifically, the build plan includes the name of a build script that runs on the build server. The build script also contains substitution variable values that are used for substitution in the build script.
For additional details, see the EGL help topics.
2. If you specify the `cicsEntries=RDO` build descriptor option, the PROGRAM entries are placed in *alias.PPT*
3. This object is produced only if the form group contains print forms.
4. This object is produced only if the form group contains text forms.

Objects Generated for Programs

Application COBOL Program

The generated program is a COBOL program that contains the following:

- Program control logic
- Logic for functions and I/O operations
- Data for both the program and program control

The program control logic performs the following functions for a program, as needed:

- Initialization
- Cleanup at end of program
- Error reporting
- Transfer of control

Sample Run-time JCL

The generator produces sample JCL for running programs in the z/OS batch environments when the build descriptor option `genRunFile` is specified during program generation. Each person using the JCL must provide a JOB statement.

The JCL is produced from model JCL templates that can be modified to enforce customer data set naming conventions.

The JCL might not be complete and should be reviewed and modified if necessary before being used. For example, the JCL for the generated program does not contain any DD statements for data sets used by other programs that can be started by CALL or TRANSFER statement. Comments in the JCL indicate where DD statements for these programs need to be added. To build the final JCL needed to run a set of programs as a run unit, you should edit the program JCL and include the DD statements for invoked programs with the JCL for the first main program. You might need to add DD statements for files that are specified during run time with the record-specific variable `resourceAssociation` or with the system variable `sysVar.printerAssociation`.

Bind Commands

Bind commands are required for an SQL program. The bind commands either reside in a bind control part that has the same name as the program or, you can specify the bind control part using the bind build descriptor option.

You are not required to supply a bind control part. If one is not supplied, EGL generates a default bind control part that may or may not meet the requirements of the program.

The bind control part generated by default cannot be affected by users. However, bind control parts provided by the user may contain references to symbolic parameters which get substituted at generation time.

Objects Generated for Tables

Table COBOL Program

The table program is a COBOL program that contains the table contents defined in program working storage. This object is produced when you specify the build descriptor option `genDataTables`. This allows tables to be generated independently of programs when the contents of a table need to be changed.

Objects Generated for Form Groups

Online Print Services Program

The online print services program is a COBOL program that performs print I/O, output formatting, and SET operations for a generated online CICS program that prints output. This object is produced when you specify the build descriptor option `genFormGroup` during program generation.

Batch Print Services Program

The batch print services program is a COBOL program that formats data for line printers and writes the data to either the printer output file (directly to the printer or a QSAM file) or to a generalized sequential access method (GSAM) file. This program is used with programs that run in the z/OS batch environment. This object is produced when you specify the build descriptor option `genFormGroup` or `genHelpFormGroup`.

Form Group Format Module

The form group format module is a generated structure that describes the layout for text forms in the form group. The generator builds the structure as a z/OS object module for the CICS environment. This object is produced when you specify the build descriptor option `genFormGroup` or `genHelpFormGroup`.

MFS Source

In the IMS environment, an MFS source file is generated at the same time as the form group format module. The build server automatically compiles this MFS source to generate IMS format, input, and output messages for each device type defined.

Chapter 10. z/OS Builds

The EGL process generates the files needed to create an executable program. After creating these files, the generation process communicates with the build server on z/OS to transfer the files to the host and then initiate the appropriate builds (compiles, link-edits, binds, etc.) for these programs.

To control the build process, the EGL generation process creates an XML file called a build plan for each generated program. This build plan contains specific information that the build server uses when building the generated program.

The type of information that the build plan contains includes:

- The name of the build script that the build server invokes to process the build
- The location on the client workstation where the server places listings and diagnostics from the build tools (for example, the compiler or linkage editor)
- The generated program
- A list of dependent files for the build process (for example, the name of the link edit file or the bind file) containing information used by the build process
- A list of environment variables that are used to override the default VARS values specified in the Pseudo-JCL build script

The environment variables defined in the build plan are set using build descriptor options and symbolic parameters specified by the user during program generation.

Using the information in the build plan, the server invokes the build script overriding any pre-defined defaults in the JCL with the appropriate values specified in the build plan.

Following the steps outlined in the build script, the build server transforms one set of files into another by invoking tools such as compilers and linkers. For example, using a build script, the build server might transform a COBOL source file into an object file. Another build script might perform the database bind.

Once the build has finished, the server places the listings and diagnostics from the build process in the location specified in the build plan or build script.

Prepared outputs are placed into PDSs on the build server machine. The high level and middle qualifiers of the PDS are controlled by the projectID and system build descriptor options. The low level qualifiers are controlled by the type of output.

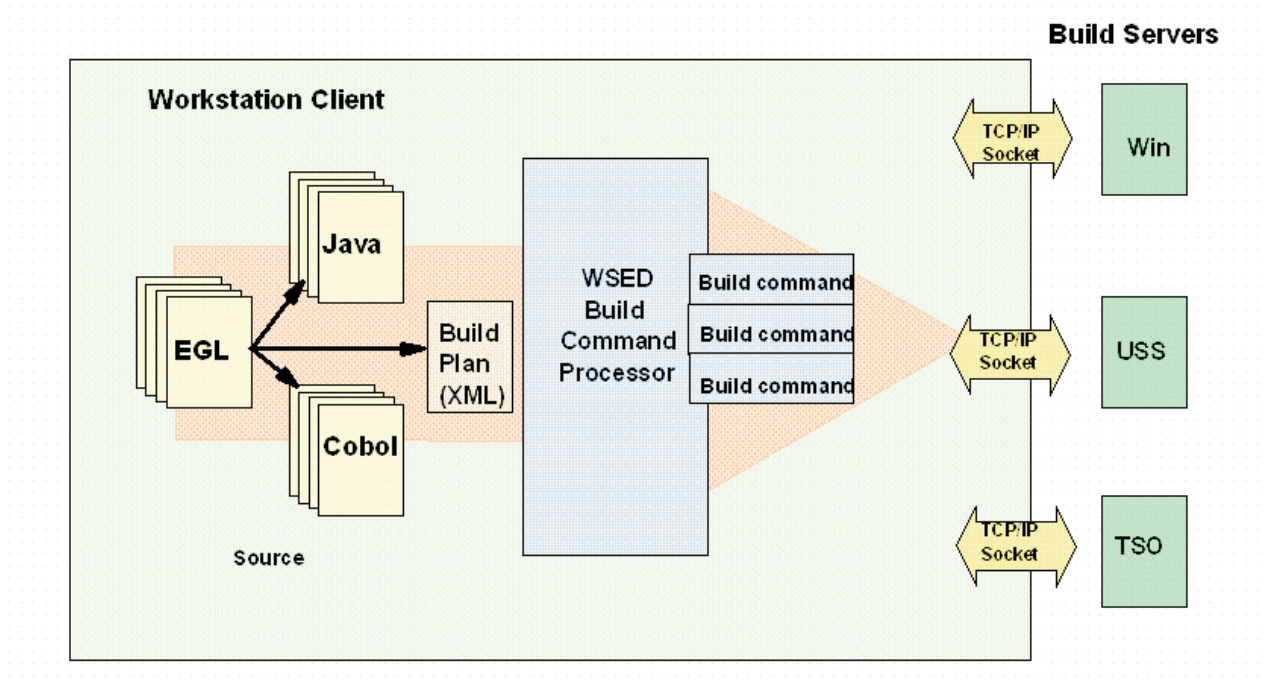


Figure 10. z/OS Build Process

z/OS Build Server

On z/OS, you can configure the build server to perform z/OS or USS builds. If you need both builds, then you need to start two servers, each listening on a unique TCP/IP port for each type.

The Remote Build server performs the following tasks:

- Receives build requests and files.
- Performs character conversions.
- Runs builds within its environment.
- Optionally collects and returns results to the client.

In z/OS, the server load module CCUBLDS receives client build requests. CCUBLDS triggers the JCL member CCUMVS, which executes the CCUBLDW module. CCUBLDW processes your build scripts.

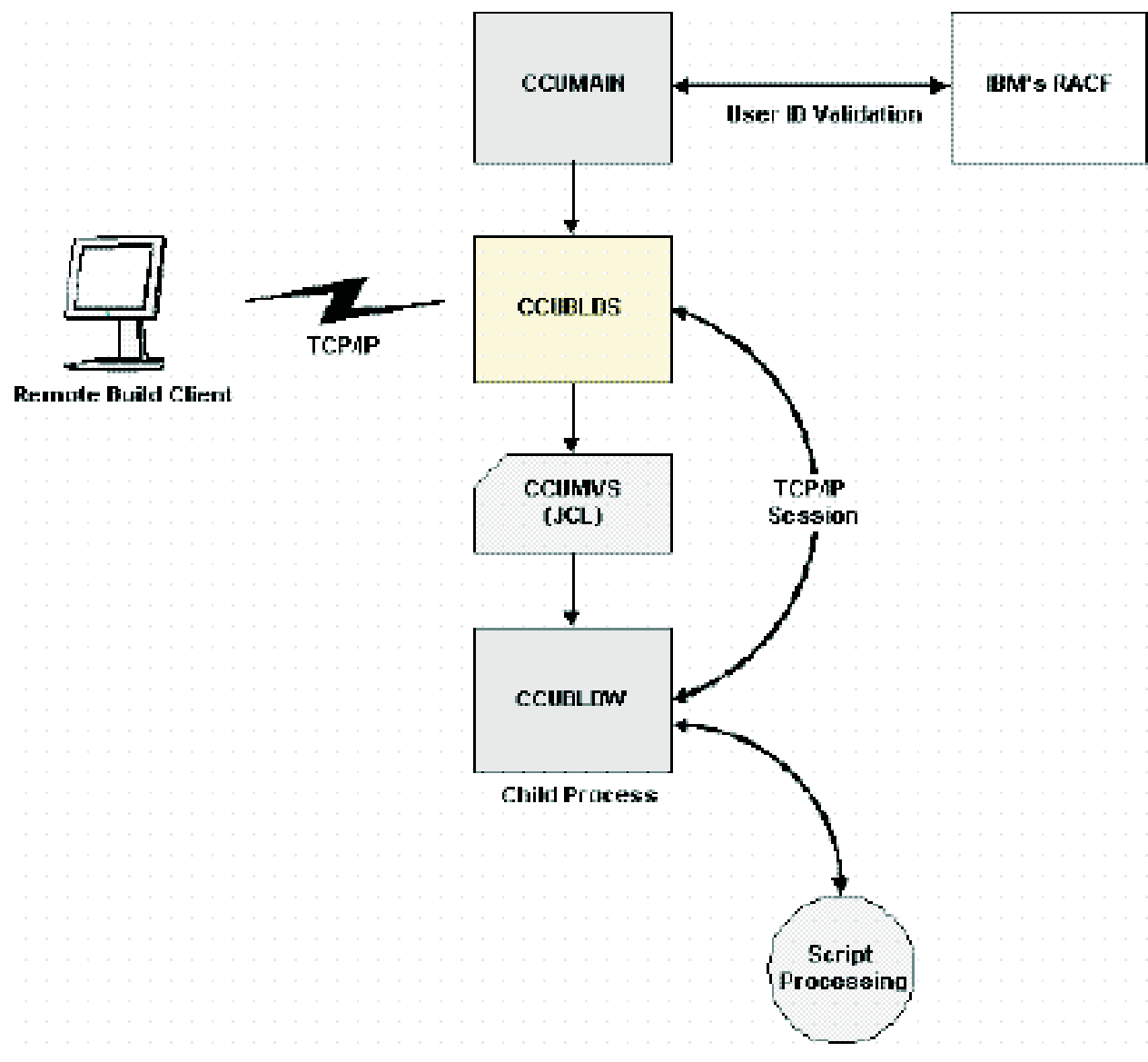


Figure 11. Processing a z/OS Build Request

For USS operations, the server load module CCUMAIN and CCUBLDS run in z/OS. CCUBLDS triggers the JCL member CCUUSS, which starts the USS shell script ccubldw. The ccubldw script starts the executable ccubldw, which processes build requests.

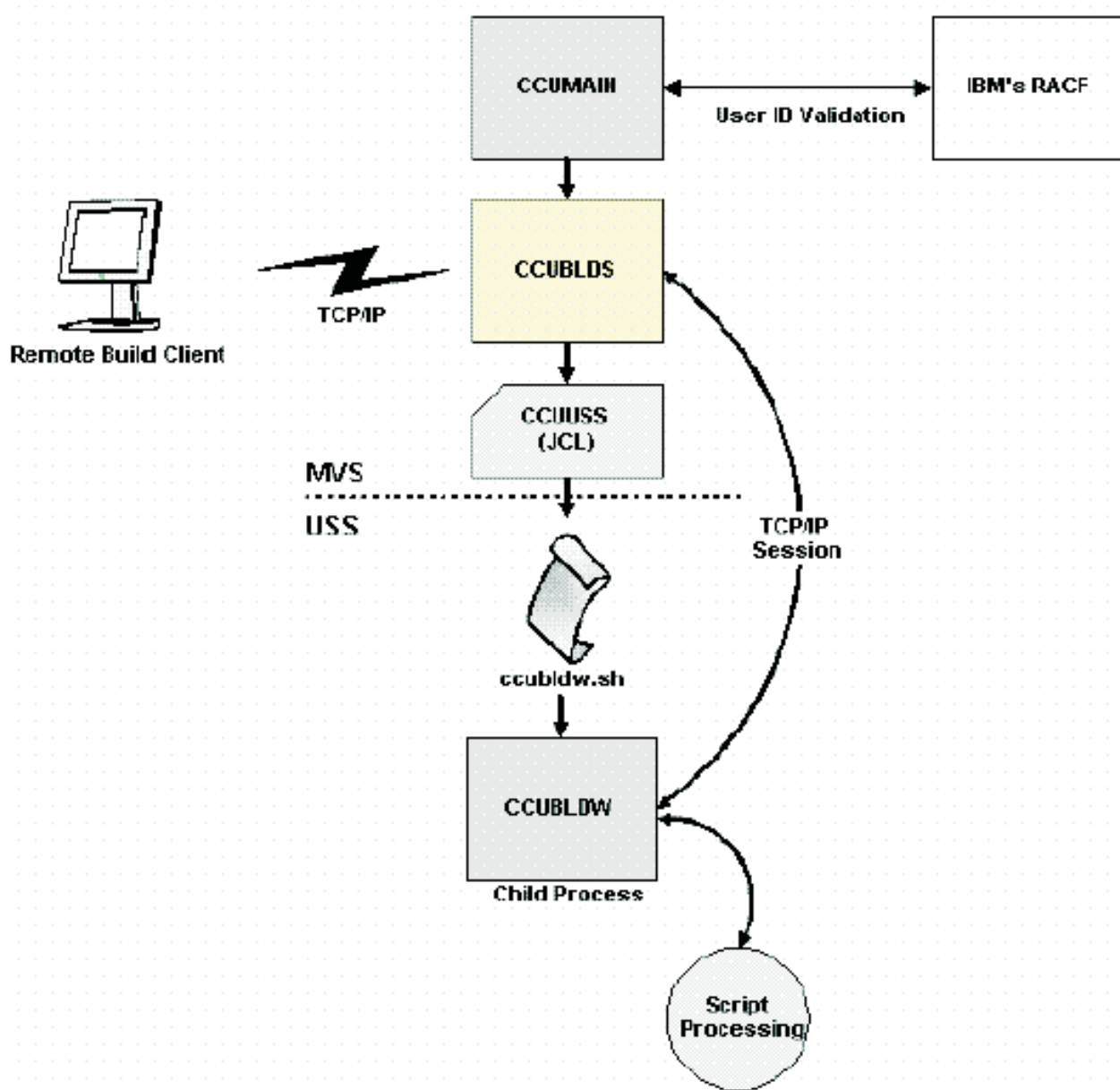


Figure 12. Processing a USS Build Request

Starting a z/OS Build Server

The z/OS build server, CCUBLDS, is an z/OS load module that you can run as a batch program.

```

//CCUBLDS JOB (ACCT#),'TEST',REGION=0M,
//      CLASS=O,MSGCLASS=T
//*-----
//* PROGRAM:  CCURUN
//* JCL to start CCU z/OS Build Server
//*
//* COPYRIGHT: Copyright (C) International Business
//*            Corp. 2001
//*
//* DISCLAIMER OF WARRANTIES:
//* The following enclosed code is sample code created
//* by IBM Corporation. This sample code is not part
//* any standard product and is provided to you solely
//* for the purpose of assisting you in the development
//* of your applications. The code is provide "AS IS",
//* without warranty of any kind. IBM shall not be
//* liable for any damages arising out of your use
//* of the sample code
//*-----
//* Some dataset names may need to be modified
//* according to your system's customization
//*-----
//RUNPGM   EXEC PGM=CCUMAIN,DYNAMNBR=30,REGION=7400K,TIME=NOLIMIT,
// -p 4112 -a 2 -n 3 -q 20 -T 20
//STEPLIB DD DSN=CUST.UCCBLD.LOAD,DISP=SHR
//CCUWJCL DD DISP=SHR,DSN=CUST.UCCBLD.JCL(CCUMVS)
//STDOUT  DD SYSOUT=*
//STDERR  DD SYSOUT=*
//CCUBLOG DD SYSOUT=*
//

```

Figure 13. An example of the JCL needed to start the build server for z/OS

The CCUBLDS job initiates a new job for each build transaction. The sample JCL for that job is in member CCUMVS.JCL of the installation data set whose low-level qualifier is SCCUSAMP. The server is multi-threaded, so these jobs run concurrently and are independent of each other. The number of concurrent jobs running at any one time is limited by system resources (such as initiators).

The server receives commands and files, performs character conversions, sets up the environment, runs builds within this environment, collects the results and returns the results.

See the program directory for Enterprise Developer Server for additional information on customizing the CCURUN, CCURUNU, CCUMVS, and CCUSS JCL and the ccubldw.sh script.

If you start the server on z/OS from an APF-authorized library (this is required in modes 1 and 2 but is optional in mode 0), the server state is authorized ('A') and the build script can specify an APF authorized program as the executable.

For additional information about installing code in an APF-authorized library to allow users to run builds under the authority of the person making a build request, refer to the EGL program directory.

Note: In this case, the build script can also specify non-APF authorized programs. However, in a multistep JCL script, an authorized program cannot be executed after an unauthorized program.

If the server is not started from an APF-authorized library, the server state is not authorized ('U') and the build script can specify only non-APF authorized programs as executables.

You start a build server by using z/OS JCL commands. The syntax for the parameters line is as follows:

Syntax: // PARM= '-p <portno> [-V ...] [-a {2|1|0}] [-n <n>] [-q <q>] [-t] [-T <n>]'

where:

- p** Specifies the port number (portno) to which the server listens to communicate with the clients.
- V** Specifies the verbosity level of the server. You may specify this parameter up to three times (maximum verbosity).

For example, to increase the verbosity to the maximum, you specify -V -V -V.
- a** Specifies the authentication mode of the CCUBLDS server. The server state is either 'A' (APF authorized) or 'U' (not APF authorized).
 - 2** Server state: A. The user submitting the build request must specify a valid user ID and password when the user initiates a build by using the remote build client. The server performs the build transaction under the access and authority of this user ID. Mode 2 is the default.
 - 1** Server state: A. The user submitting the build request can provide a valid user ID and password. The server performs the build transaction under the access and authority of this user. If the user does not provide a user ID and password, the build transaction is performed under the access and authority of the user ID assigned to the build server job.
 - 0** Server state: A or U. If U, APF-authorized build programs will fail. If the user submitting the build request specifies a TSO user ID and password, the server ignores them and the build transaction is performed under the access and authority of the user ID assigned to the build server job.

You can use modes 1 and 2 only if the server load modules are run from an APF-authorized library.

Note: For additional information about installing code in an APF-authorized library to allow users to run builds under the authority of their userid, see the program directory for Enterprise Developer Server.

- n** Specifies the number of concurrent builds. The default is 1. Set n equal to the number of concurrent builds you want to allow. Once there are n number of concurrent builds running, the build server queues any additional requests and submits them on a first come first served basis as builds are completed.
- q** Specifies the size of the queue (q) for clients. The default is 10. Each queued client uses a TCP/IP socket. Therefore setting this too high may require more sockets than are available, causing unpredictable results. If the queue is full, subsequent clients are rejected by the server. However, the build client automatically retries the build in that case.
- t** Starts tracing of this server job and writes output to STDOUT. This parameter is normally used only for debugging.
- T** Specifies the number of minutes the build server will wait for a started child process (CCUBLDW) to complete. If the system is overloaded, increase this value. The default is 5.

Note: See the program directory for Enterprise Developer Server for information about modifying the JCL necessary to start the USS and z/OS build servers

Starting a USS Build Server

You start the USS build server the same way you start the z/OS build server, except with a different dataset allocated by DD name CCUWJCL. This difference is reflected in the CCURUN and CCURUNU JCL customized at installation. The sample JCL CCURUNU needs to be modified just as CCURUN.

The CCUWJCL DD name uses the JCL member CCUUSS. As found in the installation data set whose low-level qualifier is SCCUSAMP, that member acts as a template in submitting build requests to USS using the BPXBATCH utility to submit the USS shell script ccubldw.sh.

The build server creates temporary datasets and directories in the directory where the program is initiated. It is important that the ID that starts the server has the appropriate authority to create these datasets and directories otherwise the server will not initiate properly and all transactions will fail.

Stopping servers

To stop an z/OS server, cancel the job that was used to start it.

Configuring a build server

To configure a build server, you must modify members of the installation data set whose low-level qualifier is SCCUSAMP. Those members contain JCL and are named as follows:

- CCUMVS (for z/OS builds)
- CCUUSS (for USS builds)

Note: See the program directory for Enterprise Developer Server for information about configuring the USS and z/OS build servers.

Working with Build Scripts

There is a fundamental difference between build scripts on z/OS and build scripts on USS. Build scripts on z/OS must be text files and must be written in Pseudo-JCL. On USS, you can use any executable file as a build script and the file can be either text or binary.

Working with z/OS Build Scripts

The build script processed by the z/OS server is always a text file written in Pseudo-JCL. It is specified in one of two ways. If the build script is not specified as part of the build command, then the server looks for it as a member of the PDS specified by the ddname CCUPROC for the server job. This PDS must be of RECFM=FB, LRECL=80.

The build script is parsed by the server. From the parsed results, the server allocates the specified ddnames and data sets; it then executes the programs dynamically.

On z/OS, the server also uses the JCL to determine where to store the files involved in an z/OS build.

EGL uses and Enterprise Developer Server provides build scripts in the PDS specified by DD name CCUPROC in the CCUMVS JCL. These build scripts are the defaults specified in the EGL generated build plans. The member names are FDABCL, FDABIND, FDACL, FDALINK, FDAPCL, FDAPTCL, FDATCL, and FDAMFS.

These must be members in the PDS specified in the CCUPROC DD card in the JCL used to invoke a build request (see the previous section). The members provide the following functions:

FDABCL

Compile and link EGL-generated z/OS batch programs

FDABIND

Bind generated programs that contain DB2 statements

FDACL

Compile and link of generated COBOL programs that do not contain CICS commands

FDALINK

Link of generated format module.

FDAMFS

Invocation of MFS utilities to prepare MFS source for execution in IMS/VS.

FDAPCL

DB2 precompile, compile, and link of generated z/OS batch programs that contain DB2 statements.

FDAPTCL

DB2 precompile, CICS translation, compile, and link of generated CICS COBOL programs that contain DB2 statements.

FDATCL

CICS translation, compile, and link of generated CICS COBOL programs that do not contain DB2 statements.

To override the default build scripts, use the symbolic parameter `DISTBUILD_BUILD_SCRIPT`. To identify the PDS from which to access build scripts at build time, specify the PDS name in the symbolic parameter `BUILD_SCRIPT_LIBRARY`.

See the EGL helps for more information on how to use symbolic parameters during generation.

Writing a JCL build script

JCL build scripts must be written using Pseudo-JCL. The best starting point for a JCL build script is an existing JCL fragment that is used for transforming inputs into outputs. For example, suppose you want to create a build script that compiles a Cobol source file into an OBJECT file using a z/OS compiler. You probably already have JCL that can be submitted as a batch job that does this.

When you create a build script for the z/OS environment, you specify Pseudo-JCL statements, as described in these EGL help topics:

- Pseudo-JCL syntax
- Pseudo-JCL substitution variables
- Setting and including pseudo-JCL substitution variables
- Predefined pseudo-JCL substitution variables

For more information about JCL syntax, refer to the *JCL User's Guide* and *JCL Reference* for your version of z/OS.

File Name Conversions for z/OS

Workstation file names are modified by the z/OS server according to the following rules:

- The directory path of a file name is not used. The end of a directory path of a file name is specified by a slash or left parenthesis ("/", "(", or "\"). All characters of a file name up to and including the rightmost slash or left parenthesis are discarded.
- Lowercase characters are converted to uppercase characters.
- The file extension is stripped from the right, up to and including the separating period. The extension, minus the period, is used by the z/OS server to direct the file to particular data sets according to user-specified syntax in the JCL build scripts.
- The remaining name is truncated from the left, to a maximum of 8 characters.
- Names must contain characters that are valid in z/OS. z/OS allows the following characters:
0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ\$@#
However, the name must begin with an alphabetic character.
- Underscore characters (_) in a file name are converted to at signs (@).

The following are examples of how a workstation name is converted:

- A file name of src\build\fhbldobj.CBL is converted to FHBLDOBJ on z/OS.
- A file name of src/build/fhbtruncate.cbl is converted to FHBTRUNC on z/OS.

In both of these examples, the .CBL or .cbl is split away. The z/OS server uses the resulting extension to resolve and possibly allocate the z/OS data sets needed for the build process. The extensions are required for files that participate in an z/OS build.

Converting JCL to Pseudo-JCL

The following is a JCL procedure for an z/OS compile and link:

```

//*****
//*   JCL Procedure - COBOL COMPILE AND LINK-EDIT
//*****
//*
//ELACL PROC CGHLQ='USER',
//      COBCOMP='SYS1.IGY.SIGYCOMP',
//      COBLIB='SYS1.SCEELKED',
//      ELA='VGEN.HS.V1R2M0',
//      DATA='31',
//      ENV='ZOSCICS',
//      MBR=PGMA,
//      RESLIB='SYS1.RESLIB',
//      RGN=1024K,
//      SOUT='*',
//      WSPC=500 ,
//*
//*  PARAMETERS:
//*    CGHLQ  = COBOL GENERATION USER DATA SET HIGH LEVEL QUALIFIER
//*    COBCOMP = COBOL COMPILER LIBRARY
//*    COBLIB  = LE RUN TIME LIBRARY
//*    ELA     = EGL HIGH LEVEL QUALIFIER
//*    DATA   = COMPILE OPTION FOR PLACING WORKING STORAGE
//*             ABOVE 16M LINE
//*    ENV     = COBOL GENERATION USER DATA SET ENVIRONMENT QUALIFIER
```

```

/**          (SHOULD BE EQUAL TO GENERATION TARGET ENVIRONMENT)
/**      MBR      = SOURCE NAME
/**      RESLIB   = IMS RESLIB LIBRARY
/**      RGN      = REGION SIZE
/**      SOUT     = SYSOUT ASSIGNMENT
/**      WSPC     = PRIMARY AND SECONDARY SPACE ALLOCATION
/**
/*******
/**      COMPILE THE COBOL PROGRAM
/*******
/**
/**C          EXEC PGM=IGYCRCTL,REGION=&RGN,
/**          PARM=(NOSEQ,QUOTE,OFFSET,LIB,RENT,NODYNAM,DBCS,OPT,
/**          'TRUNC(BIN)', 'NUMPROC(NOPFD)',NOCMPR2, 'DATA(&DATA)')
/**STEPLIB DD DISP=SHR,DSN=&COBCOMP
/**SYSIN   DD DISP=SHR,DSN=&CGHLQ..&ENV..EZESRC(&MBR)
/**SYSLIB  DD DISP=SHR,DSN=&ELA..SELACOPY
/**SYSLIN  DD DISP=(MOD,PASS),DSN=&&LOADSET,UNIT=VIO,
/**          SPACE=(800,(&WSPC,&WSPC))
/**SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
/**SYSUDUMP DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
/**SYSUT1  DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
/**SYSUT2  DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
/**SYSUT3  DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
/**SYSUT4  DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
/**SYSUT5  DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
/**SYSUT6  DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
/**SYSUT7  DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
/**
/*******
/**      LINK-EDIT THE COBOL PROGRAM
/**      IF THE RETURN CODE ON ALL PREVIOUS STEPS IS 4 OR LESS
/*******
/**
/**L          EXEC PGM=IEWL,COND=(5,LT,C),REGION=&RGN,
/**          PARM='RENT,REUS,LIST,XREF,MAP,AMODE(31),RMODE(ANY)'
/**SYSLIB DD DISP=SHR,DSN=&COBLIB
/**          DD DISP=SHR,DSN=&RESLIB
/**SELALMD DD DISP=SHR,DSN=&ELA..SELALMD
/**SYSLIN DD DISP=(OLD,DELETE),DSN=&&LOADSET
/**          DD DDNAME=SYSIN
/**SYSLMOD DD DISP=SHR,DSN=&CGHLQ..&ENV..LOAD(&MBR)
/**SYSPRINT DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
/**SYSUDUMP DD SYSOUT=&SOUT,DCB=BLKSIZE=13300
/**SYSUT1 DD SPACE=(1024,(&WSPC,&WSPC)),UNIT=VIO

```

The first step in converting the JCL fragment is to recognize the intent for each of the data sets and ddnames. For this Cobol compiler example, the SYSIN ddname needs to be associated with the source file, the SYSLIN ddname needs to be associated with the object file, and so on.

In each of these cases, the build script must tell the server where to pick up the input files before the execution of the specified program (PGM=IGYCRCTL) and where to put the output files after the execution of the specified program.

Assume that your source files have the extension cbl. You allocate a data set to the SYSIN ddname to contain a source file with a .cbl extension. You specify the DCB, UNIT, DISP, and SPACE attributes to dynamically create this data set every time this build script is invoked. You add CCUEXECBL to indicate that the file content comes from an input file with an extension of .cbl.

You specify the output messages that will be returned by using the CCUOUT attribute. This attribute tells the z/OS server to return the information in the data set associated with the CCUEXT=CCUOUT attribute.

The following JCL build script is the result of converting the JCL procedure.

```

//*****
//* BUILD SCRIPT - COBOL COMPILE AND LINK-EDIT
//*****
//*
//DEFAULTS VARS CGHLQ=USER,
//          COBCOMP=SYS1.IGY.V3R1M0.SIGYCOMP,
//          COBLIB=SYS1.SCEELKED,
//          COBLISTPARMS=OFFSET&COMMA.NOLIST&COMMA.MAP,
//          ELA=EDS.V5R2M0,
//          DATA=31,
//          SYSTEM=ZOSCICS,
//          MBR=PGMA,
//          RGN=4096K
//          CCUEXTC=CCUOUT,
//          CCUEXTL=CCUOUT,
//          SOUT=*,
//          DBCS=&COMMA.DBCS
//          WSPC=2500
//*
//* PARAMETERS:
//*   CGHLQ   = COBOL GENERATION USER DATA SET HIGH LEVEL QUALIFIER
//*   COBCOMP = COBOL COMPILER LIBRARY
//*   COBLIB  = LE RUN TIME LIBRARY
//*   COBLISTPARMS = LISTING OPTIONS FOR COBOL COMPILER
//*   ELA     = VISUALAGE GENERATOR SERVER HIGH LEVEL QUALIFIER
//*   DATA   = COMPILE OPTION FOR PLACING WORKING STORAGE
//*           ABOVE 16M LINE
//*   DBCS    = COMPILE OPTION FOR INDICATING SOURCE CONTAINS DBCS
//*           CHARACTERS
//*   SYSTEM  = SYSTEM GENERATING FOR. USED AS USER DATASET MIDDLE
//*           QUALIFIER
//*   MBR     = SOURCE NAME
//*   RGN     = REGION SIZE
//*   CCUEXTC = CCUEXT VALUE FOR COMPILE PRINTOUTS RETURNED TO
//*           CLIENT.
//*           CCUOUT=RETURN TO CLIENT AS FILE NAMED BY DDNAME
//*           CCUSTD=RETURN TO CLIENT AS STANDARD OUT
//*           CCUERR=RETURN TO CLIENT AS STANDARD ERROR
//*   CCUEXTL = CCUEXT VALUE FOR LINK PRINTOUTS RETURNED TO CLIENT
//*           CCUOUT=RETURN TO CLIENT AS FILE NAMED BY DDNAME
//*           CCUSTD=RETURN TO CLIENT AS STANDARD OUT
//*           CCUERR=RETURN TO CLIENT AS STANDARD ERROR
//*   SOUT    = SYSOUT ASSIGNMENT IF A SYSOUT FILE NOT RETURNED
//*           TO CLIENT
//*   WSPC    = PRIMARY AND SECONDARY SPACE ALLOCATION
//*
//*****
//*           COMPILE THE COBOL PROGRAM
//*****
//*
//C          EXEC PGM=IGYCRCTL,REGION=&RGN,
//          PARM='NOSEQ,QUOTE,LIB,RENT,NODYNAM,OPT&DBCS,
//          TRUNC(BIN),NUMPROC(NOPFD),&COBLISTPARMS.,DATA(&DATA)'
//STEPLIB DD DISP=SHR,DSN=&COBCOMP
//* COBOL SOURCE CODE UPLOADED FROM CLIENT (&MBR.CBL)
//SYSIN DD CCUEXT=CBL,DISP=(NEW,DELETE),
//      UNIT=SYSDA,SPACE=(TRK,(10,10)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSLIB DD DISP=SHR,DSN=&ELA..SELACOPY
//SYSLIN DD DISP=SHR,DSN=&CGHLQ..&SYSTEM..OBJECT(&MBR),ENQ=YES
//* RETURN COMPILER LISTING TO CLIENT AS FILE &PREFIX.C.SYSPRINT

```

```

//SYSPRINT DD CCUEXT=&CCUEXTC,DISP=(NEW,DELETE),
//      UNIT=VIO,SPACE=(CYL,(5,5)),
//      DCB=(RECFM=FB,LRECL=121,BLKSIZE=1210)
//*      UNIT=VIO,SPACE=(TRK,(30,10)),
//SYSUT1 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT2 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT3 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT4 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT5 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT6 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//SYSUT7 DD SPACE=(800,(&WSPC,&WSPC),,,ROUND),UNIT=VIO
//*
//*****
//*          LINK-EDIT THE COBOL PROGRAM
//*          IF THE RETURN CODE ON ALL PREVIOUS STEPS IS 4 OR LESS
//*****
//*
//L          EXEC PGM=IEWL,COND=(5,LT,C),REGION=&RGN,
//          PARM='RENT,REUS,LIST,XREF,MAP,AMODE(&DATA),RMODE(ANY)'
//SYSLIB DD DISP=SHR,DSN=&COBLIB
//SELALMD DD DISP=SHR,DSN=&ELA..SELALMD
//OBJLIB DD DISP=SHR,DSN=&CGHLQ..&SYSTEM..OBJECT
//* LINK EDIT CONTROL FILE UPLOADED FROM CLIENT (&MBR.LED)
//SYSLIN DD CCUEXT=LED,DISP=(NEW,DELETE),
//      UNIT=SYSDA,SPACE=(TRK,(10,10)),
//      DCB=(RECFM=FB,LRECL=80,BLKSIZE=3200)
//SYSMOD DD DISP=SHR,DSN=&CGHLQ..&SYSTEM..LOAD(&MBR),ENQ=YES
//* RETURN LINK EDIT LISTING TO CLIENT AS FILE &PREFIX.L.SYSPRINT
//SYSPRINT DD CCUEXT=&CCUEXTL,DISP=(NEW,DELETE),
//      UNIT=VIO,SPACE=(TRK,(30,10)),
//      DCB=(RECFM=FB,LRECL=121,BLKSIZE=1210)
//SYSUT1 DD SPACE=(1024,(&WSPC,&WSPC)),UNIT=VIO
//

```

Chapter 11. Preparing and Running a Generated Program in CICS

This chapter describes the unique steps required to prepare and run a generated COBOL program in an CICS environment:

- Modifying CICS startup JCL
- Making new modules available
- Making programs resident
- Running programs

Modifying CICS Resource Tables

The CICS environment uses resource definitions to identify startup parameters, transactions, programs, files, databases, transient data destinations, and system locations for proper operation. You must add to or modify these resource definitions to correctly identify all objects to be used in the new or changed program. When using CICS tables, the tables are compiled as assembler programs and stored in a run-time library. Some tables can also be maintained through an online facility as described in the resource definition online manual for your version of CICS. CICS requires that the online facility be used in place of processing program table (PPT) and program control table (PCT) entries.

Refer to the CICS resource definitions guide for additional information on providing definitions.

You can either write your own PPT, PCT, or RDO program and transaction entries or use the ones generated by Enterprise Developer. You must handle DCT, FCT, and RCT entries yourself.

Program Entries (PPT)

The entries in the PPT define programs to CICS. The EGL COBOL generation process creates programs that must be defined, as a resource definition online (RDO) PROGRAM entry or by using dynamic program entries.

An entry is required for each Enterprise Developer generated program. You can request that sample PPT or RDO entries be generated for you by specifying the `cicsEntries build descriptor` option at generation.

Either the batch program DFHCSDUP utility or the resource definition online (RDO) CEDA DEFINE PROGRAM command can be used to define the server program to CICS.

If you specify `cicsEntries=RDO`, CICS RDO DEFINE PROGRAM commands are generated for you for each program that requires an RDO PROGRAM entry. The preparation command created during generation copies the RDO command files to the z/OS library specified at generation.

The following examples show how to define the same entries using the RDO CEDA transaction DEFINE PROGRAM command.

```
CEDA DEF PROG(progname) L(LE370) REL(NO) RES(NO) S(ENABLED) GROUP(xxxx)
```

The values shown for REL, RES, and S keywords are the default values and can be omitted from the command. RES(YES) might provide better performance for frequently used programs.

Transaction Entries (PCT)

A CICS transaction entry contains the control information used by CICS for identifying and initializing a transaction. This entry is required by CICS to verify incoming requests to start transactions, and to supply information about the transaction such as the transaction priority, the security key, and the length of the transaction work area (TWA).

A CICS RDO TRANSACTION entry is required for each transaction code used to start a Enterprise Developer generated program.

EGL generated programs can be started by a remote procedure call from some remote system. The CICS support mirror program DFHMIRS, normally invoked by the CPPI transaction is used during this remote procedure call. It:

1. Determines which server program should be given control
2. Builds the COMMAREA
3. Links to the defined server program via CICS LINK

CPPI is the CICS supplied default transaction code to invoke the CICS mirror program DFHMIRS. When using CPPI to start EGL programs, you must change the transaction definition for CPPI to specify a TWASIZE of at least 1024 bytes.

To avoid making changes to the CPPI definition in the CICS supplied group, it is recommended that you copy the CICS supplied CPPI definitions to a new group or create a unique transaction ID with the same characteristics as CPPI. The new transaction or copy of CPPI should be changed and verified to ensure the following values are set.

1. The twasize is 1024
2. The profile is DFHCICSA (CICS default would be DFHCICST (T for terminal))
3. The program invoked is DFHMIRS

Example:

```
DEFINE TRANSACTION(MYMI) PROGRAM(DFHMIRS) TWASIZE(1024) PROFILE(DFHCICSA)
```

Destination Control Table Entries (DCT)

A CICS destination control table (DCT) entry is required for each program file that is assigned to a transient data queue. A DCT entry is also required for destinations specified as error destination queue names using the Enterprise Developer Server diagnostic controller utility. The parameters for DCT entries depend on your destination type. There are intrapartition, extrapartition, indirect, and remote destinations. See "Using and Allocating Data Files in CICS" on page 36 for information about defining and managing program data files and "Defining Transient Data Queues" on page 38 for information about defining the DCT entry for the error destination queue. Refer to appropriate CICS manuals for more information on DCT entries.

To provide these definitions as RDO entries, see the CICS resource definition guide.

File Control Table Entries (FCT)

A CICS file control table (FCT) entry is required for each program file that is specified as file type VSAM. You must identify all FCT entries that might be referenced at run time. Files can also be defined using RDO. See “Using and Allocating Data Files in CICS” on page 36 for more information on defining and managing program data files in the CICS environment.

Resource Control Table Entry (RCT)

If the programs running under a transaction access DB2 databases, then you must define an entry in the CICS resource control table (RCT) that associates the transaction identifier with the program plan name.

The following example shows the minimum RCT entry required: Index text: sample JCL, RCT entry

```
DSNCRCT TYPE=ENTRY TXID=tran PLAN=plan-name
```

For more information on the other parameters you can specify when you define RCT entries, refer to the chapter on connecting the CICS attachment facility in the DB2 installation manual for your version of DB2.

To provide these definitions as RDO entries, see the CICS resource definition guide.

Using Remote Programs, Transactions, or Files

Refer to the appropriate CICS manuals for information about defining remote programs, transactions, or files.

Modifying CICS Startup JCL

You must include the load library where your generated programs reside in the DFHRPL DD concatenation. Your system administrator included the LE run-time libraries and the Enterprise Developer Server load library in the DFHRPL DD concatenation when the Enterprise Developer Server product was installed.

The CICS startup JCL might need to be modified to add or change allocations for files used by EGL-generated programs. These include VSAM files and extrapartition transient data destinations.

For VSAM data sets, it is not necessary to include allocations in the startup JCL if you specify the data set name and disposition in the CICS FCT or RDO entry for the file. CICS dynamically allocates the file at open time.

Making New Modules Available in the CICS Environment

After you generate a new version of a program, you need to make the modules available to CICS.

For programs, you can use the CICS NEWCOPY command or the Enterprise Developer Server new copy utility to cause the new copy of the program to be used the next time a load request is issued for the program.

For more information on the Enterprise Developer Server new copy utility, see “New Copy” on page 112.

Making Programs Resident

You can make frequently used programs or programs with high performance requirements resident to avoid the overhead of loading the programs when they are used. To aid in deciding which programs should be made resident, you can use CICS shutdown statistics to determine how often a generated program is loaded in a CICS region.

To make a program resident, specify the program as resident in the RDO entry for the program.

Running Programs under CICS

Called programs can be started by another Enterprise Developer program, by a non-Enterprise Developer program, or through the remote CICS services.

Prior to running a generated program, the program user might be required to sign on to the CICS environment. Refer to CICS documentation for information about signing on.

Controlling Diagnostic Information in the CICS Environment

Enterprise Developer Server provides a diagnostic controller utility for the CICS environment. This utility allows you to control the type of dump, the name of the error destination queue and journal number for error messages, and whether the transaction is disabled when a run unit error occurs. See Chapter 17, “Diagnostic Control Options,” on page 117 for more information about the diagnostic controller utility.

Printing Diagnostic Messages in the CICS Environment

Enterprise Developer Server provides a way to print diagnostic messages written to a transient data queue. See “Diagnostic Message Printing Utility” on page 114 for more information.

Chapter 12. Preparing and Running Generated Programs in z/OS Batch

This chapter describes the unique steps required to prepare a generated COBOL program to run in a z/OS batch environment:

- Running main programs
- Examples of runtime JCL
- Recovery and restart

For general information on preparing your program for the runtime environment, see Chapter 9, "Output of Program Generation on z/OS Systems." For information on modifying the JCL, see Chapter 13, "Creating or Modifying Run-time JCL on z/OS Systems."

Running Main Programs under z/OS Batch

A main batch program generated for the z/OS batch environment can be started by submitting JCL. Called programs can only be started by another Enterprise Developer program or by a non-Enterprise Developer program.

The EGL COBOL generation process creates sample runtime JCL for running programs in the z/OS batch environment. The generated JCL has same name as the program. If you set the **genRunFile** build descriptor option to "YES", a sample JCL is created specifically for the program during program generation. The runtime JCL is transferred to a z/OS partitioned data set (PDS) by the Enterprise Developer prepare function.

The JCL might need to be modified to add data sets required by called or transferred-to programs. You also need to modify the JCL to add any data sets that are dynamically allocated with the EZEDEST or EZEDESTP special function words. See Chapter 13, "Creating or Modifying Run-time JCL on z/OS Systems," on page 93 for more information on modifying the sample runtime JCL.

If you get a JCL error for the runtime JCL, check the generation listing for the programs involved for any error messages related to JCL generation. In addition, ensure the tailoring that was done for the JCL templates is correct. Also check any changes you made when you customized the sample runtime JCL.

The following sections show JCL for different z/OS batch programs.

Examples of Runtime JCL for z/OS Batch Programs

The generated JCL in the following examples has these characteristics:

- The examples are based on the JCL templates shipped with Enterprise Developer. Your actual JCL templates might differ if your system administrator has tailored them for your organization. Refer to the online helps for more information about tailoring JCL templates.
- Lowercase text appears in the examples where a generic example name has been substituted for an actual program or data set name.
- EZEPRINT is always routed to SYSOUT=*

If you route EZEPRINT to a data set, you must use the following DCB attributes:

- LRECL=137, BLKSIZE=141, RECFM=VBA if the map group does not contain any DBCS maps
- LRECL=654, BLKSIZE=658, RECFM=VBA if the map group contains any DBCS maps

You cannot use map groups that do not have any DBCS maps with map groups that do have DBCS maps in a same job step.

Running a Main Batch Program with No Database Access

Figure 14 shows the JCL used to start a main batch program.

```
//jobname JOB .....MSGCLASS=A
//stepnam EXEC PGM=appl-name
//STEPLIB DD DSN=CEE.SCEERUN,DISP=SHR
//          DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//          DD DSN=cghlq.env.LOAD,DISP=SHR
//ELAPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330)
//ELASNAP DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//EZEPRINT DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//SYSABOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
/* Application specific DD statements
//file-name-1 DD .....
//file-name-n DD .....
```

Figure 14. JCL for Main Batch Program Run as z/OS Batch without DB2 or DL/I Access

Running a Main Batch Program with DB2 Access

Figure 15 shows the JCL used to start a main batch program that gains access to DB2 resources. The JCL must run the z/OS TSO terminal monitor program to run the generated program.

```
//jobname JOB USER=userid,.....
//stepname EXEC PGM=IKJEFT01,DYNAMNBR=20,REGION=4M
//STEPLIB DD DSN=DSN.SDSNLOAD,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//          DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//          DD DSN=cghlq.env.LOAD,DISP=SHR
//ELAPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330)
//ELASNAP DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//EZEPRINT DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//SYSABOUT DD SYSOUT=*
//SYSOUT DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSTSIN DD *
DSN SYSTEM (ssid)
RUN PROG (appl-name) PLAN (plan-name) -
LIB ('cghlq.env.LOAD')
END
/*
//SYSTSPRT DD SYSOUT=*
/* Application specific DD statements
//file-name-1 DD .....
//file-name-n DD .....
```

Figure 15. JCL for Main Batch Program Run as z/OS Batch with DB2 Access

Running Main Batch Program with DL/I Access

If a main batch program runs as a DL/I batch program, then all DL/I requests are handled by a private IMS region. The JCL for the step that runs the batch program

must include DD statements for the IMS log if databases are opened with update intent or if the program uses the Enterprise Developer CALL AUDIT service routine. Also, a DD statement must be included for each of the data sets associated with the DL/I databases referenced in the IMS PSB. The IMS log DD statements (IEFRDER and IEFRDER2) are normally included in the DLIBATCH procedure.

Enterprise Developer uses the JCL template FDA2MDLI to build the DD statements for program databases. This template has the DD statement commented out because the high-level program database qualifiers are not collected by Enterprise Developer. You need to provide the final tailoring of these DD statements in the sample runtime JCL. Alternatively, depending on your naming conventions, your administrator might be able to modify the FDA2MDLI template so that you can use the **symbolicParameter** build descriptor option to set high-level qualifiers for databases. Refer to the online helps for information about modifying templates and using the **symbolicParameter** build descriptor option.

Figure 16 shows the sample JCL used to run a generated program as a DL/I batch program.

```
//jobname JOB .....
//stepname EXEC DLIBATCH,DBRC=Y,
//          MBR=appl-name,PSB=ims-psb-name,BK0=Y,IRLM=N
//G.STEPLIB DD
//          DD
//          DD DSN=CEE.SCEERUN,DISP=SHR
//          DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//          DD DSN=cgh1q.env.LOAD,DISP=SHR
//* DFSVSAMP IS REQUIRED IF VSAM DATABASES - REPLACE MEMBER WITH
//* ONE THAT HAS VALID BUFFER POOL SIZES FOR YOUR APPLICATION
//G.DFSVSAMP DD DSN=ELA.V1R2M0.ELASAMP(ELAVSAMP),DISP=SHR
//G.ELAPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330)
//G.ELASNAP DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.EZEPRINT DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.SYSABOUT DD SYSOUT=*
//G.SYSOUT DD SYSOUT=*
//* Application specific DD statements including DL/I DB DD statements
//file-name-1 DD .....
//file-name-n DD .....
```

Figure 16. JCL for Main Batch Program Run as z/OS Batch with DL/I Access

Running a Main Batch Program with DB2 and DL/I Access

Figure 17 on page 92 shows the JCL that enables a program to run as a stand-alone DL/I batch processing program and to gain access to DB2 databases. Special recovery considerations are required. Refer to the DB2 documentation for your system for additional information.

The JCL for the step that runs the batch program must include DD statements for the IMS log if databases are opened with update intent or if the program uses the Enterprise Developer CALL AUDIT service routine. Also, a DD statement must be included for each of the data sets associated with the DL/I databases referenced in the IMS PSB. The IMS log DD statements (IEFRDER and IEFRDER2) are normally included in the DLIBATCH procedure.

Enterprise Developer uses the JCL template FDA2MDLI to build the DD statements for DL/I program databases. This template has the DD statement commented out because the high-level program database qualifiers are not collected by Enterprise Developer. You need to provide the final tailoring of these DD statements in the sample runtime JCL. Alternatively, depending on your

naming conventions, your administrator might be able to modify the FDA2MDLI template so that you can use the **symbolicParameter** build descriptor option to set high-level qualifiers for databases. Refer to the online helps for information about modifying templates and using the **symbolicParameter** build descriptor option.

```
//jobname JOB .....
//stepname EXEC DLIBATCH,DBRC=Y,
//          MBR=DSNMTV01,PSB=ims-psb-name,BKO=Y,IRLM=N
//G.STEPLIB DD
//          DD
//          DD DSN=DSN.SDSNLOAD,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//          DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//          DD DSN=cghlq.env.LOAD,DISP=SHR
/* DFSVSAMP IS REQUIRED IF VSAM DATABASES - REPLACE MEMBER WITH
/* ONE THAT HAS VALID BUFFER POOL SIZES FOR YOUR APPLICATION
//G.DFSVSAMP DD DSN=ELA.V1R2M0.ELASAMP(ELAVSAMP),DISP=SHR
/*
//G.DDOTV02 DD DSN=&&TEMP1,
//          DISP=(NEW,PASS,DELETE),
//          SPACE=(CYL,(1,1),RLSE),UNIT=SYSDA,
//          DCB=(RECFM=VB,BLKSIZE=4096,LRECL=4092)
//G.DDITV02 DD *
//          ssid,SYS1,DSNMIN10,,R,-,connection name,plan-name,appl-name
/*
//G.ELAPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330)
//G.ELASAP DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.EZEPRINT DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.SYSABOUT DD SYSOUT=*
//G.SYSOUT DD SYSOUT=*
/* Application specific DD statements including DL/I DB DD statements
//file-name-1 DD .....
//file-name-2 DD .....
/*
/* Attempt to print out the DDOTV02 data set created in previous step
//stepnam2 EXEC PGM=DFSERA10,COND=EVEN
//STEPLIB DD DSN=IMSVS.RESLIB,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSUT1 DD DSN=DSN=&&TEMP1,
//          DISP=(OLD,DELETE)
//SYSIN DD *
CONTROL CNTL K=000,H=8000
OPTION PRINT
/*
```

Figure 17. JCL for Main Batch Program Run as z/OS Batch with DB2 and DL/I Access

Recovery and Restart for Batch Programs

For z/OS batch programs that use DL/I, the generated runtime JCL includes the parameter BKO=Y. If the program updates databases or files, specify BKO=Y in the runtime JCL in order to have rollback (ROLB) requests honored. If you specify BKO=N, DL/I returns status code AL for the roll-back call. Enterprise Developer Server treats the AL status code as a soft error. No error message is issued, and processing continues.

You should develop recovery procedures in the event of program or system errors. Enterprise Developer does not generate JCL to perform restart or recovery procedures.

Chapter 13. Creating or Modifying Run-time JCL on z/OS Systems

This chapter contains the information you need to modify the sample run-time JCL created during program generation. You might need to modify the run-time JCL for the following reasons:

- EGL does not include DD statements in the JCL to allocate data sets accessed by programs called by or transferred to from the generated program.
- The generator does not include DD statements to allocate data sets accessed when the EGL program moves a value to the record-specific variable `resourceAssociation` or to the system variable `sysVar.printerAssociation`.
- The generator does not create any recovery or restart JCL.
- The sample JCL is based on the initial program in the run unit.

You need to ensure that the load libraries containing the initial program and any dynamically invoked programs are included in the STEPLIB concatenation unless you are using methods to put the load modules in memory. This includes program modules that are called dynamically or that receive control by a transfer and includes print services programs, form group format modules, and data-table programs.

Tailoring JCL before Generation

EGL creates sample run-time JCL for batch programs. The sample run-time JCL is based on templates that are installed in the MVSTEMPLATES subdirectory of the following plugin:

com.ibm.etools.egl.generators.cobol_*version*

version

The product version; for example, *6.0.0*.

You can specify the location of site-specific templates by setting the build descriptor option `templateDir`.

Some of the reasons to tailor the JCL templates are as follows:

- Implementing your installation's data set naming conventions
- Adding DD statements to the STEPLIB concatenation
- Specifying a different DB2 subsystem

The sample JCL is shown in Chapter 12, "Preparing and Running Generated Programs in z/OS Batch," on page 89.

Table 11. Run-time JCL Templates Based on Environment and Databases

JCL Template	Database	Environment
FDA2MEBE	None	z/OS batch
FDA2MEBD	DB2	z/OS batch
FDA2MEBA	Any, for called program	z/OS batch
FDA2MEBB	DB2 and DL/I	z/OS batch
FDA2MEBC	DL/I	z/OS batch

Table 11. Run-time JCL Templates Based on Environment and Databases (continued)

JCL Template	Database	Environment
FDA2MEIA	DB2	IMS BMP
FDA2MEIB	None	IMS BMP

Table 12 shows the JCL templates that serve as models for DD statement generation for program-dependent files and databases.

Table 12. Model DD Statement for Program-Dependent Files and Databases

JCL Template	Contents
FDA2MSDI	QSAM input file
FDA2MSDO	QSAM output file
FDA2MVSJ	VSAM input file
FDA2MVSO	VSAM output file
FDA2MGSI	GSAM input file
FDA2MGSO	GSAM output file
FDA2MIMS	GSAM IMS dataset for IMS BMP
FDA2MCAL	Comment indicating where to insert DD statements for known transferred-to and called programs
FDA2MEZA	Comment indicating where to insert DD statements for programs transferred-to using the system variable sysVar.transferName
FDA2MEZD	Comment indicating where to insert DD statements for data sets using the record-specific variable resourceAssociation or the system variable sysVar.printerAssociation
FDA2MDLI	Comment indicating where to insert DD statements for DL/I databases on z/OS batch

Modifying Run-time JCL

The sample run-time JCL for main batch programs contains EXEC statements to run a program or a cataloged procedure. The JCL for main batch programs does not include a JOB statement or the DD statements for data sets accessed by called or transferred-to programs. Before you use the JCL to run the program, you must do the following:

- Add a JOB statement.
- Insert missing DD statements as required. Comments in the generated JCL indicate where to insert the DD statements.

The sample run-time JCL for a called program contains only the DD statements that are required for the called program.

After generation, add the DD statements for any files required by called or transferred-to programs (including those named with sysVar.transferName) to the sample JCL for the main program. In addition, you must add DD statements for any files accessed by moving a value to the record-specific variable resourceAssociation or to the system variable sysVar.printerAssociation. You do not need to add DD statements for files that you access dynamically.

The type of run-time JCL generated for a main batch program varies based on the types of databases used by the main program. The generated run-time JCL does not consider the types of databases accessed by called or transferred-to programs. If the main program does not use relational databases, but it calls or transfers to programs that use relational databases, you must modify the run-time JCL for the main program.

For example, consider the following situation:

- Program A is a main batch program that does not use relational databases.
- Program B is main program that accesses relational databases.
- Programs A and B are generated for the z/OS batch environment.
- Program A transfers to program B

Because program A does not use DB2, the JCL generated for program A is for a main batch program without DB2 access (as shown in Figure 14 on page 90). This JCL will not run correctly because program B requires DB2 to run. However, the JCL generated for program B is for an z/OS batch job with DB2 access (as shown in Figure 15 on page 90). The run-time JCL for program B can serve as a starting point for creating the JCL required to run program A. The following changes are required to the run-time JCL for program B:

- Change RUN PROG(APPLB) to RUN PROG(APPLA).
- Add any DD statements for files required by program A or other programs in the job step.

If program B is a called program and program A calls B rather than transferring to B, the run-time JCL for program B consists only of DD statements. In this situation, you need to create your own program JCL. Any one of the following can serve as a starting point for the JCL:

- The run-time JCL for another main program that accesses relational databases.

You can avoid the modification just described if you include an SQL process option in the initial main program.

If you get a JCL error for the run-time JCL, check the generation listing for the programs involved for any error messages related to JCL generation. In addition, ensure the tailoring that was done for the JCL templates is correct. Also check any changes you made when you customized the sample run-time JCL.

Chapter 14. Preparing and Running Generated Programs in IMS/VS and IMS BMP

This chapter describes the steps required to prepare and run a generated COBOL program in an IMS environment:

- Modify the IMS system definition parameters
- Create the MFS control blocks
- Precompile, compile, link, and bind the generated program
- Make the new modules and MFS control blocks available to IMS
- Create or modify runtime JCL (IMS BMP only)

For general information on preparing programs for the runtime environment, see Chapter 9, “Output of Program Generation on z/OS Systems,” on page 65. For information about modifying JCL, see Chapter 13, “Creating or Modifying Run-time JCL on z/OS Systems,” on page 93.

Modifying the IMS System Definition Parameters

The following information describes the basic IMS system definition parameters that are required to prepare EGL-generated programs. You should review the performance options described in the IMS documentation for your system to determine the most effective options.

An IMS TRANSACT macro is required for each transaction code used to start a Enterprise Developer generated main program in the IMS/VS environment and for each transaction-oriented BMP program. This includes the following transactions:

- Started from a clear IMS screen
- Used as a **sysVar.transactionID**
- Used as the target of a **transfer** statement of the form *transfer to a transaction* or **sysLib.startTransaction()** statement
- Transferred to by a non-Enterprise Developer program
- Started as the result of an **add** statement that adds a transaction to a message queue
- Started by other IMS facilities

The TRANSACT macro must follow the APPLCTN macro for the IMS PSB that is to be used for the transaction.

Defining an Interactive Program

Each main transaction program must be defined as either an IMS message processing program (MPP) or a fast-path program with an associated transaction code, except when the program is started through a **transfer** statement of the form *transfer to a program* from another program.

Figure 18 on page 98 shows the system definition parameters that are required for defining an interactive Enterprise Developer program.

APPLCTN	PGMTYPE=TP,PSB=ims-psb-name.	1
TRANSACT	CODE=trancode,	X2
	INQUIRY=NO,	X3
	MODE=SNGL,	X
	MSGTYPE=(SNGLSEG,RESPONSE),	X4
	EDIT=ULC,	X5
	SPA=(size,[DASD CORE],[FIXED])	6

Figure 18. IMS System Definition for an Interactive Transaction

- 1 The IMS PSB name and the Enterprise Developer program name must match.
- 2 Multiple transactions can be associated with one program. If the program changes the value of **sysVar.transactionID** before a CONVERSE, include a TRANSACT macro for the original transaction code and a TRANSACT macro for the EZESEGTR value.
- 3 INQUIRY=NO is the default for IMS. If DL/I is used for the work database, INQUIRY=NO is required. The Enterprise Developer Server work database supports help maps and displays data again if an input error occurs, as well as the CONVERSE process option. Therefore, even if the program databases are inquiry only, INQUIRY=NO is necessary. If DB2 is used for the work database and the program's use of all DL/I databases is inquiry only, then INQUIRY=YES can be used.
- 4 SNGLSEG is required. Either RESPONSE or NONRESPONSE can be used with Enterprise Developer Server, depending on whether you want the keyboard to remain locked until the transaction completes. Even if NONRESPONSE mode is used, multiple simultaneous transactions from a single terminal are not supported.
- 5 Required for input in lowercase.
- 6 Include this parameter only if an IMS scratch pad area (SPA) is required. The SPA size is the length of the IMS SPA header (14 bytes) plus the length of the longest working storage record that might be received or sent during a **transfer** statement of the form *transfer to a transaction*. However, if you include the **spaStatusBytePosition** and omit the **spaADF** build descriptor options, then you must add an additional byte when calculating the size. The SPA size must match the number specified for the **spaSize** build descriptor option when the program is generated.

You can also include the FPATH=YES parameter on the TRANSACT macro if the program might be run in an IMS Fast Path (IFP) region. If you include FPATH=YES, be sure to include the **imsFastPath="YES"** build descriptor option when you generate the program. Refer to the IMS manuals for your system for additional information about using IFP regions.

Defining Parameters for a Batch Program as an MPP

A Enterprise Developer batch program can also run as an asynchronous MPP. For example, a Enterprise Developer batch program can be used to process the information inserted to the message queue by a **sysLib.startTransaction()** statement or an **add** statement in another program. This type of program differs from one that runs as a BMP in that the MPP cannot access any GSAM, indexed, or relative files, and cannot include any special restart logic. Figure 19 on page 99 shows the system definition parameters required for this case.

APPLCTN	PGMTYPE=TP,PSB=ims-psb-name		1
TRANSACT	CODE=trancode,	X	2
	MODE=SNGL		

Figure 19. IMS System Definition for an Asynchronous MPP Program

- 1 The IMS PSB name and the Enterprise Developer program name must match.
- 2 Multiple transactions can be associated with one program.

You can also include the FPATH=YES parameter on the TRANSACT macro if the program might be run in an IMS Fast Path (IFP) region. If you include FPATH=YES, be sure to include the **imsFastPath="YES"** build descriptor option when you generate the program. Refer to the IMS manuals for your system for additional information about using IFP regions.

Defining Parameters for a Batch-Oriented BMP Program

If a Enterprise Developer batch program is generated to run as an IMS BMP program and it does not process an input message queue, it is a batch-oriented BMP program. Figure 20 shows the system definition parameters required for defining a main batch program as a batch-oriented BMP program.

APPLCTN	PGMTYPE=BATCH,PSB=ims-psb-name
---------	--------------------------------

Figure 20. IMS System Definition for a Main Batch Program Running as a Batch-Oriented BMP Program

Defining Parameters for a Transaction-Oriented BMP Program

If a Enterprise Developer batch program is generated to run as an IMS BMP program and it processes an input message queue created by MPP programs or by other BMP programs, it is a transaction-oriented BMP program. Figure 21 shows the system definition parameters that are required to define a main batch program as a transaction-oriented BMP program.

APPLCTN	PGMTYPE=BATCH,PSB=ims-psb-name		
TRANSACT	CODE=trancode,	X	1
	MODE=SNGL,	X	
	WFI		2

Figure 21. IMS System Definition for a Main Batch Program Running as a Transaction-Oriented BMP Program

- 1 Multiple transactions can be associated with one program.
- 2 Wait-for-input (WFI) is optional. If it is specified, the program remains resident until the operator stops the transaction or shuts down the region.

Creating MFS Control Blocks

The message format services (MFS) control blocks are generated when the formGroup is generated for the IMS environment. The build script FDAMFS is used. FDAMFS has functionality similar to that of the MFSUTL and the MFSTEST JCL procedures that ship with the IMS product. When you generate the program, you specify the **mfsUseTestLibrary** build descriptor option to choose between the functionality of MFSUTL and MFSTEST. YES indicates MFSTEST.

When you set **mfsUseTestLibrary** to YES, the variable MFSTEST is set to YES in the build plan. The build script FDAMFS uses this variable to determine which of

the JCL procedures (MFSUTL or MFSTEST) to follow. Refer to the message format services documentation for your system for additional information about the MFS control blocks. Refer to the online helps for more information about the build descriptor options that control what is included in the MFS source.

If your program contains DBCS or mixed data, note that a long mixed constant field that results in multiple lines of MFS source might contain unpaired shift-in and shift-out characters. This occurs when the DBCS portion of the constant is split into more than one line. The MFS still works correctly.

Making New Modules Available in the IMS Environment

Whenever you install a new version of a program, MFS print services program, map group format module, or table, you need to recycle the message region.

If you generated with `mfsUseTestLibrary="YES"`, then the MFS control blocks were placed in the MFS test library (the TFORMAT library). To use the new version of the MFS control blocks, use the `/TEST MFS` command after you have signed on your IMS system and before you attempt to run a transaction that uses the new version of the maps.

If you generated with `mfsUseTestLibrary="NO"`, then the MFS control blocks were placed in the MFS staging library (FORMAT library). To use the new version of the MFS control blocks, you must do the following:

1. Run the IMS online change utility (OLCUTL) to copy the new MFS control blocks into the inactive format library.
2. Use the following IMS commands:

```
/MODIFY PREPARE FMTLIB  
/MODIFY COMMIT
```

Note: If the MFS control blocks and the map group format module do not have the same generation date and time, Enterprise Developer Server issues an error message.

Preloading Program, Print Services, and Table Modules

Preloading programs, MFS print services programs, map group format modules, and table modules that are frequently used might reduce the overhead of searching the STEPLIB, JOBLIB, link pack area, and link list. However, if modules are preloaded, they occupy virtual storage when they are not in use.

To improve response time, you might also preload modules associated with any transaction that might require better performance, even though the module itself is not frequently used.

To preload a program, MFS print services program, map group format module, or table program, have your system administrator do the following:

1. Put the module in a LNKLIB library.
2. Include the module name in a preload member (DFSMPLEX, where xx is a two-character ID that you select) in IMSVS.PROCLIB.
3. Indicate in the JCL for the IMS message region that the preload member is to be included.

Refer to the IMS manuals for your system to get general information on preloading modules.

Running Programs under IMS

Prior to starting a generated program, the program user might be required to sign on to the IMS environment with a `/SIGN` command. Refer to the IMS documentation for information about the `/SIGN` command.

Starting a Main Program Directly

The simplest way for a program user to start a Enterprise Developer program is by entering the IMS transaction code from an unformatted screen. The transaction code can be up to 8 characters. It is associated with the program in the IMS system definition `TRANSACT` macro. The following is an example of starting a transaction:

```
MYTRANS
```

IMS requires the transaction code to be followed by at least one blank prior to pressing the `ENTER` key.

Starting a Main Transaction Program Using the `/FORMAT` Command

A program user can use the IMS `/FORMAT` command to display a formatted screen to start a transaction if the First Map for a program is defined with the IMS transaction code for the program as an 8-byte constant with the protect and dark attributes. The attribute byte on the map becomes the attribute byte in the generated MFS. The 8-byte constant contains the name of the IMS transaction that is started when the map is processed.

The `/FORMAT` command directs IMS to display a screen format; however, the command does not cause the program to be run. After the program user enters data and presses the `Enter` key (or a function key), the message from the terminal is sent to the generated program for processing.

The syntax of the `/FORMAT` command is as follows:

```
/FORMAT modname [mapname]
      or
/FOR modname [mapname]
```

The *modname* operand is the map group name with an *O* suffix. The *mapname* operand is required if there is more than one map in the map group. It must be the map name that was specified as the First Map for the program.

Because the transaction code must be included in the map, and a transaction code can only be associated with one program in the IMS system definition, only one program using the map can be started using the `/FORMAT` command.

Running Transaction Programs as IMS MPPs

Running generated programs is similar to running non-EGL-generated programs in the IMS MPP environment, with the following differences:

IMS Commands

The `/HOLD` command should be avoided. Enterprise Developer Server uses the logical terminal identifier as the key of the work database. The data in the work database is destroyed if another generated program is run from the same terminal prior to resuming the original conversation.

Keyboard Key Operation

When the Clear key is pressed in IMS, IMS clears the screen, but does not notify the program. No transaction is scheduled, so the map is not automatically displayed again. If the program is conversational, the program user can enter the IMS /HOLD command followed immediately by an IMS /RELEASE command to display the map again.

When the EOF key is pressed in the first position of a field on a map, the data is not blanked. To blank the data, the program user must enter at least one blank before pressing the EOF key. Also, the program user should not use the DELETE CHARACTER key to erase the entire field because this is equivalent to pressing the EOF key in the first position of the field.

When typing over characters in a right-justified numeric field, any intervening spaces between the new digits entered and the original digits in the field should be deleted by pressing the DELETE CHARACTER key. Alternatively, the program user can type in all digits for the new value and use the EOF key to erase any remaining digits.

DBCS Data on a Non-DBCS Terminal

If a program inadvertently attempts to display a map with DBCS or mixed data on a non-DBCS terminal or printer, the results are unpredictable. The terminal might be logged off IMS and returned to the VTAM[®]* sign-on screen without displaying any warning or error messages. If this happens, review your use of DBCS. Also, review your values for the **mfsDevice**, **mfsExtendedAttr**, and **mfsIgnore** generation options, and compare them to the IMS system definition for the terminal that had the problem.

Error Reporting

In certain error situations, Enterprise Developer Server displays its own panel to explain the error to the program user. This occurs in the following situations:

- A message needs to be displayed, but the form field named in the form property **msgField** does not exist. Form ELAM01 in formGroup ELAxxx, where xxx is the national language code, is used.
- An unexpected program error has occurred. Form ELAM02 and (if necessary) continuation form ELAM03 are used to display the error messages. See “Using the Enterprise Developer Server Error Panel” on page 133 for an example of ELAM02.

If an error occurs information might have been written to the message queue identified by the **errorDestination** build descriptor option for the first program in the run unit. See Chapter 19, “IMS Diagnostic Message Print Utility,” on page 125 for information on printing diagnostic errors.

Responding to IMS Error Messages

If a DFS[™] message is displayed on your screen, make a note of the message. Then, depending on how your IMS system is set up, press either PA1 or PA2 to see if Enterprise Developer Server has queued an error map to the terminal with more information. This can happen in the following situations:

- If Enterprise Developer Server issues a ROLL call because of a run unit or catastrophic error, IMS issues the message:

```
DFS555I TRAN tttttttt ABEND S000,U0778 ; MSG IN PROCESS:
ttttttt mmmmmmmmmMAP                ;;;;gdate gtime rdate rtime
```

Where *ttttttt* is the IMS transaction code, *mmmmmmmmmm* is the map name, *gdate* and *gtime* are the date and time the map group was generated, and *rdate* and *rtime* are the date and time of the abend.

The DFS555I message is also used by IMS when other abends occur, including the 1600, 1601, 1602, and 1606 abends from Enterprise Developer Server.

- If Enterprise Developer Server ends the run unit for a transaction program that was generated with **imsFastPath="YES"** and is being run in an IMS fast-path region, IMS issues the message:

DFS2766I PROCESS FAILED

- If Enterprise Developer Server abnormally ends the logical unit of work (LUW) for a transaction program that was generated with **imsFastPath="YES"**, IMS might issue the message:

DFS2082I RESPONSE MODE TRANSACTION TERMINATED WITHOUT REPLY

See Chapter 20, "Diagnosing Problems for Enterprise Developer Server on z/OS Systems," on page 129 for information on diagnosing errors.

Running Batch Programs as MPPs

A Enterprise Developer main batch program can be generated to run in the IMS MPP environment. In this situation, IMS automatically starts the transaction whenever a message is written to the message queue associated with the transaction.

If an error occurs information might have been written to the message queue identified by the **errorDestination** build descriptor option for the first program in the run unit. See Chapter 19, "IMS Diagnostic Message Print Utility," on page 125 for information on printing diagnostic errors.

Running a Main Program under IMS BMP

A main batch program generated for the IMS BMP environment can be started by submitting JCL. Called programs can only be started by another Enterprise Developer program or by a non-Enterprise Developer program.

The EGL COBOL generation process creates sample runtime JCL for running programs in the IMS BMP environment. The generated JCL has the same name as the program. If you set the **genRunFile** build descriptor option to "YES", a sample JCL is created specifically for the program during program generation. The runtime JCL is transferred to a z/OS partitioned data set (PDS) by the Enterprise Developer prepare function.

The JCL might need to be modified to add data sets required by called or transferred-to programs, or for data sets used by setting **sysVar.resourceAssociation** or **converseVar.printerAssociation**. You also need to modify the JCL to add any data sets that are dynamically allocated with the EZEDEST or EZEDESTP special function words. See Chapter 13, "Creating or Modifying Run-time JCL on z/OS Systems," on page 93 for more information on modifying the sample runtime JCL.

If you get a JCL error for the runtime JCL, check the generation listing for the programs involved for any error messages related to JCL generation. In addition, ensure the tailoring that was done for the JCL templates is correct. Also check any changes you made when you customized the sample runtime JCL.

The following sections show JCL for different IMS BMP programs.

Examples of Runtime JCL for IMS BMP Programs

The generated JCL in the following examples has these characteristics:

- The examples are based on the JCL templates shipped with Enterprise Developer. Your actual JCL templates might differ if your system administrator has tailored them for your organization. Refer to the online helps for more information about tailoring JCL templates.
- Lowercase text appears in the examples where a generic example name has been substituted for an actual program or data set name.
- EZEPRINT is always routed to SYSOUT=*.

If you route EZEPRINT to a data set, you must use the following DCB attributes:

- LRECL=137, BLKSIZE=141, RECFM=VBA if the map group does not contain any DBCS maps
- LRECL=654, BLKSIZE=658, RECFM=VBA if the map group contains any DBCS maps

You cannot use form groups that do not have any DBCS forms with form groups that do have DBCS forms in a single job step.

The first library in the STEPLIB concatenation sequence must have the largest block size, or BLKSIZE=32760 can be specified on the first STEPLIB DD statement for the step.

Running a Main Batch Program as an IMS BMP Program

If a main batch program runs as a BMP program, all DL/I requests are passed to a central copy of IMS which coordinates updates to the databases across multiple BMPs and MPPs. The DD statements for the IMS log and the program databases are not required in the JCL for the BMP job step. These databases and the IMS log are allocated to the IMS control region.

Figure 22 shows a sample set of JCL to run a generated program as a BMP program.

```
//jobname JOB .....
//stepname EXEC IMSBATCH,
//      MBR=appl-name,PSB=ims-psb-name,IN=trans-name
//G.STEPLIB DD
//      DD
//      DD DSN=CEE.SCEERUN,DISP=SHR
//      DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//      DD DSN=cghlq.env.LOAD,DISP=SHR
//G.ELAPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330)
//G.ELASNAP DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.EZEPRINT DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.SYSABOUT DD SYSOUT=*
//G.SYSOUT DD SYSOUT=*
/* Application specific DD statements for files
/* No application specific DD statements for databases required
//file-name-1 DD .....
//file-name-n DD .....
```

Figure 22. JCL for Main Batch Program as an IMS BMP Program

If you run a transaction-oriented BMP program, the *trans-name* must be set to the name of the transaction for the message queue that the BMP program processes. If

not, *trans-name* should be a null value. The sample runtime JCL created by Enterprise Developer defaults *trans-name* to the program name for a transaction-oriented BMP program that uses SCAN to read the message queue. The sample runtime JCL created by Enterprise Developer defaults *trans-name* to null for batch-oriented BMP programs or for transaction-oriented BMP programs that use VGLib.VGTDLI(), AIBTDLI(), or EGLTDLI() to read the message queue.

If the BMP program uses GSAM, the following DD statements are also included in the sample runtime JCL:

```
//IMS      DD DSN=IMS.PSBLIB,DISP=SHR
//          DD DSN=IMS.DBDLIB,DISP=SHR
```

These DD statements are generated from the FDA2MIMS JCL template.

Running a Main Batch Program as an IMS BMP Program with DB2 Access

Figure 23 shows a sample set of JCL to run a generated program that accesses DB2 resources as a BMP. The DD statements for the IMS log and the DL/I program databases are not required in the JCL for the BMP job step. The DL/I databases and the IMS log are allocated to the IMS control region.

```
//jobname JOB .....
//stepname EXEC IMSBATCH,
//      MBR=appl-name,PSB=ims-psb-name,IN=trans-name
//G.STEPLIB DD
//          DD
//          DD DSN=DSN.SDSNLOAD,DISP=SHR
//          DD DSN=CEE.SCEERUN,DISP=SHR
//          DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR
//          DD DSN=cghlq.env.LOAD,DISP=SHR
//G.DFSESL DD DSN=IMS.RESLIB,DISP=SHR
//          DD DSN=DSN.SDSNLOAD,DISP=SHR
//G.ELAPRINT DD SYSOUT=*,DCB=(RECFM=FBA,BLKSIZE=1330)
//G.ELASAP DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.EZEPRINT DD SYSOUT=*,DCB=(RECFM=VBA,BLKSIZE=4096)
//G.SYSABOUT DD SYSOUT=*
//G.SYSOUT DD SYSOUT=*
//* Application specific DD statements for files
//* No application specific DD statements for databases required
//file-name-1 DD .....
//file-name-n DD .....
```

Figure 23. JCL for Main Batch Program as an IMS BMP Program with DB2

If you run a transaction-oriented BMP program, the *trans-name* must be set to the name of the transaction for the message queue that the BMP program processes. If not, *trans-name* should be a null value. The sample runtime JCL created by Enterprise Developer defaults *trans-name* to the program name for a transaction-oriented BMP program that uses SCAN to read the message queue. The sample runtime JCL created by Enterprise Developer defaults *trans-name* to null for batch-oriented BMP programs or for transaction-oriented BMP programs that use CSPTDLI to read the message queue.

If the BMP program uses GSAM, the following DD statements are also included in the sample runtime JCL:

```
//IMS      DD DSN=IMS.PSBLIB,DISP=SHR
//          DD DSN=IMS.DBDLIB,DISP=SHR
```

These DD statements are generated from the FDA2MIMS JCL template.

Recovery and Restart for IMS BMP Programs

You should develop recovery procedures in the event of program or system error. Enterprise Developer does not generate JCL to perform restart or recovery procedures.

Chapter 15. Moving Prepared Programs to Other Systems from z/OS Systems

You might need to move a prepared program from one system to another. For example you might have the compiler on one host development machine but want to run the program on several production machines.

If you use DB2, the DB2 BIND must be done on the production system.

The COBOL and Enterprise Developer Server products on the production machine must be at the same maintenance level as, or a higher level than, on the development machine.

Moving Prepared Programs To Another z/OS System

If a program has been completely prepared on one system and you want to move the prepared program to another system, perform the following steps:

1. Copy the program-related parts (including the form group and table parts) to the production system. The names of the source libraries are shown with the default naming convention used in the build scripts, where *cghlq* is the user or project-related high level qualifier and *env* is the run-time environment code.

Table 13. Parts to Copy

Data Set Name	Contents
cghlq.env.LOAD	Application module, print services program, form group format modules, and data table modules.
cghlq.env.DBRMLIB	DB2 database request modules (DBRMs) for SQL programs
cghlq.env.EZEBIND	BIND commands for SQL programs

Note:

The *cghlq* variable comes from the projectID build descriptor option. The *env* variable comes from the system build descriptor option.

2. Provide your own JCL to build the plans for DB2 programs using the BIND commands from the BIND library and the DBRMs from the DBRM library. You need to edit the EZEBIND member, and make the appropriate changes such as DB2 subsystem name or collection IDs to match the new system where you are moving the program.
3. Follow the procedures identified in this manual for defining programs to CICS.
4. Define files and databases used by the program on the new system.

Maintaining Backup Copies of Production Libraries

Follow your installation-defined guidelines and procedures for making backup copies of production libraries. Having backup copies of production libraries enables you to return to the prior level of a program in case of errors. The production libraries for which copies should be made are those listed in Table 13.

Part 4. Utilities

Chapter 16. Using Enterprise Developer Server

Utilities on z/OS Systems.	111
Using the CICS Utilities Menu.	111
New Copy	112
Diagnostic Message Printing Utility	114

Chapter 17. Diagnostic Control Options.

Change or View Diagnostic Control Options for a Transaction	118
Change or View Default Diagnostic Control Options	119

Chapter 18. Using the Parameter Group Utility

121

Chapter 19. IMS Diagnostic Message Print

Utility.	125
-----------------	-----


Chapter 16. Using Enterprise Developer Server Utilities on z/OS Systems

Enterprise Developer Server provides a set of utilities in CICS to help manage the error diagnosis and control facilities of the Enterprise Developer Server run-time environment. You can access these utilities from the CICS utilities menu.

Using the CICS Utilities Menu

To access the CICS utilities do the following:

1. Log on to CICS.
2. Type ELAM on a clear screen.
3. Press Enter. When the ELAM transaction is started, a copyright panel is displayed.
4. Press Enter. The CICS Utilities Menu (Figure 24) is displayed.



```
ELAM                               Enterprise Developer Server
                                   CICS Utilities Menu

Select one of the following utilities; then press Enter.

Action...._

_1. New Copy
_2. Diagnostic Message Printing
_3. Diagnostic Control Options

ENTER  F1=HELP  F3=EXIT
```

Figure 24. CICS Utilities Menu

Three functions are available from the CICS Utilities Menu panel (Figure 24):

New Copy

This function causes a new copy of a program, form group, or data table to be used by subsequent transactions. Use the new copy function when programs, form groups, and data tables are modified and generated again.

For programs and form groups, you can also use the Enterprise Developer Server new copy utility or the CICS NEWCOPY command to cause the new copy of the program to be used the next time a load request is issued for the program.

The Enterprise Developer Server new copy utility does a new copy for both the online print services program and the form group format module

when you specify a part type of form group. If you use the CICS NEWCOPY command for a form group, you must issue the NEWCOPY for both the online print services program and the form group format module.

For data table, you must use the Enterprise Developer Server new copy utility to cause a fresh copy of the data table to be used the next time a load request is issued for the data table. Do not use the CICS NEWCOPY command for data tables.

Diagnostic Message Printing

This function routes the diagnostic messages in an error destination transient data queue to a spool file for printing or subsequent processing.

Diagnostic Control Options

This function lets you view or change the diagnostic control options set for the installation or for individual transactions. The options include dump control, error message routing to a transient data queue or the CICS journal, and transaction disabling when serious problems occur.

New Copy

The Enterprise Developer Server new copy utility causes a new copy of a program, form group, or data table to be used by subsequent transactions. Transactions that are in progress when this function was started continue to use the copy that was current when the transaction began. Programs must end before the new copy is used.

Programs must end or reach the end of a segment before the new copy is used. The Enterprise Developer Server new copy utility must be run separately for programs, form groups, and data tables to replace the copy already in storage.

To gain access to the Enterprise Developer Server new copy utility, do the following:

1. Select option 1, New Copy, on the CICS Utilities Menu panel (Figure 24).
2. Press Enter.

The New Copy panel (Figure 25 on page 113) is displayed.

Note: You can also gain access to the Enterprise Developer Server new copy utility by doing the following:

1. Type ELAN on a clear screen.
2. Press Enter. When the ELAN transaction is started, a copyright panel is displayed.
3. Press Enter. The New Copy panel (Figure 25 on page 113) is displayed.

ELAN	Enterprise Developer Server New Copy
Type choices; then press Enter.	
Part name.....	_____
Part type.....	_
1. Program 2. Map Group 3. Table	
ENTER F1=HELP F3=EXIT	

Figure 25. New Copy panel

Enter the following on the New Copy panel:

Part name

Specifies the name of the program, form group, or data table to be used as a new copy in subsequent transactions

Part type

Specifies the type of part to be replaced

Note: Enterprise Developer Server does not validate the part type. You must specify the correct type because different processing is required for programs, form groups, and data tables. If you have problems in processing after using the Enterprise Developer Server new copy utility, try the Enterprise Developer Server new copy utility again to ensure you specified the part type correctly.

The correct type can be one of the following:

Program

This type causes the utility to issue a CICS SET PROGRAM(name) NEWCOPY command to access a new copy of the program. This command does not cause a new copy for called programs that are statically linked with their caller.

Form Group

This type causes the utility to issue a CICS SET PROGRAM(name) NEWCOPY command to access a new copy of the form group format module and the online print services program associated with the form group.

Data Table

This type causes the utility to issue a CICS SET PROGRAM(name) NEWCOPY command to access a new copy of the data-table program and sets a flag for Enterprise Developer Server, indicating that a new copy of the data table is to be used the next time a program loads the data-table contents.

If the data table has been generated as a shared data table, currently running transactions continue to use the old copy of the data table while new transactions share the new copy of the data table.

You can also access the new copy utility in batch mode. To invoke the batch new copy utility, link to program ELABNEW:

```
EXEC CICS LINK PROGRAM("ELABNEW")
COMMAREA(passed-parms)
LENGTH(174)
```

where the **passed-parms** record has the following structure:

Field	Length in Bytes	Type of Data	Description
NLS code	3	Character	NLS code identifying language
Part name	8	Character	Name of program, form group, or data table to be used as a new copy in subsequent transactions
Part type	1	Character	Type of part to be replaced: "1" Program "2" Form group "3" Data table For more information, see the description for part type in the online help discussion of the new copy utility.
Return code	2	Binary	Return code from new copy
Message 1	80	Character	Message returned from new copy
Message 2	80	Character	Message returned from new copy

The following fields must be provided by the user:

- NLS code
- Part name
- Part type

The other fields are filled in by the new copy utility.

Any nonzero return code means that the new copy operation was not successful. If a nonzero value is returned in the **return code** field, check messages 1 and 2 for details indicating what error occurred.

Note: Message 2 is not always filled in. It may be blank.

Diagnostic Message Printing Utility

Diagnostic message printing allows you to route diagnostic messages in an error destination transient data queue to a JES spool file for printing.

To gain access to the diagnostic message print utility do the following:

1. Select option 2, Diagnostic Message Printing, from the CICS Utilities Menu panel (Figure 24 on page 111).

2. Press Enter.

The Diagnostic Message Printing panel (Figure 26) is displayed.

Note: You can also access the diagnostic message print function by doing the following:

1. Type ELAU on a clear screen.
2. Press Enter. When ELAU is started, a copyright panel is displayed.
3. Press Enter. The Diagnostic Message Printing panel (Figure 26) is displayed.

```
ELAU                               Enterprise Developer Server
                                   Diagnostic Message Printing

Fill in the appropriate fields; then press Enter.
Error destination queue name.....ELAD

JES Spool File Information
Node.....*
Userid.....*
Class.....A

Clear destination queue.....Y Y=Yes, N=No

ENTER  F1=HELP  F3=EXIT
```

Figure 26. Diagnostic Message Printing panel

You can enter information in the following fields on the Diagnostic Message Printing panel:

Error destination queue name

This field specifies the name of an existing error destination.

Enter the 1 to 4 character DCT name of the ERRDEST error destination transient data queue. The queue name is initialized to the default error destination queue. The default is ELAD. You can either leave the messages in the queue or clear them after they have been printed.

JES Spool File Information

This field specifies the spool file where the messages are to be written. If you do not specify anything in these fields, the system uses the default values and sends the report to the local spool printer where your local CICS is running.

Clear destination queue

This field specifies whether to clear the error queue of all messages after the messages are written to a spool file. The default is Y.

Chapter 17. Diagnostic Control Options

The diagnostic control options utility enables you to alter the diagnostic action options taken for a given transaction code that is assigned to a generated CICS program. If multiple transaction codes are assigned to a program, each transaction code is specified independently to the diagnostic control options utility.

You can also specify a default action to take place for transactions that are not explicitly defined to the diagnostic control options utility.

To gain access to the diagnostic control options utility, do the following:

1. Select option 3, Diagnostic Control Options, from the CICS Utilities Menu (Figure 24 on page 111).
2. Press Enter. The Diagnostic Control Options panel (Figure 27) is displayed.

Note: You can also gain access to the diagnostic control options utility by doing the following:

1. Type ELAC on a clear screen.
2. Press Enter. When ELAC is started, a copyright panel is displayed.
3. Press Enter. The Diagnostic Control Options panel (Figure 27) is displayed.

```
ELAC01                Enterprise Developer Server
                        Diagnostic Control Options

Select one of the following actions; then press Enter.

Action.....1
  1. Change or View the Diagnostic Control Options for a Transaction
  2. Change or View the Default Diagnostic Control Options

ENTER  F1=HELP  F3=EXIT
```

Figure 27. Diagnostic Control Options panel

You can gain access to the following functions from the Diagnostic Control Options panel:

Change or View the Diagnostic Control Options for a Transaction

This option enables you to change or view the diagnostic options for a specific transaction code.

Change or View the Default Diagnostic Control Options

This option enables you to change or view the installation default diagnostic options.

This affects transaction codes that are not specifically identified to the diagnostic controller.

Change or View Diagnostic Control Options for a Transaction

This function enables you to change the Enterprise Developer Server error diagnostic and control options in effect for a specific CICS transaction.

To start the function do the following:

1. Select option 1, Change or View the Diagnostic Control Options for a Transaction, from the Diagnostic Control Options panel (Figure 27 on page 117).
2. Press Enter. The Change or View Diagnostic Control Options for a Transaction panel (Figure 28) is displayed.

ELAC02

Enterprise Developer Server

Change or View Diagnostic Control Options for a Transaction

Fill in the appropriate fields; then press Enter.

Transaction ID..... ____

Diagnostic Control Options

Transaction ABEND Dump

Runtime Error Dump

Error Destination Queue Name... ____

Journal Number..... ____

Journal Record Identifier..... ____

Disable on Run Unit Failure.... ____

Action..... 3

1. No Dump

2. Complete CICS dump

3. Task dump

1. No Dump

2. Complete CICS dump

3. Task dump

blank,00-99

Y=Yes, N=No

1. Change diagnostic control options

2. Use default control options

3. View diagnostic control options

ENTER F1=HELP F3=EXIT

Figure 28. Change or View Diagnostic Control Options for a Transaction panel

The following fields can be entered on the Change or View Diagnostic Control Options for a Transaction panel :

Transaction ID

Specifies the 1 to 4 character identifier of the transaction you want to change the diagnostic options for

Diagnostic Control Options

Transaction ABEND Dump

Specifies the type of dump taken on a CICS transaction ABEND

The types of dumps are:

1. No Dump
2. Complete CICS dump
3. Task dump

Runtime Error Dump

Specifies the type of dump taken on a Enterprise Developer Server-detected error for which a dump is indicated in the error message explanation

The types of dumps are:

1. No Dump
2. Complete CICS dump
3. Task dump

Error Destination Queue Name

Specifies the 1 to 4 character name of a transient data queue to which Enterprise Developer Server error diagnostic messages are written whenever a transaction ends abnormally due to an error

If this field is blank, no messages are written to a queue.

Journal Number

Specifies the journal number of the CICS journal to which error diagnostic messages are written whenever a transaction is not successful due to an error

If this field is blank, no journal messages are written.

Journal Record Identifier

Specifies the 1 to 2 character record identifier used when messages are written to the CICS journal

If this field is blank, the default identifier EZ is used.

Disable on Run Unit Failure

Specifies whether a transaction is disabled whenever an error is detected that is likely to occur each time the transaction is run

- | | |
|----------|---|
| Y | Specifies that the transaction is disabled when these errors are detected |
| N | Specifies that the transaction is not be disabled |

Action

Allows you to change the current options, view the current options, or accept the default options

To change the options currently set for a transaction do the following:

1. Specify the transaction identifier and any changes.
2. Select 1, Change diagnostic control options.
3. Press Enter.

To use the installation defaults for the transaction do the following:

1. Type the transaction identifier.
2. Select 2, Use default control options.
3. Press Enter.

To view the options currently set for a transaction do the following:

1. Type the transaction identifier.
2. Select 3, View diagnostic control options.
3. Press Enter.

Change or View Default Diagnostic Control Options

This function enables you to change or view the default diagnostic options for transactions that are not identified to the diagnostic controller. If your default options were not modified at installation, the default diagnostic options are set as follows:

- Transaction ABEND and run-time errors both cause a task dump.
- The error destination queue name is ELAD.
- Diagnostic messages are not written to a CICS journal data set.
- Transactions are not disabled on a run unit error.

To start this function do the following:

1. Select 2, Change or View the Default Diagnostic Control Options, from the Diagnostic Control Options panel (Figure 27 on page 117).
2. Press Enter. The Change or View Default Diagnostic Control Options panel is displayed:

ELAC04

Enterprise Developer Server

Change or View Default Diagnostic Control Options

Fill in the appropriate fields; then press Enter.

Default Diagnostic Control Options

Transaction ABEND Dump 3

Runtime Error Dump 3

Error Destination Queue Name... ELAD

Journal Number.....

Journal Record Identifier..... EZ

Disable on Run Unit Failure.... N

1. No Dump

2. Complete CICS dump

3. Task dump

1. No Dump

2. Complete CICS dump

3. Task dump

blank,00-99

Y=Yes, N=No

ENTER F1=HELP F3=EXIT

Figure 29. Change or View Default Diagnostic Control Options

The options on this panel are the same as those defined for changing or viewing the diagnostic control options for a transaction. They are all defined following Figure 28 on page 118.

Chapter 18. Using the Parameter Group Utility

Use the parameter group utility to create and maintain the parameter groups in the parameter group file. Each group contains parameters for controlling terminal printer utility (FZETPRT) transactions.

See “Special Parameter Group for the FZETPRT Program” on page 31 for a description of the startup parameters that can be included in the parameter group used with the FZETPRT program.

You can use the parameter group utility to perform the following operations:

- Display the contents of existing parameter groups
- View a list of existing parameter group names
- Add a new parameter group
- Change a parameter group
- Delete a parameter group

Table 14 shows the steps used to define a parameter group file.

Table 14. Defining Parameter Group Files for z/OS CICS

Procedure

1. Define the parameter group file using the IDCAMS utility.

```
DEFINE CLUSTER (NAME(PARM.GROUP.FILE)-  
RECORDS(100 100) KEYS(16 0) RECORDSIZE(272 272) INDEXED)
```
2. Initialize the parameter group file by using the IDCAMS REPRO function to insert a dummy record into the file.
3. Specify the FCT for the parameter group file utility to have access to a user-defined message file for CICS.

```
DFHFCT TYPE=DATASET, C  
DATASET=EZEPRMG, C  
ACCMETH=VSAM, C  
SERVREQ=(READ,UPDATE,ADD,DELETE,BROWSE), C  
FILESTAT=(ENABLED,CLOSED), C  
RECFORM=FIXED C
```
4. Allocate the file by adding the following statement to the z/OS CICS startup JCL:

```
//EZEPRMG DD DISP=SHR,DSN=PARM.GROUP.FILE
```

Note: The name that designates the parameter group file (EZEPRMG) is a reserved file name and cannot be used as a data file by an EGL-generated program.

When the file has been created and allocated, you can gain access the parameter group utility by doing the following:

1. Log on to CICS.
2. Type ELAP on a clear screen.
3. Press Enter.

The parameter group utility does not give message-specific tutorial help after a message is displayed and PF1 is pressed.

Once the parameter group utility has been started, the Parameter Group Specification panel (Figure 30) is displayed. You can specify the parameter group name on this panel.

```
PRGM00                                PARAMETER GROUP UTILITY
                                     ENTER = Continue    PF1 = Help    PF3 = Exit
                                     ..... PARAMETER GROUP SPECIFICATION .....
                                     Specify Parameter Group Name =>
```

Figure 30. Parameter Group Specification panel

The parameter group name can be from 1 to 4 alphanumeric characters and must be the name of the transaction that was used to start the FZETPRT program. (The utility does not verify this.)

You can enter a group name that already exists if you want to modify a parameter group, or you can enter one that does not exist if you want to define a new parameter group.

Entering a question mark (?) as the group name on the Parameter Group Specification panel displays a list of previously-defined group names on the next panel, the Parameter Group List Display panel (Figure 31). Entering some characters followed by an asterisk (*) displays a list of parameter group names that begin with the characters that you entered. Entering a specific parameter group name displays the Parameter Group Definition panel (Figure 32 on page 123).

```

PRGM01                                PARAMETER GROUP UTILITY

                                ENTER = Continue  PF3 = Exit   PF4 = Refresh  PF1 = Help
                                PF7 = Back        PF8 = Forward

..... PARAMETER GROUP LIST DISPLAY .....

_____ EZEP          _____ USRQ

```

Figure 31. Parameter Group List Display panel

From the Parameter Group List Display panel, you can select a group name to edit by typing an S in the selection field to the left of the group name. You can delete a group by typing a D in the selection field to the left of the group name.

If the specified parameter group already exists, its contents are displayed on the Parameter Group Definition panel. The parameter group can be altered. If the specified parameter group does not exist, the Parameter Group Definition panel is displayed without any data. You can define the new contents; up to 256 characters of data can be entered for a parameter group.

```

PRGM02                                PARAMETER GROUP UTILITY

                                PA2 = Cancel  PF1 = Help  PF3 = File and Exit
                                Parameter Group = CCCCCCCC

.....PARAMETER GROUP DEFINITION.....

Parameter Group:

=>PRTBUF=2048 PRTMPP=132 PRTTYP=D FORMFD=NO

```

Figure 32. Parameter Group Definition panel

The parameter group utility does not validate or format the parameters that are specified on the Parameter Group Definition panel. Any parameters that are not valid are ignored when the FZETPRT program is started.

If you press PF3 on the Parameter Group Definition panel without entering any parameters, a parameter group is stored without any associated parameters. You can store an empty parameter group to reserve parameter group names.

Empty parameter groups do not affect the initialization of the FZETPRT program.

The parameter group utility left-justifies the parameter group name and pads it to the right with blanks (X'40'). The parameter group utility uses this name as a key to index the parameter group file.

If you selected a parameter group from the Parameter Group List Display panel (Figure 31 on page 123), after the Parameter Group Definition panel is processed, the Parameter Group List Display panel is displayed again with the original request replaced by an asterisk beside the group name that was processed. An asterisk (*) is ignored as input on the Parameter Group Definition panel if more processing is done.

Chapter 19. IMS Diagnostic Message Print Utility

When a generated program ends abnormally due to an error condition in IMS environments, diagnostic error messages are written to the message queue identified by the **errorDestination** build descriptor option for the first program in the run unit.

A BMP is provided to print the messages in the message queue. The JCL needed to print the diagnostic information is supplied as member ELAMQJUD of ELA.V1R2M0.ELAJCL (see Figure 33).

The msg-queue identified by the IN parameter is the name of the queue that was specified for **errorDestination** when the program was generated.

```
//*****00000100
//** ELAMQJUD - JCL TO DRAIN AND PRINT THE ELADIAG MESSAGE QUEUE      00000200
//**          FOR VISUALAGE GENERATOR SERVER.                        00000300
//**          THIS PROGRAM RUNS AS A BMP.                            00000400
//**                                                                00000500
//** LICENSED MATERIALS - PROPERTY OF IBM                            00000600
//** 5648-B02 (C) COPYRIGHT IBM CORP. 1994, 1998                    00000700
//** SEE COPYRIGHT INSTRUCTIONS                                     00000800
//**                                                                00000900
//** STATUS = VERSION 1, RELEASE 2, LEVEL 0                         00001000
//**                                                                00001100
//** TO TAILOR THIS JOBSTREAM:                                       00001200
//**      1. COPY A JOBCARD.                                         00001300
//**      2. CHANGE IN= TO THE NAME OF YOUR ERROR DIAGNOSTIC        00001400
//**          QUEUE.                                                 00001500
//**      3. MAKE SURE THAT THE TRANSACTION SPECIFIED BY IN=        00001600
//**          AND THE ELAMPUTL PROGRAM ARE STARTED BY IMS.           00001700
//**                                                                00001800
//** RETURN CODES                                                    00001900
//**      0 - SUCCESSFUL COMPLETION                                  00002000
//**      4 - NO MESSAGES ON QUEUE TO DRAIN.                         00002100
//**      16 - FATAL ERROR. PROCESSING TERMINATED                   00002200
//**      20 - OPEN FAILED ON ELAPRINT                               00002300
//**                                                                00002400
//*****00002500
//DRAINMQ      EXEC IMSBATCH,MBR=ELAEPUTL,                            00002600
//              PSB=ELAMPUTL,IN=ELADIAG,RGN=4096K                    00002700
//G.STEPLIB    DD                                                    00002800
//              DD                                                    00002900
//              DD DSN=CEE.SCEERUN,DISP=SHR                           00003000
//              DD DSN=ELA.V1R2M0.SELALMD,DISP=SHR                   00003100
//G.ELAPRINT   DD SYSOUT=*                                           00003200
//G.SYSOUT     DD SYSOUT=*                                           00003300
//G.SYSPRINT   DD SYSOUT=*                                           00003400
/*                                                                00003500
```

Figure 33. ELAMQJUD

Part 5. Diagnosing Problems

Chapter 20. Diagnosing Problems for Enterprise

Developer Server on z/OS Systems 129

Detecting Errors 129

File and Database Errors—Category 1 129

File and Database Errors—Category 2 130

File and Database Errors—Category 3 130

Reporting Errors 131

Controlling Error Reporting in CICS. 131

Controlling Error Reporting in IMS

Environments 131

Controlling Error Reporting in z/OS Batch . . . 132

Error Reporting Summary 132

Transaction Error 132

Run Unit Error 132

Catastrophic error 133

Enterprise Developer Server Error 133

Using the Enterprise Developer Server Error

Panel 133

Printing Diagnostic Information for IMS 134

ERRDEST Message Queue 134

IMS Log Format 135

Running the Diagnostic Print Utility. 136

Printing Diagnostic Information for CICS 136

CICS Diagnostic Message Layout. 136

Running the Diagnostic Print Utility. 137

Analyzing Errors Detected while Running a

Program 138

Chapter 21. Finding Information in Dumps. 139

Enterprise Developer Server ABEND Dumps. . . 139

COBOL or Subsystem ABEND Dumps 139

Information in the Enterprise Developer Server

Control Block 140

Information in an Application. 140

How to Find the Current Position in a Program at

Time of Error 141

Chapter 22. Enterprise Developer Server Trace

Facility 143

Enabling Enterprise Developer Program

Source-Level Tracing with Build Descriptor

Options 143

Activating a Trace 144

Activating a Trace Session for CICS 144

Activating a Trace Session for z/OS Batch. . . 147

Deactivating a Trace Session 149

Printing Trace Output 149

Printing the Trace Output in CICS 149

Printing the Trace Output in z/OS Batch . . . 149

Reporting Problems for Enterprise Developer

Server 149

Chapter 23. Common Messages during

Preparation for z/OS Systems 151

Common Abend Codes during Preparation . . . 151

DB2 Precompiler and Bind Messages 151

COBOL Compilation Messages 151

Chapter 24. Common System Return Codes for

z/OS Systems 153

Common SQL Return Codes 153

Common DL/I Status Codes 155

Common VSAM Status Codes. 155

OPEN request type 156

CLOSE request type 156

GET/PUT/POINT/ERASE/CHECK/ENDREQ

request types 156

COBOL Status Key Values 157

Chapter 25. Enterprise Developer Server Return

Codes and Abend Codes for z/OS Systems . . . 159

Return Codes 159

ABEND Codes 159

CICS Environments 159

IMS, IMS BMP, and z/OS Batch Environments 161

z/OS Batch 162

Chapter 26. Codes from Other Products for

z/OS Systems. 163

Common System Abend Codes for All

Environments 163

LE Run-time Messages 164

COBOL Run-time Messages 164

Common COBOL Abend Codes 165

Common IMS Runtime Messages. 165

Common IMS Runtime Abend Codes 166

Common CICS Run-time Messages 167

Common CICS Abend Codes 167

COBOL Abends under CICS 168

Chapter 20. Diagnosing Problems for Enterprise Developer Server on z/OS Systems

The chapter contains diagnosis, modification, or tuning information. Use this information to determine the source of the problem you encountered. Some common program definition, database, and system errors that might cause problems are described. This chapter also explains how to obtain error listings and diagnose run-time errors.

Detecting Errors

You can find most logic errors by using the EGL debugger before you generate your program.

During generation, a validation step checks your program for any remaining syntax errors. In addition, Enterprise Developer also checks that your use of language elements is consistent with the resource association information you select for each file.

When you run your generated program, different types of errors are detected by Enterprise Developer Server, COBOL, CICS, or z/OS. The error handling varies depending on which product detects the error.

For diagnostic information of interest at development time, see the EGL help system.

For those errors detected by Enterprise Developer Server that result in a Run Unit Error, error messages are written to the transient data queue specified through the diagnostic control options. You can print those messages by using the diagnostic printing utility (see “Diagnostic Message Printing Utility” on page 114) or by using CICS utilities (for example, CEBR).

For more information, see Chapter 17, “Diagnostic Control Options,” on page 117.

File and Database Errors—Category 1

A hard error occurred in a file or database I/O operation. For example:

- I/O error on the file or database
- File not found

Table 15. Types of Errors

EGL COBOL Set Controls	Error Detected By	Error Handling
Use VGVar.handleHardIOErrors to indicate whether the program provides logic to override normal database and file error processing for hard errors. Hard errors are defined in the online helps within Enterprise Developer	Access method or database manager. The access method or database manager return an error code to the generated logic or Enterprise Developer Server.	<p>If VGVar.handleHardIOErrors is set to 0 and a hard error occurs, the program ends with a transaction error, as described in Table 18 on page 132. For SQL errors, the DB2 messages describing the SQL error code are obtained from DB2. Refer to the DB2 messages manual for your system for explanations of the error codes.</p> <p>If VGVar.handleHardIOErrors is set to 1 and a hard error occurs, the program is responsible for controlling the error handling.</p>

File and Database Errors—Category 2

A file or database error occurs due to incorrect program logic. For example, deleting or replacing a record without first doing an UPDATE process option is an example of this error type.

Table 16. Types of Errors

EGL COBOL Set Controls	Error Detected By	Error Handling
Follow the file management rules in the Enterprise Developer help facility.	Enterprise Developer Server.	The program ends with a run unit error, as described in Table 18 on page 132.

File and Database Errors—Category 3

Two generated programs which are being used together are defined in an inconsistent manner. For example, two programs have been generated with different file definitions.

Table 17. Types of Errors

EGL COBOL Set Controls	Error Detected By	Error Handling
Follow the rules in the EGL help facility.	COBOL or Enterprise Developer Server.	<p>If detected by Enterprise Developer Server, the program ends with a run unit error, as described in Table 18 on page 132.</p> <p>If detected by COBOL, a formatted dump is generated.</p>

Reporting Errors

Enterprise Developer Server provides functions that help you determine the cause of a run-time problem. All run-time errors that Enterprise Developer Server traps are accompanied by error messages and supporting information to help diagnose the problem. Table 18 on page 132 shows the error diagnostic actions that can be taken based on the severity of the error and type of system being used.

Controlling Error Reporting in CICS

In the CICS environment, error actions are controlled through the online diagnostic controller utility installed as transaction ELAC.

The utility allows you to specify what type of dump is requested, the name of the transient data queue to which Enterprise Developer Server diagnostic messages are written, the CICS journal number and identifier for error messages, and whether or not a transaction is disabled when a run unit error is detected. The utility lets you reset the default options for all transactions and override the default options for individual transactions.

See Chapter 17, “Diagnostic Control Options,” on page 117 for more details about the diagnostic controller utility.

Controlling Error Reporting in IMS Environments

The following error responses are controlled by build descriptor options for the IMS/VS and IMS BMP environments:

- Write error messages to the error destination. The destination is determined by the **errorDestination** build descriptor option.
- Write error messages to the system log. The log ID is determined by the **imsLogID** build descriptor option. If the **imsLogID** option does not appear in the build descriptor file, error messages will not be written to the system log.
- Put the message that caused the problem for transaction-oriented IMS BMP programs back on the message queue. **restoreCurrentMsgOnError="YES"** indicates that the message being processed when the error occurred should be placed back on the message queue before the program ends. **restoreCurrentMsgOnError="NO"** indicates that the message being processed should be deleted and not placed back on the message queue. This option is applicable only to a run unit error when Enterprise Developer Server detects the error. It does not apply to transaction-oriented BMPs that use **VGLib.VGTDLI()** to read the message queue.
- Issue ROLL call or abend for a run unit error. **imsFastPath="NO"** results in a ROLL call. **imsFastPath="YES"** results in a 1602 abend.

The actions controlled by the runtime JCL are as follows:

- Print message. This is done only if there is an ELAPRINT DD statement in the runtime JCL.
- Snap dump. If the message indicates a snap dump is taken, the snap dump is produced only if there is an ELASNAP DD statement in the runtime JCL.
- Abend 1602 or 1600. This creates a dump only if the runtime JCL contains a SYSUDUMP or SYSABEND DD statement.

Abend code 1602 is the user code issued by Enterprise Developer Server when it ends the run unit for an **imsFastPath="YES"** program because of an error.

Abend code 1600 is the user code issued by Enterprise Developer Server in all other situations when it ends program processing because of an unrecoverable error.

IMS takes the following actions, based on the way Enterprise Developer Server ends the program:

- If a rollback (ROLB) call is issued, the database changes are backed out, the logical unit of work ends, the next message is read from the message queue, and processing continues.
- If a ROLL call is issued, the database changes are backed out, the logical unit of work ends, and IMS stops the program with a user 778 abend. The transaction and PSB are not stopped and can be scheduled again without operator intervention.
- If either a 1600 or a 1602 abend is issued, the database changes are backed out, the logical unit of work ends, and IMS stops the program. The transaction and PSB are also stopped, and they require operator intervention to start them again.

Use ELASNAP so that sufficient data is captured the first time an error occurs.

Controlling Error Reporting in z/OS Batch

The actions controlled by the run-time JCL are as follows:

- Print message. This is done only if there is an ELAPRINT DD statement in the runtime JCL.
- Snap dump. If the message indicates a snap dump is taken, the snap dump is produced only if there is an ELASNAP DD statement in the runtime JCL.
- Abend 1600. This creates a dump only if the run-time JCL contains a SYSUDUMP or SYSABEND DD statement.

Error Reporting Summary

The following tables summarize the error processing actions for Enterprise Developer Server.

Transaction Error

This error affects only the current CICS task. The transaction is still available to other end users.

Table 18. Error Processing Actions For Enterprise Developer Server Detected Errors

Environment	Action
CICS	<ul style="list-style-type: none">• Write error messages to error destination (diagnostic controller option)• Write error messages to CICS journal data set (diagnostic controller option)• CICS dump, dump code ELAD, as determined by message. The type of dump issued for a particular transaction is a diagnostic control option.• Issue a rollback request• Display error messages on terminal, if possible• Set return code to 693
z/OS Batch	See run unit error

Run Unit Error

The error is likely to occur for every user. In CICS, the transaction might be disabled.

Table 19. Error Processing Actions For Enterprise Developer Server Detected Errors

Environment	Action
CICS	<ul style="list-style-type: none"> • Write error messages to error destination (diagnostic control option), if possible • Write error messages to CICS journal data set (diagnostic control option), if possible • Disable transaction (diagnostic control option) • CICS dump, dump code ELAD, as determined by message. The type of dump issued for a particular transaction is a diagnostic control option. • Issue a rollback request • Display error messages on terminal, if possible • Set return code to 693 • Return
z/OS Batch	<ul style="list-style-type: none"> • Print message (ELAPRINT DD statement) • Snap dump determined by the message (ELASNAP DD statement) • Issue a rollback request if DB2 databases were used • Set return code to 693 • Return

Catastrophic error

This error indicates storage is corrupted or standard error reporting processing ends abnormally.

Table 20. Error Processing Actions For Enterprise Developer Server Detected Errors

Environment	Action
CICS	<ul style="list-style-type: none"> • Write error messages to error destination (diagnostic control option), if possible • Write error messages to CICS journal data set (diagnostic control option), if possible • Disable transaction (diagnostic control option) • Display error messages on terminal, if possible • ABEND ELAE. The type of dump issued for a particular transaction is a diagnostic control option.
z/OS Batch	<ul style="list-style-type: none"> • Print message (ELAPRINT DD statement), if possible • Abend 1600 (SYSUDUMP or SYSABEND, DD statement)

Enterprise Developer Server Error

A Enterprise Developer Server error at a point where standard error reporting process is not active.

Table 21. Error Processing Actions For Enterprise Developer Server Detected Errors

Environment	Action
All environments	<ul style="list-style-type: none"> • Abend, ABEND code indicates the reason for the error

See Table 25 on page 139 for information concerning the contents of the registers when either a 1600, 1602, or an ELAE abend occurs.

Using the Enterprise Developer Server Error Panel

When an error occurs, Enterprise Developer Server attempts to display error messages on the current terminal. The panels used in displaying error messages are defined as map group ELAxxx where xxx is the language code.

The following figure shows the error panel (map ELAM02 in the map group) as it is shipped with the product. The panel shows the same diagnostic information that is written to the error destination queue, system log or journal, or ELAPRINT file. If there are more error messages than can fit on a single panel, the last line on the panel prompts the user to press a key to display additional error messages.

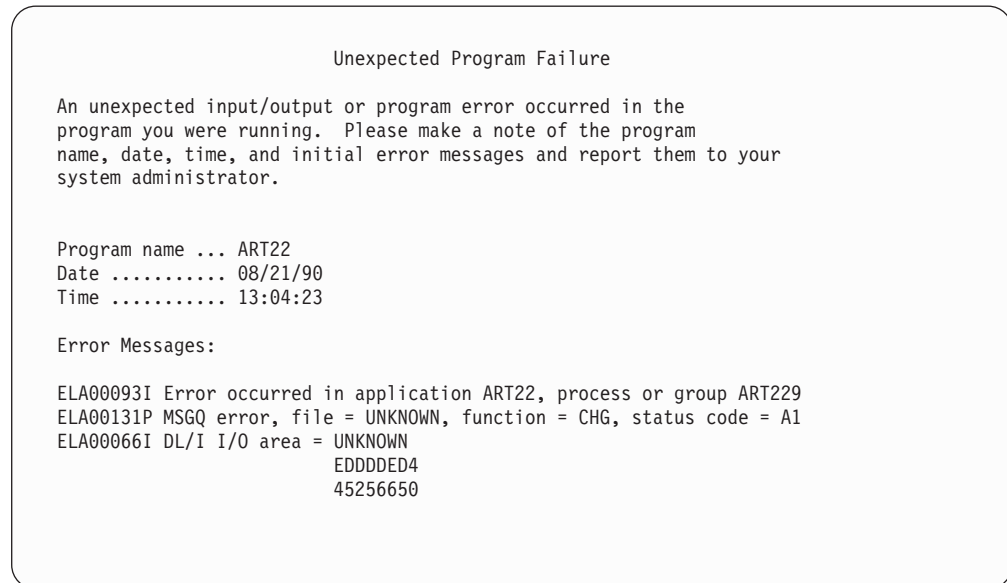


Figure 34. Panel ELAM02 (example).

Printing Diagnostic Information for IMS

Diagnostic messages are sent either to a print file for batch jobs or to a message queue for BMPs or online transactions. A diagnostic utility is provided to print messages written to a message queue. Optionally, based on the **imsLogID** build descriptor option, the diagnostic information can be written to the IMS log.

ERRDEST Message Queue

Table 22 shows the format of the information in the IMS message queue when ERRDEST is used.

Table 22. ERRDEST IMS Message Queue

Field	Length in Bytes	Type of Data	Description
Record length	2	Binary	The length of the record.
Reserved	2	Binary	A reserved field that must contain binary zeros.
IMS transaction code	8	Character	The name used to identify the IMS message queue that was specified with the errorDestination build descriptor option.
Date	8	Character	Date of the transaction from the I/O PCB (MM/DD/YY).
Time	8	Character	Time of the transaction from the I/O PCB (HH:MM:SS).

Table 22. ERRDEST IMS Message Queue (continued)

Field	Length in Bytes	Type of Data	Description
NLS	3		The value specified for the targetNLS build descriptor option
Message number	9		<p>The message number:</p> <p>Bytes 1-3 Message File Identifier (ELA)</p> <p>Byte 4 Application Identifier (0)</p> <p>Bytes 5-8 Message Number</p> <p>Byte 9 Message Type. A message type of 'C' indicates this record is a continuation of the specified message from a previous record in the queue.</p>
Message number separator (reserved position)			Byte 10 Blank
Message Text	Variable	Character	The text from the message file with specified message inserts.

IMS Log Format

Table 23 shows the format of the information in the IMS log.

Table 23. IMS Log Record

Field	Length in Bytes	Type of Data	Description
Record length	2	Binary	The length of the record.
Reserved	2	Binary	A reserved field that must contain binary zeros.
Log ID	1	Character	The value specified with the imsLogID build descriptor option.
Date	8	Character	Date of the transaction from the I/O PCB (MM/DD/YY).
Time	8	Character	Time of the transaction from the I/O PCB (HH:MM:SS).
NLS	3		The value specified for the targetNLS build descriptor option

Table 23. IMS Log Record (continued)

Field	Length in Bytes	Type of Data	Description
Message number	9		The message number: Bytes 1-3 Message File Identifier (ELA) Byte 4 Application Identifier (0) Bytes 5-8 Message Number Byte 9 Message Type. A message type of 'C' indicates this record is a continuation of the specified message from a previous record in the queue.
Message number separator (reserved position)			Byte 10 Blank
Message Text	Variable	Character	The text from the message file with specified message inserts.

Running the Diagnostic Print Utility

A BMP is provided to print diagnostic information that is written to the message queue specified by the **errorDestination** build descriptor option. The JCL needed to print the diagnostic information is supplied as member ELAMQJUD of 'ELA.V1R2M0.ELAJCL.

The msg-queue identified by the IN parameter is the name of the queue that was specified in the **errorDestination** option when the application was generated. See "Diagnostic Message Printing Utility" on page 114 for more information.

Printing Diagnostic Information for CICS

Diagnostic messages are sent to a transient data queue for CICS transactions. A diagnostic print utility is provided to print messages written to a transient data queue. Optionally, as specified by the diagnostic controller utility, the diagnostic information can also be written to an CICS journal data set.

CICS Diagnostic Message Layout

Table 24 shows the format of the information in each error message record written to a transient data queue or CICS journal.

Table 24. Diagnostic Message Layout

Field	Length in Bytes	Type of Data	Description
SYSID name	4	Character	The name of the CICS system that the error message was created on.

Table 24. Diagnostic Message Layout (continued)

Field	Length in Bytes	Type of Data	Description
TRANID name	4	Character	The name of the CICS transaction code that started the logical unit-of-work.
Task identifier	8	Character	The task identifier assigned by CICS to each transaction instance that is processed. This number is reset to 0 when CICS is cold-started. This is taken from EIB field EIBTASKN.
ERRDEST name	4	Character	The name of the CICS transient data queue. This field is blank if the record is written to the CICS journal.
Date	8	Character	Date of the transaction (MM/DD/YY)
Time	8	Character	Time of the transaction (HH:MM:SS)
NLS	3	Character	The value specified for the targetNLS build descriptor option
Message number	9	Character	The message number: Bytes 1-3 Message File Identifier (ELA) Byte 4 Application Identifier (0) Bytes 5-8 Message Number Byte 9 Message Type. A message type of 'C' indicates this record is a continuation of the specified message from a previous record in the queue.
Message number separator (reserved position)			Byte 10 Blank
Message text	110	Character	The text from the message file with specified message inserts

Running the Diagnostic Print Utility

Use the ELAU transaction to print the messages routed to a transient data queue. You enter the following information on the ELAU panel:

Error destination queue name

The name of the queue containing the messages. The default name is ELAD.

JES spool file information

The node, user ID, and class to which the messages on the queue are spooled. The default class is A and the default node and user ID are * which routes the printed messages to the local spool printer.

Clear queue

Y specifies the queue is deleted after its contents are printed. N specifies the messages are left in the queue after they are printed. Y is the default value.

See “Diagnostic Message Printing Utility” on page 114 for more information about running the diagnostic print utility.

Analyzing Errors Detected while Running a Program

Use the error messages and diagnostic messages to determine the cause of the problem. If the error is detected by another product (for example, COBOL), check the information in Chapter 24, “Common System Return Codes for z/OS Systems” and Chapter 26, “Codes from Other Products for z/OS Systems” and the documentation for the other product. the cause

For environmental debugging, you can use the run-time diagnostic facility (EDF) for CICS programs. In addition, if you use the TEST COBOL compile option, you can use the COBOL debugging facilities.

Refer to the CICS and COBOL manuals for your versions of these products for additional information on their debugging facilities.

If you get a JCL error for the run-time JCL, check the generation output for any programs involved for any error messages related to JCL generation. In addition, ensure the tailoring that was done on the JCL templates and EGL build scripts is correct. Also check any changes made to customize the sample runtime JCL.

When abends occur, the problem determination might require assistance from the IBM Support Center. In this case, be prepared to provide IBM with the following information:

- Enterprise Developer print of the problem program
- COBOL source file created using the commentLevel=1 build descriptor option.
- Formatted dump
- Enterprise Developer Server diagnostic information written to either the error diagnostic queue or listed in the printout
- CICS journal

IBM requests a COBOL debugger trace listing only if the information is needed for problem determination. IBM will give you the information on how to specify the trace options if the information is necessary.

Chapter 21. Finding Information in Dumps

Information about the problem program can be determined by finding the address of the Enterprise Developer Server control block in a dump.

Enterprise Developer Server ABEND Dumps

If the dump code is 1600, 1602, or ELAE, the dump was initiated because Enterprise Developer Server detected an error. Register 2 at ABEND points to the Enterprise Developer Server control block. Register 4 points to a linked list of messages formatted as shown in Figure 35.

Table 25. Registers when a SNAP dump is taken or a Enterprise Developer Server ABEND occurs.

Reg.	Value
2	Points to Enterprise Developer Server control block. At offset 272 (hexadecimal offset 110) from the start of the Enterprise Developer Server control block is the address of the initial program profile block, which provides information about the first generated program that was started. At offset 276 (hexadecimal offset 114) from the start of the Enterprise Developer Server control block is the address of the current program profile block, which provides information about the Enterprise Developer program that was running at the time of the abend.
4	Points to the message buffer that contains all messages.

The following diagram shows the format of the message buffer that contains all the messages in the dump.

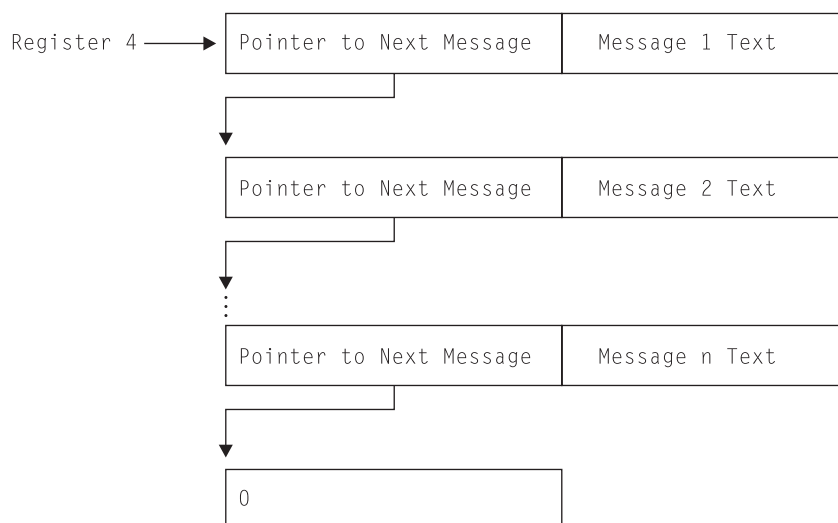


Figure 35. Message Buffer Format

COBOL or Subsystem ABEND Dumps

If the dump is not a Enterprise Developer Server abend, you can use the following method to locate the Enterprise Developer Server control block:

- On CICS systems, locate the CICS Task Work Area (TWA) in the dump. Locate the string *EZERTS-CONTROL* in the TWA. This string is the identifier at the start of the Enterprise Developer Server control block. The * and - characters might be converted to periods in a formatted dump.
- On other systems, locate the string ELARHAPP followed immediately by a program name. ELARHAPP is the identifier at the start of a program profile block. The four-byte address at hex offset 20 in the program profile is the Enterprise Developer Server control block address. If 0, the program might not yet be activated. Do a search for another ELARHAPP control block followed by a program name.

For information in the program profile control Block, see Table 27.

Information in the Enterprise Developer Server Control Block

The following information is in the Enterprise Developer Server control block:

Table 26. Information in the Enterprise Developer Server Control Block

Offset in hex	Length in bytes	Contents
0	16	Control block identifier - *EZERTS-CONTROL*
104	4	CICS EIB Pointer
110	4	Program profile address for current program
114	4	Program profile address for initial program
118	8	Terminal identifier
120	8	User identifier
128	8	Transaction identifier
150	12	DLILib.psbData
1CC	18	Current process or group

Information in an Application

Each generated COBOL program contains a profile control block in COBOL working storage initialized with information about the program. The first eight bytes contain an eye-catcher constant identifying whether the program is an application. The second eight bytes contain the program name. Other information in the profile block is shown in the following table:

Table 27. Locator Format for Generated COBOL Program Dumps

Offset in hex	Length in hex	Contents
00	08	Program type identifier: ELARHAPP — Application program
08	08	Program name
10	08	Program generation date (MM/DD/YY)
18	08	Program generation time (HH:MM:SS)
20	04	Enterprise Developer Server control block address
24	02	Generator version
26	02	Generator release
28	02	Generator modification level

Table 27. Locator Format for Generated COBOL Program Dumps (continued)

Offset in hex	Length in hex	Contents
2A	10	Reserved
34	08	Target run-time system

How to Find the Current Position in a Program at Time of Error

The Enterprise Developer Server control block identifies the currently running program and function at the time of the error (Table 26 on page 140). Associated error messages identify the Enterprise Developer statement number for errors detected by Enterprise Developer Server that need statement identification to resolve the problem. For performance reasons, the generated COBOL program does not keep track of the Enterprise Developer statement number for each generated statement. If a program exception occurs in a generated program, you can determine the Enterprise Developer statement number by finding the COBOL statement that was not successful in a COBOL program listing that contains the Enterprise Developer statements generated as comments.

Chapter 22. Enterprise Developer Server Trace Facility

The Enterprise Developer Server trace facility can be used by the IBM Support Center to aid in problem determination, or by the program user to trace program activity.

There are two levels of tracing available:

- Enterprise Developer program source-level tracing
- Enterprise Developer Server run-time level tracing

With source-level tracing, you can request traces of Enterprise Developer statements, traces of the data, and error codes after every SQL call in a program, except SQL calls made with the SQLEXEC process option. Source-level tracing is enabled with the use of the sqlIOTrace, sqlErrorTrace, and statementTrace build descriptor options. Source-level tracing is activated in the run-time environment by specifying trace filter criteria. See “Activating a Trace” on page 144 for more information on activating traces.

With run-time level tracing, you can request a data stream trace, a Enterprise Developer Server internal dump trace, or a service routine trace. Run-time level tracing does not require the use of a build descriptor option. Run-time level tracing is activated in run-time environment by specifying trace filter criteria. See “Activating a Trace” on page 144 for more information on activating traces.

Use these functions only with the assistance of the IBM Support Center. If you use these functions without assistance, large amounts of trace output might be produced based on trace option selection.

Enabling Enterprise Developer Program Source-Level Tracing with Build Descriptor Options

You must specify the sqlIOTrace, sqlErrorTrace, and statementTrace build descriptor options in order to get source-level trace output.

The Enterprise Developer preprocessor validates the build descriptor option and its parameters. Enterprise Developer creates the necessary COBOL code to accomplish the type of tracing that you request.

The trace build descriptor options are sqlIOTrace, sqlErrorTrace, and statementTrace. When using these options, you must specify a value of YES or NO. Each of these build descriptor options tells the COBOL generator whether or not to generate code to allow execution time tracing of a particular aspect of execution - SQL I/O, SQL Errors, and EGL statement execution path.

Note: These options are intended for the use of support personnel and should only be used when a trace is requested as part of a support effort. Normal application debugging should be done through the use of the EGL Debugger.

Activating a Trace

Tracing is activated during run time either by using the ELAZ transaction in the CICS environment.

Activating a Trace Session for CICS

A utility is supplied to activate tracing in the CICS environment. To start the utility, enter the utility transaction code, ELAZ. The utility transaction must start prior to running the transaction to be traced.

The ELAZ transaction must run in the same region as the transactions to be traced. In CICS, enter the ELAZ transaction and the transaction to be traced from terminals attached to the same CICS region.

Figure 36 shows the initial panel for the ELAZ transaction that enables you to specify which transactions are to be traced. You use a secondary panel to specify filter criteria for a specific transaction that control what information is traced for that transaction.

ELAZ01 Enterprise Developer Server
Trace Transaction Selection

Specify the transaction you want to trace; then press Enter.

To select specific programs and services for tracing, place the cursor on a transaction name and press F4.

Transaction codes or initial program names			
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____
_____	_____	_____	_____

ENTER F1=HELP F3=EXIT F4=FILTER F9=REFRESH F10=STOP TRACE

Figure 36. Enterprise Developer Server Trace Transaction Selection Panel

Enterprise Developer Server then presents the panel shown in Figure 37 on page 145 for trace filter selection:.

ELA02		Enterprise Developer Server	
		Trace Filter Selection	
Transaction code or Initial Program _____			
Fill in the appropriate fields, then press Enter.			
3270 Data Stream.....N		APP Statement Trace.....N	
Terminal ID.....N	_____	SQL/IO Trace.....N	
Trace to File.....N		SQL/ERR Trace.....N	
EDS IDUMP Trace.....N			
FILENAME	ELATOUT	NODE	* USERID EZEUSRID CLASS A FORM *
Programs			
_____	_____	_____	_____
Services			
_____	_____	_____	_____
_____	_____	_____	_____
ENTER F1=HELP F3=RETURN F9=REFRESH			

Figure 37. Enterprise Developer Server Trace Filter Selection Panel

The filter criteria include the following:

3270 Data Stream (Y or N)

Specifies whether to trace 3270 data streams

If yes (Y), the 3270 data streams built or received by Enterprise Developer Server are traced. The default is no (N).

Terminal ID

Specifies a terminal identifier

If specified, only transactions initiated from that terminal are traced. If not specified, service requests from any terminal are traced.

Trace to File (Y or N)

Specifies whether the trace output goes to a file

If yes (Y), the trace output of Enterprise Developer Server is sent to the ELAT transient data queue in CICS. This trace is also written to an in-storage wrap-around trace buffer.

If no (N) the trace output goes to an in-storage wrap-around trace buffer. The size of this trace buffer is defined during customization of Enterprise Developer Server.

Y must be specified if you specify Y (yes) for the SQL/IO Trace or SQL/ERR Trace options. All trace output for SQL/IO and error tracing is sent to a file, not to the in-storage wrap-around trace buffer.

EDS IDUMP Trace (Y or N)

Specifies whether to dump Enterprise Developer Server internal storage areas

If yes (Y), the trace facility provides dumps of certain Enterprise Developer Server internal storage areas. The default is no (N), no internal storage dumps.

APP Statement Trace (Y or N)

Specifies whether to trace Enterprise Developer statements in a program

If yes (Y), the trace facility provides the process name and the statement for each Enterprise Developer statement that the program processes. Specify statementTrace=YES to enable this type of tracing. The default is no (N).

SQL/IO Trace (Y or N)

Specifies whether to trace SQL/IO

If yes (Y), the trace facility provides traces of the data and error codes on the return from the SQL call. The Enterprise Developer SQL process name, the process option, the process object, the SQL function name, and the Enterprise Developer data item name, length, type, and contents are given. Specify statementTrace=YES to enable this type of tracing. The default is no (N).

SQL/ERR Trace (Y or N)

Specifies whether to trace SQL error information

If yes (Y), the trace facility provides traces of the error information that comes back from SQL on every database call. The SQLCODE, SQLERRP, SQLSTATE, SQLWARN, SQLERRD, SQLEXT, and SQLERRMC codes are given. Specify statementTrace=YES to enable this type of tracing. The default is no (N).

FILENAME

The system resource name for the trace output. The default is ELATOUT.

NODE

1 to 8 characters that specify the system node ID. The default is the current system node ID.

USERID

1 to 8 characters that specify the user ID. The default is the value of the EZEUSRID special function word.

CLASS

A single character that specifies the print class. The default is A.

FORM

1 to 4 characters that specify the form number for print output. The default is your location's standard form.

Applications

Specifies whether to limit the trace to certain application programs or print services programs

If specified, only the requested programs are traced.

Services

Specifies whether to limit the trace to certain services

If specified, only the requested services are traced. Otherwise all service numbers are traced if the other criteria are met.

Note: The entry to ELARSINI (initialization service) and the exit from ELARSTRM (cleanup service) are not traced. ELARSINI initializes the trace facility. ELARSTRM ends the trace facility.

If you are running a trace to aid in problem determination, enter the filter criteria as directed by the IBM support center.

Activating a Trace Session for z/OS Batch

Tracing is activated by providing trace filters in a preallocated data set with the DD name ELATRACE before starting the program or job to be traced. ELATRACE contains control statements which control the programs and events to be traced. The attributes for the data set are LRECL=80, DSORG=PS, and RECFM=FB. If the ELATRACE data set is empty or allocated as DD DUMMY, all services are traced, data streams are not traced, and SQL I/O, SQL errors, and program statement are not traced even if enabled through **sqlIOTrace**, **sqlErrorTrace**, or **statementTrace** build descriptor options. Figure 38 shows the correct syntax for the trace control statements.

```
:FILTER DATASTREAM=Y|N
:FILTER TRACETOFILE=Y|N
:FILTER APPSTMT=Y|N
:FILTER SQLIO=Y|N
:FILTER SQLERR=Y|N
:FILTER IDUMP=Y|N
:APPLS

:
[ name ]

:
: EAPPLS
: SERVICES

:
[ service number ]

:
: ESERVICES
: EFILTER
```

Figure 38. ELATRACE Data Set Entries

Notes:

1. Only one program name or service number can be entered on each line.
2. The :FILTER and :EFILTER tags are required if any other tags are included in the ELATRACE data set.
3. More than one filter can be specified on a line. The filters must be separated by 0 or more blanks. The example below shows sample :FILTER statements that are valid and equivalent:

```
:FILTER APPSTMT=Y
:FILTER SQLERR=Y

:FILTER APPSTMT=YSQLERR=Y

:FILTER APPSTMT=Y SQLERR=Y

:FILTER APPSTMT=Y          SQLERR=Y
```

The filters cannot be continued on the next line. The statement shown below is not valid:

```
:FILTER APPSTMT=Y SQLERR=
Y
```

The control card tags and attributes that control filtering have the following meaning:

:FILTER

Options controlling what information is traced and where trace data is written

The following attributes can be used with the :FILTER statement:

- DATASTREAM=Y|N

If DATASTREAM=Y is specified, the 3270 data streams built or received by Enterprise Developer Server are traced. The default value is N, no data stream tracing.

- TRACETOFILE=Y|N

If TRACETOFILE=Y is specified, the trace output is directed to the preallocated data set named ELATOUT in addition to being directed to an in-storage wrap-around trace buffer.

If TRACETOFILE=N is specified, the trace output goes to an in-storage wrap-around trace buffer. The size of this trace buffer is defined during customization of Enterprise Developer Server. The default for the TRACETOFILE option is N.

TRACETOFILE=Y must be specified if SQLIO=Y or SQLERR=Y are specified. All trace output for SQL I/O and SQL errors is directed to the ELATOUT data set, not to the in-storage wrap-around trace buffer.

- APPSTMT=Y|N

If APPSTMT=Y is specified, the trace facility provides the process name and the statement for each Enterprise Developer statement that the program processes. You must use the **statementTrace="YES"** build descriptor option to enable this type of tracing. The default for the APPSTMT option is N.

- SQLIO=Y|N

If SQLIO=Y is specified, the trace facility provides traces of the data and error codes on the return from the SQL call. The Enterprise Developer SQL process name, the process option, the process object, the SQL function name, and the Enterprise Developer data item name, length, type, and contents are given. You must use the **sqlIOTrace="YES"** build descriptor option to enable this type of tracing. The default for the SQLIO option is N.

- SQLERR=Y|N

If SQLERR=Y is specified, the trace facility provides traces of the error information that comes back from SQL on every database call. The SQLCODE, SQLERRP, SQLSTATE, SQLWARN, SQLERRD, SQLEXT, and SQLERRMC codes are given. You must use the **sqlErrorTrace="YES"** or the **sqlIOTrace="YES"** build descriptor option to enable this type of tracing. The default for the SQLERR option is N.

- IDUMP=Y|N

If IDUMP=Y is specified, the trace facility provides dumps of certain Enterprise Developer Server internal storage areas. The default for the IDUMP option is N, no internal storage dumps.

:APPLS

Application program names or print service program names

If program names are specified, only the specified programs are traced. Otherwise service requests from each generated program are traced. Up to 16 program names can be specified.

:SERVICES Service numbers

If service numbers are specified, only those specific services are traced. To trace all service numbers, 999 must be specified. Otherwise, up to 32 service numbers can be specified.

Note: The entry to ELARSINI (initialization service) and the exit from ELARSTRM (cleanup service) are not traced. ELARSINI initializes the trace facility. ELARSTRM ends the trace facility.

Deactivating a Trace Session

To stop all trace activity for CICS, use the ELAZ transaction to delete the transaction codes from the list of transactions to be traced by using the F10 function key. When a transaction ends and is subsequently restarted, tracing does not start if the transaction code no longer appears in the transaction list.

To stop tracing in z/OS batch, cancel the program and remove the ELATRACE and ELATOUT DD cards from the run-time JCL.

Printing Trace Output

If the trace output is not directed to a file for the CICS environment, trace output is written to a wrap-around trace buffer in memory. The trace output can be seen in dumps taken when programs end abnormally.

Printing the Trace Output in CICS

Trace output for CICS is routed to an extrapartition transient data queue which is directed to a data set named ELATOUT if you direct the trace output to a file by specifying yes (Y) on the ELAZ02 panel. The ELATOUT data set has the attributes of LRECL=133, RECFM=FBA. The file can be printed as directed on the DD statement for ELATOUT in the CICS startup JCL.

Printing the Trace Output in z/OS Batch

Trace output is directed to the ELATOUT DD statement and is printed as directed on the DD statement.

Reporting Problems for Enterprise Developer Server

For instructions on reporting problems, visit the following Web site, click **Support**, and click **Submit and track problems**:

<http://www.ibm.com/software/awdtools/studioenterprisedev>

Chapter 23. Common Messages during Preparation for z/OS Systems

This chapter contains some error messages from other products. It is not a complete list. For a complete explanation of product messages, refer to the documentation provided with that product.

Common Abend Codes during Preparation

Only the most frequently occurring preparation abend codes are listed in this section. If you receive any other abend code or need a more complete explanation of one of the abend codes, refer to the documentation for that product.

System B37

The temporary work space is filling up. The WSPC parameter is used in the build scripts to prepare generation outputs specifies the amount of temporary space allocated. The build script names used by default are fdac1, fdac1, and fdap1, depending on whether you are preparing a CICS server with no SQL access, or a CICS server with SQL access.

To solve the abend, use a symbolic descriptor option named WSPC and set it to a larger value.

System 213, or System 230

Two program developers tried to update the directory of a PDS at the same time. Submit the job again.

This problem can also be prevented by specifying ENQ=YES for the DD statement for the PDS for which the 213 occurred. However, this serializes preparation of servers when their preparation outputs are placed in the same PDS's.

DB2 Precompiler and Bind Messages

Only the most frequently occurring DB2 precompiler and bind messages are listed in this section. If you receive other messages that start with DSN or if you need a more complete explanation of one of the messages, refer to the documentation for your release of DB2.

DSNX100I BIND SQL WARNING

Explanation: One or more DB2 tables have not been created. The tables that do not exist will be identified in an explanation associated with the message by:

xxxxxxx IS NOT DEFINED
where xxxxxx is the table name.

User response: Create the necessary DB2 tables and synonyms.

COBOL Compilation Messages

Only the most frequently occurring COBOL compilation messages are listed in this section. If you receive other compilation messages that start with IGY or if you need a more complete explanation of one of the messages, refer to the documentation for your release of COBOL.

IGYPS2015I The paragraph or section prior to paragraph or section EZEMAIN-PROCESS did not contain any statements.

IGYPS2023I Paragraphs prior to section EZEMAIN-PROCESS were not contained in a section

Explanation: These two messages occur if your program has been processed by the DB2 precompiler.

User response: They are normal messages that you can ignore.

IGYOP3091W Code from "?" to "?" can never be executed, and was therefore discarded.

IGYOP3093W The "PERFORM" statement at "?" cannot reach its exit.

IGYOP3094W There may be a loop from the "PERFORM" statement at "?" to itself. "PERFORM" statement optimization was not attempted.

Explanation: These messages occur if your program has been processed using the OPTIMIZE compiler option.

User response: These are normal messages that you can ignore.

IGYPA3013W Data item "?" and "?" had overlapping storage. An overlapping move will occur at execution time.

Explanation: This message occurs if your program attempts to assign the value of a data item to the same data item.

User response: You might want to check that you really intended to do this.

IGYPG3113W Truncation of high-order digit positions may occur due to precision of intermediate results exceeding 30.

Explanation: This message might occur if your program was generated with the math=COBOL build descriptor option.

User response: You might want to change the arithmetic expression identified in the message. For example, you could split the expression into several smaller ones.

If you do not change the expression, ensure that the intermediate values will fall within the precision that COBOL supports. Refer to the programming guide for your release of COBOL for more information about the

precision of intermediate results.

IGYSC2025W "EZPCB-?" or one of its subordinates was referenced, but "EZPCB-?" was a "LINKAGE SECTION" item that did not have addressability. This reference will not be resolved successfully at execution.

Explanation: This warning message occurs when PCBs or any data structure is generated in the linkage section, but is not used in a statement.

User response: Ignore the messages and the program will work correctly.

Chapter 24. Common System Return Codes for z/OS Systems

The information within this chapter is diagnosis, modification, or tuning information.

Enterprise Developer Server messages might include return codes from databases or operating systems that are being used. This could include DB2, z/OS VSAM, or CICS EXEC Interface Block (EIB) codes.

This chapter contains only the most common errors that occur during file input and output operations.

The return codes included in this chapter are for the following databases and operating systems:

- CICS
- DB2
- VSAM
- COBOL

For details on values returned to the program, see the EGL help topic for the system variable `sysVar.returnValue`.

Common SQL Return Codes

Only the most frequently occurring SQL codes are listed in this section. If you receive other SQL codes or if you need a more complete explanation of one of the SQL codes, refer to the documentation for your release of DB2.

RC	Meaning
----	---------

100	No rows were found by SQL that meet the search criteria specified in the WHERE clause of the SQL statement, or if processing in conjunction with a setting or setupd statement, the end of the selected rows has been reached. The possible causes are the following:
-----	---

- | | |
|--|--|
| | <ul style="list-style-type: none">• The key value(s) were not moved correctly to the host variable(s) used in the WHERE clause.• No rows meet the search criteria specified in the WHERE clause.• Enterprise Developer stripped trailing blanks for the character host variables used in a LIKE predicate in the WHERE clause. You can use the <code>sqlIOTrace=YES</code> build descriptor option to enable tracing of the data sent to SQL and the data coming back from SQL. See Chapter 22, "Enterprise Developer Server Trace Facility," on page 143 for more information about using the Enterprise Developer Server trace facility. |
|--|--|

-301, -302, -303, -304	
------------------------	--

	The Enterprise Developer data item definition does not match the definition of the same column in the DB2 table. This can be caused by defining a column as variable length, but not defining the data item in Enterprise Developer with a variable-length SQL code. This can also be caused by specifying a different length to Enterprise Developer from what was defined in the DB2 table.
--	---

Make the necessary changes in the Enterprise Developer data item definition to match the DB2 table and generate the program again.

- 302 Refer to the program directory in the appendix of this manual for information about installing a DB2 work database.
- 805 The DBRM for the current program was not bound as part of the current DB2 plan. Possible causes are:
- The BIND process was never run for the program.
 - An incorrect plan name was specified at startup.
 - The plan name specified in the RCT on CICS did not match the plan name used in the BIND process.
 - All programs that run together under a single transaction or job step must be bound into the same DB2 plan.

Look at the message inserts to see what DB2 returned as the program name and plan name. If these are what you expect, review the steps used for preparing the program.

- 818 The DB2 precompiler-generated time stamp in the load module is different from the database request module (DBRM) used on the most recent BIND for the PLAN being used. Both of these precompiler outputs must match and one of them is not from the most recently-run precompile. This typically happens when the precompile, link-edit, and bind process is run more than once and either the DBRM library or the load library used for the load module is changed. This creates the opportunity to pick up the old load module at run time if the old load library is first in the search sequence at run time. Alternatively, the BIND process might be using an old DBRM library that contains an old copy of that member.

Ensure that you are running with the most recent copy of the load module and that you are using the same DBRM library on the precompile and BIND steps. On CICS ensure that the latest copy of the load module has been picked up by issuing an CICS NEWCOPY command or by using the Enterprise Developer Server new copy utility.

-911,-913

A deadlock condition occurred. Possible causes are:

- The isolation level was set for repeatable read.
- There were long periods of time between commit points.
- In EGL, the program issued a get statement with the hold option, but failed to issue a related replace statement. In VisualAge Generator, the program issued an update without a replace.

Note: The program should be coded to handle these conditions.

- 922 Connection authorization was not successful. The type of error is indicated in the SQL error message. Some typical causes are not granting authority for the DB2 plan or not creating a synonym for one or more of the DB2 tables.

Make the necessary changes to provide authorization to the DB2 plan and then run the program again. You might also want to refer to the documentation for your release of DB2 for additional causes of the authorization error.

Common DL/I Status Codes

Only the most frequently occurring DL/I status codes are listed in this section. If you receive other DL/I status codes or if you need a more complete explanation of one of the DL/I status codes, refer to the application programming manual for your release of IMS.

Code Meaning

- | | |
|-----------|---|
| AD | The function parameter on the call is not valid. If the function code is correct, the status code can be from an I/O or alternate PCB for a database call. You might have a mismatch between the PSB you defined in Enterprise Developer and the IMS PSB definition. |
| AI | A data management open error occurred. Either no DD statements were supplied for logically related databases, or the DD name is not the same as the name specified on the DATASET statement of the DBD. The segment name area in the DB PCB has the DD name of the data set that could not be opened. |
| AJ | The format of one of your SSAs is not valid. Either the SSA contains a command code for that call that is not valid, or the SSA uses an R, S, W, or M command code for a segment for which there are no subset pointers defined in the DBD. |
| AK | An SSA contains either a field name that is not valid or a name that is not defined in the DBD, or the Enterprise Developer data item for DL/I segment does not match the name defined to DL/I. |
| AM | The call function is not compatible with the processing option in the PCB, the segment sensitivity, the transaction-code definition, or the program type. |
| GA | In trying to satisfy an unqualified GN or GNP call, DL/I crossed a hierarchic boundary into a higher level. |
| GB | In trying to satisfy a GN, DL/I reached the end of the database. |
| GD | The program issued an ISRT that was not qualified for all levels above the level of the segment being inserted. The segment might have been deleted by a DLET using a different DB PCB. |
| GE | DL/I is unable to find a segment that satisfies the segment described in a get call. |
| GK | DL/I has returned a different segment type at the same hierarchic level for an unqualified GN or GNP. |
| GP | The program issued a GNP when parentage is not established, or the segment level specified in the GNP is not lower than the level of the established parent. |
| II | The program issued an ISRT that tried to insert a segment that already exists in the database. |

Common VSAM Status Codes

Only the most frequently occurring VSAM codes are listed in this section. If you receive other VSAM codes or if you need a more complete explanation of one of these values, refer to the *z/OS VSAM Administration: Macro Instruction Reference* manual.

OPEN request type

Code	Meaning
64	Warning message: OPEN encountered an empty alternate index that is part of an upgrade set.
74	This is a warning message indicating the data set was not properly closed. Either the implicit verify for the OPEN was unsuccessful, or the user specified that the implicit verify should not be attempted for the OPEN. A previous VSAM program might have ended abnormally. The VERIFY command of Access Method Services can be used to properly close the data set.
80	The DD statement for this access method control block is either missing or not valid.
94	Either no record for the data set to be opened was found in the available catalog or catalogs, or an unidentified error occurred while VSAM was searching the catalog.
98	Security verification was not successful; the password specified in the access method control block for a specified level of access does not match the password in the catalog for that level of access.
A0	The operands specified in the ACB or GENCB macro are inconsistent either with each other or with the information in the catalog record. You might have attempted to open an empty data set for input only (SCAN).
A8	The data set was not available for the type of processing you specified, or an attempt was made to open a reusable data set with the reset option while another user had the data set open.
BC	The data set indicated by the access method control block is not a valid type of data set for specification by an access method control block. You might have used a sequential data set as the physical file, but specified VSAM or VSAMRS as the file type for resource association when you generated the program.
C0	An unusable data set was opened for output.
C4	Access to data was requested using an empty path.

CLOSE request type

Code	Meaning
04	The data set indicated by the access method control block is already closed.
88	Not enough virtual storage was available in the address space of your program for the work area required by CLOSE.
94	An unidentified error occurred while VSAM was searching the catalog.

GET/PUT/POINT/ERASE/CHECK/ENDREQ request types

Note: The following occur when register 15=8(8).

Code	Meaning
08	An attempt is made to store a record with a duplicate key, or there is a duplicate record for an alternate index with the unique key option.

- 6C The RECLLEN specified was one of the following:
 - Larger than the maximum allowed
 - Equal to 0
 - Smaller than the sum of the length and the displacement of the key field
 - Not equal to the record(slot) size specified for a relative record data set
- 70 The KEYLEN specified was too large or equal to 0.
- C0 A relative record number that is not valid was encountered.

COBOL Status Key Values

This shows the most frequently occurring COBOL status key values. If you receive other status key values or if you need a more complete explanation for one of these values, refer to the application programming language reference for your release of COBOL.

Status Key	Explanation
10	The end of a file was reached.
22	An attempt was made to write a record with a key that duplicated one that was already in the file.
23	Record not found. This can also be caused by an optional file not being allocated.
35	No DD statement was included in the JCL. This can occur if the program calls another program or transfers to another program using a DXFR statement, but the DD statements for the second program have not been added to the sample runtime JCL for the main program.
39	The physical file that you specified during resource association does not match the file characteristics that you specified during record definition. The file characteristics include file organization (sequential, relative or indexed), the prime record key, the alternate record keys, and the maximum record size.
44	A variable-length record was written that is not valid. This can occur if the value in the number of occurrences field is larger than the maximum value, or the value in the record length item field is larger than the maximum length of the record.
96	No DD statement was included in the JCL for a VSAM file.

Chapter 25. Enterprise Developer Server Return Codes and Abend Codes for z/OS Systems

The information within this chapter is diagnosis, modification, or tuning information.

Only the most frequently occurring system abend codes are listed in this section. If you receive other abend codes or if you need a more complete explanation of one of the codes, refer to the z/OS messages and codes manual for your release of z/OS.

Return Codes

This section contains a listing of codes set by Enterprise Developer Server and returned in the COBOL return code of a program.

If a generated program completes normally, the COBOL return code is set to the value in the **sysVar.returnCode**. This code must be less than or equal to 512. Return codes greater than 512 are reserved for Enterprise Developer Server. The return codes set by Enterprise Developer Server are:

- 693** The program ended due to an error detected by Enterprise Developer Server. The error description is reported as described in Chapter 20, "Diagnosing Problems for Enterprise Developer Server on z/OS Systems."
- 4093** A program generated using Enterprise Developer Server Version 1.2 ended due to an error detected by Enterprise Developer Server.

If LE detects an error and returns to the operating system, the LE return code modifier (2000 - error, 3000 - severe error, or 4000 - critical error) is added to the user or Enterprise Developer Server return code.

ABEND Codes

Enterprise Developer Server reports errors by error messages whenever possible. ABENDs are issued only in situations where initialization has not progressed to the point where messages can be issued or when the error messages cannot be written to their normal destination.

CICS Environments

For CICS, you can control whether or not a core dump is taken by using the diagnostic controller utility. If a core dump is taken, the dump code is ELAD. See "Controlling Error Reporting in CICS" on page 131 for information on the diagnostic controller utility.

- ELA1** This abend code should never be received. However, if register 1 in a dump contains "ELA1", then a database manager or subsystem interface module was not linked with a Enterprise Developer Server program at product installation. Registers 3 and 4 in the dump usually contain the name of the stub program. The abending load module is the module that was not linked correctly.
- ELA2** The Task Work Area (TWA) does not exist or is not long enough to be used by Enterprise Developer Server. The TWA length must be greater than or

equal to the sum of 1024 plus the twaOffset (TWA offset) build descriptor option specified when the initial program in the transaction was generated.

Use the TWASIZE parameter in the transaction definition to define a TWA with an adequate length for the transaction.

When this is a client/server program set, ensure that the CICS mirror transactions on the server CICS have a TWA size at least equal to the TWA size specified for the Enterprise Developer transaction on the client system.

- ELA3** Load for module ELARSCNT was not successful. Enterprise Developer Server has not been installed correctly.

Ensure the CICS region can gain access to the Enterprise Developer Server run-time library and that module ELARSCNT is defined in the program definition.

When this is a client/server program set, ensure that the security level-checking parameters (RSL and RSLC) for the CICS mirror transactions on the server CICS are the same as those specified for the Enterprise Developer transaction on the client system.

- ELA4** Load for module ELARPRTX was not successful. Enterprise Developer Server has not been installed correctly.

Ensure the CICS region can gain access to the Enterprise Developer Server run-time library and that module ELARPRTRX is defined in the program definition.

- ELA5** Load for module ELARPRTC was not successful. Enterprise Developer Server has not been installed correctly.

Ensure the CICS region can gain access to the Enterprise Developer Server run-time library and that module ELARPRTC is defined in the program definition.

- ELA6** The dynamic storage stack used for working storage for Enterprise Developer Server modules was exhausted and Enterprise Developer Server could not continue.

This problem should not occur. Report the problem to the IBM support center.

- ELA7** A GETMAIN was not successful. There was not enough storage for the program to complete.

Try the program again when the region is less busy or try it again in a larger region.

- ELA9** Load or link for a Enterprise Developer Server module was not successful. Enterprise Developer Server has not been installed correctly. Use CEDF to determine the module name. Look for a PGMIDERR on a CICS LOAD or CICS LINK command.

Ensure that the CICS region can gain access to the Enterprise Developer Server run-time library and the module name being loaded is defined in the PPT.

- ELAB** A call was made to a main program, which is not allowed.

- ELAC** Enterprise Developer Server has detected a FREEMAIN request that is not valid. Collect the dump and contact the IBM Support Center for assistance.

- ELAE** A generated program has ended because of a serious error. This occurs for one of the following reasons:

- Storage has been corrupted so that a dump is necessary to debug the abend.
- Error handling was unable to write messages to the error destination queue or to the user at the terminal. The dump is necessary to make the diagnostic information available. The situation can occur if the error destination queue specified for the transaction using the diagnostic controller utility is not defined to CICS. In CICS, if the error destination queue is defined as an intrapartition queue, this situation occurs when there is no more space on the intrapartition queue and the error messages cannot be written.
- A severe error has occurred. Refer to the error destination queue for the corresponding error messages. The default name is ELAD. The queue name can be changed using the diagnostic controller utility.

See “Enterprise Developer Server ABEND Dumps” on page 139 for information on how to find error messages in the dump on an ELAE abend.

ELAF ELATSRST has detected one of the following errors:

- ELATSRST was not initiated with a CICS XCTL command (for example, the restart transaction ID was associated directly to ELATSRST).
- The COMMAREA length on entry was not 0 or 10.
- The Enterprise Developer Server portion of the TWA had been initialized, indicating that a **converse** was not in process or the non-Enterprise Developer program uses the TWA and the program was not generated with the proper TWA offset.

ELAW A program was generated using incompatible versions of COBOL generators.

IMS, IMS BMP, and z/OS Batch Environments

1600 A generated program has ended because of a serious error. This occurs for one of the following reasons:

- Storage has been corrupted so that a dump is necessary to debug the abend.
- Error handling was unable to write messages to the error destination queue or to the user at the terminal. The dump is necessary to make the diagnostic information available. In IMS, the situation can occur if the error destination queue specified using the **errorDestination** build descriptor option is not defined to IMS.
- A severe error has occurred. In IMS, refer to the error destination queue specified using the **errorDestination** build descriptor option for the corresponding error messages. In z/OS Batch, refer to the data set ELAPRINT for the messages.

See “Enterprise Developer Server ABEND Dumps” on page 139 for information on how to find error messages in the dump on a 1600 abend.

1601 A database manager or subsystem interface module (for example, ASMTDLI for DL/I access) was not linked with a Enterprise Developer Server program at product installation. Registers 3 and 4 in the dump contain the name of the stub program. The abending load module is the module that was not linked correctly.

Refer to the *Program Directory for Enterprise Developer Server for z/OS* for information on correctly linking the abending load module.

- 1602** A program generated with the `imsFastPath="YES"` build descriptor option ended because of a run unit error. The abend is issued to prevent any further scheduling of the program in error.
- See “Enterprise Developer Server ABEND Dumps” on page 139 for information on how to find error messages in the dump on a 1602 abend. Depending on the generation options specified for the program, the message might also have been written to an error diagnostic message queue, on the IMS log, or to an ELAPRINT file. See Chapter 20, “Diagnosing Problems for Enterprise Developer Server on z/OS Systems” for more information on Enterprise Developer Server error reporting.
- 1606** The dynamic storage stack used for working storage for Enterprise Developer Server modules was exhausted and Enterprise Developer Server could not continue.
- This problem should not occur. Report the problem to the IBM Support Center.
- 1608** Enterprise Developer Server has detected a FREEMAIN request that is not valid. Collect the dump and contact the IBM Support Center for assistance.

z/OS Batch

- 1600** A generated program has ended because of a serious error. This occurs for one of the following reasons:
- Storage has been corrupted so that a dump is necessary to debug the abend.
 - Error handling was unable to write messages to the error destination queue or to the user at the terminal. The dump is necessary to make the diagnostic information available.
 - A severe error has occurred. Refer to the data set ELAPRINT for the messages.
- See “Enterprise Developer Server ABEND Dumps” on page 139 for information on how to find error messages in the dump on a 1600 abend.
- 1601** A database manager or subsystem interface module was not linked with a Enterprise Developer Server program at product installation. Registers 3 and 4 in the dump contain the name of the stub program. The abending load module is the module that was not linked correctly.
- Refer to the *Program Directory for Enterprise Developer Server for z/OS* for information on correctly linking the abending load module.
- 1606** The dynamic storage stack used for working storage for Enterprise Developer Server modules was exhausted and Enterprise Developer Server could not continue.
- This problem should not occur. Report the problem to the IBM Support Center.
- 1608** Enterprise Developer Server has detected a FREEMAIN request that is not valid. Collect the dump and contact the IBM Support Center for assistance.

Chapter 26. Codes from Other Products for z/OS Systems

The chapter contains lists of common system abend codes, COBOL run-time messages, LE abend codes, and common run-time messages from CICS.

Common System Abend Codes for All Environments

Only the most frequently occurring abend codes are listed in this section. If you receive another abend code or if you need a more complete explanation of one of the abend codes, refer to the *System Codes* manual for your release of z/OS.

System 0C7 Data exception. The abend occurs when fields defined as decimal or packed decimal are retrieved from a database and are found to contain data of a different format.

The abend can also occur if fields that are not initialized are used in calculations or comparisons. This happens if the program attempts to read a record from a database and the record is not found, but the program uses fields in the record anyway. To ensure that records are initialized, use a SET record EMPTY statement in the program or specify `initAdditionalWS` and `initIORecords` as build descriptor options. Refer to the Enterprise Developer online help system for additional information on how to initialize records using a SET record EMPTY statement in the program. Scan the helps using the phrase SET record EMPTY.

The abend can also occur when SET record EMPTY is used or when `initAdditionalWS` and `initIORecords` are used if one of the following is true:

- There are redefined records with different data types or variable field boundary alignments from the original record.
- The primary working storage record receives a transferred record that contains different data types or variable-field boundary alignments from the original record.

System 806 Module not found in a library. This can occur if a new version of a module is put into a load library and is placed in secondary extents. To avoid this when you allocate load libraries, specify a large primary allocation and 0 for the secondary allocation. This insures that if there is enough space for the load module it will be placed in the primary extent. If there is not enough space, there will be an abend (for example, a B37 abend for insufficient space) when you link the module into the load library. Using this technique detects the space problem during the preparation step rather than at run time.

In other environments, this can occur if the module is not in a library defined in your link list, JOBLIB, or STEPLIB concatenation sequence.

If the missing module name is ELACxxx, the NLS language code identified by the last 3 characters of the module name is not installed on the system. This language code was specified with the `targetNLS` build descriptor option when the program was generated.

If you try to run an EGL-generated program under Enterprise Developer Server and cannot load the module ELARSCNT, the system abends with an 806.

LE Run-time Messages

Only the most frequently occurring LE run-time messages are listed in this section. If you receive other run-time messages that start with IGZ or if you need a more complete explanation of one of the messages, refer to the debugging manual for your release of LE.

IGZ0033S **An attempt was made to pass a parameter address above 16 megabytes to AMODE(24) program program-name.**

Explanation: An attempt was made to pass a parameter located above the 16-megabyte storage line to a program in AMODE(24). The called program will not be able to address the parameter.

Programmer response: If the calling program is compiled with the RENT option, the DATA(24) option may be used in the calling program to make sure that its data is located in storage accessible to an AMODE(24) program. If the calling program is compiled with the NORENT option, the RMODE(24) option may be used in the calling program to make sure that its data is located in storage accessible to an AMODE(24) program. Verify that no linkedit, binder or genmod overrides are responsible for this error.

System action: The application was terminated

IGZ0064S **A recursive call to active program program-name in compilation unit compilation-unit was attempted.**

Explanation: COBOL does not allow reinvocation of an internal program which has begun execution, but has not yet terminated. For example, if internal programs A and B are siblings of a containing program, and A calls B and B calls A, this message will be issued.

Programmer response: Examine your program to eliminate calls to active internal programs.

System action: The application was terminated.

IGZ0066S **The length of external data record data-record in program program-name did not match the existing length of the record.**

Explanation: While processing External data records during program initialization, it was determined that an External data record was previously defined in another program in the run unit, and the length of the record as specified in the current program was not the same as the previously defined length.

Programmer response: Examine the current file and ensure the External data records are specified correctly.

System action: The application was terminated.

IGZ0075S **Inconsistencies were found in EXTERNAL file file-name in program program-name. The following file attributes did not match those of the established external file: attribute-1 attribute-2 attribute-3 attribute-4 attribute-5 attribute-6 attribute-7**

Explanation: One or more attributes of an external file did not match between two programs that defined it.

Programmer response: Correct the external file. For a summary of file attributes which must match between definitions of the same external file, see *IBM COBOL Language Reference*.

System action: The application was terminated.

COBOL Run-time Messages

Only the most frequently occurring COBOL run-time messages are listed in this section. If you receive other run-time messages that start with IGZ or if you need a more complete explanation of one of the messages, refer to the *Debugging and Run-time Messages Guide* for your release of LE.

IGZ033S **An attempt was made to pass a parameter address above 16 megabytes to AMODE(24) program program-name.**

Explanation: An attempt was made to pass a parameter above the 16-megabyte storage line to a

program in AMODE(24). The called program will not be able to address the parameter.

IGZ064S **A recursive call to active program program-name in compilation unit**

compilation-unit was attempted.

Explanation: COBOL does not allow an internal program that has started to run, but has not completed, to be invoked again. For example, if internal programs A and B are siblings of a containing program, and A calls B and B calls A, this message will be issued.

IGZ066S **The length of external data record data-record in program program-name did not match the existing length of the record.**

Explanation: While processing external data records during program initialization, it was determined that an external data record was previously defined in

another program in the run unit, and the length of the record as specified in the current program was not the same as the previously defined length.

IGZ075S **Inconsistencies were found in EXTERNAL file file-name in program program-name. The following file attributes did not match those of the established external file: attribute-1 attribute-2 attribute-3 attribute-4 attribute-5 attribute-6 attribute-7**

Explanation: One or more attributes of an external file did not match between two programs that defined it.

Common COBOL Abend Codes

Only the most frequently occurring abend codes are listed in this section. If you receive another abend code or if you need a more complete explanation of one of the messages, refer to the debugging manual for your release of LE.

User 4087 This is an LE abend code. If reason code is 7, the error could be due to the region size not being large enough to run the COBOL program.

Common IMS Runtime Messages

Only the most frequently occurring IMS runtime messages are listed in this section. If you receive another runtime message that starts with DFS or if you need a more complete explanation of one of the messages, refer to the IMS messages and codes manual for your release of IMS.

DFS057I **REQUESTED BLOCK NOT AVAILABLE: blockname RC = reason code**

Explanation: The blockname is either the MOD or the DOF name. If it is the DOF name, the first 2 bytes of the name are the device type and features printed in hexadecimal. Refer to the message format services manual for your release of IMS for an interpretation of these 2 bytes. If it is a MOD name, it will be the name of a map group.

User response: If a DOF name was specified, review the values you specified for the **mfsDevice**, **mfsExtendedAttr**, and **mfsIgnore** build descriptor options, and compare them to the IMS system definition for the terminal that had the problem.

If a MOD name was specified, ensure that you installed the MFS control blocks into the correct library. If you specified the **mfsUseTestLibrary="YES"** build descriptor option, ensure that you used the **/TEST MFS** command. If you specified **mfsUseTestLibrary="NO"**, ensure that your system administrator has run the IMS online change utility to copy in the new format definitions.

DFS064 **NO SUCH TRANSACTION CODE**

Explanation: This message is sent to a terminal when the transaction code requested by the user is not defined to IMS. An example of a situation that results in this message is when a program does an XFER with a map to a transaction that is not defined to IMS. The map is written to the terminal, but when the user enters data, the transferred-to transaction cannot be scheduled because it is not defined to IMS.

User response: Either ensure the transaction code is defined to IMS or change the XFER statement in the transferring program to reference the correct IMS transaction code.

DFS182 **INVALID OR MISSING PARAMETER**

Explanation: An IMS reserved word (for example, LTERM) was used as a map name in a **/FORMAT** command.

User response: If you need to use the **/FORMAT** command to display this map, you need to change the map name and generate the map group and any programs that use this map again.

DFS555I **TRAN ttttttt ABEND S000,Uaaaa; MSG
IN PROCESS: (up to 78 bytes of data)
time stamp**

Explanation: This message indicates that the transaction running in IMS has ended abnormally. Typical abend codes are shown below:

0778 IMS user abend, indicating that a ROLL request was issued

1602 Enterprise Developer Server abend because a rununit error occurred in a program that was generated with the **imsFastPath="YES"** build descriptor option

1600 Enterprise Developer Server abend because an unrecoverable error occurred in situations other than rununit errors for programs generated with **imsFastPath="YES"**

User response: Press the PA1 or PA2 key to display the error map that contains the error diagnostics that describe the error.

DFS2082 **RESPONSE MODE TRAN
TERMINATED WITHOUT REPLY**

Explanation: Enterprise Developer Server has ended the logical unit of work for a program that was generated with the **imsFastPath="YES"** build descriptor option.

User response: Press the PA1 key to display the error map that contains the error diagnostics that describe the error.

DFS2766I **PROCESS FAILED**

Explanation: IMS issues this message if Enterprise Developer Server ends the run unit for a transaction program that was generated with **imsFastPath="YES"** and run in an IMS fast-path region.

User response: Press the PA1 or PA2 key to display the error map that contains error diagnostics that describe the error. See Chapter 20, "Diagnosing Problems for Enterprise Developer Server on z/OS Systems" for additional information.

(none) **Logged off IMS and returned to the
VTAM sign-on screen without any
warning or error message being
displayed.**

Explanation: One of the following might have occurred:

- The program attempted to display a map with DBCS or mixed data on a non-DBCS terminal or printer.
- The values specified for the **mfsDevice**, **mfsExtendedAttr**, and **mfsIgnore** build descriptor options do not match the IMS system definition for the terminal that had the problem.

User response: Correct the program or generation options, generate the program and map group again, and then run the program again.

Common IMS Runtime Abend Codes

Only the most frequently occurring IMS abend codes are listed in this section. If you receive another abend code or if you need a more complete explanation of one of the abend codes, refer to the messages and codes manual for your release of IMS.

IMS 259 A program has been compiled with the DATA(31) compile option and is being run in a non-IMS/ESA environment. The program should be recompiled with the DATA(24) compile option.

IMS 462 A program was scheduled in a message region, but the program ended without successfully issuing a get unique for an input message. This can occur if Enterprise Developer Server detects an error that would prevent the program from processing properly. Examples of these errors are:

- The IMS PSB does not match the PSB defined in Enterprise Developer.
- The print services program is missing.

IMS 778 A ROLL call has been issued Enterprise Developer Server because of a rununit error or a catastrophic error in the IMS/VS environment. The ROLL is issued to prevent further scheduling of the program in error. IMS displays message DFS555I indicating that abend 778 has occurred. The Enterprise Developer Server error message panel can be displayed by pressing PA1.

Based on your generation options and the JCL for your message region, additional diagnostic information might be provided on an error diagnostic message queue, on the IMS log, or in ELAPRINT. See “Controlling Error Reporting in IMS Environments” on page 131 for additional information.

Note: Press PA2 if PA1 does not cause the Enterprise Developer Server error map to display.

IMS 1008

A program that was running as a BMP and that obtained access to fast-path databases did not issue a SYNC or CHKP call at the end of the job step. You can force the CHKP call to occur by:

- Using the EZECOMIT special function word in a batch-oriented BMP
- Ensuring that the transaction-oriented BMP ends with an EOF (QC status) for the file being used for input from the IMS message queue

IMS 3042

Access to DB2 cannot be obtained. Possible causes of this are:

- The terminal ID is not defined to DB2.
- The DB2 program plan is not valid or access to the DB2 program plan cannot be obtained.

If the program was being run as a BMP, see Figure 23 on page 105 for sample JCL.

Common CICS Run-time Messages

Only the most frequently occurring CICS run-time messages are listed in this section. If you receive another CICS run-time message that starts with DFH or if you need a more complete explanation of one of the messages, refer to the CICS messages and codes manual for your release of CICS.

DFHAC2016 date time **applied Transaction tranid cannot run because program program-name is not available.**

Explanation: The transaction tranid cannot be run because the initial program for the transaction is not available. This could occur because the transaction is defined in the PCT, but the program is not defined in the PPT or is not in a library in the DFHRPL concatenation.

User response: Have your system administrator check the PPT entries. Be sure the program is in a library in the DFHRPL concatenation.

DFHAC2206 time applied **Transaction tranid has failed with abend abcode. Resource backout was successful.**

Explanation: The transaction tranid has ended abnormally with abend code abcode. abcode is either an CICS transaction abend code or a user abend code.

User response: If the user abend code starts with ELA, see “CICS Environments” on page 159. If it is an CICS abend code, see “Common CICS Abend Codes” to see if it is included there. If not, refer to the CICS messages and codes manual for your release of CICS.

Common CICS Abend Codes

Only the most frequently occurring CICS abend codes are listed in this section. If you receive another CICS abend or if you need a more complete explanation of one of the abend codes, refer to the CICS messages and codes manual for your release of CICS.

Depending on your diagnostic options, information might be available on an error destination queue or in an CICS journal. See “Controlling Error Reporting in CICS” on page 131 and the appropriate ELA messages in Chapter 2.

ADLD

A program isolation deadlock occurred and a transaction was

selected for an abend. Refer to the *VisualAge Generator Design Guide* for information on using the EZEDLRST special function word and for information on designing restartable transactions.

AEY9	Access to DB2 cannot be obtained. This occurs if DB2 is not running.
AFCY	A transaction was purged when a deadlock occurred because a file is defined with LSRPOOLID not equal to NONE in the FCT, and one process within a program has performed a SCAN against a file and another process requested an update or add to the same file (or its alternate index) without ending the SCAN. Change the LSRPOOLID to NONE, or change the program design to end the SCAN before the update or add is requested.
APCT	A requested module cannot be located in the program definitions or in the program library.
ASRA	<p>A program check occurred. Some of the reasons this can occur for a Enterprise Developer program are as follows:</p> <ul style="list-style-type: none"> • Incorrectly linked Enterprise Developer Server modules. If register 1 contains ELA1, see the information for ELA1 in "CICS Environments" on page 159. • Data not initialized or data initialized to incorrect values. If the error occurred as a result of a data exception, see the explanation for "System 0C7" in "Common System Abend Codes for All Environments" on page 163.
ATDD	The program attempted to process a transient data queue that is disabled. This can occur for a program file associated with a transient data queue or for the transient data queue used for error diagnostic information.
AXFQ	The most common cause is the result of INBFMH not being specified equal to ALL in the profile associated with the CICS mirror program (CPMI).

Note: CICS users that receive abend codes ADLD, ADCP, AKCT, or D106 might see four question marks in place of the CICS abend code for the resulting Enterprise Developer Server message. The CSMT console log contains the true CICS abend code that was issued.

COBOL Abends under CICS

1009	A program has a dynamic storage requirement greater than 64KB, but was compiled with the DATA(24) compiler option. Compile the module again with the DATA(31) compiler option.
1029	Either a PPT entry for a program attached through a COBOL dynamic call is not found or the module being invoked cannot be found in the CICS region program library search string. Additional information can be retrieved by entering transaction CEBR on the terminal where the error occurred.

Part 6. Appendixes

Appendix. Enterprise Developer Server Run-time Messages

This section describes a series of messages that are given by Enterprise Developer Server.

ELA00002P Enterprise Developer Server is required for program %01C08

Explanation: The generated COBOL program is not compatible with the installed version of Enterprise Developer Server.

Enterprise Developer Server ends the program with a user abend.

User response: Contact the system administrator. Enterprise Developer Server should be installed.

ELA00003P PCB %01D03 DL/I error, function = %02C04, status code = %03C02

Explanation: The program control logic attempted a DL/I call to a teleprocessing PCB and received an error status code from IMS on the call. The message specifies the PCB that was used on the call (0 is the I/O PCB, 1 is the modifiable alternate PCB, and 2 is the express modifiable alternate PCB). The message also specifies the function code and the status code. For ISRT calls, the message is accompanied by message ELA00066I, which displays the first 255 bytes of the DL/I I/O area.

The run unit ends. If the ELASnap data set is allocated, Enterprise Developer Server issues a SNAP dump for all status codes other than AI.

User response: Look up the status code in the IMS messages and codes documentation for your system.

ELA00005A Date entered is not valid for defined date format %01C10

Explanation: Data entered into a form field defined with a date edit either does not meet the requirements of the format specification, or the month or day of the month is not valid.

It is not necessary to enter the separator characters shown in the message, but if they are omitted, enter leading zeros. For example, if the date format is MM/DD/YY, you can enter 070491.

User response: Enter the date in the format shown in the message.

ELA00007P File OPEN error on file %01C08, file status = %02C08

Explanation: The specified file did not open successfully.

The format of the file status depends on the file type.

For SEQ files, the file status is the 2-character COBOL status code followed by six zeros.

For VSAM files, the file status is composed of the 2-character COBOL status code followed by the VSAM return code (two characters), VSAM function code (one character), and the VSAM feedback code (three characters). The VSAM codes could be blank if the file OPEN was not completed.

For VSAMRS files, the file status is composed of the 2-character ACB (access control block) return code in hexadecimal format followed by six zeros.

The run unit ends.

User response: First see the table of common COBOL and VSAM status codes in the *IBM Enterprise Developer Server Guide for z/OS*. If the codes in the message are not listed in the table, refer to the COBOL programming language reference and VSAM administration guide for your system for a definition of other file status and VSAM codes. Also look for system error messages pertaining to the specified DD name or DLBL name. Correct the error and run the program again.

ELA00008P File CLOSE error on file %01C08, file status = %02C08

Explanation: The specified file did not close successfully, and the run unit ends.

The format of the file status depends on the file type.

For SEQ files, the file status is the 2-character COBOL status code followed by six zeros.

For VSAM files, the file status is composed of the 2-character COBOL status code followed by the VSAM return code (two characters), VSAM function code (one character), and the VSAM feedback code (three characters).

For VSAMRS files, the file status is composed of the 2-character ACB (access control block) return code in hexadecimal format followed by six zeros.

User response: First see the table of common COBOL and VSAM status codes in the *IBM Enterprise Developer Server Guide for z/OS*. If the codes in the message are not listed in the table, refer to the COBOL programming language reference and VSAM administration guide for your system for a definition of other file status and VSAM codes. Also look for system

error messages pertaining to the DD name. Correct the error and run the program again.

ELA00009P Overflow occurred because the target item is too short

Explanation: The target of a move or assignment statement is not large enough to hold the result without truncating significant digits. The value of sysVar.handleOverflow is 1, and the run unit ends if the overflow condition occurs.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, the Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Do as follows:

- Increase the number of significant digits in the target data item; or
- Define the program logic to handle the overflow condition by using sysVar.handleOverflow and sysVar.overflowIndicator.

ELA00014P A replace was attempted without a preceding get for update on %01C18

Explanation: This error occurs in these cases:

- A replace statement was issued against a record that was not successfully read for update; or
- A replace statement is associated with a specific get statement, but a different get statement was used to select the record.

The read for update might have been cancelled as the result of a converse statement in a segmented program.

The run unit ends.

User response: Make sure that in the get and replace statements, the program correctly used record names or resultSetID.

Also make sure that the sequence of statements is appropriate. To step through the program, you can use the EGL debugger or (for CICS-based programs) CEDF.

ELA00015P READ/WRITE error for file %01C08, file status = %02C08

Explanation: An I/O operation was not successful for the specified file. Program processing ends on any nonzero status code if the I/O statement is not in a try block; and ends on a hard error if the I/O statement is in a try block when sysVar.handleHardIOErrors is set to 0.

The format of the file status depends on the file type.

For SEQ files, the file status is the 2-character COBOL

status code followed by six zeros.

For VSAM files, the file status is composed of the 2-character COBOL status code followed by the VSAM return code (two characters), VSAM function code (one character), and the VSAM feedback code (three characters).

The run unit ends.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: First see the table of common COBOL and VSAM status codes in the *IBM Enterprise Developer Server Guide for z/OS*. If the codes in the message are not listed in the table, refer to the COBOL programming language reference and VSAM administration guide for your system for a definition of the other file status and VSAM codes. Also look for system error messages pertaining to the specified DD name. Correct the error and run the program again.

ELA00016P %01C08 error for file %02C08, %03C44, file status = %04C08

Explanation: An I/O operation was not successful for the specified file. Program processing ends on any nonzero status code if the I/O statement is not in a try block; and ends on a hard error if the I/O statement is in a try block when sysVar.handleHardIOErrors is set to 0.

The message identifies the VSAM operation that was not successful, the Enterprise Developer file name associated with the record, the system resource name, and the file status. The file status is composed of two zeros followed by the VSAM return code (two characters), VSAM function code (one character), and the VSAM feedback code (three characters).

The run unit ends.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: First see the table of common VSAM status codes in the *IBM Enterprise Developer Server Guide for z/OS*. If the codes in the messages are not listed in the table, refer to the VSAM administration guide for your system for a definition of other VSAM codes. Also look for system error messages pertaining to the specified system resource. Correct the error and run the program again.

ELA00021I An error occurred in program %01C08 on statement number %02D06

Explanation: An error occurred in the specified program on the specified statement. The actual error that occurred is identified in the messages following this message.

User response: Refer to a listing of the program, correct the statement, and generate the program again.

ELA00022P Form group format module %01C08 could not be loaded

Explanation: The specified form group format module could not be loaded. The module is a generated object module linked as a program that contains data tables that describe the format and constant fields for text forms in a form group. The module name is the form group alias (or a variation to conform with length and character restrictions) followed by the characters FM.

If the format module name uses the format ELAxxxFM, where xxx is the language code, the definitions for the Enterprise Developer Server error forms could not be loaded.

The run unit ends.

User response: Have the system administrator verify that the specified program has been generated, compiled, and linked into a library defined in the library search order.

For z/OS CICS, the search order includes the DFHRPL data sets, and you should verify that the program has been defined to the system.

ELA00023P Call to data-table program %01C07 was not successful

Explanation: A dynamic COBOL call to the specified data-table program was not successful. The run unit ends.

User response: Make sure that the specified program was generated, compiled, and linked into a library defined in the library search order. For z/OS CICS, the search order includes the DFHRPL data sets.

Also for z/OS CICS, make sure that the program was generated with data (a build descriptor option) set to 31 and that the program is defined to the CICS region.

ELA00024P Conversion table %01C08 could not be loaded

Explanation: Either the specified table program could not be loaded or the program that was loaded is not a Enterprise Developer Server conversion table.

The run unit ends.

User response: Verify that the correct conversion table name was specified in the generation-time linkage options part; that a correct conversion table has been moved into the system variable sysVar.callConversionTable at run time; or that a correct conversion table has been specified on any call to the system function sysLib.convert. For details, see the EGL help topic on data conversion.

If the conversion table was properly specified in the

program, make sure that the table program was generated, compiled, and linked into a library defined in the library search order. For z/OS CICS, the search order includes the DFHRPL data sets.

Also for z/OS CICS, make sure that the table program was generated with data (a build descriptor option) set to 31 and that the table program is defined to the CICS region.

ELA00026P A calculation caused a maximum-value overflow

Explanation: During a calculation in an arithmetic statement, an intermediate result exceeded the maximum value (18 significant digits). This condition also occurs when division by zero occurs. If sysVar.handleOverflow is set to 0 or 1, the program ends.

This error can only occur when you specify the build descriptor option checkNumericOverflow.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Correct the program logic either to avoid the error or to use sysVar.handleOverflow and sysVar.overflowIndicator to handle the error.

ELA00027P The data on a character-to-numeric move is not valid

Explanation: The statement in error involves a move from a character to a numeric data item. The character data item contains nonnumeric data.

The run unit ends.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Change the program to ensure that the source operand contains valid numeric data.

ELA00029P Transfer to %01C08 was not successful

Explanation: The transfer to another program was not successful. Usually, the program being transferred to could not be found.

The run unit ends.

User response: Make sure that the program was generated, compiled, and linked into a library defined in the library search order. For z/OS CICS, the search order includes the DFHRPL data sets.

Also for z/OS CICS, make sure that the program was generated with data (a build descriptor option) set to 31 and that the table program is defined to the CICS region.

ELA00031P Call to %01C08 was not successful

Explanation: A dynamic call to the specified program failed, ending the run unit.

User response: Make sure that the program was generated, compiled, and linked into a library defined in the library search order. For z/OS CICS, the search order includes the DFHRPL data sets.

Also for z/OS CICS, make sure that the program was generated with data (a build descriptor option) set to 31 and that the program is defined to the CICS region.

ELA00032P Called program %01C07 received a parameter list that is not valid

Explanation: A call to the specified program was not successful for one of the following reasons:

- The calling program passed too many or too few parameters.
- Different values are in the linkage-options part, callLink element, parmform property for the called and calling programs.
- The parmform value COMMDATA was specified for the call, and the COMMAREA passed has a different length than the length expected by the called program.

If the called program is a remote program, a CICS abend occurs. Because the COMMAREA is too small, the called program cannot notify the calling program of the error.

In all other cases, the run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Verify that the argument list in the call statement matches the parameter list for the program being called, and then generate the called and calling program with the same parmform value in the linkage options part, callLink element.

ELA00033P Call to program %01C08 returned exception code %02D05.

Explanation: An exception code was returned on a call to the specified program, indicating that one of the arguments passed to the program was not valid. The run unit ended because the call was not in a try block.

User response: Place the call statement in a try block and make sure that all the passed arguments are valid.

ELA00034P Program %01C07 was declared as a main program and cannot be called

Explanation: The specified program was not declared as a called program.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Declare the program as a called program.

ELA00035A Data type error in input - enter again

Explanation: The data in the first highlighted field is not valid numeric data. The field was defined as numeric.

User response: Enter only numeric data in this field, or press a bypass edit key to bypass the edit check. In either situation, the program continues.

ELA00036A Input minimum length error - enter again

Explanation: The data in the first highlighted field does not contain enough characters to meet the required minimum length.

User response: Enter enough characters to meet the required minimum length, or press a bypass edit key to bypass the edit check. In either situation, the program continues.

ELA00037A Input not within defined range - enter again

Explanation: The data in the first highlighted field is not within the range of valid data defined for this item.

User response: Enter data that conforms to the required range, or press a bypass edit key to bypass the edit check. In either situation, the program continues.

ELA00038A Table edit validity error - enter again

Explanation: The data in the first highlighted field does not meet the table edit requirement defined for the variable field.

User response: Enter data that conforms to the table edit requirement, or press the bypass edit key to bypass the edit check. In either situation, the program continues.

ELA00039A Modulus check error on input - enter again

Explanation: The data in the first highlighted field does not meet the modulus check defined for the variable field.

User response: Enter data that conforms to the modulus check requirements, or press a bypass edit key to bypass the edit check. In either situation, the program continues.

ELA00040A No input received for required field - enter again

Explanation: No data was typed in the field designated by the cursor. The field is required.

User response: Enter data in this field, or press a bypass edit key to bypass the edit check. Blanks or nulls will not satisfy the data input requirement for any type of field. In addition, zeros will not satisfy the data input requirement for numeric fields. The program continues.

ELA00041P Property msgTablePrefix was not specified for a program: Message %01C04, NLS code %02C03

Explanation: The program tried to display a message from a message table but lacks a value for the property msgTablePrefix.

The run unit ends.

User response: Do any of the following:

- Assign a valid value in the message table property and generate the program again.
- Change the program to avoid requesting the user message, then generate the program again.
- Remove the user message number from the validation property validatorTableMsgKey and generate the program or form group again.

ELA00042P The expected number of inserts for message %01C08, NLS code %02C03 was not received

Explanation: The expected number of variable inserts for an Enterprise Developer Server message did not match the number received. The message text is in the language-dependent message table program, ELACxxx, where xxx is the language code.

The program is generated from a data-table part that might have been modified and generated specifically for your installation.

The inserts show the original error message number that occurred and the language code being used. Message ELA00163P shows the original error message number that occurred and the message inserts that would have been displayed for that message.

The run unit ends.

User response: Correct the problem identified by the original message.

If the language-dependent message table was modified, correct the modified message so that the inserts are the same as the inserts defined in the default data table that was shipped with Enterprise Developer Server.

ELA00043P %01C08, %02C03

Explanation: The Enterprise Developer Server message table program ELACxxx (where xxx is the language code) did not contain a runtime message.

The program is generated from a data-table part that might have been modified and generated specifically for your installation.

The inserts show the original error message number that occurred and the language code being used. Message ELA00163P shows the original error message number that occurred and the message inserts that would have been displayed for that message.

The run unit ends.

User response: Correct the problem identified by the original message.

If the language-dependent message table was modified, verify that the message numbers in the modified table match the message numbers in the message table as shipped in the product. Also, verify that the program loaded is at the same maintenance and release level as the message table shipped in the product.

ELA00044P Message %01C08, NLS code %02C03, not found

Explanation: The Enterprise Developer Server message table program ELANCxxx (where xxx is the NLS code) did not contain a runtime message.

The program is generated from a data-table part that might have been modified and generated specifically for your installation.

The inserts show the original error message number that occurred and the NLS language code that was being used. The message is accompanied by message ELA00163P, which shows the original error message number that occurred and the message inserts that would have been displayed for that message.

The original error message that occurred determines if (and how) the program ends and if a SNAP dump is issued.

User response: Correct the error identified by the first message insert.

If the message table was modified, check that the message numbers in the modified table match the message numbers in the message table as shipped in

the product. Also, check that the program loaded by the program is at the same maintenance and release level as the message table shipped in the product.

ELA00045P Error reading message %01C08, NLS code %02C03, status %03C08

Explanation: The user message file or database did not contain a user-defined message for the language associated with the language code. Message files and databases are used only in COBOL programs generated using CSP/370 Runtime Services Version 1 Release 1.

The format of the message ID is as follows:

- Positions 1-3 = User message file
- Positions 4-8 = Message number

The status code varies depending on the type of user message file or database being used:

- For VSAM, status is eight characters. The first two bytes of code are either 08 (to specify a relative message within a record is not used) or 12 (to specify a record was not found in the VSAM file). The remaining six bytes of code are the VSAM return code (two characters), function (one character), and feedback code (three characters), all in decimal format. Refer to the VSAM administration guide for your system for a definition of the VSAM codes.
- For DB2, status is the 4-character SQL code. Refer to the DB2 manuals for your system for a description of the SQL code.

User response: Make sure that the message is defined in the program message file in one of two ways:

- Convert the message file to an EGL message table. Generate the program and the message table again using Enterprise Developer.
- If a message database is being used, add or replace the message in the message database using the Cross System Product/370 Runtime Services Version 1 Release 1 message database utility.

ELA00046P Call to print services program %01C08 was not successful

Explanation: A dynamic COBOL call to the specified print services program was not successful.

The run unit ends.

User response: Make sure that the program was generated, compiled, and linked into a library defined in the library search order. For z/OS CICS, the search order includes the DFHRPL data sets.

Also for z/OS CICS, make sure that the program was generated with data (a build descriptor option) set to 31 and that the program is defined to the CICS region.

ELA00047P Message %01D04 was not found in message table program %02C07

Explanation: A user message could not be found in the program message table.

In all z/OS environments, the Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

The run unit ends.

User response: Either add the message to the table or modify the program to use a message that is defined in the table.

ELA00050A Number of allowable significant digits exceeded - enter again

Explanation: The user entered data into a numeric field that was defined with decimal places, a sign, currency symbol, or numeric separator edits. The number of significant digits that can be displayed within the editing criteria was exceeded by the input data; the number entered is too large. The number of significant digits cannot exceed the field length, minus the number of decimal places, minus the places required for editing characters.

User response: Enter a number with fewer significant digits.

ELA00051P Form %01C08 was not found in form group %02C06

Explanation: The specified form name is not in the form group.

The run unit ends.

User response: Re-generate the form group and the program.

ELA00057P Delete attempted without preceding update on record %01C18

Explanation: This error occurs in these cases:

- A delete statement was issued against a record that was not successfully read for update; or
- A delete statement is associated with a specific get statement, but a different get statement was used to select the record.

The read for update might have been cancelled as the result of a converse statement in a segmented program.

The run unit ends.

User response: Make sure that in the get and delete statements, the program correctly used record names or resultSetID.

Also make sure that the sequence of statements is appropriate. To step through the program, you can use

the EGL debugger or (for CICS-based programs) CEDF.

ELA00061P DL/I error, function = %01C04, status code = %02C02

Explanation: DL/I returned an error status code in response to the DL/I call for the current function and either of the following occurred:

- There was no error routine specified for the function.
- Both special function words EZEFECE and EZEDLERR were set to 0 (this indicates that the program should end on abnormal DL/I conditions), and the status code specified either an abnormal condition, or a condition that was not expected.

The status code in the message comes from the DL/I PCB used for the DL/I call.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

This is either a program error or a database definition error.

User response: Do the following:

1. Locate the specified error code. Refer to the IMS messages and codes or the IMS application programming manuals for a description of the specified status code.
2. Correct the error.
3. Generate the program again.

ELA00062P DL/I call overlaid storage area, record %01C18

Explanation: A DL/I call read a block of data that was larger than the record defined to hold the data. The storage area immediately following the record buffer was overlaid.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, the Enterprise Developer Server issues a SNAP dump if the ELASAP data set is allocated.

User response: This is a program error. Define the record so that its length matches the length of the segment it represents and generate the program again.

ELA00063I PCB DB %01C08, segment %02C08, level %03D02, options %04C04

Explanation: This message provides additional diagnostic information for a database I/O error. The

PCB passed in the DL/I call contained the specified information.

For unsuccessful DL/I I/O call, the segment name field contains the last segment along with the path to the requested segment that satisfied the call. When a program is initially scheduled, the name of the database might be put in the segment name field if no segment is satisfied.

User response: Refer to message ELA00061P.

ELA00064I PCB key feedback area length %01D04

Explanation: This message provides additional diagnostic information for a database I/O error. The PCB passed in the DL/I call contained the specified key feedback length. This is the length of the concatenated key of the hierarchical database path.

User response: Refer to message ELA00061P.

ELA00065I PCB key feedback area = %01C255

Explanation: This message provides additional diagnostic information for a database I/O error. The PCB passed in the DL/I call contained the specified key feedback area.

The first 255 bytes are displayed. If necessary, because of the line and data lengths, the message wraps around to display all 255 bytes. The data is displayed as character data in the message. The message is followed by two lines that give the hexadecimal value under each character.

User response: Refer to message ELA00061P.

ELA00066I DL/I I/O area = %01C255

Explanation: This message provides additional diagnostic information for a hard DL/I I/O error. The message displays the contents of the DL/I I/O area.

The first 255 bytes are displayed. If necessary, because of the line and data lengths, the message wraps around to display all 255 bytes. The data is displayed as character data in the message. The message is followed by two lines that give the hexadecimal value under each character.

User response: This message is always accompanied by another message (for example, ELA00003P or ELA00061P) that specifies the error. See the explanation and user response of the accompanying message.

ELA00067I DL/I SSA %01D02: %02C255

Explanation: This message provides additional diagnostic information for a DL/I I/O error. The message displays the contents of a segment search argument (SSA) for the DL/I call. The first message insert gives the number of the SSA. The second insert gives the first 255 bytes of the SSA.

If necessary, because of the line and data lengths, the message wraps around to display all 255 bytes. The data is displayed as character data in the message. The message is followed by two lines that give the hexadecimal value under each character.

This message is repeated once for each SSA used in the DL/I call.

User response: Refer to message ELA00061P.

ELA00068P DL/I variable segment length is not valid, segment %01C08

Explanation: A DL/I segment I/O area is shorter than the segment returned in a DL/I retrieval, or the computed segment length on an ADD or REPLACE I/O option is not valid.

If the I/O option was an INQUIRY, UPDATE, or SCAN, the BYTES parameter in the DBD is greater than the length of the record defined to Enterprise Developer.

If the I/O option was an ADD or REPLACE, the program has erroneously set the length of the segment. If this error occurs for a path call, the DL/I I/O area shown in message ELA00061I contains only segments before the segment with the error. Because the length is in error, the segment with the error cannot be moved to the DL/I I/O area.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, the Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: If the error occurred in a retrieval, have the database administrator correct either the DBD or VisualAge Generator record definition, and generate the program again.

If the error occurred on an update, correct the logic associated with calculating the length of the segment. Generate the program again.

ELA00069P The value of an input variable is too large for the target SQL column

Explanation: The run unit ends.

In CICS environments, Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, the Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Modify the program to ensure that values that overflow the even-numbered length of the

record item are detected and rectified before executing any function that has an SQL row record as its object, and that uses the record item as an input host variable in its SQL statement.

This condition is not detected in programs that have the build descriptor option checkNumericOverflow; instead the high-order digit of the record item's value is truncated before being used in the SQL statement.

ELA00070P %01C04 error, status code %02C02

Explanation: DL/I returned an error status code other than QC or AL in non-VSE environments or HX or XR on VSE environments in response to a CHKP (checkpoint) or ROLB (rollback) DL/I call.

CHKP and ROLB calls are issued for the following reasons:

- The program calls the commit or rollback functions.
- The program ends abnormally and a PSB is active.
- The program causes a commit to be taken at a CONVERSE I/O option, a First Map, or because of the /SYNCFER generation option.

The status code in the message is taken from the I/O PCB used with the DL/I call.

The run unit ends.

In the VSE batch environment, the Enterprise Developer Server issues a SNAP dump that is directed either to the system logical unit SYSLST or the dump data set of the partition.

In all z/OS and VSE environments, the Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Make a note of the message and notify the system programmer. On z/OS systems, refer to the application programming manual or the IMS messages and codes manual for a description of the status code. On VSE systems, refer to the DOS DL/I messages and codes manual for a description of the status code.

ELA00072P %01C18, set record position is not supported

Explanation: The SET SCAN indicator was on for a DL/I segment record when a SCAN function with a user-modified SSA list was used with that record. The SET SCAN indicator is not supported for DL/I calls with modified SSA lists.

The run unit ends.

User response: Modify the program logic so that it does not set the SET SCAN indicator for a segment with a modified DL/I call.

ELA00073P SQL error, command = %01C08, SQL code = %02D04

Explanation: The SQL database manager returned an error code for an SQL statement. Program processing ends following an SQL request whenever the sqlcode in the SQL communications area (SQLCA) is not 0, and either of the following is true:

- The I/O statement is not in a try block; or
- The sqlcode indicated a hard error and the system variable sysVar.handleHardIOErrors was set to 0.

The message is followed by message ELA00074I which displays the substitution variables associated with the sqlcode. (Those substitution variables are also available to the program by way of the system variable sysVar.sqlerrmc.)

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Determine the cause of the problem from the SQL code and the SQL error information.

Either correct the program or the database definition. Refer to the appropriate database manager messages and codes manual for information on the SQL code and SQL error information.

ELA00074I SQL error message: %01C70

Explanation: This message accompanies message ELA00073P when an SQL error occurs. It displays the relational database manager error information returned in the SQLCA field SQLERRM and is repeated as many times as necessary to display the complete description.

User response: Use the information from this message and ELA00073P to correct the error.

ELA00076P Invalid data is used in a character-to-hexadecimal assignment or comparison

Explanation: The current statement involves either a move from a character data item to a hexadecimal data item, or a comparison between a character data item and a hexadecimal data item. The characters in the character data item all must occur in the following set for the move or compare to complete successfully:

a b c d e f A B C D E F 0 1 2 3 4 5 6 7 8 9

One or more of the characters in the character data item is not in this set. This condition causes a program error.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Change the program to ensure that the character data item contains valid data when the character-to-hexadecimal move or compare operation occurs. In text-form fields, you can use the isHexDigit validation property to ensure that user input contains only valid characters.

ELA00080A Hexadecimal data is not valid

Explanation: The data in the variable field identified by the cursor must be in hexadecimal format. One or more of the characters you entered does not occur in the following set:

a b c d e f A B C D E F 0 1 2 3 4 5 6 7 8 9

User response: Enter only hexadecimal characters in the variable field. The characters are left-justified and padded with the character zero. Embedded blanks are not allowed.

ELA00086P %01C18 - No active open or get for update is in effect

Explanation: One of these case applies:

- A get next statement cannot run because a related open did not run previously in the same program; or
- A replace or delete statement cannot run because a related open or get for update did not run previously in the same program.

All rows selected for update are released when a called program returns to the calling program.

The run unit ends.

User response: Make sure that in the second statement (get next, replace, or update), the program correctly used record names or resultSetID to match the first statement (open or get).

Also make sure that the sequence of statements is appropriate. To step through the program, you can use the EGL debugger or (for CICS-based programs) CEDF.

ELA00093I An error occurred in program %01C08, function %02C18

Explanation: An error occurred in the specified function for the specified program. Other information about the error is given in the messages that follow this message.

If a function is not active, the second insert contains the name of a section in the generated initialization or ending logic of the program.

User response: Refer to the error messages following this message to determine the cause of the error.

ELA00096P A data operand of type MBCHAR is not valid

Explanation: An operand in a move or assignment statement contains mixed double-byte and single-byte data that is not valid.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Verify that the data is valid in variables that are of type MBCHAR and are in a move or assignment statement.

ELA00105I Error occurred at terminal %01C08, date %02C08, time %03C08, user %04C08

Explanation: An error occurred at the specified logical terminal on the specified date and time. This message precedes any error diagnostic information routed to an alternate error destination.

For a program running in z/OS batch, the first insert is ***** , which indicates that the terminal identifier is not known.

For z/OS CICS, the last insert is only provided if sign-on security is active on or provided in the system.

User response: Examine all error messages that follow this message and precede the next occurrence of this message. Use the information from these messages to diagnose and correct the error.

ELA00106P Program %01C08 PSB does not match the generated PSB definition

Explanation: The PCBs passed to the program at program initialization time did not match the PSB defined for the program. The number of PCBs passed was less than the number of PCBs defined in the Enterprise Developer definition.

The run unit ends. In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, the Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Either correct the program definition

of the PSB and generate the program again, or correct the IMS PSB and generate it again.

ELA00109P Input form must be form %01C08 rather than form %02C08, for program %03C07

Explanation: The form received by the program is not the form specified as the value of program property inputForm. This error occurs when the program starts.

The run unit ends.

User response: Make sure that the transferring program specifies the correct form on the show statement and that the receiving program has the correct value in property inputForm.

ELA00110P Shared data table %01C07 cannot be updated

Explanation: The program modifies a data table that was generated as a shared table. Shared data tables cannot be updated.

The run unit ends.

User response: Either generate the data table as non-shared or change the program to avoid modifying the data table.

ELA00111P Length of input form %01C08 is not valid

Explanation: The length of an input form received by a program is not the length defined for the form in the program.

The run unit ends.

User response: Use the same form declaration when generating both the program that receives the input form and the program that issues the show statement.

ELA00114P A transfer to called program %01C07 is not allowed

Explanation: A program cannot transfer to a called program.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Replace the transfer statement with a call statement.

ELA00115P Use of a transfer statement is invalid because the receiving program (%01C07) has an input form

Explanation: Only a show statement can transfer to a program that requires an input form.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

The Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response:

Do either of these actions:

- Use a show statement to invoke the receiving program indirectly; or
- Remove the value in the inputForm property of the receiving program. The program can show the form after receiving control.

ELA00118P Missing PSB for program %01C07

Explanation: A PSB was specified for the named program during definition. However, the program ran as a z/OS batch job. This can happen if you do not use the sample JCL or CLIST created by the generation function.

The run unit ends.

User response: If the program contains DL/I I/O or other DL/I functions, change the runtime JCL or CLIST to run DL/I programs. If the program does not use DL/I, remove the PSB name from the program definition.

ELA00119P Programs %01C07 and %02C07 are not compatible

Explanation: A program started by a transfer or call statement is not compatible with the starting program because the programs were generated for different target systems.

The run unit ends.

User response: Re-generate the program for which the target system was wrong.

ELA00120P sysLib.startTransaction failed, logical terminal ID = %01C08, status code = %02C02

Explanation: Common status codes are as follows:

- QH** Unknown output destination
- A1** Unknown output destination

Both status codes indicate that the 8-character logical terminal ID was not defined to the IMS system as either a terminal or transaction.

The run unit ends.

User response: Do as follows:

1. Ensure that the transaction code field of the record specified on sysLib.startTransaction is defined to the IMS system.
2. Review the program logic ensure that the transaction code file is set correctly.
3. Refer to the IMS application programming manual or the IMS messages and codes manual for your system for an explanation of status codes other than the ones listed above.

ELA00121P sysLib.audit was not successful, logical terminal ID = %01C08, status code=%02C04

Explanation: The status code is the 2-character status from the I/O PCB.

The run unit ends.

User response: Refer to the IMS application programming manual or the IMS messages and codes manual for your system.

ELA00125P Error number %01D04 is not valid

Explanation: The error handler was called with an error number that it did not recognize. This is a product error.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Ensure that the generated COBOL code has not been modified by generating the program again. Afterwards, run the program again. If the problem persists, do as follows:

1. Record the message number
2. Obtain the dump
3. Record the scenario under which this message occurs
4. Obtain the COBOL source for the problem program
5. Use your electronic link with IBM Service if one is available, or contact the IBM Support Center

ELA00127P A requested function is not supported for form %01C08, form group %02C06

Explanation: A form field referenced a validation function, but the program does not include that function.

The run unit ends.

User response: Check the field properties and the program, then re-generate the program with build descriptor option genFormGroup set to YES.

ELA00129I Form %01C08 was received

Explanation: Related messages give further details.

User response: Refer to the related error messages.

ELA00130P GSAM error, file = %01C08, function = %02C04, status code = %03C02

Explanation: An I/O error occurred on an ADD, SCAN, or CLOSE I/O option for a file associated with a GSAM database. Program processing ends on a hard status code if EZEFECE is set to 0, or on any error status code if there is no process error routine.

This message can also occur on an implicit OPEN or CLSE call to the GSAM database. An implicit OPEN or CLSE call occurs as a result of an ADD or SCAN process. Program processing ends on a hard status code if EZEFECE is set to 0, or on any error status code if there is no process error routine for the ADD or SCAN that caused the implicit OPEN or CLSE call.

An AI status code for an implicit OPEN might be caused by specifying a file name during Enterprise Developer resource association or with the ASSOCIATE command that is different from the DD name specified in the GSAM DBD.

For an ADD, message ELA00066I accompanies this message and provides the DL/I I/O area that was used for the call.

The run unit ends. If ELASNAP is allocated, the Enterprise Developer Server issues a SNAP dump.

User response: Determine the cause of the I/O error from the DL/I status code and either correct the program or the database definition. Refer to the IMS application programming manual or the IMS messages and codes manual for your system for an explanation of the DL/I status code.

ELA00131P MSGQ error, file = %01C08, function = %02C04, status code = %03C02

Explanation: An error occurred on a SCAN or ADD function for a file or a DISPLAY function for a print map when the file or EZEPRINT is associated with an IMS message queue (I/O or TP PCB). Program processing ends on a hard status code, if EZEFECE is set to 0, or on any error status code, if there is no I/O error routine.

Common status codes are:

QH Unknown output destination (ADD, DISPLAY, or CONVERSE)

A1 Unknown output destination (ADD, DISPLAY, or CONVERSE)

A6 Output segment limit exceeded (ADD, DISPLAY, or CONVERSE)

FD Deadlock occurred (SCAN).

For an ADD, DISPLAY, or CONVERSE, the listed status codes specify that the 8-character system resource name associated with the file or EZEPRINT at generation or in the EZEDEST or EZEDESTP special function words was not defined to the IMS system as either a terminal or a transaction.

For an ADD, DISPLAY, or CONVERSE, message ELA00066I accompanies this message and shows the DL/I I/O area that was used for the call.

The run unit ends. If ELASNAP is allocated, the Enterprise Developer Server issues a SNAP dump.

User response: If the output destination is not valid, ensure that it is defined to the IMS system. Also review the program logic to ensure that EZEDEST, if used, is set correctly. For an explanation of status codes other than the ones listed above, refer to the IMS application programming manual or the IMS messages and codes manual for your system.

ELA00135P The program is not expecting an input form

Explanation: A program issued a show statement that included an input form, but the receiving program has no value for property inputForm.

The run unit ends.

User response: Either change the invoking program to avoid sending a form or change the receiving program to specify an input form.

ELA00136P DL/I error occurred in work database operation

Explanation: An error occurred during use of the work database when it was implemented using DL/I. This message is accompanied by additional DL/I diagnostic messages, including ELA00061P, that provide additional information about the error. Message ELA00061P includes the DL/I function and status code. Refer to the IMS messages and codes or IMS application programming manual for your system for a description of the status code.

The run unit ends. If ELASNAP is allocated, the Enterprise Developer Server issues a SNAP dump.

User response: This is a database definition error or an error in the definition of the work database PCB in your IMS PSB. Record this information and any other diagnostic messages, and notify the system administrator.

ELA00137P SQL error occurred in work database operation

Explanation: An error occurred during use of the work database when it was implemented using SQL. This message is accompanied by additional SQL diagnostic messages, including ELA00073P, that provide additional information about the error.

The run unit ends. If ELASNAP is allocated, the Enterprise Developer Server issues a SNAP dump.

User response: Determine the cause of the problem from the SQL code and the SQL error information in related message ELA00074I, and correct the database definition.

ELA00138P %01C07 was replaced in the middle of a conversation

Explanation: The program was running in segmented mode and ran a converse statement, but was replaced in the load library during user think time (the time between writing the form to the terminal and receiving the user's input).

The program conversation with the user started with the original version of the program and cannot be resumed.

The run unit ends. In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

User response: Run the program again.

ELA00139P MFS map program %01C06 and MFS map %02C08 have different versions

Explanation: An MFS mapping services program attempted to process a message input descriptor for an MFS map that was generated at a different time than the MFS mapping services program. Both the MFS mapping services program and the map it works with must be built in the same generation step.

This is probably a problem with the installation of either the program or the MFS map after generation of a map group. One of the following might have occurred:

- The MFS mapping services program might have been compiled and linked without installing the MFS maps, or vice versa.
- The MFS map might have been installed in an MFS test library, but you did not enter an IMS /TEST MFS command prior to starting the transaction.
- The MFS map might have been installed in the MFS production library, and you entered a /TEST MFS command prior to starting the transaction.
- The MFS map might have been used in an XFER with a map from another program. The transfer-from program used a different map group, but the map name on the XFER is the same as the First Map name for the transfer-to program.

In the IMS/VS environment, the transaction (logical unit of work) ends and processing continues with the next message. In all other environments, the run unit ends.

User response: Ensure that the same version of the MFS mapping services program and the MFS control blocks are installed in the correct libraries. If an XFER and First Map are involved, ensure that the transfer-from and transfer-to programs use the same map group.

ELA00140P Segmentation storage size discrepancy for %01C07

Explanation: The size of the segmentation storage record is not valid for the specified program.

Possible causes for the error include:

- The program is replaced in the load library in the middle of a program conversation with the user
- The program issues a show statement that includes a form, but the receiving program expects an input form that has different characteristics
- The program is segmented and issues a converse statement when sysVar.transactionID contains a transaction code, but that code is associated with a program that has no relationship to the issuing program. If the sysVar.transactionID is used to switch transaction codes, the new transaction must start either the same program that was started by the old transaction or the program that issued the converse statement.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Try the transaction again. If the program works correctly, the error was caused by a re-link in the middle of the conversation. If the error still occurs, determine why there is a mismatch and correct the situation that caused the error.

ELA00141P Data table %01C07 cannot be modified. Delete %02D06 bytes.

Explanation: The program's attempt to modify a shared data table would cause an increase in data-table size beyond the CICS limit, which is 65535 bytes.

The run unit ends.

User response: Either change the logic of the program so that the data table is not modified or decrease the

size of the data-table content by the specified number of bytes.

ELA00142P Map %01C08 in group %02C06 not supported on this device

Explanation: A map has been sent to a device using IMS Message Format Services, but the device type was not in the list of devices specified for the map using the Enterprise Developer device selection function. The message appears when either of the following occurs:

- A printer map was sent to a destination that is defined as a terminal in the IMS System Generation. The destination is the system resource name specified for EZEPRINT at generation or an override value loaded into the EZEDESTP special function word at run time. The message appears at the terminal where the printer map was directed, not at the terminal that originated the transaction. Program processing continues.
- A terminal map is defined in a map group that contains multiple maps with different device selections. The device to which the map was directed was not specified using the Enterprise Developer device selection function. The message appears at the terminal that originated the transaction as the result of a CONVERSE or an XFER with a map. The program conversation with the user at this device ends because there is no way for the user to enter data. The program continues processing with the next input message on the message queue.

The program is not notified by MFS that a problem has occurred. Therefore, message ELA00142P is built into the MFS source to provide a method of notifying you when an error occurs. A SNAP dump is not issued.

User response: If the error occurred for a printer map, review the resource association information specified during generation, the program logic used to set the value of EZEDESTP, and the MFS generation options (/MFSDEV, /MFSIGNORE, and /MFSEATTR) to determine the appropriate corrections to make. Depending on the corrections required, generate either the program or map group again. In addition, if the printer map was sent to a terminal device, it might be necessary for the system administrator to purge the messages pending for the terminal using the IMS /DEQ command.

If the error occurred for a terminal map, review the terminal device types specified for this map and the MFS generation options (/MFSDEV, /MFSIGNORE, and /MFSEATTR) to determine the appropriate corrections to make. Generate the map group again.

If the program using the terminal map is a nonconversational program (/SPA=0 generation option), the user only needs to clear the screen and type another transaction code to resume work.

If the program that used the terminal map is a

conversational program (/SPA generation option greater than 0), the user must clear the screen, type /EXIT to end the conversation and then type another transaction code to resume work.

ELA00143P Data table %01C07 is not a message table

Explanation: A message table was specified for the program. The data table specified is not a message table.

The run unit ends.

User response: Either declare the data table as a message table and generate the data table again, or correct the message table name specified for the program and generate the program again.

ELA00144P Segmentation storage error

Explanation: Segmentation storage has an internal error mapping memory.

The run unit ends. In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: This is an internal system error. Contact the system administrator for assistance.

ELA00145A Map name required - enter /FOR %01C06O map-name

Explanation: The map group has more than one map, but a valid map name was not entered when the IMS /FOR command was used to display the map.

User response: Enter the /FOR command again, using the following format:

/FOR map-group0 map-name

ELA00146P Segmentation status error

Explanation: The status byte for segmentation storage management is lost and the program has no way to recover.

This error occurs when a PA key is pressed prior to pressing the ENTER key or a PF key for an IMS conversational transaction.

If the program was generated with /SPA=n or /SPA=(n,ADF), then there was no recovery feature generated in the program.

If the program was generated with /SPA=(n,m) or /SPA=(n,ADF,m) then the recovery feature was generated in the program, but was bypassed. A bypass of the recovery feature occurs when a deferred message switch comes from a non-generated program or a

generated program that was not generated with the same /SPA generation option.

In the IMS/VS environment, the transaction (logical unit of work) ends and processing continues with the next message.

User response: Restart the transaction sequence and avoid using PA keys while on a Enterprise Developer-generated screen.

Consider generating the Enterprise Developer programs with one of the /SPA generation options that will allow recovery from pressing a PA key.

ELA00147A Key sequence is not valid. Last screen will display - enter the data again

Explanation: A PA key was pressed prior to pressing the ENTER key or a PF key. IMS has reserved the use of the PA keys. All modifications on the previous screen are lost.

User response: Enter the data again and avoid use of PA keys while on a generated screen.

ELA00149I %01C07 command ignored during message database load

Explanation: The PSB for the message database specifies that the database is being initially loaded. Only ADD commands are supported during initial load of a DL/I message database.

User response: Run the message utility again, specifying the PSB for the database.

ELA00151P %01C07 of message record failed for the message database

Explanation: The message utility program encountered an error inserting or deleting a message in the message database. This message is accompanied by diagnostic messages describing the error.

If an ELASNAP DD statement is specified in the JCL, Runtime Services issues a snap dump. The run unit ends.

User response: Review the diagnostic messages. Verify that the database has been successfully defined by checking the DB2 message database create job (ELAMSJL2) messages. Correct the problem and run the job again.

ELA00152I Message file %01C03 has been added

Explanation: The indicated user message file has been successfully added to the message database.

User response: Test the programs that use this user message file.

ELA00153P %01C08 failed on file %02C08

Explanation: While running the message utility, an attempt was made to access (open, close, read, or write) the indicated file. The access failed and the message utility ended. The first message insert indicates the type of access that failed. The most common errors are a missing DD card for the file.

User response: Refer to the job listing for system error messages pertaining to the indicated DD name. Correct the error and run the job again, starting with the command that caused the error.

ELA00154I Message file %01C03 has been replaced

Explanation: The indicated user message file has been successfully replaced in the message database.

User response: Test the programs that use this user message file.

ELA00155I Message file %01C03 has been deleted

Explanation: The indicated user message file has been successfully deleted from the message database.

User response: Change the program using this user message file to use another message file and generate the program again.

ELA00156I Replace on non-existent message file %01C03, file was added

Explanation: A REPLACE command was issued for the indicated message file, but the file did not exist in the message database. The file was added instead.

User response: None, provided the file was added to the correct message database.

ELA00157P %01C08 failed on file %02C08, file status = %03C06

Explanation: While running of the message utility, an attempt was made to access (open, close, read, or write) the indicated VSAM file. The file identifies the DD name. The file status consists of the VSAM return code (2 characters), function (1 character), and feedback code (3 characters). The access failed and the message utility terminated. The first message insert indicates that type of access that failed.

User response: Refer to the VSAM administration guide for your system for a definition of the status codes. Also look at the job listing for system error messages pertaining to the indicated DD name. Correct the error and run the job again, starting with the command that caused the error.

ELA00158P Syntax error on command

Explanation: A command being processed by the message utility did not follow the correct syntax. The message utility ends.

User response: Correct the command and rerun the job, starting with the command that had the incorrect syntax.

ELA00159P Message file %01C03 already exists in the message database

Explanation: An attempt to add a user message file failed because the message file already existed in the message database for the language specified in the current message utility command. The return code is set to 08.

User response: Use the REPLACE command to update the message file in the message database.

ELA00160P Message file %01C03 does not exist in the message database

Explanation: An attempt to remove or list a user message file failed because the message file does not exist in the message database for the language specified in the current message utility command. The return code is set to 08. If the insert is an asterisk, you attempted to list all messages in an empty message database.

User response: Correct the message file ID in the command and run the job again.

ELA00162P Message I/O error, type %01C04, file %02C08, code %03C08

Explanation: An error occurred when a program generated using Cross System Product/370 Runtime Services Version 1 Release 1 attempted to open or close a user message file. The type variable insert specifies VSAM as the message file type. The file insert specifies the DD name. The first two bytes of the code insert are either 08 (to specify an OPEN) or 16 (to specify a CLOSE). The next two bytes are the ACB (Access control block) return code in hexadecimal format. The remaining bytes in the code insert are zero.

The run unit ends.

User response: Have the administrator do one of the following:

- Determine the cause of the problem from the VSAM error code. First see the table of common VSAM codes in the *IBM Enterprise Developer Server Guide for z/OS*. If the codes are not listed in the table, refer to the VSAM administration guide for your system for a definition of other VSAM codes. Also verify that the user message file is allocated correctly.

- Convert the message file to a message table and generate the program again under VisualAge Generator or CSP/370AD Version 4 Release 1.

ELA00163P %01C08, %02C60

Explanation: This message is used when a Enterprise Developer Server message cannot be found in the language-dependent message table program ELACxxx, where xxx is the language code.

The first variable insert in this message is the error message number for the error that actually occurred. The second insert in this message contains one of the message inserts that is used by the error that actually occurred. This message is repeated as many times as necessary to report all inserts. The inserts are reported in order by their number: %01, %02, and so on.

User response: See the message with the corresponding message number in this manual. Take the action appropriate for that message. Also, contact the system administrator to determine why the message could not be found in the Enterprise Developer Server language-dependent message table program.

ELA00164P %01C08, %02C04, %03C02, %04X08

Explanation: The error handler was not successful in using a DL/I call to write diagnostic information about another error to normal destinations for error information. The variable inserts contain the following information:

- Destination from the terminal identifier field of the PCB used in the call.

The destination can be the error destination specified at program generation, the user terminal ID, or the IMS log.

- DL/I function
- DL/I status code
- PCB Address

Enterprise Developer Server ends the program with a user abend.

User response: See Chapter 20, "Diagnosing Problems for Enterprise Developer Server on z/OS Systems," on page 129 for information about locating the diagnostic messages in the dump. These messages relate to the original error that ended the program. Also verify that the **errorDestination** value specified in your build descriptor options is included in the IMS system generation.

ELA00166P The recursion stack exceeds the maximum size allowed

Explanation: The stack that contains information to support recursion or segmentation has become too large.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Check for an infinite loop that is causing a large number of recursions. Either limit the number of recursions, or reduce the number of functions in the program.

ELA00168P %01C03

Explanation: The NLS language code in the file allocated to ELAMSG as shown in the insert is not valid. The Enterprise Developer Server utility ends because the language code for messages and report headings cannot be determined.

User response: Correct the JCL so that the ELAMSG DD statement references a sequential file or in-stream data that contains a valid NLS code in columns 1 through 3 of the first record. Refer to the IBM Enterprise Developer Server Guide for z/OS for a list of the valid NLS codes.

**ELA00169I Work database purged of %01D08
records older than day %02C06, time
%03C06**

Explanation: The utility that purges obsolete records from the work database has completed normally.

User response: None required.

ELA00170P Input is not valid

Explanation: Either the date or the time provided to the utility that purges obsolete records from the work database was nonnumeric or was not valid.

The run unit ends.

User response: Ensure that the date is in Julian format (YYDDD - two positions for the year and three positions for the day of the year). Ensure that the time is in HHMMSS format (two position for the hour, two positions for the minutes, and two positions for the seconds). The date and time specified must be at least 24 hours before the time that the purge program is run.

ELA00172I CICS error, system identifier %01C08

Explanation: An error occurred on a CICS function to be performed on a remote system. The message displays the CICS identifier for the remote system.

This message is always issued along with other messages that identify the function being performed and the CICS error return information.

User response: None required.

**ELA00173P An error occurred in remote program
%01C08, date %02C08, time %03C08**

Explanation: An error occurred in a remote program that caused the remote program to stop running. Diagnostic messages might have been logged at the remote location giving information about the error. The date and time stamp on this message can be used to associate the messages logged at the remote system with this error message.

The run unit ends.

User response: Report the error to the system administrator.

**ELA00179P An error occurred starting transaction
%01C08**

Explanation: IMS or CICS indicates that an error occurred when a program attempted to start the specified transaction. A message following this message gives the IMS or CICS error codes.

The run unit ends.

User response: Determine the cause of the error from the following message and correct the error.

**ELA00184P Program %01C07 and mapping services
program %02C08 are not compatible**

Explanation: The specified program and mapping services program are generated for different systems.

The run unit ends.

User response: Generate the mapping services program for the same environment as the program.

**ELA00185P Length of %01D02 for record %02C18 is
not valid and conversion ended**

Explanation: Conversion of a variable length record between the workstation format and host format cannot be performed because of one of the following conditions:

- The record length for the current record indicates that the record ends in one of the following:
 - The middle of a numeric field
 - The middle of a DBCHAR character
 - The middle of an SO/SI string.
- The record is longer than the maximum length defined for the record.

The run unit ends. In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Modify the program to set the record length so that it ends on a valid field boundary.

ELA00186P An operand of type MBCHAR in a conversion operation is not valid

Explanation: Conversion of an item from EBCDIC to ASCII or from ASCII to EBCDIC cannot be performed because a double-byte data value is not valid.

The run unit ends. In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Modify the program to ensure that any items of type MBCHAR are valid in the records to be converted.

ELA00187P Conversion table %01C08 does not support double-byte character conversion

Explanation: Conversion of an item of type MBCHAR or DBCHAR from ASCII to EBCDIC or from EBCDIC to ASCII cannot be performed because the specified conversion table does not include conversion tables for double-byte characters.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Modify the program to specify a conversion table that contains the double-byte conversion tables that are valid for data of types DBCHAR and MBCHAR. For background information, see the EGL help topic *Data Conversion*.

ELA00188P Conversion Error. Function: %01C25, Return Code: %02C05, Table: %03C08

Explanation: A system function was called to perform code page conversion for data used in a client/server program. The function failed.

Possible causes for the failure are:

- The code pages identified in the conversion table are not supported by the conversion functions on your system.

- For double-byte character conversion where the source data is in ASCII format, the source data was created under a different DBCS code page than the code page that is currently in effect on the system.

User response: Consider the EGL help topic *Data conversion*.

ELA00191I Program %01C07, generation date %02C08, time %03C08

Explanation: An error in the specified program has occurred. The error is identified in other messages preceding this message. The error might be caused by changes to individually generated components of the program.

User response: Verify the generation date and time of the program with that of other generated components.

ELA00192I Print services program %01C08, generation date %02C08, time %03C08

Explanation: An error in the specified print services program has occurred. The error is identified in other messages preceding this message. The error might be caused by changes to individually generated components of the controlling program.

User response: Verify the generation date and time of the print services program with that of other generated components in the program.

ELA00195I Form group format module %01C08, generation date %02C08, time %03C08

Explanation: An error in the specified form group format module has occurred. The error is identified in other messages preceding this message. The error might be caused by changes to individually generated components of the controlling program.

User response: Verify the generation date and time of the form group format module with that of other generated components in the program.

ELA00201P z/OS %01C08 error in service %02C08, RC = %03D04

Explanation: Enterprise Developer Server received an error return from a z/OS macro. The inserts identify the macro name, the Enterprise Developer Server program name, and the return code.

The run unit ends.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Contact the system administrator.

ELA00202P The file name %01C65 is not valid in the record-specific variable resourceAssociation or in sysVar.printerAssociation

Explanation: The format of the record name is not valid. The run unit ends.

User response: Refer to the EGL help to determine the valid syntax, then correct and regenerate the program.

ELA00203P CICS I/O error on file %01C08, resource %02C08

Explanation: The current program has attempted to gain access to a CICS file, and CICS returned a status code that indicated an I/O error occurred. The file is the logical file name specified in the record part declaration. The resource is the CICS file control table (FCT) or destination control table (DCT) name.

Possible causes of the error are the following:

- The file does not exist on disk.
- The file is not defined in the CICS FCT or DCT.
- The file was specified to be opened when first referenced.
- On z/OS CICS, the file was closed using the CSMT or CEMT transactions.
- For z/OS CICS, the DD statement for the file in the CICS startup JCL is missing, does not match the FCT name, or is in error.
- The file has been changed or otherwise corrupted.

Message ELA00204I is also displayed with the information from the EXEC interface block (EIB).

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: Have the CICS administrator refer to the CICS messages and codes manual for an explanation of the EIB codes. Correct the error and run the program again.

ELA00204I CICS EIBFN %01X04, RCODE %02X12, RESP %03D04, RESP2 %04D04

Explanation: The current program has received an error code for a CICS command. The run unit ends.

User response: Refer to the CICS application programmers' guide for an explanation of the EXEC interface block (EIB) codes. Correct the error and run the program again.

ELA00205P A CICS %01C22 error occurred in service %02C08

Explanation: Enterprise Developer Server received an error status code for a CICS command. This message identifies the command and the service program that

issued the command. This message is accompanied by message ELA00204I, which contains the response codes from the EXEC interface block (EIB).

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: Have the system administrator use the CICS diagnostic information in this message and in message ELA00204I to determine the cause of the error. Correct the error and run the program again.

ELA00206P Format of file %01C08 is not valid, reason code %02C01, resource %03C56

Explanation: The attributes of the system resource associated with the specified file name are not compatible with the properties defined for the record in the program. The reason code identifies the problematic attribute, as follows:

- | | |
|---|---------------|
| 1 | Key offset |
| 2 | Key length |
| 3 | Access method |
| 4 | Record format |
| 5 | Record length |

An access method mismatch occurs when the type of data set allocated does not match what the program expects. For example, a VSAM file is allocated as a system sequential file or a partitioned data set is allocated as a sequential file without specifying a part name.

The run unit ends.

User response: Change the record part declaration, the resource association part, or both, so that the record properties match the system resource attributes. Generate and test the affected programs again.

ELA00207P The attributes for file %01C08 are not compatible, reason code %02C01

Explanation: A program has attempted to use a file having file attributes that differ from another program in the run unit. All programs in a run unit must use the same attributes for a file. The reason code identifies the problematic attribute, as follows:

- | | |
|---|---------------|
| 1 | Key offset |
| 2 | Key length |
| 3 | Access method |
| 4 | Record format |
| 5 | Record length |

- 6 Use the system variable
sysVar.remoteSystemID to identify the location
of a remote file

The run unit ends.

User response: Change the record-part declarations, the resource association part, or both, so that all programs in the run unit have identical attributes for the file; then regenerate the affected programs.

ELA00208P Print services program %01C06 and form group format module %02C08 were generated separately

Explanation: The specified print services program attempted to process a form that was generated at a time different from the form group format module. Both the print services program and the form group format module must be generated at the same time.

The run unit ends.

User response: Make sure that the print services program and the form group format module were generated at the same time and are installed in the correct libraries.

ELA00210P Service number %01D04 is not valid

Explanation: An attempt was made to start a Enterprise Developer Server routine that does not exist or that is not valid.

The run unit ends.

In CICS environments Enterprise Developer Server issues a dump based on options selected using the diagnostic controller utility.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASNAP data set is allocated.

User response: Regenerate and test the program. If the problem persists, do as follows:

1. Record the message number
2. Obtain the dump
3. Record the scenario under which this message occurs
4. Obtain the COBOL source for the problem program
5. Use your electronic link with IBM Service if one is available, or contact the IBM Support Center

ELA00212P Error encountered gaining access to file %01C08, spool resource %02C65

Explanation: An error was received when attempting to gain access to a spool file. The message is accompanied by message ELA00204I, which contains response codes from the CICS EXEC interface block (EIB).

If the function was a write spool request (EIBFN 5602)

and the spool resource name was specified as node ID without being qualified by user ID, an error will occur if the user did not log on using the CICS logon procedure.

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: If the spool resource name specifies node ID without specifying user ID, log on using the CICS logon procedure before running the program again. Otherwise, refer to the CICS customization documentation for an explanation of the codes that are returned by the spool interface; then, correct the problem specified in the response codes.

Refer to the EGL help for additional information on the format of the system resource name.

ELA00215P PSB does not match the development-time PSB definition

Explanation: The number of PCBs passed to the program at program initialization time was less than the number of PCBs in the development-time PSB definition. This message is accompanied by ELA00217I.

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: Do as follows:

- Correct the DL/I PSB; or
- Correct the development-time PSB definition and generate the program again

ELA00216P CICS DL/I error, function %01C04, UIBFCTR %02X02, UIBDLTR %03X02

Explanation: CICS detected an error in a DL/I call. The message variable inserts specify the function being requested and the return codes from the CICS user interface block (UIB). If the function code is PCB, the program was attempting to schedule the program PSB. The message is accompanied by message ELA00217I.

Common return codes are as follows:

UIBFCTR	UIBDLTR	Description
08	00	Argument on DL/I call not valid. This error can occur if the IMSESA installation option in module ELARPIOP is specified as YES, but the IMS environment is not IMS/ESA.
08	01	PSB not found. The PSB must be defined to CICS.

UIBFCTR	UIDLTR	Description
08	03	The calling program has already successfully issued a scheduling (PCB) call that has not been followed by a TERM call.
08	05	PSB initialization was not successful.
08	06	The PSB in the scheduling call is not defined in the program control table (DLZACT).
08	07	A TERM call was issued when the task had already been terminated.
08	09	An MPS batch program attempted to issue a PCB call for a read-only PSB or for a nonexclusive PSB if program isolation was active.
08	FF	DL/I not active
0C	02	Intent scheduling conflict

The run unit ends.

User response: Check the definition of the call to the CSPTDLI service routine in the program, if the DL/I call is not valid. Otherwise, correct the problem specified by the error code. For additional codes, refer to the CICS application programmers' guide for your system to determine the meaning of the error codes.

ELA00217I Program %01C07, PSB name %02C08

Explanation: An error was detected in the specified DL/I program. The message is accompanied by messages ELA00215P or ELA00216P, which identify the problem.

The run unit ends.

User response: Refer to the accompanying messages for the problem cause.

ELA00218P Invocation of sysLib.audit not successful, journal id = %01D05, journal type = %02C02

Explanation: This message is accompanied by ELA00204I, which displays the contents of EIBRESP.

Common EIBRESP codes for CICS are as follows:

22 LENGERR

The computed length for the journal record exceeds the total buffer space allocated for the journal data set as specified in the journal control table (JCT) entry for the data set

43 JIDERR

Occurs if the specified journal identifier does not exist in the JCT

The run unit ends.

User response: Refer to the CICS programming documentation to define journal data sets, or contact the system administrator.

ELA00219P %01C22 error for %02C06 file %03C08, %04C56

Explanation: An I/O operation was not successful for the specified file.

Program processing ends on any nonzero status code if the I/O statement is not in a try block; and ends on a hard error if the I/O statement is in a try block when sysVar.handleHardIOErrors is set to 0.

The message identifies the I/O operation, the file type, the file name as specified in the record part, and the system resource name associated with the file.

The run unit ends.

In all z/OS environments, Enterprise Developer Server issues a SNAP dump if the ELASnap data set is allocated.

User response: Check that the correct data set has been allocated for this file.

ELA00220P Dynamic allocation was not successful, file %01C08, return %02D04, error reason code %03X04.

Explanation: Enterprise Developer Server was not successful on an attempt to perform dynamic allocation for the specified file. The other inserts are the return code in register 15 and the error reason code returned by the SVC 99 instruction.

The most common cause is that the file was not available. If you want your program to receive control after getting the status fileNotAvailable, place the I/O operation in a try block and set sysVar.handleHardIOErrors to 1.

The run unit ends.

User response: Contact the system administrator. Refer to the MVS System Programming: System Macros and Facilities manual for an explanation of the codes.

ELA00221P File %01C08, system resource name %02C56, not found

Explanation: Enterprise Developer Server attempted to dynamically allocate the file with the system resource name shown in the message. The file could not be found.

If the system resource name is a 1- to 8-character DD

name, then there is no DD card for the file in the job JCL. If the system resource name is a data set name, then the data set either does not exist or is not cataloged.

The run unit ends.

User response: If the name is a DD name, allocate a file to the DD name in the JCL. If the name is a data set name, ensure that the file exists and is cataloged.

ELA00222P Transaction %01C04 ended abnormally with CICS abend code %02C04

Explanation: The specified CICS transaction ended abnormally with the specified code.

On z/OS CICS systems, the following additional information is provided:

- On CICS Version 2 systems, if the ABEND code is ASRA or ASRB, this message is accompanied by the message ELA00223P and the ABEND exit can determine the module within which the error occurred.
- On later CICS systems, if the abend code is ASRA or ASRB, CICS message DFHAP0001 identifies the offset in the module at which the error occurred. The diagnostic control option specified for transaction abends using the Enterprise Developer Server diagnostic control utility determines whether a dump occurs.

The Enterprise Developer Server abend handler ends the program by issuing another ABEND command using the same code.

User response: See "ABEND Codes" in the *IBM Enterprise Developer Server Guide for z/OS* for a description of abend codes using the format ELAx. Refer to CICS or user program documentation for an explanation of other abend codes.

ELA00223P Program %01C08 abended at offset %02X08

Explanation: The specified program has abended with an ASRA or ASRB abend code. This indicates that a program check has occurred at the specified hexadecimal offset.

Enterprise Developer Server ends the program with a user abend.

User response: If the program is a generated COBOL program, use the compile listing to find the COBOL verb that was running when the program ended abnormally. The COBOL comments identify the EGL statements associated with the COBOL verb. Determine from the dump whether the problem was caused by bad data passed to the program. If the generated COBOL is in error, use your electronic link with IBM Service or contact the IBM Support Center.

ELA00225P Temporary storage queue name %01C08 is not valid

Explanation: The record-specific variable sysVar.resourceAssociation is set to a temporary storage queue name that is not valid. The name conflicts with a queue name that is reserved for Enterprise Developer Server. Names cannot begin with EZE.

The run unit ends.

User response: Specify a valid temporary storage queue name in the program.

ELA00228P The program attempted to use the resource %01C65 with file %02C07 and file %03C07

Explanation: The program attempted to associate the same system resource with two different files. The resource cannot be associated with two different files at the same time.

The run unit ends.

User response: Examine the program and correct the logic. Generate and test the affected programs again.

ELA00229P Invocation of sysVar.startTransaction failed, transID = %01C04, terminal ID = %02C08

Explanation: This message is accompanied by the message ELA00204I, which displays the contents of EIBRESP.

Common codes are as follows:

- | | |
|----|--|
| 11 | TERMINID error |
| | The specified terminal ID is not known to CICS. |
| 28 | TRANSID error |
| | The specified transaction ID is not known to CICS. |

The run unit ends.

User response: Have the system administrator define the terminal or transaction to CICS.

ELA00230P An error was encountered accessing CICS queue %01C08

Explanation: An error was received when attempting to access a CICS queue. The queue can be a transient data queue or temporary storage queue. This message is accompanied by message ELA00204I, which contains response codes from the CICS EXEC interface block (EIB).

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: Refer to the CICS application programmers' guide for an explanation of the response codes.

ELA00231P Error encountered retrieving data passed to program %01C08

Explanation: An error was received when attempting to retrieve data being passed to this program by a transfer statement or by sysVar.startTransaction. This message is accompanied by message ELA00204I, which contains response codes from the CICS EXEC interface block (EIB).

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: Refer to the CICS application programmers' guide for an explanation of the codes that are returned.

ELA00232P Form %01C08 in form group %02C06 is not declared or is not supported

Explanation: The specified form does not exist or is not defined for the type of device being used.

The run unit ends.

User response: Either define the map for your device type or select the device for the map. Generate the map group again.

If you are running on a CICS system, have the system administrator check that the alternate screen size for your device type is specified in the PCT entry for your transaction.

If the map group name uses the format ELAxxx, where xxx is the language code, the map group might have been modified incorrectly. The ELAxxx map group contains the Enterprise Developer Server error maps.

ELA00237P CICS TS Queue %01X16 error occurred in work database operation for program %02C07

Explanation: An error was received when attempting to access a CICS temporary storage queue. This message is accompanied by message ELA00204I, which contains response codes from the CICS EXEC interface block (EIB).

If the error is an INVREQ (EIBRESP=16), the problem might be caused by Enterprise Developer Server attempting to write a record that is longer than the control interval size for the VSAM data sets used for the auxiliary storage queue. The maximum segmentation record size written by Enterprise Developer Server is set by the TSQUE option in the installation options module ELARPIOP. TSQUE specifies the maximum size as the number of kilobytes; the default value is 16 KB.

The run unit ends.

User response: Refer to the CICS application programmers' guide for an explanation of the codes.

If the control interval size is the problem, have the system administrator assemble the installation module again after setting the TSQUE value to a value less than the control interval size.

Refer to the Server program directory for your system for more information.

ELA00249P Mapping services program %01C08 compiled with DATA(31) cannot be used by program

Explanation: A mapping services program compiled with the DATA(31) compiler option has been loaded for a program link-edited as AMODE(24).

User response: Compile the mapping services program again with the COBOL DATA(24) option; and make sure that data (a build descriptor option) is set to 24 whenever the form group is generated.

ELA00250P Program cannot process data with 31-bit addresses

Explanation: The initial program in the run unit was compiled with DATA(31). The current program was link-edited as AMODE(24). This is not compatible.

User response: Do one of the following:

- Compile the initial program in the run unit as DATA(24).
- Link-edit the current program as AMODE(31).

ELA00251P Data table %01C08 compiled with DATA(31) cannot be used by program

Explanation: A data table compiled with the DATA(31) compiler option has been loaded for a program link-edited as AMODE(24).

User response: Compile the table program again with the COBOL DATA(24) option. Also ensure the /DATA=24 generation option is specified whenever the table is generated.

ELA00252P Error on file %01C08, queue name %02C08, RC = %03C08

Explanation: An I/O logic error was detected by Enterprise Developer Server during processing of an I/O option for a CICS temporary storage queue.

Program processing ends on any nonzero status code if the I/O statement is not in a try block; and ends on a hard error if the I/O statement is in a try block when sysVar.handleHardIOErrors is set to 0.

Because the error was detected by Enterprise Developer Server instead of the access method, the return code

value consists of the characters RS (for runtime services) followed by a Enterprise Developer return code number.

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: See the section on return codes in the *IBM Enterprise Developer Server Guide for z/OS* to determine the meaning of the Enterprise Developer return code, and take the appropriate action.

ELA00253P Program %01C08 was not generated to receive form %02C08

Explanation: The specified program received a form as an input form, but the program does not contain processing logic for handling segmented programs. Either the wrong transaction name was specified when the program was started, or the wrong program was specified in the transaction definition.

The run unit ends.

User response: Make sure that the following are specified correctly:

- The transaction ID in the show statement
- The form name in the program property inputForm

Re-generate the modified program.

ELA00254P Invalid values for sysLib.audit, journal ID = %01D05, type = %02C02, length = %03D05

Explanation: A parameter in sysLib.audit is not valid:

- The journal ID must be between 1 and 99
- The third byte in the record must be in the range X'A0' to X'FF'
- The record length must be between 28 and 32767

The run unit ends. Enterprise Developer Server issues a dump based on information supplied for the transaction with the diagnostic controller utility.

User response: Correct the error and regenerate the program.

ELA00255P Invalid values for sysLib.audit, type = %01C02, length = %02D05

Explanation: A parameter in sysLib.audit is not valid:

- The third byte in the record must be in the range X'A0' to X'FF'
- The record length must be between 28 and 32767

The run unit ends.

User response: Correct the error and regenerate the program.

ELA00265E Segmented converse is not supported when local variables or function parameters are in the run-time stack

Explanation: The message indicates that a converse statement is invalid because the EGL run time cannot restore the values of parameters or local variables after the converse runs.

A segmented converse is described in the help topic on *Segmentation*.

The run-time stack is a list of functions; specifically, the current function plus the series of functions whose running made possible the running of the current function.

User response: Modify the program in one of two ways:

- Make sure that the functions on the run-time stack have neither parameters nor local variables; or
- Make sure that the converse is not segmented.

ELA00266E MQ function \$01C08, Completion Code \$02C02, Reason Code \$03C08.

Explanation: The MQ function did not complete successfully, as indicated by the following completion codes: 1 (MQCC_WARNING) 2 (MQCC_FAILED) The reason for the completion code is set in the reason code field by MQSeries®. Some common reason codes are: 2009 (Connection broken) 2042 (Object already open with conflicting options) 2045 (Options not valid for object type) 2046 (Options not valid or not consistent) 2058 (Queue manager name not valid or not known) 2059 (Queue manager not available for connection) 2085 (Unknown object name) 2086 (Unknown object queue manager) 2087 (Unknown remote queue manager) 2152 (Object name not valid) 2153 (Object queue-manager name not valid) 2161 (Queue manager quiescing) 2162 (Queue manager shutting down) 2201 (Not authorized for access) 2203 (Connection shutting down)

The run unit ends.

User response: Please refer to the *MQSeries Application Programming Reference* for further information on MQSeries completion and reason codes.

ELA00267E Queue Manager Name %01C48.

Explanation: This is the name of the queue manager associated with the failing MQ function call listed in message ELA00266. If the failing MQ function was MQOPEN, MQCLOSE, MQGET, or MQPUT, the name identifies the queue manager specified with the object name when the queue was opened. Otherwise, the name is the name of the queue manager to which the program is connected (or trying to connect). If the queue manager name is blank, the queue manager is the default queue manager for your system.

The run unit ends.

User response: Please refer to the *MQSeries Application Programming Reference* for further information on the MQSeries completion and reason codes that are listed in message ELA00266.

ELA00268E Queue Name %01C48.

Explanation: This is the name of the queue object associated with the failing MQ function call listed in message ELA00266.

The run unit ends.

User response: Please refer to the *MQSeries Application Programming Reference* for further information on MQSeries completion and reason codes that are list in message ELA00266.

ELA00269E Array index value %01D07 out of range for array %02C18 with size of %03D07

Explanation: The index specified for the dynamic array is out of bounds.

User response: Specify an index between 1 and the current number of elements in the array.

ELA00270E An attempt was made to exceed the maximum size of array %01C18

Explanation: An attempt was made to add an element to a dynamic array that already contains the maximum allowed number of elements.

User response: Modify the program in either of two ways:

- Increase the value of the dynamic-array property `maxSize`; or
- Change the logic so that the number of elements is always less than or equal to the value of `maxSize`.

ELA00300I A new copy was requested for part %01C08

Explanation: A new copy was requested for the programs associated with the specified part. Newly started transactions use the new copy of the program.

User response: None required.

ELA00301I The diagnostic control options were changed

Explanation: The diagnostic control options were changed after a user request from the Enterprise Developer Server Diagnostic Control utility.

User response: None required.

ELA00302I Error message queue sent to print destination

Explanation: The contents of the transient data queue containing the error messages were sent to the spooling system after a user request from the Enterprise Developer Server Diagnostic Print utility.

User response: None required.

ELA00303I Error message queue sent to print destination and deleted

Explanation: The contents of the transient data queue containing the error messages were sent to the spooling system after a user request from the Enterprise Developer Server Diagnostic Print utility. The contents of the transient data queue were then deleted.

User response: None required.

ELA00304A Type a valid selection number, then press Enter

Explanation: The selection number entered for a field on one of the Enterprise Developer Server utility panels is not valid. The cursor is positioned at the field in error.

User response: Type a valid selection and press Enter.

ELA00305A Type a name, then press Enter

Explanation: A required field was left blank on one of the Enterprise Developer Server utility panels. The cursor is positioned at the empty field.

User response: Type a valid name and press Enter.

ELA00306P CICS new copy was not successful for program %01C08. Press F2.

Explanation: The CICS SET NEWCOPY command was not successful for the specified part. The specified part was requested on the Enterprise Developer Server New Copy panel.

User response: Press F2 to view message ELA00204I, which contains the CICS response information from the EXEC interface block (EIB). Verify that the part name is correct. Refer to the CICS application programmers' guide for an explanation of the EXEC interface block (EIB) codes.

ELA00308P I/O error on error message queue. Press F2.

Explanation: A CICS error occurred when attempting to gain access to the error destination queue identified on the Enterprise Developer Server Diagnostic Print panel.

User response: Press F2 to view message ELA00204I,

which contains the CICS response information from the EXEC interface block (EIB). Verify that the error destination name is correct. Refer to the CICS application programmers' guide for an explanation of the EXEC interface block (EIB) codes.

ELA00309A Error message queue was not found

Explanation: The error destination queue identified on the Enterprise Developer Server Diagnostic Print panel was not found.

User response: Specify the correct error destination queue name on the panel.

ELA00310A Type a valid response, then press Enter.

Explanation: A value that was not recognized was specified in the field where the cursor is positioned. Valid values are shown following the field on the form.

User response: Type a valid value in the field and press Enter.

ELA00313I Default options are in effect for this transaction

Explanation: You made a request to view the diagnostic control options in effect for a specific transaction. The options currently in effect for the transaction are the default options.

User response: To exit, press F3. To change the options for this transaction do as follows:

1. Type the new options
2. Select action 1
3. Press Enter

ELA00314I Error message queue was empty

Explanation: A request was made to print an error message queue that does not contain any messages.

User response: None required.

ELA00315I Trace transaction list was updated successfully

Explanation: The list of transactions you specified to be traced has been processed successfully.

User response: None required.

ELA00316I Trace filter criteria updated successfully

Explanation: The list of trace filter criteria you specified has been processed successfully.

User response: None required.

ELA00317P Service number is not valid

Explanation: The trace filter criteria contains a service number that is not valid. If this error is detected during ELATRACE data set parsing, the run unit ends.

User response: Correct the service number specification in the ELATRACE data set and run the program again.

ELA00318P Tag in %01C08 is not valid

Explanation: The filter criteria contains a tag that is not valid. Valid tags are FILTER, EFILTER, APPLS, EAPPLS, SERVICES, and ESERVICES. The run unit ends.

User response: Correct the tag specification and run the program again.

ELA00319P Missing or misplaced tag in %01C08

Explanation: The filter criteria contains a missing or misplaced tag. The run unit ends.

User response: Correct the filter criteria and run the program again.

ELA00320P Too many programs in %01C08

Explanation: The filter criteria contains too many programs. The maximum number is 16. The run unit ends.

User response: Reduce the number of programs or remove all program filter criteria, then run the program again.

ELA00321P Too many services in %01C08

Explanation: The filter criteria contains too many services. The maximum number is 32. The run unit ends.

User response: Reduce the number of services or remove all service filter criteria, then run the program again.

ELA00322P One or more filters has a invalid value

Explanation: One or more codes entered for the DATASTREAM, TRACETOFILE, APPSTMT, SQLIO, SQLERR or IDUMP filters is not valid. The valid code that is entered must be either Y (yes) or N (no).

For z/OS batch, the run unit ends.

If you are defining filters online on z/OS CICS, the filter containing the value that is not correct is highlighted.

User response: For z/OS batch, specify either Y or N for these filters and run the program again. For CICS, type one of the valid values for the highlighted filter as

shown on the form, then press Enter.

**ELA00323P I/O error on storage queue %01C08.
Press F2.**

Explanation: An error was received when attempting to access a temporary storage queue in the diagnostic message print utility. Press F2 to view message ELA00204I, which contains response codes from the CICS EXEC interface block (EIB).

User response: Refer to the CICS application programmers' guide for an explanation of the codes.

ELA00324P Error reading trace control record. Press F2.

Explanation: An error was encountered when attempting to read or write to the trace control record in CICS. Press F2 to view more information.

For z/OS CICS, message ELA00204I is displayed, which contains response codes from the CICS EXEC interface block (EIB).

User response: Review the accompanying error messages.

ELA00325P Error opening %01C08

Explanation: An error was encountered when attempting to open the specified data set.

User response: Make sure that the data set has the correct attributes.

ELA00326P Error reading %01C08

Explanation: An error was encountered when attempting to read the specified data set.

User response: Make sure that the data set has the correct attributes.

ELA00342A The maximum number of copies already exists for the data table

Explanation: The maximum number of copies of a data table that can be used in a CICS region at one time is 5. The request for a new copy of the data table was rejected.

User response: Old copies of a data table that are in use are freed when all the transactions that are using the data table end. Retry the new copy request later.

ELA03001I F3=EXIT F8=CONTINUE

Explanation: None.

User response: None required.

ELA03002I F3=EXIT

Explanation: None.

User response: None required.

ELA03003I CLEAR=EXIT

Explanation: None.

User response: None required.

ELA03004I PF3=EXIT PF8=FORWARD

Explanation: None.

User response: None required.

ELA03005I PF3=EXIT

Explanation: None.

User response: None required.

ELA03006I PA1=CONTINUE

Explanation: None.

User response: None required.

ELA03007I ENTERPRISE DEVELOPER SERVER

Explanation: None.

User response: None required.

Notices

This information was developed for products and services offered in the U.S.A. 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 give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

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

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, 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.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites 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
P.O. Box 12195, Dept. TL3B/B503/B313
3039 Cornwallis Rd.
Research Triangle Park, NC 27709-2195
U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurement may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

COPYRIGHT LICENSE: This information contains sample application programs in source language, which illustrates programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows: 9999 (your company name) (year). Portions

of this code are derived from IBM Corp. Sample Programs. 9999 Copyright IBM Corp. _enter the year or years_. All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

Index

Special characters

/FORMAT command 101
/HOLD command 101
/MODIFY command 100
/WORKDB generation option
IMS 13

A

abend
ASPE
CICS 34
codes
CICS 159, 167
COBOL 165
IMS runtime 166
non-CICS environments 161, 162
preparation 151
system 163
dumps
COBOL 139
Enterprise Developer Server 139
recovery considerations
z/OS 40, 44, 45, 54, 55
activating trace sessions
CICS 144
adding
file name to the CICS file control table
z/OS 37
job control statements
z/OS 37
addressing, extended 27
alternate index, defining 24
alternate PCB, using 48
American National Standards printer
control character
z/OS 29, 31
AMODE 6
analyzing
detected errors 138
application
load module storage for Enterprise
Developer Server 5
plan for DB2 14
applying maintenance to
Enterprise Developer Server 3
ASA (see also American National
Standards printer control character) 31
ASPE abend, preventing 34
attributes for DBCS, hardware 27

B

backing up data 28
backup, maintaining copies of production
libraries 107
batch
print services program 71

BIND
command
data set 66
default 70
defining 70
DB2 programs 28
precompile messages 151
buffer size, printing
CICS 32
build descriptor
and compiler options that affect
performance 23
options
commentLevel 138
errorDestination 131, 134, 136
imsFastPath 131, 166
imsLogID 131, 135
initAdditionalWS 163
initIORecords 163
language code 135, 137
mfsDevice 165, 166
mfsExtendedAttr 165, 166
mfsIgnore 165, 166
mfsUseTestLibrary 165
performance considerations 23
restoreCurrentMsgOnError 131
spaSize 49, 98
targetNLS 163
trace 143
output files 67

C

catastrophic error 133
cautions
empty KSDS data set, VSAM
restriction 26
PRTMPP parameter, line skip
malfunction 33
CEDA transaction, RDO 85
change or view
defaults - ELAC04 120
options - ELAC02 118
checking
access authorization
z/OS 28
database authorization
CICS 28
IMS 54
CICS
abend codes 167
activating trace sessions 144
database
recovery considerations 40
DB2 considerations 10, 40
destination control table (DCT)
printing, DBCS 34
sample entry 39
transient data queue name 30
diagnostic control options 117
DL/I considerations 10, 40

CICS (*continued*)
ELAC transaction 117
ELAM transaction 111
ELAN transaction 112
ELAU transaction 114
EZEP transaction 31
EZEZ transaction 31
file descriptions 29
installation considerations 10
mode, pseudoconversational,
residency consideration 36
monitoring and tuning 10
new modules 87
parameter group
print file 29
parameter group, creating and
maintaining 121
PCT (program control table), printing,
DBCS 34
performance
considerations 35
preparation 85
print destination, specifying in
DCT 39
printing
buffer size 32
DBCS (double-byte character
set) 31, 34
DCT (destination control
table) 30, 31
destination control table
(DCT) 30, 31, 39
double-byte character set
(DBCS) 31
EZEP transaction 31
EZEZ transaction 31
file description 29
form-feed 31
FORMFD=NO parameter 31
FZETPRT program 31, 39
parameter, PRTTYP 33
PCT (program control table),
FZETPRT program 34
printer destination 39
program control table (PCT) 34,
39
PRTBUF parameter 31
PRTMPP parameter 31
PRTTYP parameter 31
SEND command 32
terminal control table (TCT),
entry 39
transient data queue 39
processing mode
types 30
program control table (PCT)
DTB=YES and DBP value 39
printing, DBCS 34
pseudoconversational
processing mode 36
programs and residency 36

CICS (*continued*)

- residency
 - considerations 35, 36
 - general rules 35
- resource tables 85
- security considerations 10
- spool files 11
- startup JCL 87
- storage facilities used by Enterprise Developer Server 7
- system considerations 29
- temporary storage queues for Enterprise Developer Server 11
- terminal control table (TCT), entry 39
- terminal printing 31
- transaction
 - EZEP 31
 - EZEZ 31
 - PR01 transient data queue 39
- transactions, passing transient data between 39
- transient data queue 30
- utilities
 - (see also CICS, utilities) 111
 - diagnostic control facility, ELAM 111
 - diagnostic control options, ELAC 117
 - diagnostic message printing, ELAU 114
 - menu 11
 - new copy utility, ELAN 112
- CICS, PRGM transaction 122
- CICS, utilities, change diagnostic control options 118
- CICS, utilities, default diagnostic control options 119
- CICS, utilities, parameter group utility, PRGM 122
- CICS, utilities, PRGM, parameter group utility 122
- CICS, utilities, view diagnostic control options 118
- CICS/ESA
 - monitoring and tuning 10
- clearing records from databases 55, 56
- client/server 87
- CLIST
 - modifying 93
 - templates 93
- CMPAT parameter, IMS 48
- COBOL
 - abend codes 165
 - abend dumps 139
 - abends under CICS 168
 - DATA compiler option 6, 11
 - status key values 157
 - WSCLEAR option 16
- COBOL dynamic storage
 - for Enterprise Developer Server 6
- COBOL/370 runtime messages 164
- codes
 - abend, IMS 166
 - return
 - SQL 153

- commit point
 - IMS 50
- common system return codes 153
- compiler options that affect performance 23
- considerations
 - batch
 - DB2 44
 - DL/I 44
 - program runtime support 44
 - system 43
 - customization 15
 - database integrity
 - DB2, CICS 40
 - DB2, IMS 54
 - IMS 49
 - database recovery
 - IMS 49
 - DB2
 - CICS 40
 - DB2 database recovery
 - CICS 40
 - IMS 54
 - DL/I
 - CICS 40
 - IMS 54
 - z/OS batch 44
 - DL/I database integrity and recovery
 - CICS 40
 - IMS 55
 - z/OS batch 45
 - message format services 61
 - performance
 - CICS 35
 - compiler options 23
 - IMS 50, 52
 - link pack area 52
 - printing
 - IMS 49
 - recovery
 - IMS 49
 - residency
 - CICS 35
 - system
 - backing up data 28
 - CICS 29
 - DBCS 27
 - extended addressing 27
 - IMS 47
 - tuning IMS 53
 - z/OS/XA 27
- control block 140
- control character, American National Standards, printer 29
- control region in IMS 13
- controlling error reporting
 - CICS 131
 - IMS 131
- conversational processing mode, CICS 30
- creating
 - MFS control blocks 99
- CREATX service routine, print destination 30
- customizing
 - Enterprise Developer Server 15
 - JCL procedures 16

D

- DATA compiler option 6, 11, 12
- data file
 - backing up 28
 - defining 36
 - program, defining 24
- data queue
 - extrapartition 39
 - intrapartition 39
 - transient 38
- data set
 - bind command 66
- CICS
 - PCT entries 67
 - PPT entries 67
 - DB2 database request module 66
 - DBRMLIB 66
 - EZEBIND 66
 - EZEJCLX 66, 107
 - EZEPCT 67
 - EZEPPT 67
 - EZEPRINT 43, 89, 104
 - EZESRC 66
 - load library 66
 - loading KSDS files 26
 - object library 66
 - SYSLIN 66
 - SYSLMOD 66
 - user 66
- database
 - expanding 57
 - multiple
 - work 60
 - request module, DB2 66
 - work
 - clearing records 55
 - expanding 57
 - maintaining 55
- database authorization
 - checking
 - IMS 54
 - z/OS 28
- database integrity and recovery
 - considerations
 - DB2
 - CICS 40
 - IMS 54
 - DL/I
 - CICS 40
 - IMS 55
 - z/OS batch 45
 - IMS 49
- DB Tools product 53
- DB2
 - application plan 14
 - checking authorization
 - IMS 54
 - z/OS 28
 - considerations
 - CICS 10, 40
 - IMS 12
 - TSO 9
 - database
 - request module data set 66
 - table space 58, 59

- DB2 (*continued*)
 - database integrity and recovery
 - considerations
 - CICS 40
 - IMS 54
 - precompile
 - messages 151
 - programs
 - bind 28
 - work database
 - clearing records 56
 - expanding the table space 58
 - IMS 13
 - multiple 60
- DBCS (double-byte character set)
 - data on a non-DBCS terminal 102
 - hardware attributes 27
 - printing
 - CICS 31, 34, 39
- DBRMLIB 66
- DCAPRMG file, parameter group for
 - FZETPRT 31
- DCT (destination control table)
 - entries 86
 - printing, DBCS 34
 - sample entry 39
 - transient data queue 30, 31
 - trigger level 31
- DD statements by file type 94
- deactivating a trace session 149
- default
 - print destination, IMS 49
- defining
 - alternate index 24
 - data files 36
 - ESDS (serial) data set 24
 - KSDS (indexed) data set 24
 - program data files 36
 - program specification block (PSB)
 - IMS 48
 - RRDS (relative) data set 24
 - transient data
 - extrapartition 39
 - intrapartition 39
 - transient data files
 - extrapartition 39
 - intrapartition 39
 - transient data queues
 - extrapartition 38, 39
 - intrapartition 38, 39
 - VSAM data files 24
- deleting old records from the work
 - database 55
- descriptions
 - CICS files 29
 - IMS files 47
- destination control table (DCT)
 - entries 86
 - printing, DBCS 34
 - sample entry 39
 - transient data queue 30, 31
 - trigger level 31
- destination, default print, IMS 49
- detecting errors 129
- determining position in program 141
- DFHAC2016 messages 167
- DFHAC2206 messages 167

- DFS057I error message 165
- DFS064 error message 165
- DFS182 error message 165
- DFS2082 error message 103, 166
- DFS2766I error message 103, 166
- DFS555I error message 102, 166
- diagnosing problems 129
- diagnostic control
 - facility
 - CICS utilities 111
 - options
 - change or view defaults 120
 - change or view options 118
 - ELAC transaction 117
- diagnostic message print utility,
 - ELAU 114
- disk storage requirements
 - for Enterprise Developer Server 7
- DL/I
 - considerations
 - CICS 10, 40
 - IMS 54
 - TSO 9
 - z/OS batch 44
 - integrity and recovery considerations
 - CICS 40
 - IMS 55
 - z/OS batch 45
 - status codes 155
 - work database
 - clearing records 55
 - expanding the database 57
 - in IMS 13
 - multiple 60
- double-byte character set (DBCS)
 - hardware attributes 27
 - printer 39
- DSNX100I messages 151
- dumps
 - snap, listing file on IMS 47
- dynamic
 - interface plan 28
 - storage utilization in Enterprise
 - Developer Server 6

E

- ELA2SSQL module 51
- ELA2SSQX module 51
- ELA2SSQY module 51
- ELAC, diagnostic control options 117
- ELAC02 panel, change or view
 - options 118
- ELAC04 panel, change or view
 - defaults 120
- ELACJWKD member 60
- ELADIAG file 47
- ELAM, CICS utilities menu 111
- ELAN, new copy utility 112
- ELANCCCC module 51
- ELAPCB macro 48
- ELAPRINT system output file 43, 47
- ELARPRTM load module 51
- ELARPRTR load module 51
- ELARSDCB load module 51
- ELASNAP file 47

- ELAU, diagnostic message printing
 - utility 114
- ELAWKJC2 member 56
- ELAWKJCD member 55
- ELAWORK work database PCB 48
- ELAWORK2 DL/I work database 60
- emulating IBM 3270 devices 27
- Enterprise Developer Server
 - abend dumps 139
 - application load module storage 5
 - applying maintenance 3
 - COBOL dynamic storage 6
 - control block 140
 - control options by transaction 118
 - customizing JCL procedures 16
 - DB2 considerations
 - CICS 10
 - IMS 12
 - IMS work database 13
 - TSO 9
 - default control options 120
 - diagnostic control options 117
 - disk storage requirements 7
 - DL/I considerations
 - CICS 10
 - IMS work database 13
 - TSO 9
 - dynamic storage 6
 - error 133
 - extended addressing 27
 - generated programs
 - using with PL/I programs 16
 - IMS/ESA exploitation 12
 - installation considerations
 - CICS 10
 - IMS 11
 - preparing to install 3
 - load module
 - reentrant 5
 - storage 5
 - storage estimates, statically
 - linked 6
 - new copy 112
 - performance considerations 15
 - security considerations
 - all systems 15
 - CICS 10
 - IMS 12
 - storage facilities for CICS, using 7
 - storage requirements 5
 - temporary storage queues 11
 - utilities
 - diagnostic control facility
 - (ELAM) 111
 - diagnostic control options
 - (ELAC) 117
 - diagnostic message printing utility
 - (ELAU) 114
 - for CICS 11
 - new copy (ELAN) 112
 - virtual storage requirements 5
 - work database space for segmented
 - applications 7
 - WSCLEAR option for COBOL,
 - specifying 16
- Enterprise Developer Server, utilities,
 - parameter group utility, PRGM 122

- Enterprise Developer Server, utilities, PRGM, parameter group utility 122
- ERRDEST message queue 134
- error
 - detection 129
 - file
 - definition 130
 - I/O 129
 - processing 130
 - message
 - file 47
 - panel 133
 - reporting 131
 - IMS 131
 - in IMS 102
 - summary 132
- ESDS (serial) define cluster 24
- expanding
 - the table space (DB2) 58
 - work database 57
- express alternate PCB 48
- extended addressing considerations
 - z/OS 27
- external work file, backing up 28
- extrapartition transient data, defining 39
- EZEBIND data set 66
- EZEDESTP special function word 43
- EZEJCLX data set 66
- EZEP transaction 29, 31, 39
- EZEPCT data set 67
- EZEPT data set 67
- EZEPRINT data set
 - IMS 49
 - specify as PRO1 39
- EZEPRMG file
 - CICS 29
 - parameter group for FZETPRT 31
- EZEROLLB service routine
 - IMS 50
- EZESRC data set 66
- EZETRACE data set 43
- EZEZ transaction 31, 39

F

- FCT (file control table)
 - entries 87
 - user data file 37
- file
 - control table (FCT)
 - described 87
 - default message queue, IMS 47
 - definition errors 130
 - description
 - CICS 29
 - IMS 47
 - descriptions 29
 - error message 47
 - from generation 67
 - I/O errors 129
 - parameter group 29
 - processing errors 130
 - snap dump listing, IMS 47
 - system output 43
 - trace 43
- file control table (FCT)
 - entries 87

- file control table (FCT) *(continued)*
 - user data file 37
- form feed
 - order (see American National Standards printer control character) 31
 - printing 31
- FORMFD parameter
 - option=NO, forms alignment 31
 - parameter group for EZEP or EZEZ 33
 - used with FZETPRT program 31
- function
 - new copy 34
 - preload, IMS 50
- FZETPRT program 34
 - DBCS considerations 34
 - EZEP or EZEZ transaction 39
 - special parameter group 31
 - terminal printing support in CICS 31
- FZEZREBO utility, initializing indexed files 26

G

- generated applications
 - with PL/I programs 16
- generating
 - application control block 48

H

- hardware attributes for DBCS 27

I

- IBM 3270 device, emulating 27
- IBM 5550 family of terminals 27
- IDCAMS program
 - BLDINDEX command 24
 - DEFINE PATH command 24
 - loading indexed files 27
 - REPRO command 24, 26
- IGYOP3091W error message 152
- IGYOP3093W error message 152
- IGYOP3094W error message 152
- IGYPA3013W error message 152
- IGYPG3113W error message 152
- IGYPS2015I error message 152
- IGYPS2023I error message 152
- IGYSC2025W error message 152
- IGZ033S error message 164
- IGZ064S error message 164
- IGZ066S error message 165
- IGZ075S error message 165
- improving
 - performance 53
 - library lookaside (LLA) 24
 - link pack area (LPA) 24
 - virtual lookaside facility (VLF) 24
 - response time 52
- IMS
 - commit point 50
 - control region 13
 - database
 - authorization checking 54

- IMS *(continued)*
 - database *(continued)*
 - integrity considerations 49
 - recovery considerations, DB2 54
 - recovery considerations, DL/I 49
 - DB2 considerations 12
 - default
 - message queue file 47
 - print destination 49
 - DL/I considerations 54
 - ELAPCB macro 48
 - error
 - controlling, generation
 - options 131
 - messages 102
 - reporting 102
 - file descriptions 47
 - HIPERSPACE buffer usage 52
 - installation considerations 11
 - integrity considerations, DB2 54
 - log format 135
 - logical unit of work 55
 - monitoring and tuning 12, 53
 - new modules 100
 - performance considerations 52
 - preload function 50
 - preloading
 - Enterprise Developer Server
 - modules 51
 - program modules 52
 - preparation 97
 - processing modes 49
 - program specification block,
 - defining 48
 - residency considerations 50
 - rollback 50
 - runtime
 - abend codes 166
 - messages 165
 - security considerations 12
 - segmented mode 49
 - single-segmented mode 49
 - snap dump listing file 47
 - system considerations 47
 - system definition
 - batch program as an MPP 98
 - batch-oriented BMP program 99
 - general 13
 - interactive program 97
 - parameters 97
 - transaction-oriented BMP 99
 - system printing considerations 49
 - work database considerations
 - DB2 13
 - DL/I 13
- IMS DC monitor facilities 12
- IMS Performance Analysis and Reporting System 12
- IMS/ESA exploitation 12
- IMS/VS, message format service (MFS)
 - Control Blocks 53
- IMSPARS 12, 53
- indexed (KSDS) data set
 - define cluster 24
 - loading 26
- installation considerations
 - preparing to install 3

integrity considerations, database

DB2

CICS 40

IMS 54

DL/I

CICS 40

IMS 55

z/OS batch 45

IMS 49

intrapartition transient data

defining 39

J

JCL

by environment 93

examples of runtime 90, 91, 92, 104, 105

modifying 93, 94

modifying runtime 94

tailoring before generation 93

templates 93

job stream data set

runtime 66

K

KSDS (indexed) define cluster 24

L

LE

runtime messages 164

library

backup 28

production copies, maintaining
backup 107

link pack area

loading 52

performance considerations 52

listing file

IMS, snap dump 47

load library data set 66

load module

preloading 51

storage for Enterprise Developer
Server 5

storage for Enterprise Developer
Server application 5

loading

modules into link pack area 52

logical unit of work (LUW)

IMS 54, 55

M

macro, ELAPCB 48

maintaining

backup copies of production
libraries 107

work database 55

maintenance, applying to

Enterprise Developer Server 3

map group

format module 71

message

format services

considerations 61

description 27, 61

queue file, default, IMS 47

message format service (MFS) control

blocks in IMS 53

messages

COBOL/370 runtime 164

DFHAC2016 167

DFHAC2206 167

DFS057I 165

DFS064 165

DFS182 165

DFS2082 103, 166

DFS2766I 103, 166

DFS555I 102, 166

DSNX100I 151

IGYOP3091W 152

IGYOP3093W 152

IGYOP3094W 152

IGYPA3013W 152

IGYPG3113W 152

IGYPS2015I 152

IGYPS2023I 152

IGYSC2025W 152

IGZ0033S 164

IGZ0064S 164

IGZ0066S 165

IGZ0075S 165

IMS runtime 165

preparation 151

runtime

COBOL/370 164

IMS 165

z/OS 167

z/OS runtime 167

MFS

control blocks 99

mode

CICS execution, performance
considerations 36

processing

CICS 30

IMS 49

models

JCL 93

modifying

IMS system definition parameters 97

JCL or CLISTs 93

runtime

JCL 94

modules

CICS 87

IMS 100

in memory 23

loading into link pack area 52

preloading 52

monitoring and tuning

CICS 10

IMS system 12, 53

performance 53

moving prepared programs

z/OS 107

multiple work databases 60

N

new copy

function 34

new copy utility 112

new copy utility, ELAN 112

new modules

CICS 87

IMS 100

nonsegmented processing mode,

CICS 30

O

object library data set 66

objects generated

application COBOL program 70

batch print services program 71

BIND command 70

from generation 67

map group format module 71

online print services program 71

runtime

JCL 70

table program 71

online print services program 71

option

preloading

Enterprise Developer Server

modules, IMS 51

program modules, IMS 52

recovery 34

SPA 49

output of program generation 67

P

panels

Parameter Group Definition
(PRGM02) 123

Parameter Group Specification
(PRGM00) 122

panels, Parameter Group List Display
(PRGM01) 123

parameter

group associated with FZETPRT
program

DCAPRMG file 31

EZEPRMG file 31

resident 35

WORK in ELAPCB 48

Parameter Group Definition panel
(PRGM02) 123

parameter group file, EZEPRMG data set,
CICS 29

Parameter Group List Display panel
(PRGM01) 123

Parameter Group Specification panel
(PRGM00) 122

passing transient data between CICS
transactions 39

PCT (program control table)

entries 86

FZETPRT program 34

performance

considerations 23

CICS 35

- performance (*continued*)
 - considerations (*continued*)
 - general 15, 23
 - IMS 50, 53
 - IMS/ESA 52, 53
 - z/OS batch 45
 - generation and compiler options 23
 - HIPERSPACE buffers for IMS 52
 - library lookaside (LLA) 24
 - limiting MFS control blocks 53
 - link pack area 23
 - monitoring and tuning
 - IMS 12, 53
 - preload modules 100
 - RES(YES) parameter, RDO DEFINE PROGRAM command 86
 - tuning IMS 53
 - virtual lookaside facility (VLF) 24
- Performance Analysis and Reporting System (PARS) 53
- PL/I programs 16
- plan, DB2 28
- PPT (processing program table)
 - defining programs to CICS 67
 - entries 85
- PR01 transient data queue 39
- precompile messages
 - BIND 151
 - DB2 151
- preloading
 - Enterprise Developer Server modules, IMS 51
 - objects, IMS 50
 - print services
 - description 100
 - module 51
 - program 52
 - program 100
 - program modules 51, 52
 - service module 51
 - table modules 51, 100
- preparation
 - abend codes 151
 - messages 151
- preparing
 - and running programs
 - CICS 85
 - IMS 97
 - z/OS batch 89
 - to install Enterprise Developer Server 3
- PRGM00 (Parameter Group List Display panel) 123
- PRGM00 (Parameter Group Specification panel) 122
- PRGM02 (Parameter Group Definition panel) 123
- print destination
 - CICS, specifying in DCT 39
 - default
 - IMS 49
- print file, utilities 29
- print services program
 - object of generation 71
 - preloading 52
- printing
 - buffer size 32

- printing (*continued*)
 - CICS
 - considerations 29
 - file descriptions 29
 - CICS, destination control table (DCT) 31
 - considerations
 - IMS 49
 - CREATX call for print destination 30
 - DBCS (double-byte character set), printer 39
 - DCT (destination control table)
 - transient data queue name 30
 - trigger level 31
 - default, print destination 30
 - destination control table (DCT)
 - transient data queue name 30
 - trigger level 31
 - destination, using CREATX call 30
 - diagnostic information
 - CICS 136
 - IMS 134
 - EZEP transaction 31
 - EZEZ transaction 31
 - file descriptions, CICS 29
 - form-feed 31
 - FORMFD=NO parameter 31, 33
 - FZETPRT program 31
 - parameter
 - FORMFD 31, 33
 - group associated with FZETPRT program 31
 - PRTBUF 31
 - PRTMPP 31, 33
 - PRTTY 31, 33
 - PCT (program control table), FZETPRT program 34
 - PR01 transient data queue 39
 - print destination, default 30
 - printer destination 39
 - program control table (PCT), FZETPRT program 34
 - SEND command 32
 - transient data
 - at a terminal device 39
 - transient data queue 30, 39
- problem
 - diagnosis 129
- processing
 - batch 43
- processing mode
 - CICS
 - types 30
 - IMS 49
- processing program table (PPT)
 - entries 85
- production libraries, maintaining copies for backup 107
- profile block
 - program 140
- program
 - bind DB2 28
 - data files, defining 24
 - entries 85
 - module, preloading 51
 - preloading 52
 - profile block 140

- program (*continued*)
 - return codes 159
- program communication block (PCB)
 - alternate 48
 - ELAPCB macro 48
- program control table (PCT)
 - DTB=YES and DBP value 39
 - entries 86
 - FZETPRT program 34
- program specification block (PSB)
 - defining 48
 - generation 48
- PRTBUF parameter
 - specifying print buffer size 31
 - using with the FZETPRT program 31
- PRTMPP parameter
 - specifying maximum print positions 33
 - using with FZETPRT program 33
- PRTTY 31, 33
- PRTTY parameter
 - DBCS printing 33
 - using with the FZETPRT program 31
- pseudoconversational
 - processing mode
 - CICS 30, 36

R

- RCT 87
- RDO (resource definition online), generation output 69
- RDO CEDA transaction 85
- reading transient data from tape 39
- recovery
 - options
 - specifying 34
- recovery considerations
 - DB2
 - CICS 40
 - IMS 54
 - DL/I
 - CICS 40
 - IMS 55
 - z/OS batch 45
 - IMS 49
- reentrant code 23
- reentrant load module storage estimates for Enterprise Developer Server 5
- relative (RRDS) define cluster 24
- reporting
 - errors 131
 - problems 149
- request module, DB2 66
- residency
 - considerations
 - CICS 35
 - IMS 50
 - general rules, CICS 35
- resident
 - parameter 35
 - programs 88
- resource
 - control table 87
 - tables for CICS 85
- Resource Measurement Facility II 13, 53
- response time, improving 52

- return codes
 - SQL 153
 - system 153
- RMF 13, 53
- rollback
 - IMS 50
- RRDS, data set definition 24
- running
 - main programs under z/OS batch 89
 - programs under IMS 101
- running under
 - CICS 88
 - IMS
 - BMP with DB2 105
 - main batch as BMP 104
 - main program under BMP 103
 - z/OS batch
 - main batch with DL/I 90
 - main batch with no database 90
 - main batch with no DB2 90
- runtime
 - JCL 70, 94
 - job stream data set 66
 - messages
 - COBOL/370 164
 - IMS 165
 - z/OS 167
 - messages, LE 164

S

- sample JCL
 - BMP with DB2 105
 - IMS BMP program 104
 - RCT entry 87
 - z/OS Batch with DB2 Access 90
 - z/OS Batch with DB2 and DL/I 92
 - z/OS batch with DL/I Access 91
 - z/OS batch without DB2 90
- saving storage space 52
- security considerations
 - CICS 10
 - general 15
 - IMS 12
- segmented processing mode
 - CICS 30
 - IMS 49
- SEND command, printing 32
- serial (ESDS) define cluster 24
- service module, preloading 51
- services, message format 27
- sharing modules 52
- single-segment mode, IMS 49
- snap dump listing file, IMS 47
- spa build descriptor option 49, 98
- spool files, CICS 11
- SQL
 - considerations 28
 - return codes 153
- starting
 - IMS programs
 - /FORMAT command
 - (transaction) 101
 - directly (main) 101
 - MPPs (transactions) 101
- startup JCL for CICS 87
- statistics, performance 53

- status 13
 - codes
 - DL/I 155
 - key values, COBOL 157
 - storage requirements
 - for Enterprise Developer Server
 - COBOL dynamic storage 6
 - subsystem ABEND dumps 139
 - support for DBCS terminals 27
 - SYSLIN 66
 - SYSLMOD 66
 - SYSOUT system output file 43
 - SYSPRINT system output file 43
 - system
 - abend codes 163
 - considerations
 - CICS 29
 - general 23
 - IMS 47
 - definition, IMS 13
 - output file 43
 - return codes 153
- SYSUDUMP system output file 43

T

- table
 - modules, preloading 51
 - preloading 52
 - program 71
 - space
 - expanding 58, 59
 - requirements 58
- TCT (terminal control table) 39
- templates
 - CLIST 93
 - JCL 93
- temporary storage queues 11
- terminal control table (TCT) 39
- terminal printing
 - CICS 31
- trace facility 143
- trace file 43
- tracing
 - activating 144
 - deactivating 149
- transaction
 - entries 86
 - error 132
- transient data
 - defining extrapartition 39
 - printing 39
 - queue
 - defining 38
 - printing, CICS 30
 - TYPE=INTRA entry in DCT 31
 - reading from tape 39
- tuning
 - IMS 12, 53

U

- unit of work, logical
 - IMS 54, 55
- user data set 66

- using
 - data build descriptor option 11
 - generated applications with PL/I
 - programs 16
 - multiple work databases 60
 - remote files, CICS 38
 - using spool files 11
- utilities
 - diagnostic control options
 - (ELAC) 111, 117
 - diagnostic message printing
 - (ELAU) 114
 - for CICS with Enterprise Developer
 - Server 11
 - IMS diagnostic message print
 - new copy (ELAN) 112
 - utilities, diagnostic, message print utility,
 - CICS 114
 - utilities, parameter group utility,
 - PRGM 122

V

- virtual storage
 - considerations and residency 35
 - requirements
 - Enterprise Developer Server 5
- VSAM
 - data set definition 24
 - defining an alternate index 24
 - file loading 26
 - indexed (KSDS) data set 24
 - relative (RRDS) data set 24
 - serial (ESDS) data set 24
 - status codes 155

W

- warnings
 - empty KSDS data set, VSAM
 - restriction 26
 - PRTMPP parameter, line skip
 - malfunction 33
- work database
 - clearing records 55
 - deleting old records 55
 - ELAPCB macro 48
 - expanding 57
 - IMS 13
 - maintaining 55
 - multiple 60
 - space for segmented applications 7
- WORK parameter in ELAPCB 48
- WSCLEAR option for COBOL 16

Z

- z/OS
 - DB2 considerations for Enterprise
 - Developer Server 9
 - DL/I considerations 9
 - DL/I considerations for Enterprise
 - Developer Server 9
 - installation considerations 3
 - preparation 89
 - runtime messages 167

z/OS batch	
DL/I considerations	44
z/OS/XA considerations	27



Program Number: 5655-I57

Printed in USA

SC31-6306-03

